APPLICATION NOTE

4D SYSTEMS

TURNING TECHNOLOGY INTO ART

Application Note: 4D-AN-P5001

Serial - Displaying Third-Party Fonts

Document Date: 15th March 2013

Document Revision: 1.0

Description

This Application Note explains how custom fonts can be used on a PICASO module in ViSi Environment:

Here is the list of items required to replicate this application,

- 4D Workshop 4 IDE
- A PICASO based Display Module
- 2GB uSD Card
- 4D Programming Cable (For simulation only)

NOTE: 4D Programming Cable is used in this application to connect the Display Module with the PC where the Serial Commander Test software simulates the Host Controller on the PC.

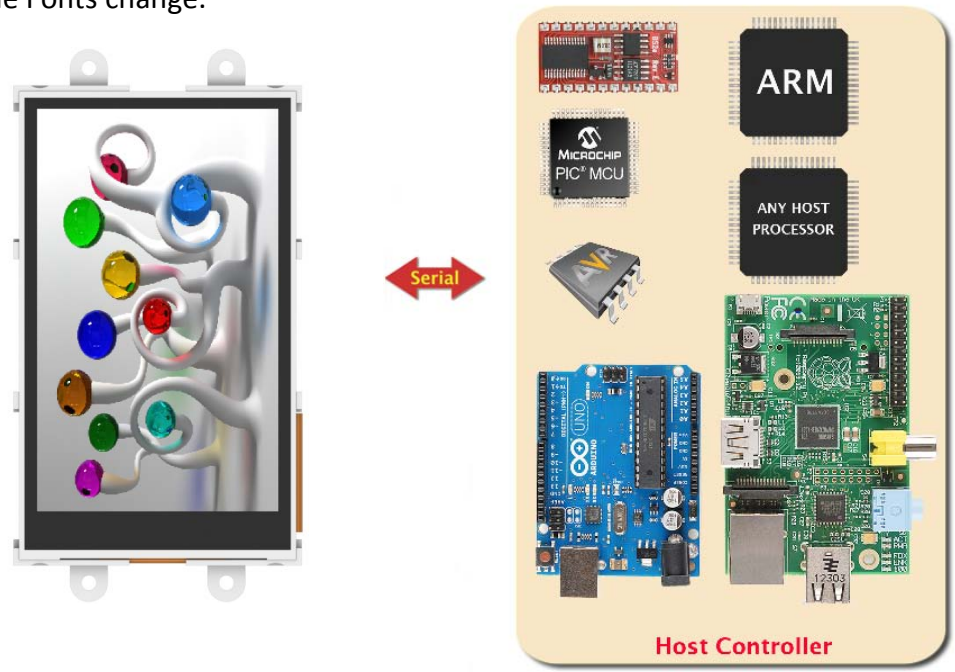
Application Overview

There are 3 built in fonts in the PICASO processor, those are.

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

User 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.

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.



Setup Procedure

Foremost, the 4D Workshop 4 IDE has to be downloaded and installed. This is available from the 4D Systems website through the following link:

http://www.4dsystems.com.au/prod.php?id=172

Documentation regarding Workshop 4 and its environments, such as Serial Environment, can also be downloaded from this site.

These are the documents that you will need to refer to, to replicate this application.

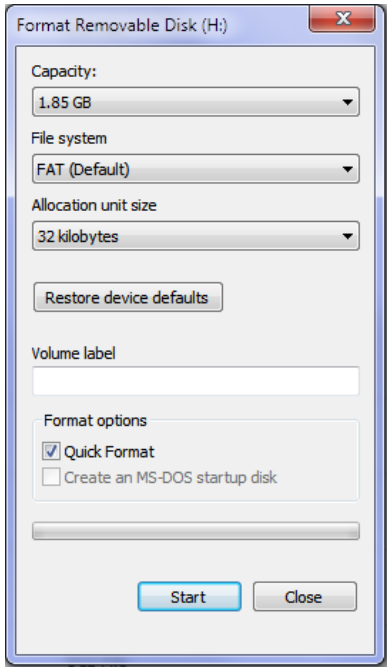
Workshop 4 - PICASO Serial Command Set Reference Manual

The PICASO Serial Environment Libraries are also available from the following links,

- Arduino Library (PICASO)
- C Library (PICASO)
- Pascal Library (PICASO)
- PicAxe Library (PICASO)

Customizing Fonts using the ViSi environment

FAT (aka FAT16) format the uSD card using Windows formatting tool,



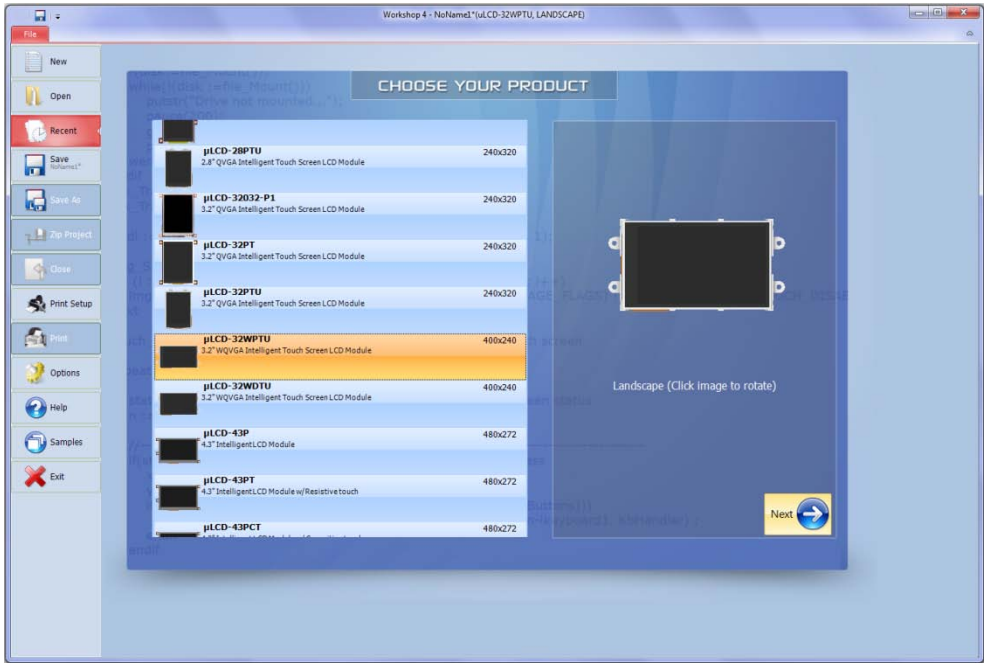
Adding Fonts Procedure

Customizing Fonts using the ViSi environment

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

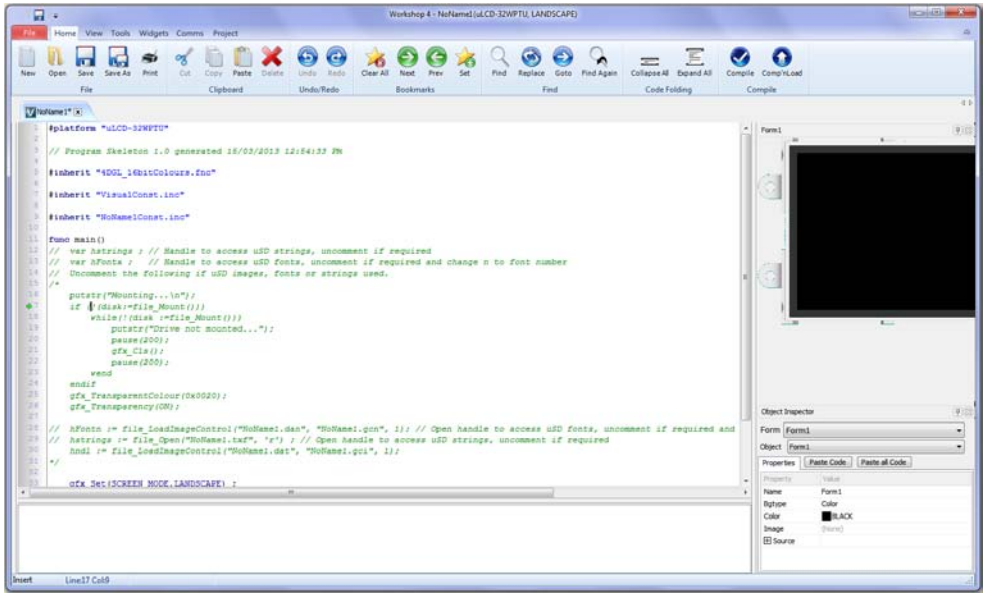
Choose the Display module you want to use. uLCD-32WPTU will be used for this application.

Note: Orientation of the Display is irrelevant here.



Select the **ViSi** evironment

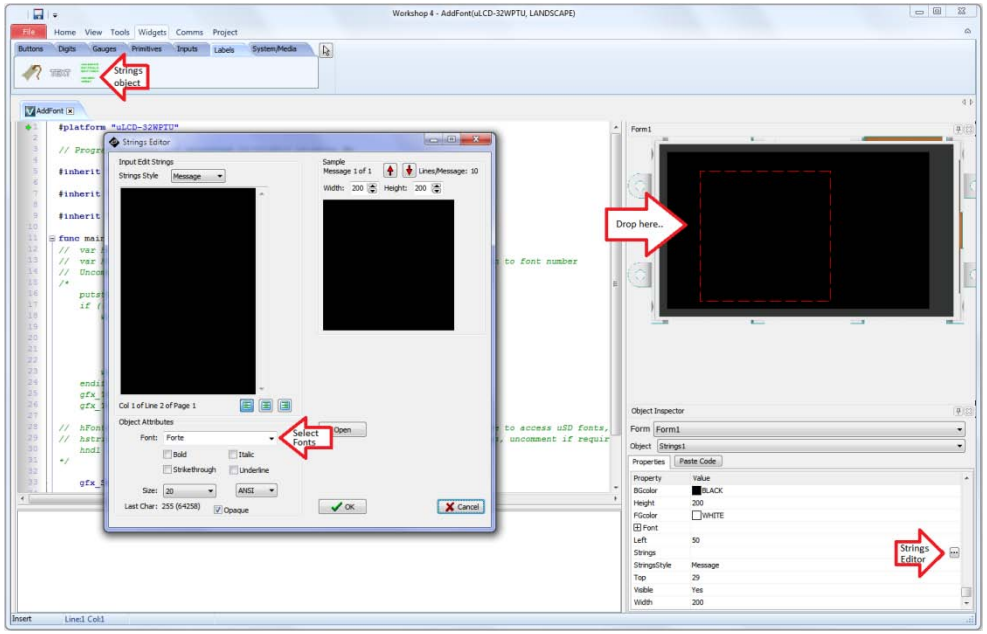
This will open the ViSi development environment window within the WS4 IDE as shown below.



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 screen to drop the Strings object.
- Click **Strings** property in the Properties section at the bottom right to setup the Fonts.
- Write something on the left window.
- Select the Fonts and adjust other properties as required.
- For this application, we have only adjusted the following properties,
    - Set Fonts to 'Forte'
    - Font size is set to 20

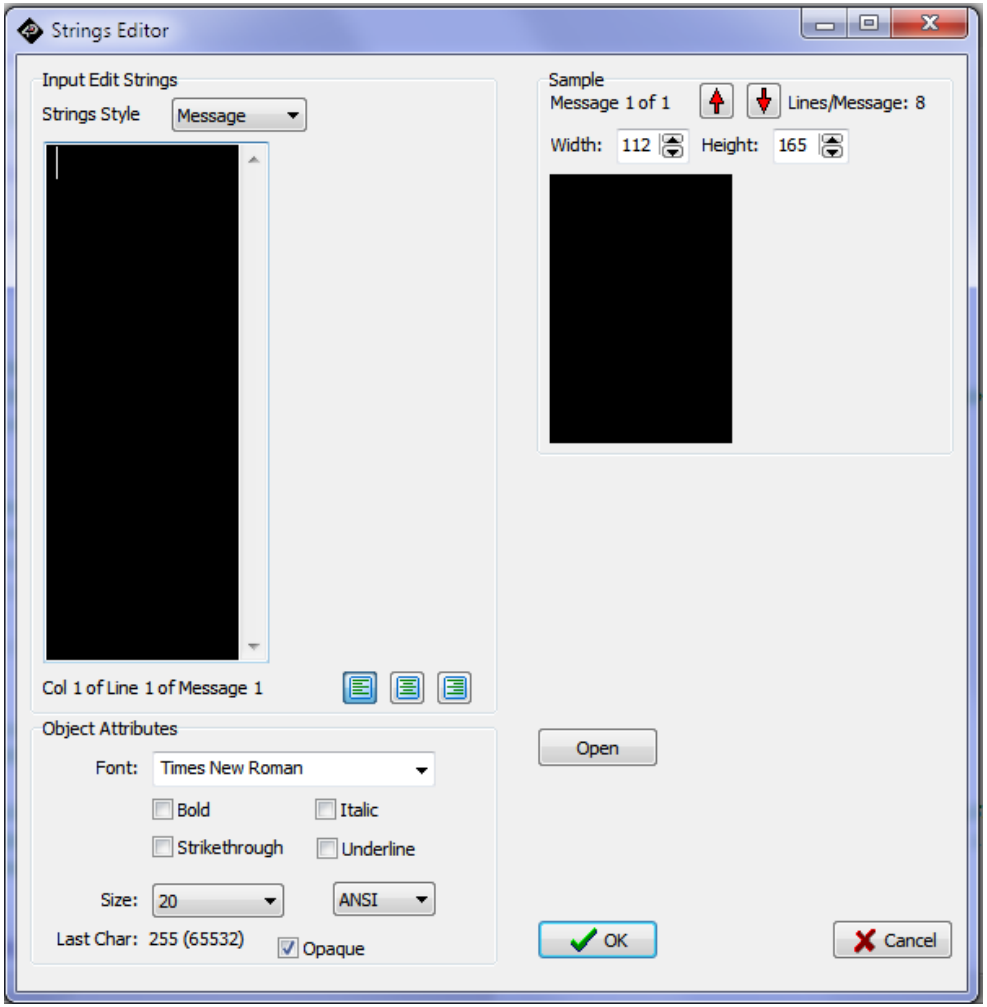
Press OK.



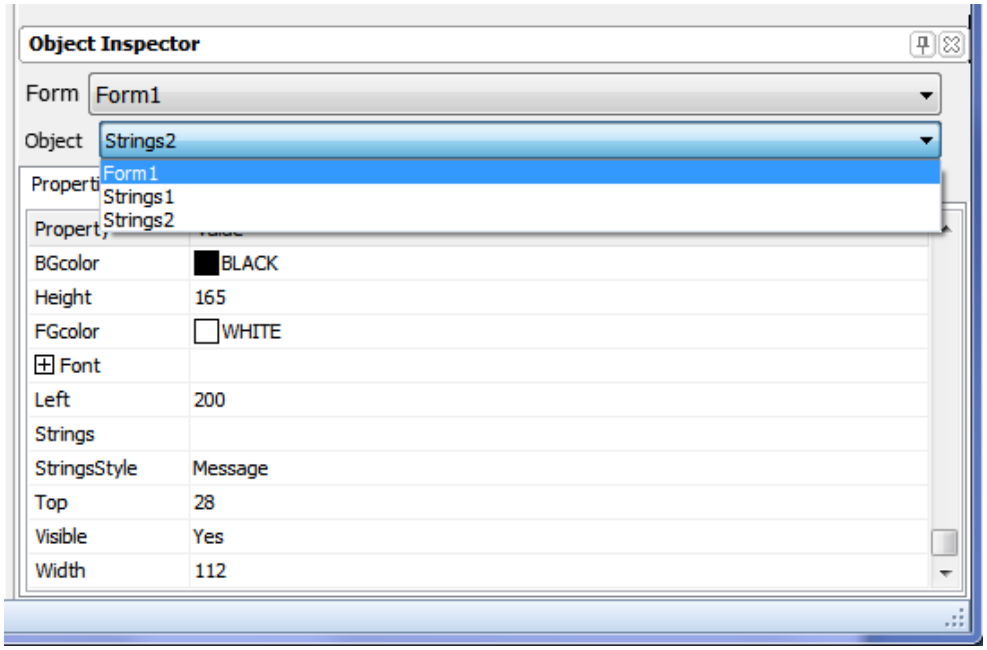
Similarly, drop another Strings object on screen and set it to,

- Set Fonts to 'Times New Roman'
- Font size is set to 20

Press OK.

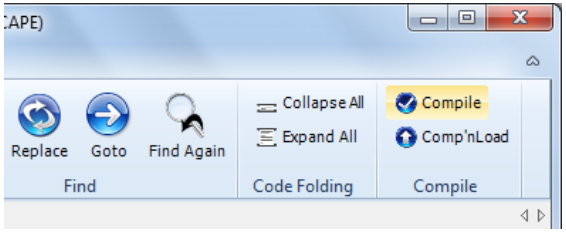


Now, there are two strings in the object.

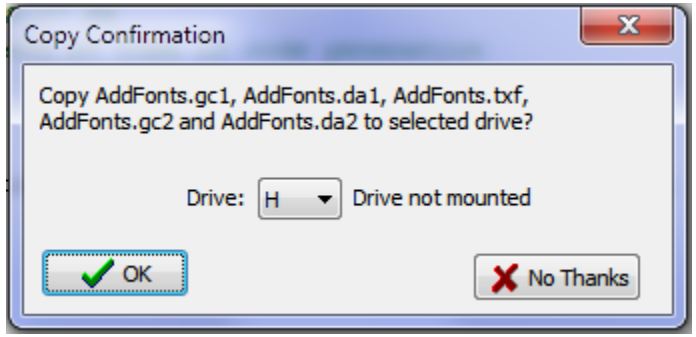


Completing the ViSi project

- 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.



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 Procedure

Converting a 4D Picaso Display to a Serial display

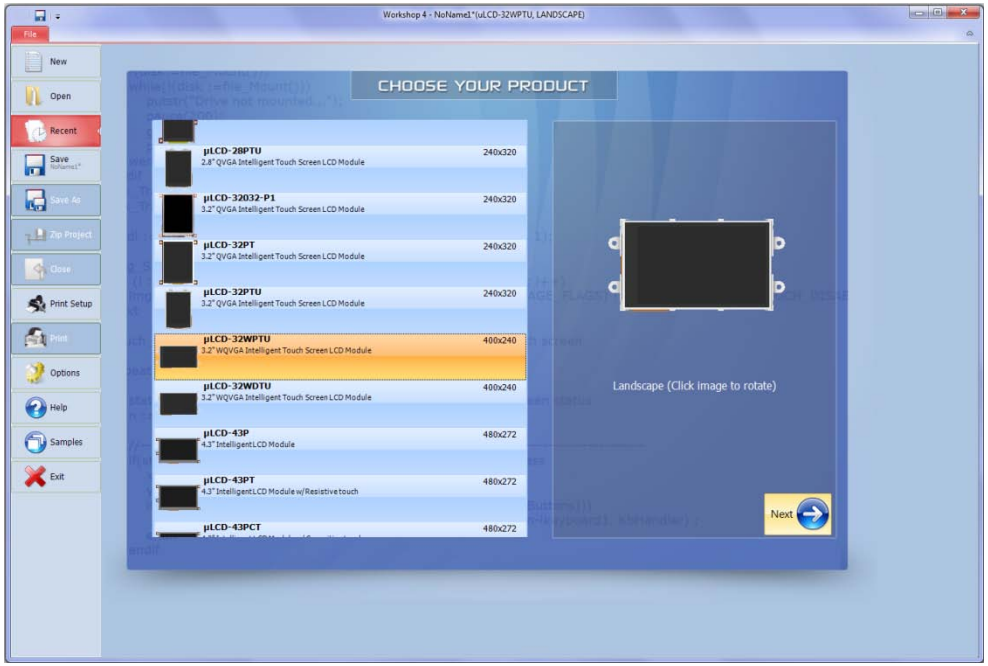
Note: This step is not needed if the user is using a newly purchased Picaso module, as the module is in Serial Environment configuration (i.e. pre-loaded with the SPE) by default.

Connect the 4D Programming Cable to the PC, you will be prompted for the driver installation, if not then you may have to manually install the driver available here,

http://www.4dsystems.com.au/prod.php?id=138

Create a new project in Workshop 4 (WS4),

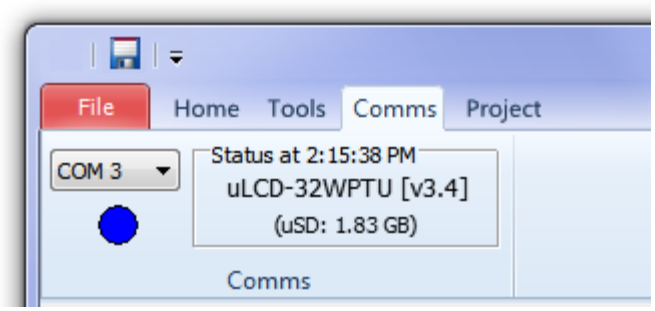
We are using a uOLED-128-G2 module for demonstration; you may wish to choose any other Goldelox Display module from the products list.



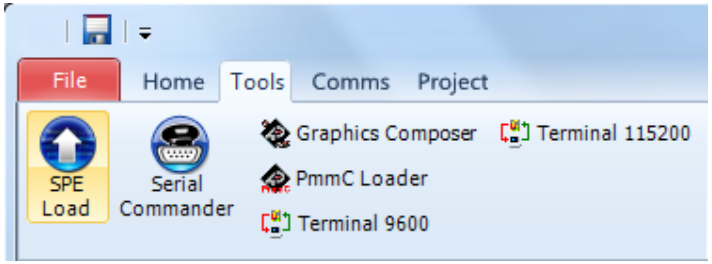
Press Next.

Now, select the serial environment.

Connect the display module to the 4D programming cable. Click COMMs tab on WS4. Ensure that the "Indicator Dot" is blue otherwise select appropriate COM port, click the dot and wait until it is blue indicating that communications with the module was established.



Open Tools menu and click SPE Load. This procedure would load the necessary firmware and configuration to convert the display in Serial (SPE) mode.

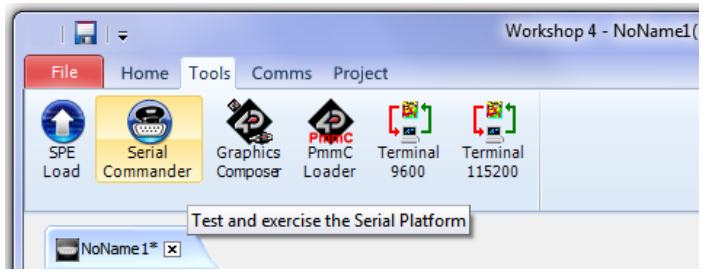


On the 4D display, you should see a splash screen scrolling at this point.

Note: SPE is factory programmed on to the module. I.e. the Picaso display modules are configured for “Serial Environment” by default. Customers can switch the environments, if they need to.

Printing text in custom fonts

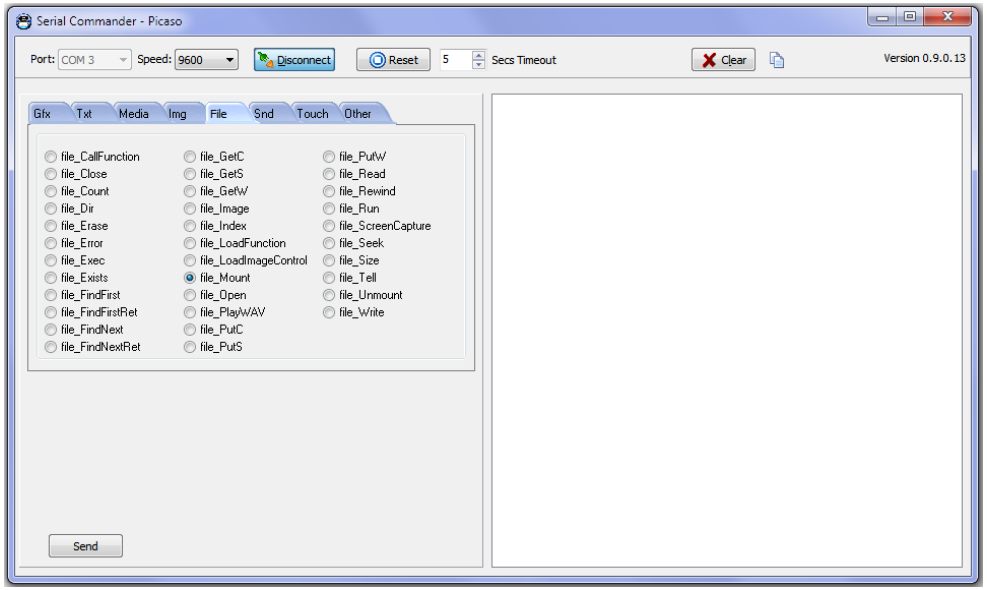
As mentioned earlier, the Host controller can now send the commands to the display module over the serial interface. We are using Serial Commander Software tool to simulate the Host controller. Serial Commander is a part of 4D Workshop4 IDE.



Please also note, although, 4D Programming Cable is not needed by the Host Controller to connect to the Display module, it’s needed to test the module with the Serial Commander.

Note: You could be using your Host controller to send the commands illustrated in the following section. Make sure the Baud rate is set correctly. SPE loaded at the Factory is set to 9600 by default. If you wish to change it, you can go to Options, change the “Serial Environment Initial Baud Rate” under the Serial tab and reload the SPE.

See the image below.



- Let's **clear the screen** first.

  0xFF 0xCD
- Make sure uSD card is inserted with the font files. Send **File Mount** command

  0xFF 0x03
- Use **Load image control** for the Fonts as set in Strings1 in the ViSi Project.

  0x00 0x09 0x41 0x64 0x64 0x46 0x6F 0x6E 0x74 0x2E 0x64 0x61 0x31 0x00 0x41 0x64 0x64 0x46 0x6F 0x6E 0x74 0x2E 0x67 0x61 0x31 0x00 0x00 0x01

Note: "AddFont.da1" and "AddFont.gc1" are the files needed for fonts off the Strings1 object.

- You should get the 16bit 'handle' in return. We received the following address in return. It could be different in your case.

  0x14 0x52
- Now set the fonts using **Set Font** command and the above address,

  0xFF 0xE5 0x14 0x52
- Let's print something using **Put String**

  0x00 0x18 0x48 0x65 0x6C 0x6C 0x6F 0x20 0x34 0x44 0x20 0x53 0x79 0x73 0x74 0x65 0x6D 0x73 0x2E 0x00 0x00 0x11

Note: The string is, "Hello 4D Systems."

- You should see "Hello 4D Systems." On your screen in 'Forte' fonts.

- Let's try the second font set in the Strings2.
- Let's change the line first using **Put Character** command.

  0xFF 0xFE 0x00 0x0A

- Use **Load image control** for the Fonts as set in Strings2 in the ViSi Project.

  0x00 0x09 0x41 0x64 0x64 0x46 0x6F 0x6E 0x74 0x2E 0x64 0x61 0x32 0x00 0x41 0x64 0x64 0x46 0x6F 0x6E 0x74 0x2E 0x67 0x61 0x32 0x00 0x00 0x01

Note: "AddFont.da2" and "AddFont.gc2" are the files needed for fonts off the Strings2 object.

- You should get the 16bit 'handle' in return. We received the following address in return. It could be different in your case.

0x13 0x02

- Now set the fonts using **Set Font** command and the above address,
  0xFF 0xE5 0x13 0x02
- Let's print something using **Put String**
  0x00 0x18 0x48 0x65 0x6C 0x6C 0x6F 0x20 0x34 0x44 0x20 0x53 0x79 0x73 0x74 0x65 0x6D 0x73 0x2E 0x00 0x00 0x11

Note: The string is, "Hello 4D Systems."

- You should see "Hello 4D Systems." on your screen in 'Times New Roman' fonts in the next line.

- Let's change the line again using **Put Character** command.
  0xFF 0xFE 0x00 0x0A
- Let's print something using **Put Character**
  0xFF 0xFE 0x00 0x39

Note: The Character is, "9".

- You should see "9" on your screen in 'Times New Roman' fonts in the next line.

- Let's change the line again using **Put Character** command.
  0xFF 0xFE 0x00 0x0A
- Let's print the list of files using **List Filenames** command.
  0x00 0x02 0x2A 0x2E 0x2A 0x00

Note: Wildcard (*.*) is used in this command instead of the file name.

- You should see the list of files on your screen in 'Times New Roman' fonts.

Log of commands sent from the Serial Commander

gfx_Cls[FFCD] 0.024 (ACK)

file_Mount[FF03] 0.138 (ACK 5505 0x1581)

file_LoadImageControl[0009 "AddFont.da1" "AddFont.gc1" 0001] 0.047 (ACK 5202 0x1452)

txt_FontID[FFE5 1452] 0.023 (ACK 2 0x0002)

putstr[0018 "Hello 4D Systems."] 0.045 (ACK 17 0x0011)

putCH[FFFE 000A] 0.007 (ACK)

file_LoadImageControl[0009 "AddFont.da2" "AddFont.gc2" 0001] 0.047 (ACK 4866 0x1302)

txt_FontID[FFE5 1302] 0.009 (ACK 5202 0x1452)

putstr[0018 "Hello 4D Systems."] 0.046 (ACK 17 0x0011)

putCH[FFFE 000A] 0.007 (ACK)

putCH[FFFE 0039] 0.008 (ACK)

putCH[FFFE 000A] 0.007 (ACK)

file_Dir[0002 "*.*"] 0.243 (ACK 16 0x0010)

See the snapshot of the display module.

© 2012 4D Systems

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.