APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

Serial Goldelox Displaying Third Party Fonts

DOCUMENT DATE:	**20th May 2019**
DOCUMENT REVISION:	**1.1**

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

Description

This application note is intended to provide users of 4D Systems Goldelox Display Module and beginners, information on how to serially display third party fonts or customized fonts. In order to carry out this application note the following items are required:

- Any of the following 4D Goldelox display modules:

  uOLED-96-G2 uOLED-128-G2 uOLED-160-G2
  uLCD-144-G2

  or any superseded module that supports the Serial environment. Visit www.4dsystems.com.au to see the latest products using the Goldelox graphics processor.

- 4D Programming Cable / µUSB-PA5/µUSB-PA5-II
- 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 2
Application Overview 3
Setup Procedure 3
Adding Fonts 4
Customizing Fonts using the ViSi Environment 4
Completing the ViSi Project 7
Writing Font Data to the uSD Card 8
Simulation 10
Displaying the Fonts 10
Choosing and Displaying another Font 13
User Experiments 14
Conclusion 14
Proprietary Information 15
Disclaimer of Warranties & Limitation of Liability 15

Application Overview

This application note walks the user through the steps to get a 4D Systems Goldelox Display up and running for the first-timer, with the guarantee at the end of displaying any text or string typed and rendered using third-party or even customized fonts other than the 4D System fonts that are displayed by default when writing text, labels or strings.

This application note starts with a basic tutorial in the creation of a simple Visi project to generate a sample media font base on a uSD card which will be used by a Goldelox display of choice.

Furthermore, this application note would serve as a guide for any beginner in converting any Goldelox display to a Serial display using the Serial Environment of Workshop 4, using the SPE application from 4D Systems.

At the end, the user will be introduced in operating the Serial Commander application tool inside the Workshop 4 IDE to stream data, commands and functions to control and show the custom fonts or third party fonts created using the ViSi Environment, on the display module connected to the host PC using a 4D programming cable.

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 Goldelox

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.

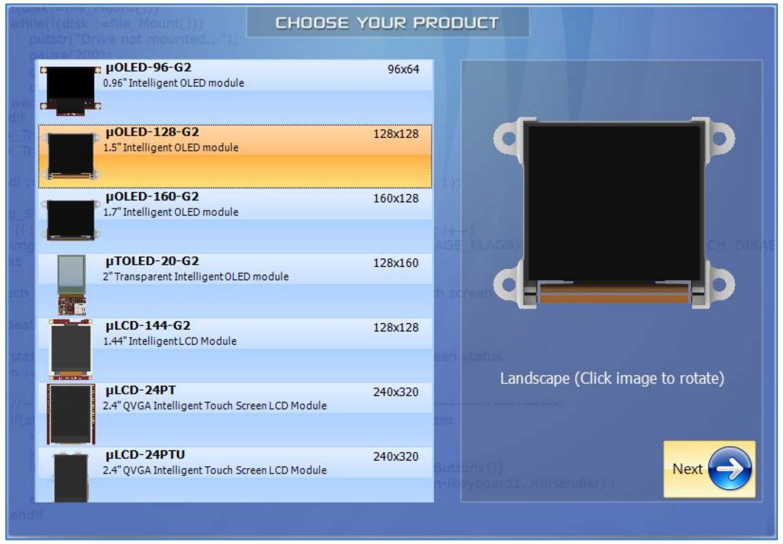
Adding Fonts

Customizing Fonts using the ViSi Environment

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

Create a new 4D Systems Project
Start building a new Visi, Genie, Designer or Serial program.

Choose the Display module you want to use. For this example we select the **uOLED-128-G2**, and click **Next**.

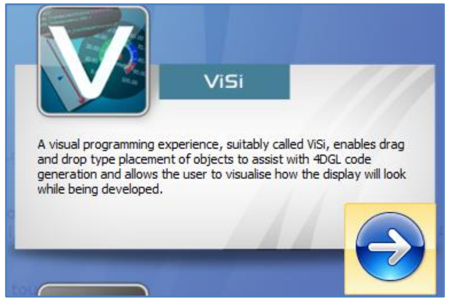


CHOOSE YOUR PRODUCT

µOLED-96-G2	0.96" Intelligent OLED module	96x64
µOLED-128-G2	1.5" Intelligent OLED module	128x128
µOLED-160-G2	1.7" Intelligent OLED module	160x128
µTOLED-20-G2	2" Transparent Intelligent OLED module	128x160
µLCD-144-G2	1.44" Intelligent LCD Module	128x128
µLCD-24PT	2.4" QVGA Intelligent Touch Screen LCD Module	240x320
µLCD-24PTU	2.4" QVGA Intelligent Touch Screen LCD Module	240x320

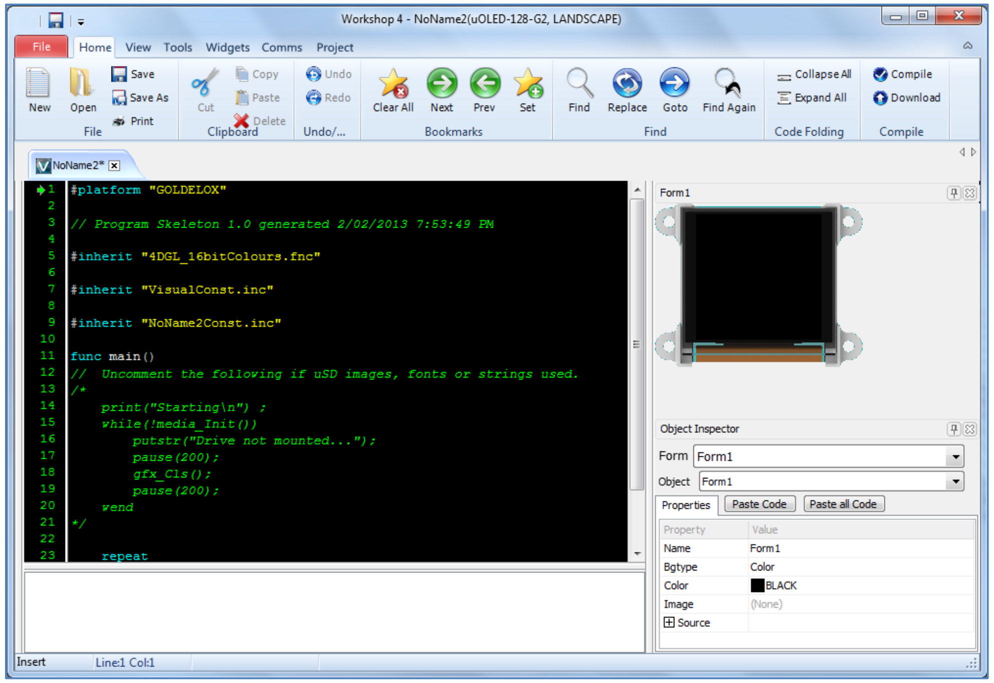
Landscape (Click image to rotate)

Select the **ViSi** evironment



A visual programming experience, suitably called ViSi, enables drag and drop type placement of objects to assist with 4DGL code generation and allows the user to visualise how the display will look while being developed.

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

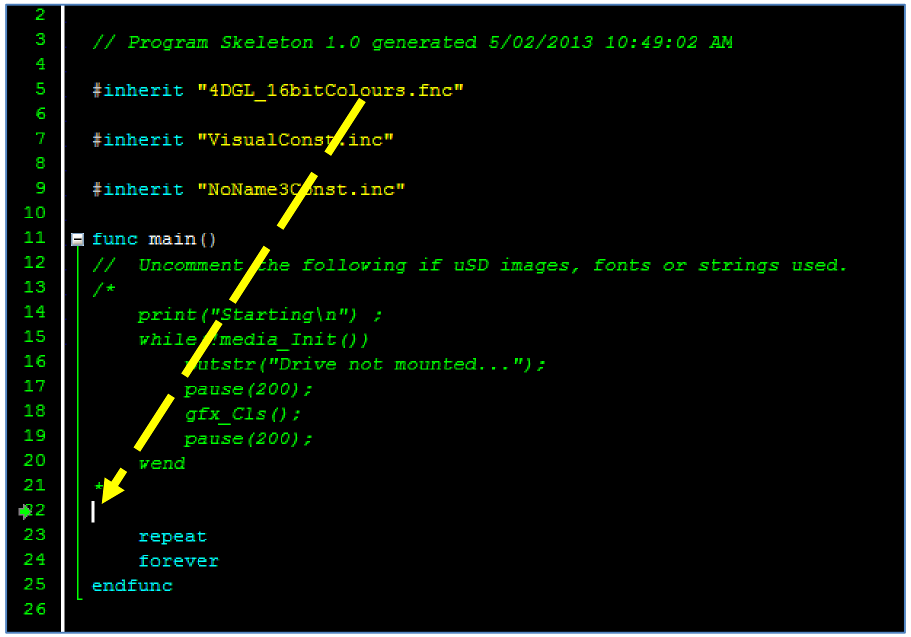


```
#platform "GOLDELOX"

// Program Skeleton 1.0 generated 2/02/2013 7:53:49 PM

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"

#inherit "NoName2Const.inc"

func main()
//  Uncomment the following if uSD images, fonts or strings used.
/*
    print("Starting\n") ;
    while(!media_Init())
        putstr("Drive not mounted...");
        pause(200);
        gfx_Cls();
        pause(200);
    wend
*/

    repeat
```

Form	Form1
Object	Form1

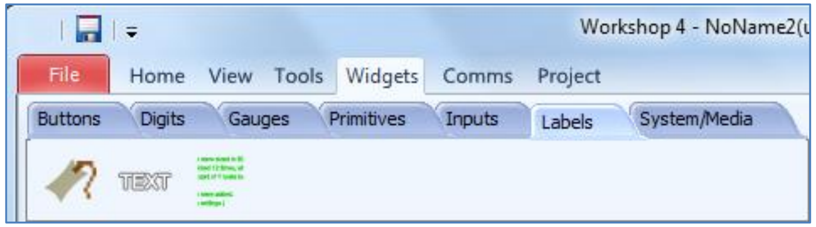
Property	Value
Name	Form1
Bgtype	Color
Color	■BLACK
Image	(None)
⊞ Source	

On the ViSi code area, position the cursor as shown below:

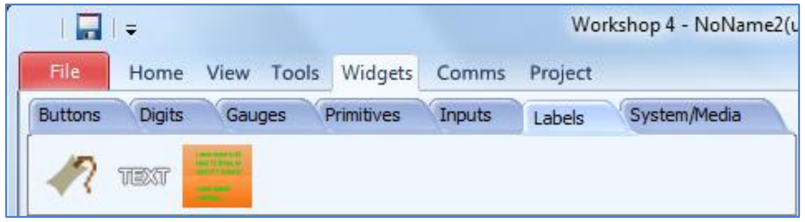


```
// Program Skeleton 1.0 generated 5/02/2013 10:49:02 AM

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"

#inherit "NoName3Const.inc"

func main()
//  Uncomment the following if uSD images, fonts or strings used.
/*
    print("Starting\n") ;
    while(!media_Init())
        putstr("Drive not mounted...");
        pause(200);
        gfx_Cls();
        pause(200);
    wend
*/

    repeat
    forever
endfunc
```

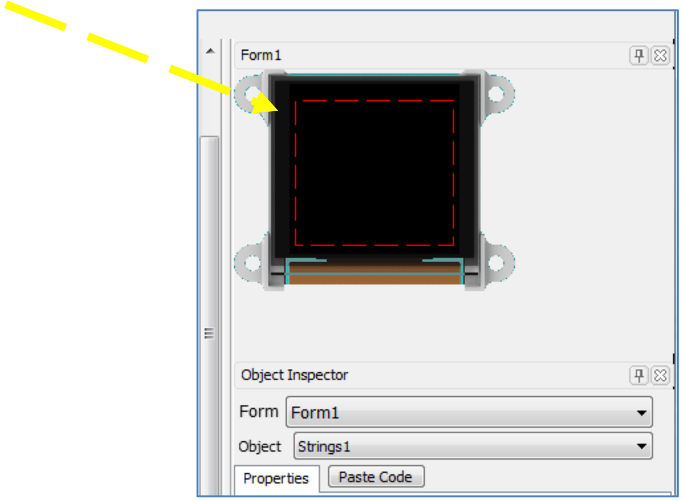
On the WS4 main menu, click **Widgets** and then select **Labels** tab on the objects ribbon.



Next, select and click on the **Strings** Icon



Click inside the display area in the Form or WYSIWYG section of WS4 IDE to drop the **String** placeholder represented by a dashed red box.



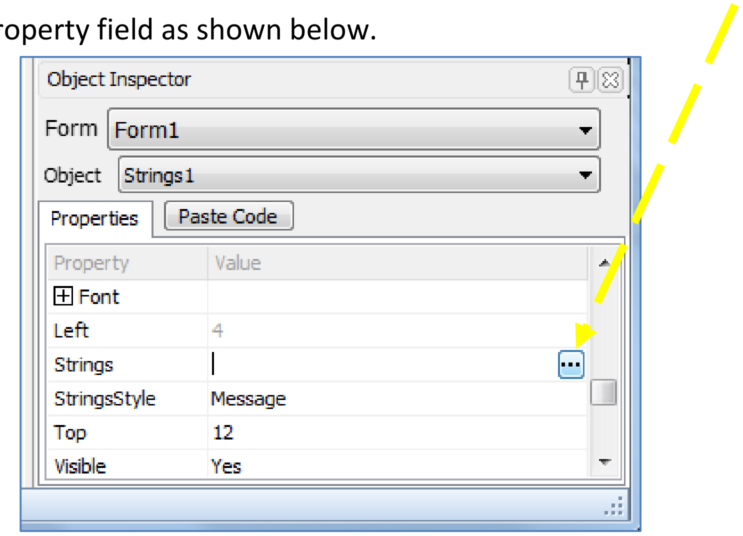
Form1

Object Inspector

Form: Form1

Object: Strings1

Scroll down to the Object Inspector area and find the String property field. To add a font and open the string editor, simply double click on the (...) of the String property field as shown below.

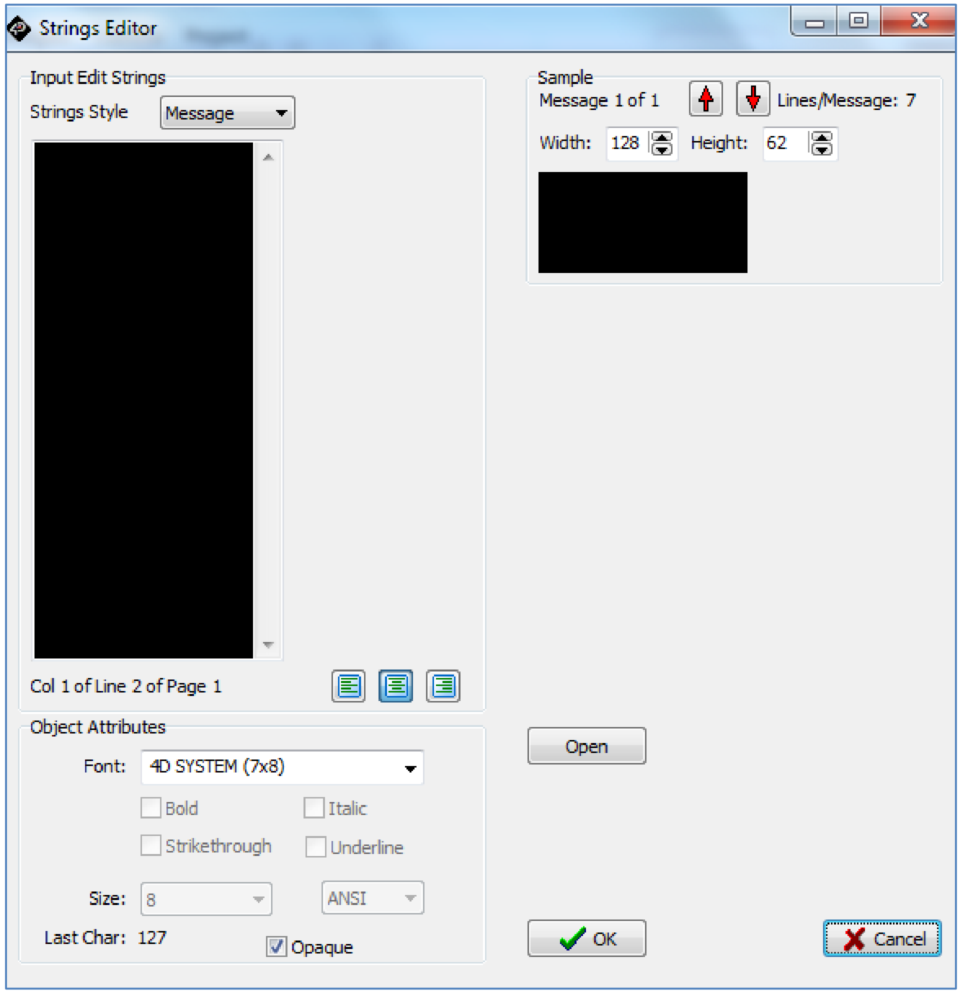

Object Inspector

Form: Form1

Object: Strings1

Property	Value
Font	
Left	4
Strings	
StringsStyle	Message
Top	12
Visible	Yes

This opens the string editor where you can select the font type, and text style, size and other attributes. See next photo below for details.


Strings Editor

Strings Style: Message

Col 1 of Line 2 of Page 1

Message 1 of 1 Lines/Message: 7

Width: 128 Height: 62

Font: 4D SYSTEM (7x8)

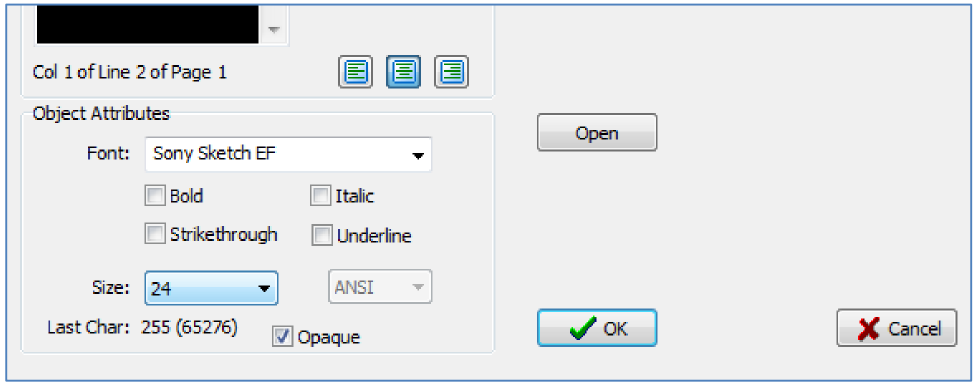
Bold, Italic, Strikethrough, Underline (unchecked)

Size: 8 ANSI

Last Char: 127

Opaque [✓]

In the font selection area of the string editor, click on the down arrow and select the desired font type. As an example, we use ***the Sony Sketch EF*** font and we set the size to **24** as shown below. The user may at his own desire select another font.



Col 1 of Line 2 of Page 1

Object Attributes

Font: Sony Sketch EF

☐ Bold ☐ Italic

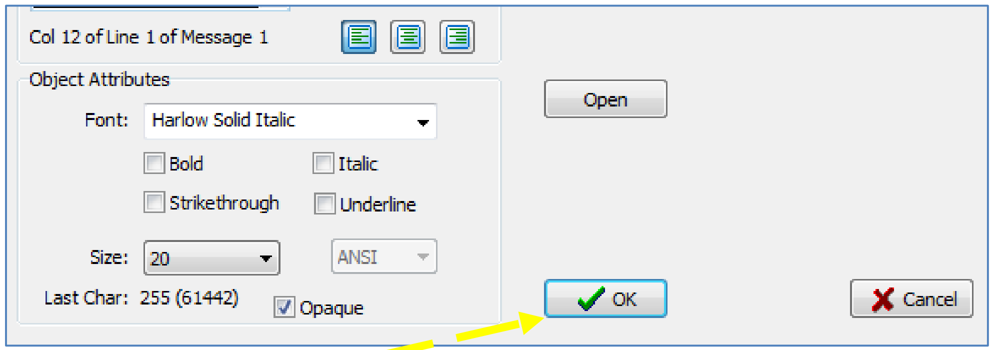
☐ Strikethrough ☐ Underline

Size: 24 ANSI

Last Char: 255 (65276) ☑ Opaque

We will now add the second string. To do so, we repeat the steps starting from the selection of the String icon on the Labels menu. Again click on the Form or WYSIWYG section to drop the new string placeholder.

In the same manner as in String1, on the Object Inspector area, go to the String2 properties and click (...) to open the String Editor window. For this exercise, select **Harlow Solid Italic** with size of **20**.



Col 12 of Line 1 of Message 1

Object Attributes

Font: Harlow Solid Italic

☐ Bold ☐ Italic

☐ Strikethrough ☐ Underline

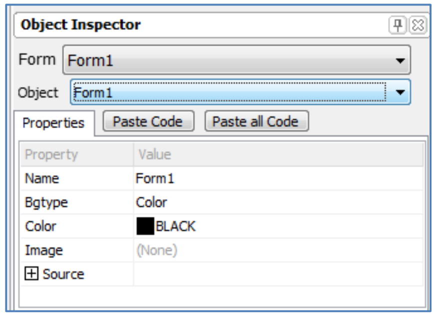
Size: 20 ANSI

Last Char: 255 (61442) ☑ Opaque

Click **OK** to finalize and exit the String Editor window.

Completing the ViSi Project

Next go to the **Object Inspector** area and on the **Object** menu select **Form1** as shown below:



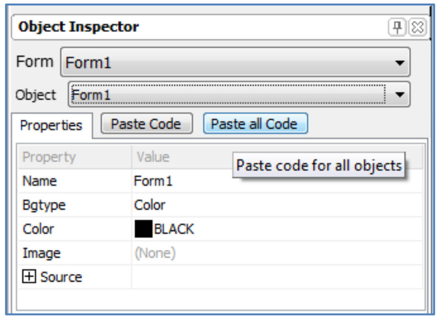
Object Inspector

Form: Form1

Object: Form1

Property	Value
Name	Form1
Bgtype	Color
Color	■ BLACK
Image	(None)
⊞ Source	

Next, click "Paste all Code".



Property	Value
Name	Form1
Bgtype	Color
Color	■ BLACK
Image	(None)
⊞ Source	

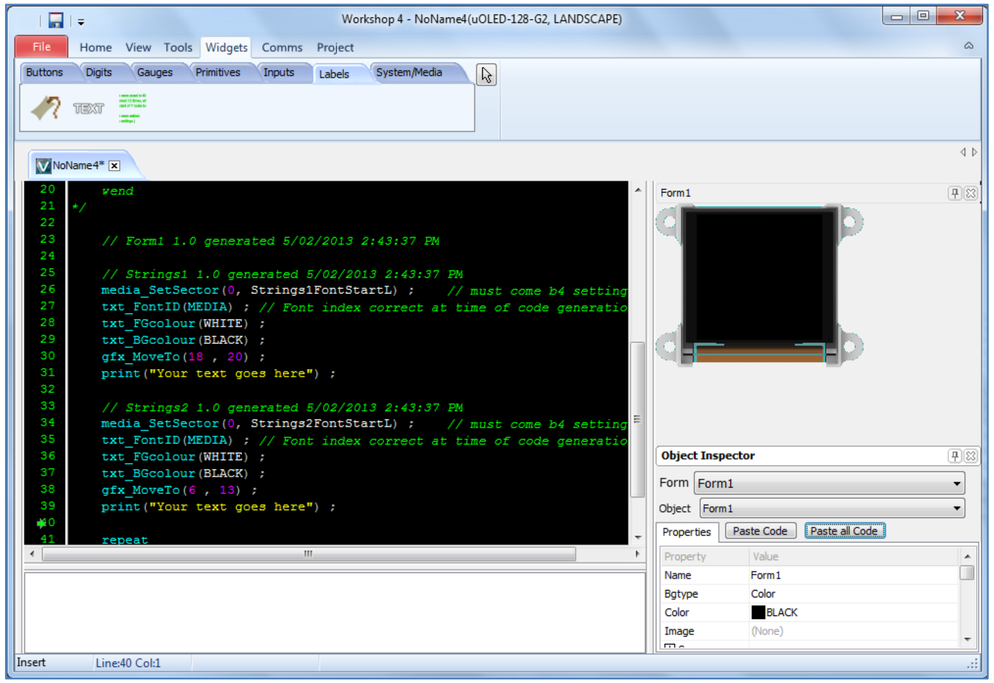
This completes the **ViSi** code which generates the data for the fonts we will be using for the simulation.

Writing Font Data to the uSD Card

NOTE: During this procedure, **the 4D programming cable** or the **Display module** should **NOT be connected** yet.

First step is to copy the font data to a uSD card. Insert uSD to USB adapter and plug to PC.

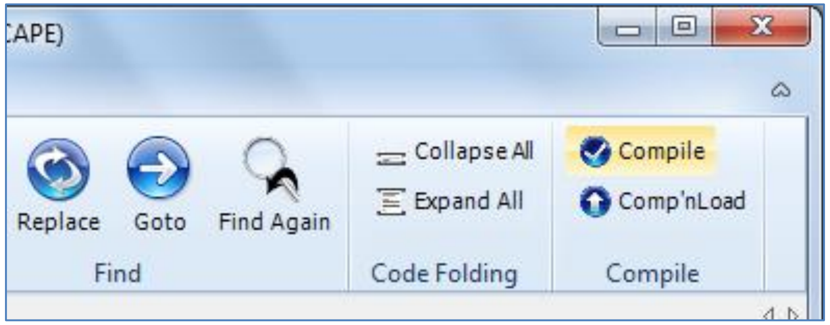
FAT format the uSD card then go back to **ViSi** project window.



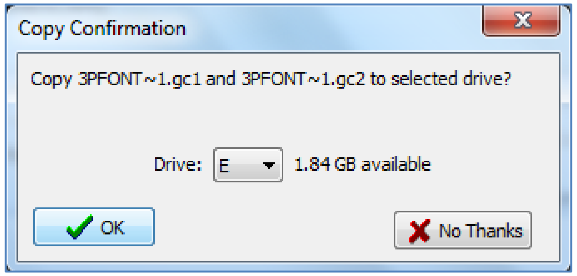
```
    wend
*/

    // Form1 1.0 generated 5/02/2013 2:43:37 PM

    // Strings1 1.0 generated 5/02/2013 2:43:37 PM
    media_SetSector(0, Strings1FontStartL) ;    // must come b4 setting
    txt_FontID(MEDIA) ; // Font index correct at time of code generatio
    txt_FGcolour(WHITE) ;
    txt_BGcolour(BLACK) ;
    gfx_MoveTo(18 , 20) ;
    print("Your text goes here") ;

    // Strings2 1.0 generated 5/02/2013 2:43:37 PM
    media_SetSector(0, Strings2FontStartL) ;    // must come b4 setting
    txt_FontID(MEDIA) ; // Font index correct at time of code generatio
    txt_FGcolour(WHITE) ;
    txt_BGcolour(BLACK) ;
    gfx_MoveTo(6 , 13) ;
    print("Your text goes here") ;

    repeat
```

Property	Value
Name	Form1
Bgtype	Color
Color	■ BLACK
Image	(None)

On the **ViSi** window **Home** ribbon in WS4 click the **Compile** button. At this point you might be prompted to save the file. Choose a location and type in the filename you desire (3PFontsDemo was used for this example).



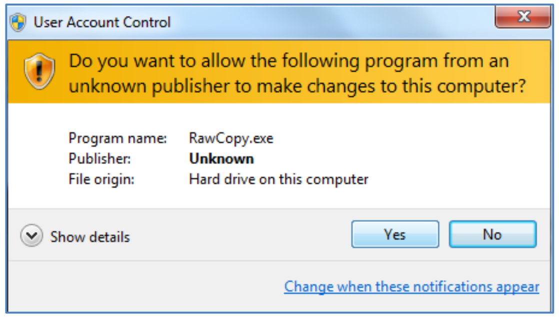
After clicking on **Compile**, you will be prompted to select and confirm the drive to where the font files will be saved. Click **OK** to confirm and start copying.


Copy Confirmation

Copy 3PFONT~1.gc1 and 3PFONT~1.gc2 to selected drive?

Drive: E 1.84 GB available

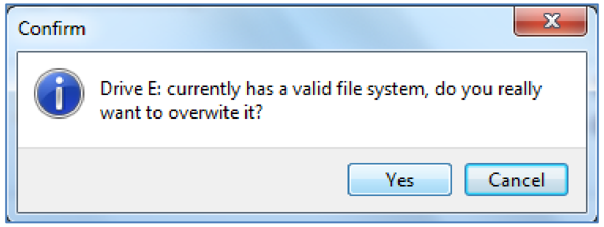
Windows security might prompt whether to allow RawCopy.exe to run, click **Yes**.


User Account Control

Do you want to allow the following program from an unknown publisher to make changes to this computer?

Program name: RawCopy.exe
Publisher: **Unknown**
File origin: Hard drive on this computer

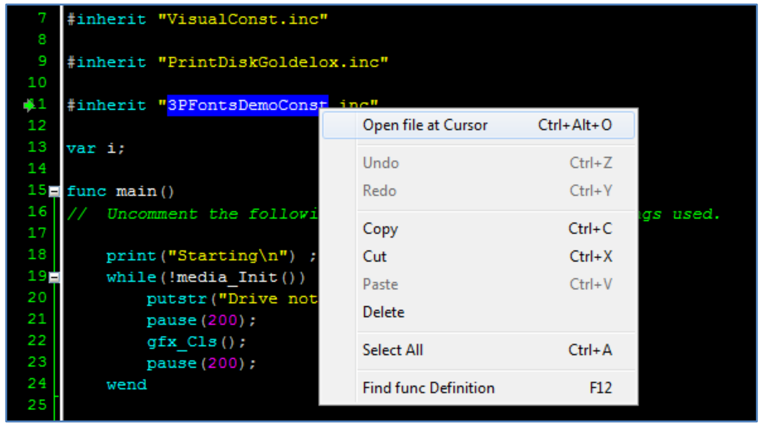
Another prompt will appear asking you if you want to overwrite the files on the selected drive, click **Yes.**


Confirm

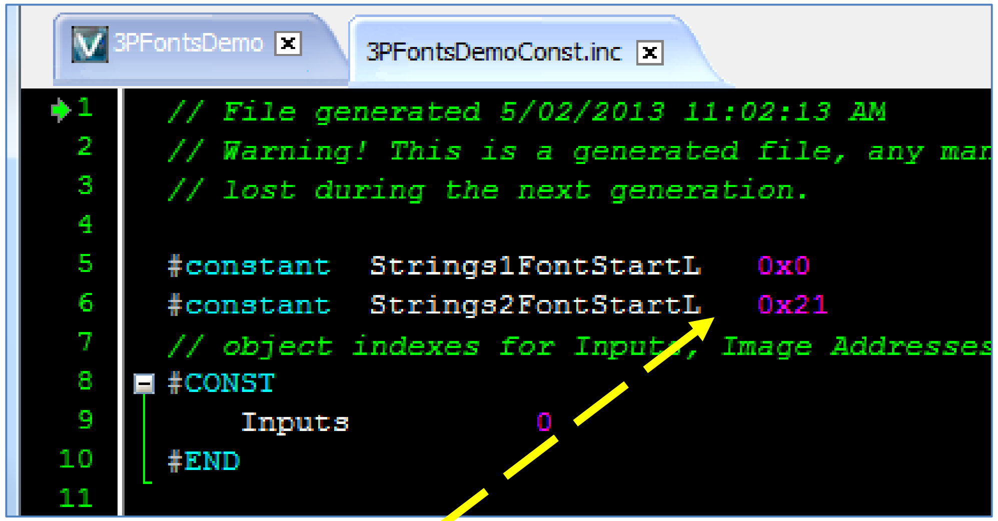
Drive E: currently has a valid file system, do you really want to overwrite it?

That's it! You have successfully copied the font data to the uSD card. Eject from the USB and adapter and insert it to the Display module you are using for this demo. (uOLED-128-G2 in this example)

Go back to the WS4 IDE, **right click** on the **3PFontsDemo** in the statement #inherit "3PFontsDemoConst.inc".


```
#inherit "VisualConst.inc"

#inherit "PrintDiskGoldelox.inc"

#inherit "3PFontsDemoConst.inc"

var i;

func main()
//  Uncomment the followi                                   gs used.

    print("Starting\n") ;
    while(!media_Init())
        putstr("Drive not
        pause(200);
        gfx_Cls();
        pause(200);
    wend
```

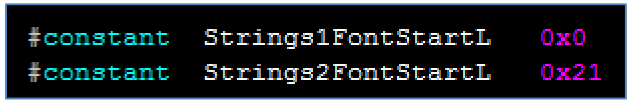
Next, click "Open file at Cursor" and this should open a new screen as shown below:



3PFontsDemo 3PFontsDemoConst.inc

```
// File generated 5/02/2013 11:02:13 AM
// Warning! This is a generated file, any mar
// lost during the next generation.

#constant   Strings1FontStartL    0x0
#constant   Strings2FontStartL    0x21
// object indexes for Inputs, Image Addresses
#CONST
    Inputs          0
#END
```

Please take note of the values at the arrow tip in the foregoing picture which appears on your screen, write them down as we need it later for the font simulation. They represent the sector in the uSD card where the font data is stored in Hex format.



```
#constant   Strings1FontStartL    0x0
#constant   Strings2FontStartL    0x21
```

Here, the first font points to sector 0x0 and the second is on 0x21. In case you see different values, that's not a problem, just note them down. This ends the first part of this application note.

Simulation

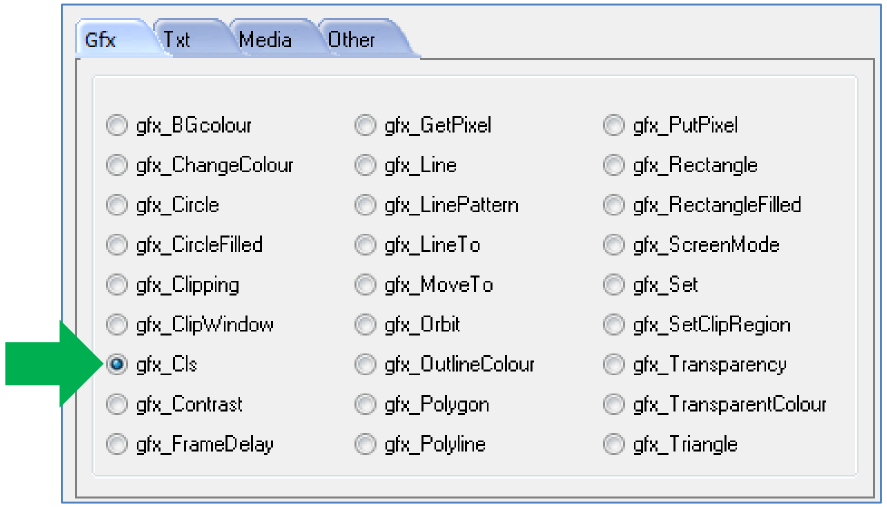
The display must be configured as a slave device first before it can be controlled by a host. For instructions on how to launch Workshop 4, how to connect the display module to the PC, and how to configure the display as a slave device, kindly refer to the section "**Setup Procedure**" of the application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

Serial Goldelox Getting Started - The SPE Application

This application note also introduces the user to the Serial Protocol thru the use of the Serial Commander.

Displaying the Fonts

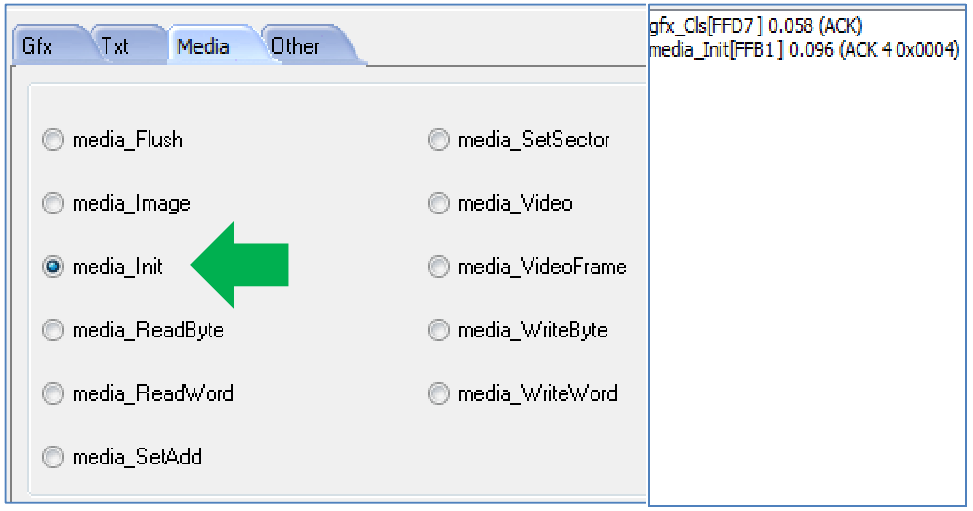
Click to open the **Gfx** tab. Select **gfx_Cls** and press the **Send** button below.



Gfx Txt Media Other

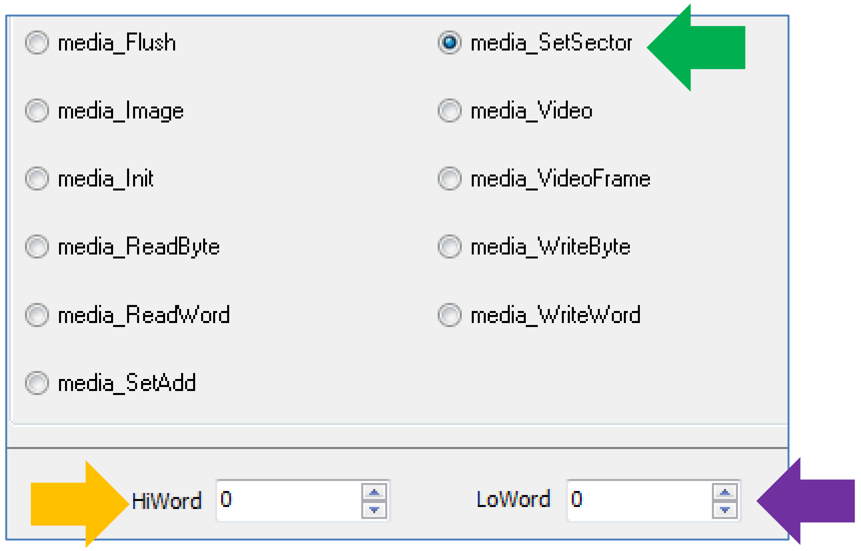
gfx_BGcolour	gfx_GetPixel	gfx_PutPixel
gfx_ChangeColour	gfx_Line	gfx_Rectangle
gfx_Circle	gfx_LinePattern	gfx_RectangleFilled
gfx_CircleFilled	gfx_LineTo	gfx_ScreenMode
gfx_Clipping	gfx_MoveTo	gfx_Set
gfx_ClipWindow	gfx_Orbit	gfx_SetClipRegion
[✓ gfx_Cls]	gfx_OutlineColour	gfx_Transparency
gfx_Contrast	gfx_Polygon	gfx_TransparentColour
gfx_FrameDelay	gfx_Polyline	gfx_Triangle

This will clear the 4D display screen. Next click on the Media tab and select media_Init and press Send.



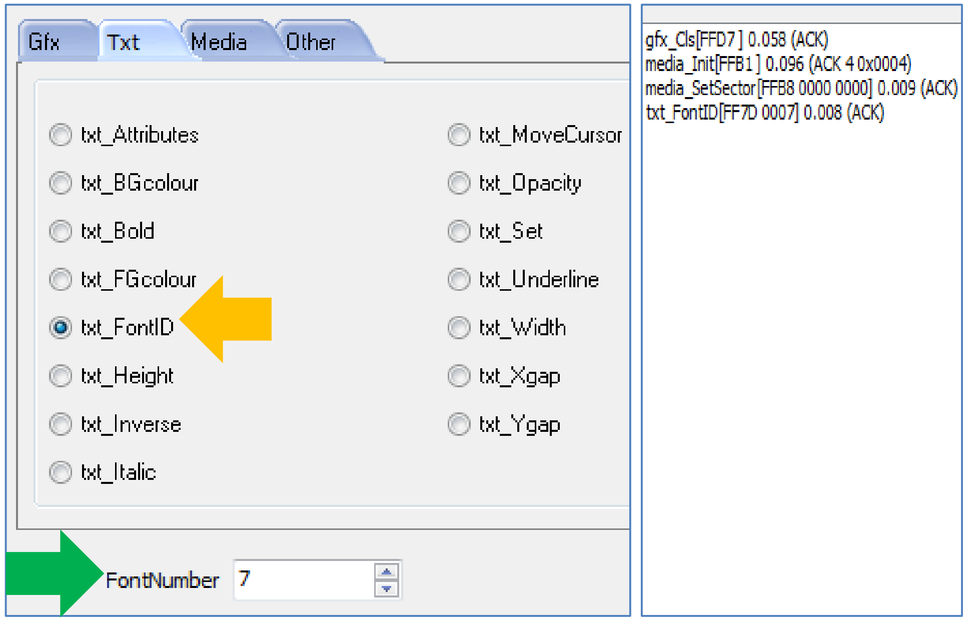
```
gfx_Cls[FFD7] 0.058 (ACK)
media_Init[FFB1] 0.096 (ACK 4 0x0004)
```

Next, from the same menu, select media_SetSector. Here you are prompted to enter the Sector addresses. Remember the Hex values from part 1 of this section? The values are 0x0 and 0x21. Now we need to use these values. Inside the 3PFontsDemo.4DViSi are two third-party font types namely ***Sony Sketch EF*** located at **0x0** and ***Harlow Solid Italic*** located at **0x21**. If you get different numbers, use those what you got instead.
The first exercise will be to use the *Sony Sketch EF* font, here located at 0x0, meaning **HiWord = 0** and **LoWord = 0** in decimal form. 0x0 is "0" and "0" for HiWord and LoWord respectively. The screen defaults is zero for both thus we need not enter anything on the boxes below and simply press **Send**.



HiWord 0 LoWord 0

Now open the **Txt** menu by clicking on the Txt tab. Select **txt_FontID** and enter "7" in the **FontNumber** box. See below photo for details:

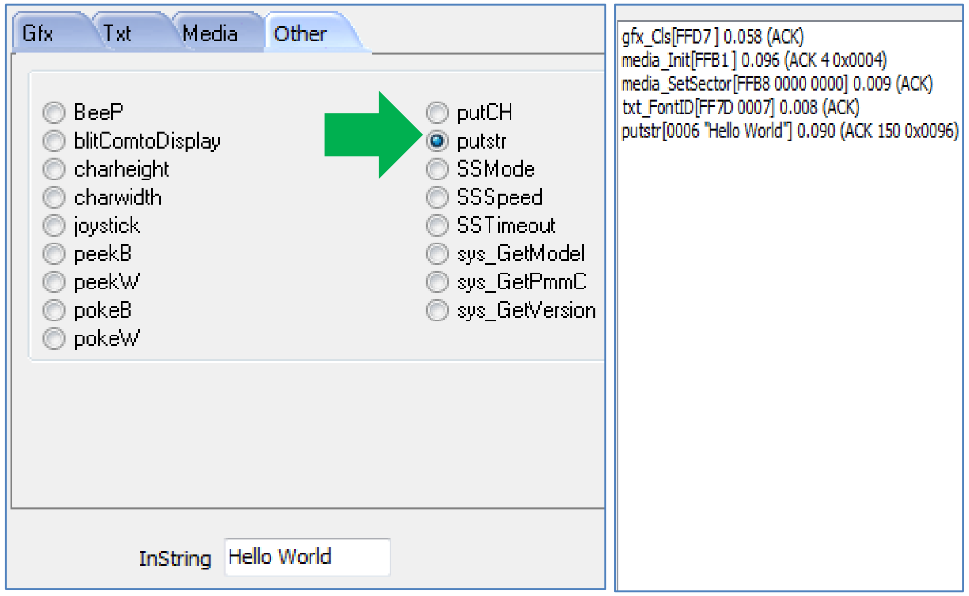


FontNumber 7

```
gfx_Cls[FFD7 ] 0.058 (ACK)
media_Init[FFB1 ] 0.096 (ACK 4 0x0004)
media_SetSector[FFB8 0000 0000] 0.009 (ACK)
txt_FontID[FF7D 0007] 0.008 (ACK)
```

FontNumber = 7 denotes media(i.e. uSD_Card) fonts which are fonts not included in the 4DSystem fonts.

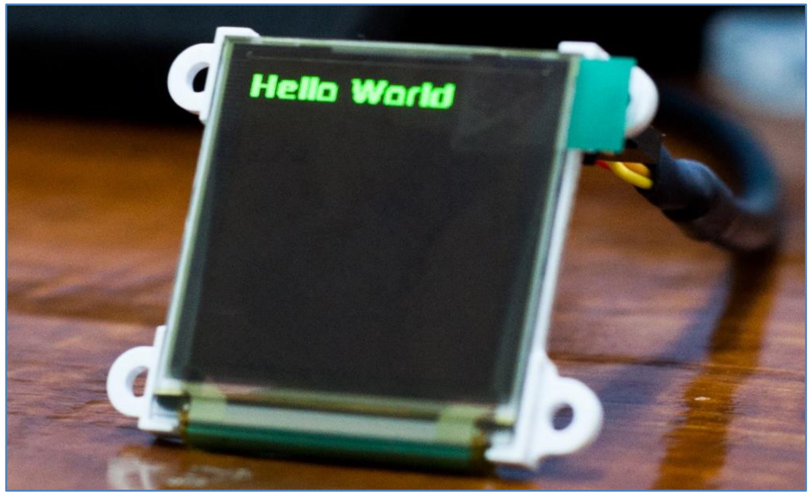
Next step, click on the Other tab to open the Other menu. At this point, we are ready to write and display text on the display module screen. Select **putstr** and type "Hello World" or any text you may prefer. For this example, we type "Hello World". See photo below for details.



InString Hello World

```
gfx_Cls[FFD7 ] 0.058 (ACK)
media_Init[FFB1 ] 0.096 (ACK 4 0x0004)
media_SetSector[FFB8 0000 0000] 0.009 (ACK)
txt_FontID[FF7D 0007] 0.008 (ACK)
putstr[0006 "Hello World"] 0.090 (ACK 150 0x0096)
```

Looking at your display, it should look and show these:

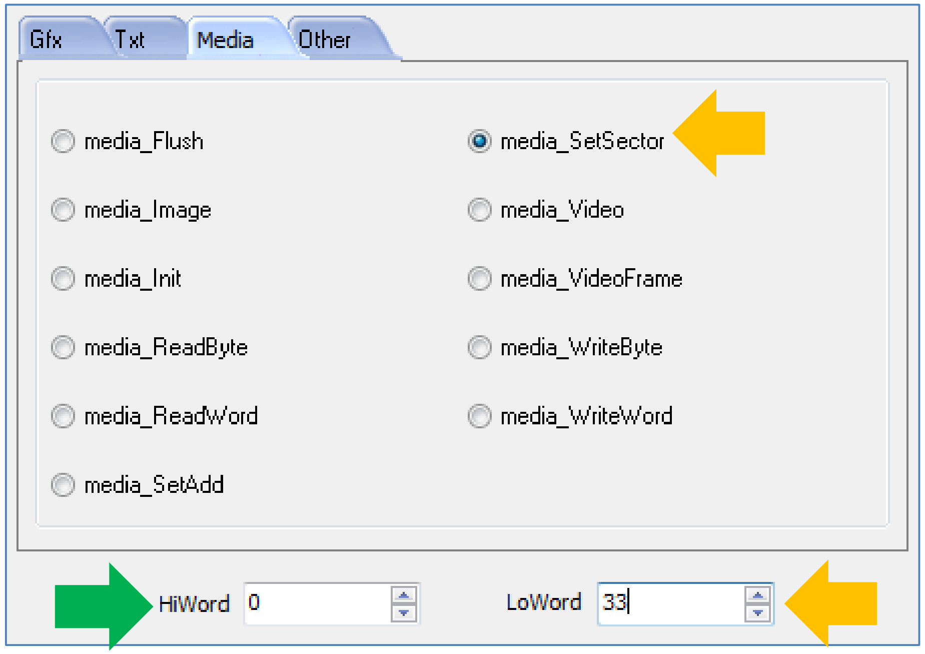


If the display scrolls, just send **SSTimeout** from the **Other** tab.

Choosing and Displaying another Font

As mentioned earlier, a second third-party font has been embedded in the 3PFontsDemo Visi application which is the ***Harlow Solid Italic*** type font. To do so, repeat all the previous steps starting with the **gfx-Cls** command. When you reach the **media_SetSector** portion enter **HiWord = 0** and **LoWord = 33** in the boxes provided. This points now to the sector of the uSD where the second font data is stored.

Note: 33 is the decimal equivalent of 0x21 which is hexadecimal notation. In case you got a different value convert it to decimal and enter this in the LoWord box.

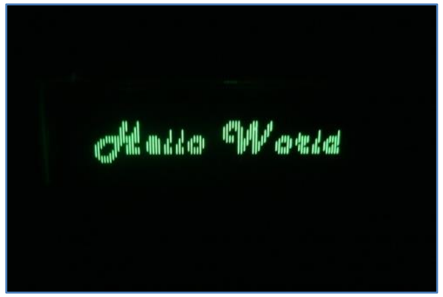


media_SetSector selected

HiWord: 0

LoWord: 33

Follow the succeeding steps as in the first font exercise. Sending the next string to the display assuming we type again "Hello World" will show:



User Experiments

The user may use the Serial Commander to change the text style, FGcolour, BGcolour, Bold, Italic, underline or other attributes and properties as the user may prefer and be able to view the results on the display.

Conclusion

This application note has shown the user the versatility of the 4D Systems Intelligent displays in particular the Goldelox series which is featured in this application note being capable of displaying custom or third party fonts other than the 4D System fonts available by default.

Further, it is our hope that we have delivered useful information which guarantees the user the ease-of-use in setting up 4D systems displays for serial display applications.

Proprietary Information

Disclaimer of Warranties & Limitation of Liability