

Designer Displaying Images from the uSD Card - GC FAT16

DOCUMENT DATE:1st MAY 2020DOCUMENT REVISION:1.2



W W W . 4 D S Y S T E M S . C O M . A U

Description

This application note shows how to display an image in Designer. Before getting started, the following are required:

• Any of the following 4D Picaso display modules:

gen4-uLCD-24PTgen4-uLCD-28PTgen4-uLCD-32PTuLCD-24PTUuLCD-28PTUuVGA-III

and other superseded modules which support the Designer environment.

• The target module can also be a Diablo16 display

gen4-uLCD-24D series	gen4-uLCD-28D series	gen4-uLCD-32D series
gen4-uLCD-38D series	gen4-uLCD-43D series	gen4-uLCD-50D series
gen4-uLCD-70D series		
<u>uLCD-35DT</u>	uLCD-43D Series	<u>uLCD-70DT</u>

Visit <u>www.4dsystems.com.au/products</u> to see the latest display module products that use the Diablo16 processor.

- <u>4D Programming Cable</u> / <u>μUSB-PA5/uUSBPA5-II</u> for non-gen4 displays (uLCD-xxx)
- <u>4D Programming Cable & gen4-IB / 4D-UPA / gen4-PA</u> for gen4 displays (gen4-uLCD-xxx)
- micro-SD (µSD) memory card
- <u>Workshop 4 IDE</u> (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	5
Create a New Project	5
Design the Project	5
The DAT and GCI Files	5
Using the Graphics Composer	6
Set the Screen Size	7
How to Add an Image	8
Add More Images	11
Save the Graphics Composer File	11
Build the GCI and DAT Files	12
View the DAT File	12
Insert the μ SD Card to the PC	14
Copy the DAT and GCI Files to the uSD Card	14
Insert the μ SD Card to the Display Module	14
Understanding the Demo Code	15
Checking the uSD Card	15
Clearing the Screen	15
Defining Variables	15

The Image Control	15
Use of Index Values to Display Images	16
Displaying Images	17
The for Loop	17
Build and Upload the Project	
Proprietary Information	
Disclaimer of Warranties & Limitation of Liability	

Application Overview

For this application note, the images to be displayed on the screen will come from the uSD card. These images need to be saved onto the uSD card in a format readable by 4D-LAB's processors. The images therefore are not saved onto the uSD card in their original format (jpeg/gif/png/bmp/etc) but are converted and combined by Workshop into a single graphics file, which is called the "GCI" file. The same concept applies to animations, movie clips, widgets, and other objects (common animation and movie file formats are flv, mov, mpeg, etc). GCI stands for Graphics Composer Image and it has its own format. The Graphics Composer or GC is a program used by Workshop to convert multimedia graphics files into a GCI file.

When designing in the Designer environment, the user will need to manually use the Graphics Composer. The general procedure is:

- 1. Add the images to the GC window.
- 2. Insert a uSD card into the PC.
- 3. Build or generate the GCI file and other supporting files and copy them to the uSD card.
- 4. Write the 4DGL code for accessing the uSD card and displaying the images on the screen.
- 5. Compile the code and upload the program to the display.
- 6. Unmount the uSD card from the PC and insert it to the display module.
- 7. The program should now run on the display module and the images should now be shown.

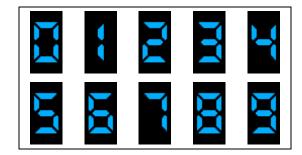
When designing in both the ViSi and ViSi-Genie environments, the Graphics Composer is invisible to the user. Instead, the user sees only the WYSIWYG (What-You-See-Is-What-You-Get) screen during design time. The general procedure is:

- 1. Place the images on the WYSIWYG screen.
- 2. For the codeless ViSi-Genie environment, configure the properties of the image objects using the Object Inspector.
- 3. For the ViSi environment, the code area is beside the WYSIWYG screen. Write the code for the program.
- 4. Click on the Build/Compile and Upload button.
- 5. Workshop will prompt for the uSD card drive to which the GCI file and other supporting files will be copied. Insert the uSD card to the PC and select the correct drive. Workshop transfers the files to the uSD card.
- 6. Workshop uploads the program to the display module.
- 7. Remove the uSD card from the PC and insert it to the display module.
- 8. The program should now run on the display module and the images should now be shown.

For more information on the use of the Visi and ViSi-Genie environments, refer to the relevant application notes. The user will realize that using the WYSIWYG screen is much easier than using the Graphics Composer. For this reason, ViSi is the recommended environment for users intending to design GUIs (Graphical User Interface) thru coding and drag-and-drop methods at the same time.

A simple program is developed for this application note. It flashes the digits 0 to 9 on the screen. The images for the digits can come from image files of

any of the common formats (in this example, png image files are used). The png files are added as image entries into the Graphics Composer window, the graphics file is generated, and a program for accessing this graphics file is written. Attached to this document is a zip file containing all the files used in the project. The digits are shown below.



Manual use of the Graphics Composer is not recommended since the ViSi and ViSi-Genie environments are meant to "automate" the process for the user.

Setup Procedure

For instructions on how to launch Workshop 4, how to open a **Designer** project, and how to change the target display, kindly refer to the section "**Setup Procedure**" of the application note **Designer Getting Started - First Project**

Create a New Project

For instructions on how to create a new **Designer** project, please refer to the section "**Create a New Project**" of the application note **Designer Getting Started - First Project**

Design the Project

The DAT and GCI Files

The images or objects inside the GCI file are arranged in a sequential order – that is according to when the user adds them. A list of these objects is created by the Graphics Composer along with the GCI file. This list, which is another separate file – the "DAT" file, contains the names of the object entries, their locations inside the GCI file, and their initial X/Y positions onscreen. To illustrate:

DAT file – list na of the objects na inside the GCI na

name1 address1 x1 y1 name2 address2 x2 y2 name3 address3 x3 y3 name4 address4 x4 y4 name5 address5 x5 y5



4D-AN-00044

APPLICATION NOTES

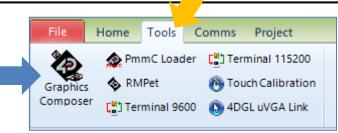


When the user intends to display a hundred images for instance, the Graphics Composer is first used to combine them into a GCI file and to create a DAT file. The user will then write a program to access these two files to be able to display the images on the screen. Below is a screenshot of the Graphics Composer, the use of which is discussed in the following section.

\$ 2	GCI_DEMO1 - Graphics Composer 3	- 🗆 ×
File Quality About		
🖹 🚵 🔒	GC DAT	
Entries TSEGORANGE_18X36.GIF 10-GEAR.GIF EXCLAIM.BMP INFO.BMP WARN.BMP WARN.BMP	7SEGORANGE_18X36.GIF Name: 7SEGORANGE_18X36.GIF Source: 7SEGORANGE_18X36.GIF Position Default X: Default X: 51 Image: Source window Image: Source window Left: 0 Width: 18 Video Attributes Start Frame Start Frame 0 End Frame 11 ms/Frame 255 Bit Depth 16	Screen size
Add	Offset: 0x0 Size: 13,832(0x3608)	● 128 x 128 32 ○ 160 x 128 CL

Using the Graphics Composer

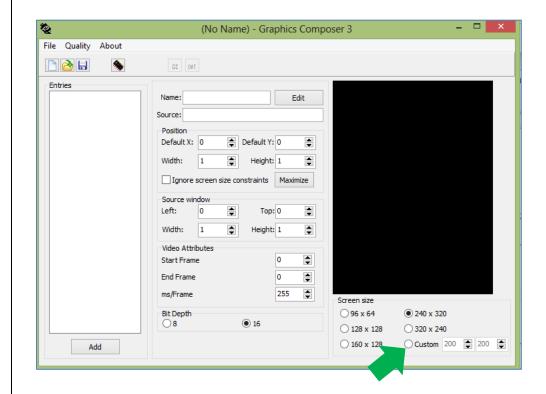
The Graphics Composer has a dedicated user guide which can be downloaded <u>here</u>. In this section only the basics of using the GC are covered. To open the Graphics Composer, go to the Tools menu and click on the Graphics Composer icon.



Depending on the user's PC User Account Control settings, Windows might ask for a confirmation to run the program **gc.exe**. This is the Graphics Composer. Click Yes.

🚱 User Account Control	E		
	Do you want to allow the following program from an unknown publisher to make changes to this computer?		
Program name: gc.exe Publisher: Unknown File origin: Hard drive	on this computer		
Show details	Yes No		
	Change when these notifications appear		

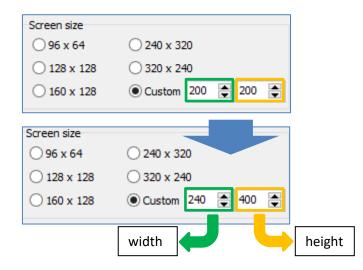
The Graphics Composer now opens.



Set the Screen Size

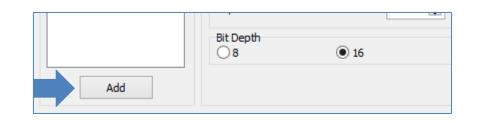
To make the screen size consistent with the selected project display (uLCD-32WPTU Portrait orientation), click on the custom button and change the width from 200 to 240 and the height from 200 to 400. To know the pixel resolution of your display module, consult the datasheet. When creating a new project, the Choose-Your-Product window also displays the pixel resolution of the display module.

CHOOSE YOUR PRODUCT
een LCD Module
480x272



How to Add an Image

To add an image, click on the Add button on the left lower part of the window.



The standard Open window opens. Select the desired image file. The screen will be updated. The image files used in this tutorial are inside the attached zip file.

www.4dsystems.com.au

\$	(No Name) - Graphics Composer 3 – 🗖 🗙
File Quality About	
	GC DRT
Entries digit_0.png	digit_0.png Name: digit_0.png Edit Source: C: Users Position Default X: 0 Default X: 0 Ignore screen size constraints Maximize Source window Left: 0 Top: Width: 32 Height: 62 Image Attributes Tile Tile Height: 2 (Not int size) Tile Height: 2 0 16 Offset: 0x0 Size: 3,974(0xF86)
	Screen size 0 96 x 64 0 240 x 320
	◯ 128 x 128 ◯ 320 x 240
	◯ 160 x 128

In this example, a seven segment display digit is chosen. Notice that under Entries, the image file **digit_0.png** is listed.

٠		
File	Quality	About
	2	
Ent dig	ries it_0.png	

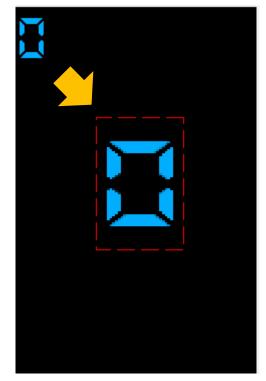
The middle part of the window shows the properties of the image.

digit_0.png			
Name: digi	t_0.png		Edit
Source: C:\	Jsers\4d-doff	Pictures (bi	uttons\digit_0.
Position			
Default X:	0	Default Y:	0
Width:	32	Height:	62
Ignore	screen size co	onstraints	Maximize
Source wir	ndow		
Left:	0	Top:	0
Width:	32	Height:	62
Image Att	ributes		
Tiled	Tile Width:	2	(Not int size)
	The Widdh	-	
	Tile Height:	2	(Not int size)
Bit Depth		-	
08		16	
Offset: 0x0 Size: 3,974(0xF86)			

The object entry name can be changed by removing the file extension.

_digit_0.	png	
Name	digit_0.png	Edit
digit_0		
Name:	digit_0	Edit

The user can drag and resize the image to the desired location and dimensions.



4D-AN-00044

APPLICATION NOTES

The properties will be updated accordingly. The figure below shows the new X/Y position and dimension properties of the image.

_digit_0						
Name:	digit	_0			Edit	
Source:	C:\L	Jsers\4	4d-doff	\Pictures\b	uttons	digit_0.
Positio	n					
Defaul	tX:	75	▲ ▼	Default Y:	115	•
Width:		94	•	Height:	122	
🗌 Ign	ore	screen	size co	onstraints	Maxin	nize

Add More Images

Add the rest of the **.png** files (digits 1 to 9). Note that the files have the same pixel dimensions. Resize and drag all the entries to the same size and location on the screen. It is also possible to directly edit the property values. The objective here is to have all the entries, which have a common size, located at a common position on the screen. A program will then be written to display each of these image entries. When finished, the Graphics Composer window will look similar to that shown below.

2	ssd - Graphics Composer 3	- 🗆 🗙
File Quality About		
🖹 🎦 🔒	EC DAT	
Entries	digit_9	
digit_0 digit_1	Name: digit_9 Edit	
digit_2	Source: digit_9.png	
digit_3 digit_4	Position	
digit_5 digit_6	Default X: 75 Default Y: 115	
digit_7 digit_8	Width: 94 ਦ Height: 122 퉂	
digit_9	Ignore screen size constraints Maximize	
	Source window	
	Left: 0 😴 Top: 0 🚔	
	Width: 32 🗭 Height: 62	
	Image Attributes	
	Tiled Tile Width: 2 (Not int size)	
	Tile Height: 2 😭 (Not int size) 5	Screen size
	Bit Depth	○ 96 x 64 ○ 240 x 320
	8 16	○ 128 x 128 ○ 320 x 240
Add	Offset: 0x32A00 Size: 22,942(0x599E)	◯ 160 x 128
Add		

Save the Graphics Composer File

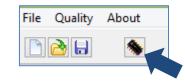
Click on the Save icon under the File menu.



The Save As window opens. Type in the desired file name and click Save. The file in this example is saved with the name "**ssd**". Note that the file extension is "**.gcs**".

Build the GCI and DAT Files

Click on the Build icon.



The Build Output window appears. Choose the first option for the build type. Select a convenient folder for saving the GCI and DAT files. By default, the folder where the Graphics Composer file was saved is selected. The DAT and the GCI files will be transferred later to a FAT-formatted uSD card. Click OK when finished.

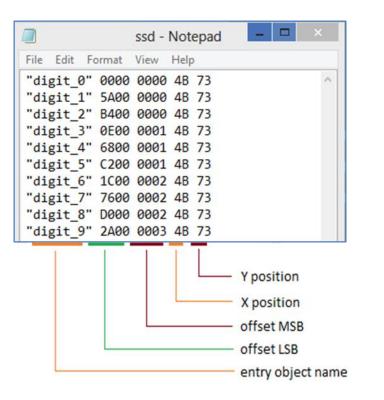
2	Build outpu	t –		×
Build Type				
🖲 4DGL -	GCI - FAT Selected Folder	r		
0 4DGL -	uSD Raw - GCI at Specifie	d Offset		
Drive:	¥			
Folder: C:\	Jsers\Public\Documents\4	D Labs\De	esigner l	
File:				~
Sector offse	t (0x or \$ for hex) 0x0			
Approx Spa	ce required: 20.50 KB (41	Sectors)		
🖌 ок			🗙 Ca	ncel
FAT to C:\U	sers\Public\Documents\4) Labs\De	signer D	off\

View the DAT File

To view the DAT file (which is a list of the object entries inside the GCI file), click on the DAT icon.



The DAT file opens with Notepad. The contents are labelled below.



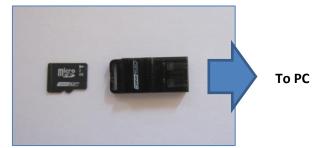
Offset is the location of the entry in the GCI file. Since **digit_0** is the first entry, it has an offset of 0x00000000. Notation is in hexadecimal. This also applies to the X and Y coordinates.

Note that **digit_1** has an offset of **0x00005A00**. The offset of an object entry may depend on the size of the previously added entry. Also, each new entry begins on a sector (512 byte) boundary. The lower middle part of the Graphics Composer window shows the size and offset of a highlighted entry.

digit_0.png			
Name: digit	t_0.png		Edit
Source: C:\	Jsers\4d-doff	Pictures b	uttons\digit_0.
Position			
Default X:	0	Default Y:	0
Width:	32 🚔	Height:	62 🚔
Ignore :	screen size co	onstraints	Maximize
Source win	dow		
Left:	0	Top:	0
Width:	32	Height:	62
Image Attr	ibutes		
Tiled	Tile Width:	2	(Not int size)
	Tile Height:	2	(Not int size)
Bit Depth			
08		16	
Offset: 0x0	Size: 3,974(0xF86)	

Insert the μSD Card to the PC

Insert the μ SD card into the USB adaptor and plug the USB adaptor into a USB port of the PC.



OR insert the μ SD card into a μ SD to SD card converter and plug the SD card converter into the SD card slot of the PC.

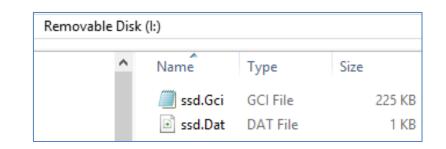


Check if the μ SD card is mounted, here it is mounted as drive I:



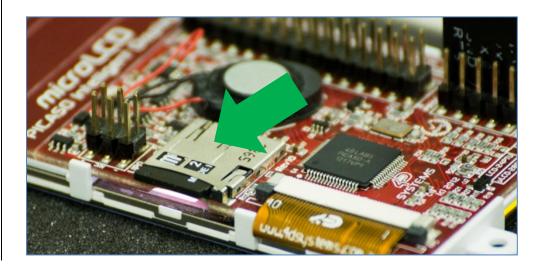
Copy the DAT and GCI Files to the uSD Card

Go to the folder where the DAT and GCI files are saved. Copy these to the uSD card.



Insert the µSD Card to the Display Module

Properly disconnect the μ SD card from the PC and plug it to the μ SD Card slot of the display module. The next step now is to create a program that will access the DAT and GCI files and display the images.



Understanding the Demo Code

Attached is a ZIP file containing the Designer file, the DAT, the GCI, and the image files used in this application note. Open the Designer file to have a better view of the code. An explanation now follows, giving emphasis to the more important parts.

Checking the uSD Card

It is necessary to check that the uSD card is mounted before using it. The block containing lines 7 to 14 defines the behaviour of the screen while waiting for a μ SD card to be mounted on the slot. If no uSD card is detected, the screen will constantly flash the message "**Drive not mounted...**". The user can consult the <u>Picaso Internal Functions Manual</u> for a detailed explanation of the functions used in this block.

7	<pre>if (!(disk:=file Mount()))</pre>
8	<pre>while(!(disk :=file_Mount()))</pre>
9	<pre>putstr("Drive not mounted");</pre>
10	pause(200);
11	gfx_Cls();
12	pause(200);
13	wend
14	endif

Clearing the Screen

The function gfx_Cls() used in lines 11 and 15 is used to clear the screen.

5 gfx_Cls(); //clear the screen

Defining Variables

Lines 17 to 19 show how variables are declared in 4DGL. The purpose of these variables will be explained later.

17	<pre>var i, handle, idigit_0, idigit_1, idigit_2, idigit_3;</pre>
18 19	<pre>var idigit_4, idigit_5, idigit_6, idigit_7, idigit_8;</pre>
19	<pre>var idigit_9;</pre>

Lines 21 to 23 show how variables are initialized.

21 22 23	<pre>idigit_0 := 0;</pre>	<pre>idigit_3 := 3;</pre>	<pre>idigit_6 := 6;</pre>
22	idigit_1 := 1;	idigit_4 := 4;	idigit_7 := 7;
23	idigit_2 := 2;	<pre>idigit_5 := 5;</pre>	idigit_8 := 8;
A A			

The Image Control

To access the images inside the GCI file, the function below is used.

handle := file_LoadImageControl("ssd.dat", "ssd.gci", 1);

In the sample code, this function is used in line 26. It requires two files –the DAT file and GCI file. The names of the DAT and GCI files are passed to the function as the first and second arguments. The third argument (mode) defines the manner by which the image control is built. Remember that the GCI file contains the actual graphics to be displayed and that the DAT file is a list of all the individual images in the GCI file. In the DAT file are lines, each of which contains the name of an image entry, its location (or offset) in the memory card, and its initial x and y coordinates when displayed onscreen. The function

file_LoadImageControl("file.dat", "file.gci", mode);

therefore, takes the offsets and x/y coordinates of the images from the DAT file and saves them in to the image control. The program can then use the image control to access the actual images in the GCI file. The function returns a handle (pointer to the memory allocation) to the image control list that has been created. This handle will be used to display an image. To learn more about the function for loading an image control, read section 2.14.28 of the Picaso Internal Functions Manual.

Use of Index Values to Display Images

After having created the image control, it is now possible to display an image using the function

img_Show(handle, index)

The first argument is the handle, which points to the memory allocation of the image control. The second argument is the **index**. Remember that image entries in the GCI file are indexed by the order by which they are added in the Graphics Composer window. The image **digit_0**, for example, has an index value of 0 since it is the first image added. The image **digit_1** has an index value of 1 since it is the second image added. The third image entry has an index value of 2, and so on. In the Graphics Composer:

Sequence		Index
First	image added	0
Second	image added	1

Entries	Third	image added	2
diata o	Fourth	image added	3
digit_0	Fifth	image added	4
digit_1	Sixth	image added	5
digit_2 digit 3	Seventh	image added	6
digit 4	Eighth	image added	7
digit 5	Ninth	image added	8
digit_6 digit_7 digit_8 digit_9	Tenth	image added	9

It is also convenient to think of the index value as the line number of the image entry in the DAT file.

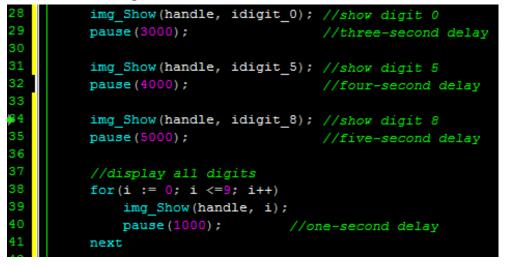
🧾 ssd - Notepad 🗕 🗖	Line	Index
File Edit Format View Help	2	mach
"digit_0" 0000 0000 4B 73	First	0
"digit_1" 5A00 0000 4B 73	Second	1
"digit_2" B400 0000 4B 73	Third	2
"digit_3" 0E00 0001 4B 73	Fourth	3
"digit_4" 6800 0001 4B 73	Fifth	4
"digit_5" C200 0001 4B 73	Sixth	5
"digit_6" 1C00 0002 4B 73	Seventh	6
"digit_7" 7600 0002 4B 73	Eighth	7
"digit_8" D000 0002 4B 73	Ninth	8
"digit_9" 2A00 0003 4B 73	Tenth	9

After knowing how index values are determined, the user can now use variables to define the index values of images. To illustrate:

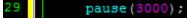
<pre>idigit_0 := 0;</pre>	idigit_3 := 3;	<pre>idigit_6 := 6;</pre>
<pre>idigit_1 := 1;</pre>	<pre>idigit_4 := 4;</pre>	<pre>idigit_7 := 7;</pre>
<pre>idigit_2 := 2;</pre>	<pre>idigit_5 := 5;</pre>	<pre>idigit_8 := 8;</pre>
	idigit_1 := 1;	<pre>idigit_0 := 0; idigit_3 := 3; idigit_1 := 1; idigit_4 := 4; idigit_2 := 2; idigit_5 := 5;</pre>

Displaying Images

The lines below show how to display the images after creating the image control and defining the index values.



Note that a delay is used after an image is displayed.



//three-second delay

This allows the observer to see the image before it is replaced with something else.

The for Loop

The block below shows how a for loop is used in 4DGL.

37	<pre>//display all digits</pre>
38	<pre>for(i := 0; i <=9; i++)</pre>
39	<pre>img_Show(handle, i);</pre>
40	<pre>pause(1000); //one-second delay</pre>
37 38 39 40 41	next

The loop displays all the digits with a one-second interval.

Build and Upload the Project

For instructions on how to save a **Designer** project, how to connect the target display to the PC, how to select the program destination, and how to compile and upload a program, please refer to the section "**Run the Program**" of the application note

Designer Getting Started - First Project

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.

4D-AN-00044