

# **Serial Displaying Third Party Fonts FAT16**

**DOCUMENT DATE: DOCUMENT REVISION:** 

24<sup>th</sup> May 2019 1.1



# Description

This application note explains how custom fonts can be used on a Picaso or a Diablo16 display module in the Serial environment. Here is a list of items required to replicate this application:

• Any of the following Picaso touch display modules:

uLCD-24PTU	<u>uLCD-32PTU</u>
uLCD-28PTU	uLCD-32WPTL

and other superseded touch display modules which support the ViSi environment

• The target module can also be a Diablo16 touch display

<u>uLCD-35DT</u>

#### uLCD-70DT

uLCD-43(PT/PCT)

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> or <u>µUSB-PA5</u>
- 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

Description2
Content2
Application Overview
Setup Procedure
Adding Fonts4
Customizing Fonts using the ViSi environment4
Completing the ViSi project8
Simulation8
Clear the Screen9
Mount the uSD Card9
Load the Font Files10
Set the Font ID11
Send a String11
The New Line Character12
Choose another Font13
Print the Contents of the uSD Card14
Unmount the uSD Card16
Proprietary Information17
Disclaimer of Warranties & Limitation of Liability

# Application Overview

There are three built in fonts in the Picaso processor:

- 0 or FONT1 = System font
- 1 or FONT2
- 2 or FONT3 = Default font

Users might need more stylish and larger size fonts which is a need addressed in this application. User can import ANSII or UNICODE fonts. The text could be displayed using **Put Character** or **Put String** command.

For the Diablo16 processor, the available system font IDs are:

- 1 for FONT\_1 = System 5x7
- 2 for FONT\_2 = System 8x8
- 3 for FONT\_3 = System 8x12 (Default)
- 4 for FONT\_4 = System 12x16
- 5 for FONT\_5 = MS San Serif 8x12
- 6 for FONT\_6 = Deja Vu Sans Condensed 9pt
- 7 for FONT\_7 = Deja Vu Sans 9pt
- 8 for FONT\_8 = Deja Vu Sans Bold 9pt
- 9 for FONT\_9 = System 3x6
- 10 Not currently available for SPE Serial, N/A
- 11 for FONT\_11 = EGA 8x12 font

NOTE: The **List Filenames** command is the only other command that writes the list of directory directly to the screen. This command is also affected by the Fonts change.

This application note uses a Picaso display module as the sample model, but the procedures are also applicable to Diablo16 display modules. The main difference between a Picaso display and a Diablo16 display with respect to the Serial environment is the bytes used for the serial commands. The serial command bytes for clearing the screen and sending a string, for instance, are different for each graphics processor.

	Picaso	Diablo16
Clear screen	0xFFCD	0xFF82
Put string	0x0018	0x0018

A user with a Diablo16 display module must have to consider this fact when using this application note as a reference.

# Setup Procedure

This application note, although written for Serial, requires the use of the ViSi environment to generate the necessary files which will be copied to the uSD card. The display module is then configured as a slave device by loading it with the SPE application. With the uSD card mounted onto the display, the host, which is the Serial Commander in this application note, will then be able to control the display and access the contents of the uSD card.

This application note starts with the creation of a basic ViSi project. Users who want to learn more about the ViSi environment may consult the application note

## ViSi Getting Started - First Project for Picaso and Diablo16

Topics discussed include instructions on how to launch Workshop 4, how to open a ViSi project, how to change the target display, how to create a new ViSi project, how to save a ViSi project, how to connect the target display to the PC, and how to compile and download a program.

# Adding Fonts

## **Customizing Fonts using the ViSi environment**

Open the Workshop 4 (WS4) IDE and click "Create a new project".



Choose the Display module you want to use. A uLCD-32WPTU will be used for this application.

**Note:** Orientation of the Display is irrelevant here.



# Select the ViSi evironment



This will open the ViSi development environment window within the WS4 IDE as shown below.

Workshop 4 - NoName1(ul.CD-32WPTU, LANDSCAPE)	
File Home View Tools Widgets Comms Project	۵. ا
New Open Save SaveAs Prot         %         Image: Save SaveAs Prot         %         Image: Save SaveAs Prot         SaveAs Prot         Image: SaveAs Prot         <	Compile Comp'nLoad
Prie Cilposito Orico/neuo Bookmaiks Prina ColePositing	Compae ⊲ ⊳
1 fplatform "uLCD-32NFTU"	A Form1
// Program Skeleton 1.0 generated 15/03/2013 12:54:33 PM	
<pre>5 #inherit "4DGL_16bitColours.fnc"</pre>	
7 #inherit "VisualConst.inc"	
#inherit "NoNamelConst.inc" 10	
11 func main()	
12 // var hstrings ; // Handle to access uSD strings, uncomment if required	
14 // Var nonck ; // manuae to access usb ionts, uncomment if require and change n to iont number 14 // Uncomment the following if usb images, fonts or strings used.	. 6
15 /*	
<pre>pubstr("Mounting\n"); f d(disk:=file Mount()); </pre>	
<pre>18 while(!(disk :=file_Nount())) 18</pre>	
<pre>19 putstr("Drive not mounted");</pre>	
20 pause (200) :	
22 pause (200) ;	
23 wend	
24 endif	
<pre>26 gfx_irensparence(0x0020); 26 gfx_irensparence(0x01);</pre>	
27	Object Inspector
<pre>28 // hFontn := file LoadImageControl("NoName1.dan", "NoName1.gcn", 1); // Open handle to access uSD fonts, uncomment if required an</pre>	nd Form Form1 •
29 // hstrings == file_Open("NoNamel.txf", 'r'); // Open handle to access uSD strings, uncomment if required bndl:s= file_Load Iman@Control("NoNamel.dxf", "NoNamel.dci", i);	Object Form1 ·
31 */	Properties Paste Code Paste all Code
32 31 after Sec (SCREEN MODE TANDECADE) .	Property Value
OLA DECIDENCER RULE.LANDOLARE) /	Name Form1
	Bgtype Color
	Color BLACK
	E Source
Inset Line17 Col9	la.

The new project starts as 'noname' project. Save the project as, say AddFont.

Go to Widgets, select Strings object under the Labels tab.



Click on to the WYSIWYG screen to drop the string object. A string object has now been created. This is Strings1.



Click on the ellipsis dots of the **Strings** property of the Object Inspector.

Object Inspecto	or	<b>4</b>
Form Form1		~
Object Strings	s1	~
Properties	Paste Code	
Property	Value	~
FGcolor	WHITE	
1 Font		
Left	50	
Strings		

The Strings Editor window appears. Select the font and adjust other properties as required.

		Strings E	Editor	
Input Edit Str Strings Style	Message V		Sample Message 1 of 1	↓ Lines/Message: 13
		^		. 105 💌
	2 of Page 1			
COLLOT LINC .	ites			
Object Attrib	nes		0.000	
Object Attribu Font:	4D Font3 (8x12) Bold Strikethrough	▼ ] Italic ] Underline	Open	
Object Attribu Font: Size:	4D Font3 (8x12) Bold Strikethrough	✓ Italic Underline ANSI ✓	Open	

For this application, we have only adjusted the following properties,

Set Fonts to 'Forte'

Font size is set to 20

#### Press OK.



# Create another string object. This object will be Strings2.

Object Inspector		<b>P</b> 83
Form Form1		¥
Object Strings2		
Properties Pa	ste Code	
Property	Value	^
FGcolor	WHITE	
🛨 Font		
Left	200	
Strings		

Set the properties in the Strings Editor window. For Strings2, the properties are:

Object Attributes	Open
Font: Times New Roman	open
Bold Italic	
Strikethrough Underline	
Size: 20 🗸 ANSI 🗸	
Last Char: 255 (65532)	🗸 ок

Now, there should be two strings in the object inspector.

Object Inspec	or	
orm Form1		•
bject Strings		•
Properti Form 1 Strings 1 Propert 7		
BGcolor	BLACK	
Height	165	
FGcolor	WHITE	
+ Font		
Left	200	
Strings		
StringsStyle	Message	
Тор	28	
Visible	Yes	Γ
Minister .	112	

#### Completing the ViSi project

- Now, insert the uSD card in the uSD card reader on the PC/Notebook.
- Make sure the uSD card is FAT (aka FAT16) formatted.
- On the ViSi window Home ribbon in WS4 click Compile.



After clicking on **Compile**, you will be prompted to select and confirm the drive to where the font files will be saved. Click **OK** to confirm and start copying.

Copy Confirmation	×
Copy AddFonts.gc1, AddFonts.da1, AddFonts.txf, AddFonts.gc2 and AddFonts.da2 to selected drive?	
Drive: H   Drive not mounted	
СК	lo Thanks

Press OK to start writing the Font files to the uSD card. Once the writing is completed, remove the uSD card from the PC/Notebook and insert it in to the display module.

# Simulation

The display must be configured as a slave device first before it can be controlled by a host. For instructions on how to launch Workshop 4, how to connect the display module to the PC, and how to configure the display as a slave device, kindly refer to the section "**Setup Procedure**" of any of the application notes below. Choose according to your display module's processor.

Serial Picaso Getting Started - the SPE Application

Serial Diablo16 Getting Started - the SPE Application

These application notes also introduce the user to the Serial Protocol thru the use of the Serial Commander

#### **Clear the Screen**



The bytes sent and received are:

gfx\_Cls[FFCD ] 0.020 (ACK)

The screen should now be cleared.

## Mount the uSD Card

Mount the uSD card on to the display module. Go to the **File** tab and select **file\_Mount**.

Gfx Txt Media	Img File Snd Tou	uch Other
	0 7 0 10	0 a. p. w
		⊖ file_PutW
O file_Close	O file_GetS	O file_Read
◯ file_Count	O file_GetW	○ file_Rewind
◯ file_Dir	○ file_Image	◯ file_Run
file_Erase	○ file_Index	<ul> <li>file_ScreenCapture</li> </ul>
file_Error	file_LoadFunction	◯ file_Seek
◯ file_Exec	O file_LoadImageControl	○ file_Size
○ file_Exists	file_Mount	◯ file_Tell
O file_FindFirst	◯ file_Open	◯ file_Unmount
O file_FindFirstRet	◯ file_PlayWAV	◯ file_Write
◯ file_FindNext	○ file_PutC	
◯ file_FindNextRet	⊖ file_PutS	
<u> </u>		

The bytes sent and received are:



#### Load the Font Files

In the **File** tab, select **file\_LoadImageControl** then enter the font file names and mode.

ifx Txt	Media	lmg File	e Snd	Touch	Other
◯ file_CallFund	ction	⊖ file_0	ietC	С	file_PutW
○ file_Close		⊖ file_0	ietS	0	file_Read
○ file_Count		⊖ file_0	àetW	0	file_Rewind
⊖ file_Dir		O file_I	mage	0	file_Run
O file_Erase		O file_I	ndex	0	hile_ScreenCapture
		O file_L	oadFunction		l file_Seek
○ file_Exec		● file_t	.oadimageud Aount		rile_bize File_Tell
file_Exists	r F	⊖ file_f	loen	ĕ	file_linmount
○ file_FindFirst	tRet	O file F	PlavWAV	ŏ	file Write
⊖ file_FindNex	(t	◯ file_F	PutC	0	_
◯ file_FindNex	dRet	◯ file_F	PutS		
Datna	ame AddF	ont.da1	GCI	Name A	ddFont.gc1
Mo	ode 1	×	•		
Send					



Take note of the reply from the display – "**5144**" in decimal or "0x1418" in hexadecimal. This is the 16-bit file handle. It could be different in your case.

The bytes sent and received are:

## Set the Font ID

Go to the Txt tab and select **txt\_FontID**. Enter the file handle into the **FontNumber** box.



# Send a String

Go the **Other** tab and select **putstr**. Input "Hello 4D Systems." to the **InString** box.

Gfx Txt	Media	Img	File	Snd	Touch Other
<ul> <li>blitComtoD</li> <li>bus_In</li> <li>bus_Out</li> <li>bus_Read</li> <li>bus_Set</li> <li>bus_Write</li> <li>charheight</li> <li>charwidth</li> <li>mem_Free</li> <li>mem_Hea</li> <li>peekM</li> <li>pin_HI</li> </ul>	Pisplay			<ul> <li>pin_</li> <li>pin_</li> <li>pok</li> <li>pok</li> <li>pott</li> <li>putt</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>writ</li> </ul>	_LO _Read _Set .eM CH str _GetModel _GetPmmC _GetVersion _Sleep eString
InS	tring lell	o 4D Sys	stems.		
Send					

The bytes sent and received are:

putstr[0018 "Hello 4D Systems."] 0.047 (ACK 17 0x0011)

The string "Hello 4D Systems." should now be shown on the screen.



# The New Line Character

The ASCII code for the new line character is 0x0A in hexadecimal or 10 in decimal. In the tab **Other**, select **putCH**. Input the value "10" to the **WordChar** box and click on the send button.

Gfx	Txt	Media	Img	File	Snd	Touch	Other
000000000000000000000000000000000000000	blitComto bus_In bus_Out bus_Rea bus_Set bus_Writ charheigl charwidtł mem_Fre mem_He peekM pin_HI	Display d e ht e ap			<ul> <li>pin_</li> <li>pin_</li> <li>pin_</li> <li>pok</li> <li>put(</li> <li>put(</li> <li>puts</li> <li>reac</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>write</li> </ul>	LO Read Set eM CH String GetModel GetPmmC GetVersior Sleep eString	1
	Wor	dChar 10		<b></b>			
	Send						

The bytes sent and received are:

putCH[FFFE 000A] 0.007 (ACK)

## **Choose another Font**

Choose a new font by loading the correct files.

file_CallFunction       file_GetC       file_PutW         file_Close       file_GetS       file_Read         file_Count       file_GetW       file_Rewind         file_Dir       file_Image       file_Run         file_Erase       file_Index       file_ScreenCapture         file_Error       file_LoadFunction       file_Size         file_Exec       file_Mount       file_Size         file_FindFirst       file_Open       file_Unmount         file_FindFirstRet       file_PutC       file_Write         file_FindNext       file_PutS       file_PutS	Gfx Txt Media	Img File (	Snd Tou	ch Other
Datname AddFont.da2 GCIName AddFont.gc2 Mode 1  v Send	<ul> <li>file_CallFunction</li> <li>file_Close</li> <li>file_Count</li> <li>file_Dir</li> <li>file_Erase</li> <li>file_Error</li> <li>file_Exec</li> <li>file_Exists</li> <li>file_FindFirst</li> <li>file_FindFirstRet</li> <li>file_FindNext</li> <li>file_FindNextRet</li> </ul>	<ul> <li>file_GetC</li> <li>file_GetS</li> <li>file_GetW</li> <li>file_Image</li> <li>file_Index</li> <li>file_LoadFu</li> <li>file_LoadIm</li> <li>file_Mount</li> <li>file_Open</li> <li>file_PlayWA</li> <li>file_PutC</li> <li>file_PutS</li> </ul>	nction ageControl W	<ul> <li>file_PutW</li> <li>file_Read</li> <li>file_Rewind</li> <li>file_Run</li> <li>file_ScreenCapture</li> <li>file_Seek</li> <li>file_Size</li> <li>file_Tell</li> <li>file_Unmount</li> <li>file_Write</li> </ul>
Send	Datname A Mode 1	ddFont.da2 . ✓	GCIName	AddFont.gc2
	Send			



file\_LoadImageControl[0009 "AddFont.da2" "AddFont.gc2" 0001] 0.045 (ACK 4808 0x12C8)

0.045 (ACK 4808 0x12C8)

Again, take note of the returned value, which is the 16-bit file handle. This value may be different in your case. Use this value when setting the new font ID.

Gfx	Txt Media	Img	File	Snd	Touch	Other		
				<u> </u>				
0	xt_Attributes	$\bigcirc$ txt	_MoveCurs	sor				
0	xt_BGcolour			⊖ txt_	Opacity			
0	xt_Bold	⊖ txt_	_Set					
0	xt_FGcolour	◯ txt_Underline						
٥	xt_FontID		⊖ txt_Width					
0	xt_Height		⊖ txt_Wrap					
0	xt_Inverse			⊖ txt_Xgap				
Ot	xt_Italic		⊖ txt_Ygap					
	FontNumber 48	08	•					
	Send							

When successful, the new string should now be in a new line and using the new font.



# Print the Contents of the uSD Card

Send another new line character to the display.

Gfx	Txt	Media	Img	File	Snd	Touch	Other
000000000000000000000000000000000000000	blitComto bus_In bus_Out bus_Set bus_Writ charheig charwidtl mem_Fre mem_He peekM pin_HI	Display d ht n ap			<ul> <li>pin_</li> <li>pin_</li> <li>pin_</li> <li>pok</li> <li>put(</li> <li>puts</li> <li>read</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>sys_</li> <li>write</li> </ul>	LO Read Set eM CH String GetModel GetPmmC GetVersion Sleep eString	n
	Wor	dChar 10		•			
	Send						

Filename b	оох.					
Gfx Txt	Media	Img	File	Snd	Touck	n Other
<ul> <li>file_Call</li> <li>file_Clos</li> <li>file_Cout</li> <li>file_Dir</li> <li>file_Erat</li> <li>file_Erat</li> <li>file_Erat</li> <li>file_Exat</li> <li>file_Exat</li> <li>file_Find</li> <li>file_Find</li> <li>file_Find</li> <li>file_Find</li> <li>file_Find</li> </ul>	Function se int se or c sts dFirst dFirstRet dNextRet	000000000000000000000000000000000000000	file_GetC file_GetS file_Image file_Index file_LoadI file_LoadI file_Open file_Open file_PlayW file_PutC file_PutS	Function mageCon t /AV	(( (( (( (( ((	<ul> <li>file_PutW</li> <li>file_Read</li> <li>file_Rewind</li> <li>file_Run</li> <li>file_ScreenCaptur</li> <li>file_Size</li> <li>file_Tell</li> <li>file_Unmount</li> <li>file_Write</li> </ul>
F	ilename 🔭	•				
Send				<u> </u>		
bytes sent	and receiv	ved are:				
file	Dir [0002	** **	0.096	(ACK	5 0xt	0005)

The file names should now be printed on the display.



# Unmount the uSD Card

To properly unmount the uSD card, go to the **File** tab and select **file\_Unmount**. Then click on the send button.

						_	
Gfx	Txt	Media	Img	File	Snd	Tou	ich Other
	ile_CallFur ile_Close ile_Dount ile_Erase ile_Error ile_Exec ile_FindFir ile_FindRe ile_FindNe	st stRet extRet	000000000000000000000000000000000000000	file_GetC file_GetS file_Indey file_Indey file_Load file_Load file_Oper file_PlayV file_PutC file_PutS	/ e Function ImageCo nt N VAV	ntrol	<ul> <li>file_PutW</li> <li>file_Read</li> <li>file_Rewind</li> <li>file_Run</li> <li>file_ScreenCapture</li> <li>file_ScreenCapture</li> <li>file_Size</li> <li>file_Tell</li> <li>file_Unmount</li> <li>file_Write</li> </ul>
	Send						
he by	tes sent	and re	ceived	are:			
		file	e_Unmo	ount[Fl	F02]0	).004	4 (ACK)

# **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.