APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

ViSi: Pixxi Flash File System

DOCUMENT DATE: **23rd November 2020**
DOCUMENT REVISION: **1.0**

Description

This application note provides a basic example on utilizing Flash File System for Pixxi which allows projects to store small files using Workshop4 and read it during runtime.

Before getting started, the following are required:

Hardware

- Any 4D Systems display module powered by any of the following processors:
  - Pixxi28
  - Pixxi44
- Programming Adaptor for target display module

Software

- Workshop4

This application note comes with one (1) ViSi project:

- PixxiFlashFS.4DViSi

Note: Using a non-4D programming interface could damage the processor and void the warranty.

Content

Application Overview

This document is mainly focused on showing the utilization of the Flash chip for additional small files in ViSi environment of the Workshop4 IDE. This feature is very useful in storing default settings for ViSi projects.

For more specific details regarding accessing flash chip, please refer to the following manual:

- **Pixxi Internal Functions Manual**

The simple project developed in this application note demonstrates reading strings from an additional settings file stored in the flash chip.

Setup Procedure

For instructions on how to launch Workshop4, how to open a **ViSi** project, and how to change the target display, kindly refer to the section "**Setup Procedure**" of the application note:

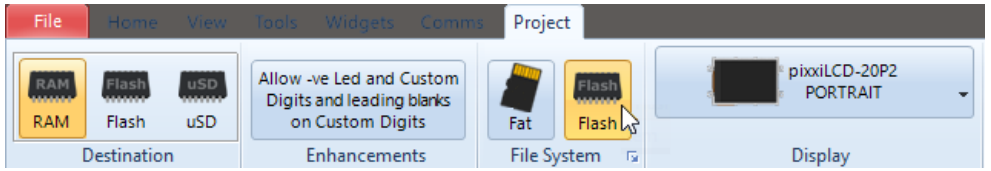
- **ViSi Getting Started - First Project for Pixxi Display Modules**

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 Pixxi Display Modules**

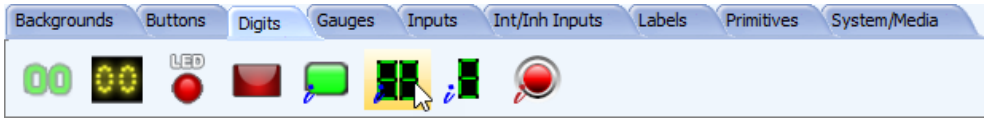
After creating your project, under *Project Menu* tab, set the File System to **Flash** as shown below:



Design the Project

Adding an Internal LedDigits

Add the widget to the Form by clicking on the **Digits** tab, then selecting the icon as shown below.



Click the WYSIWYG screen to place it, then drag it to the desired position.

The Internal Leddigits widget can be modified as needed through the **Object Inspector**.

Adding an Inherent Media Gauge

Add the widget to the Form by clicking on the **Int/Inh Inputs** tab, then selecting the icon as shown below.

Click the WYSIWYG screen to place it, then drag it to the desired position.

The Gauge widget can be modified as needed through the **Object Inspector**.

Programming the Display

It is always ideal to prepare the graphics interface before moving to programming the display. This lessens the number of times the users will have to proceed on doing a paste code action to regenerate the widget parameters in the code editor.

When the user interface is near its final stages, it is the best time to proceed with programming.

Initial ViSi Code

When starting a ViSi project, it includes a bare minimum project that contains commented initialization code for connected storage device.

```
func main()
//  var hstrings ; // Handle to access uSD strings, uncomment if required
//  var hFontx ;   // Handle to access uSD fonts, uncomment if required and change n to font number
//  Uncomment the following if uSD images, fonts or strings used.
/*
    // for Fat File System
    putstr("Mounting...\n");
    if (!(file_Mount()))
        while(!(file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
    endif
    // for (GCI) Flash File System
    media_Init() ;                  // or media_Init4(FLASH_ADDR_DEF_COMMAND) if flash > 16MB
//    gfx_TransparentColour(0x0020);  // uncomment if transparency required, please understand wh
//    gfx_Transparency(ON);           // uncomment if transparency required, as generally there i
    // for Fat File System
//  hFontn := file_LoadImageControl("NoName1.dan", "NoName1.gcn", 1); // Open handle to access uSD
//  hstrings := file_Open("NoName1.txf", 'r') ; // Open handle to access uSD strings, uncomment if
    hndl := file_LoadImageControl("NoName1.dat", "NoName1.gci", 1);
    // for (GCI) Flash File System
    hndl := file_LoadImageControl(0, 0, 3); // fonts, strings and WAV files are all entries in here
*/


    repeat
    forever
endfunc
```

Since this project will be utilizing the Flash File System. We only need to use the line related to it as highlighted in the preview image.

```
16  func main()
17
18      media_Init() ;                          // or media_Init4(FLASH_ADDR_DEF_COMMAND) if flash > 16MB
19      hndl := file_LoadImageControl(0, 0, 3); // fonts, strings and WAV files are all entries in here.
20
21      repeat
22      forever
23  endfunc
```

Feel free to remove the other unnecessary lines of code as done in this application note project.

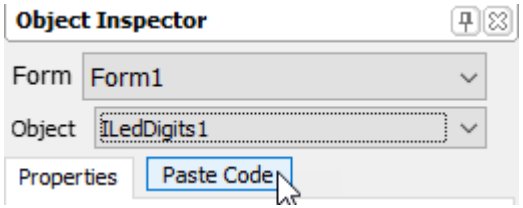
Paste the Internal LedDigits Code

ViSi provides Paste Code feature for widgets.

Place the blinking cursor before the repeat-forever loop.

```
16  func main()
17
18      media_Init() ;                          // or media_Init4(FLASH_ADDR_DEF_COMMAND) if flash > 16MB
19      hndl := file_LoadImageControl(0, 0, 3); // fonts, strings and WAV files are all entries in here.
20
21      repeat
22      forever
23  endfunc
```

Go to the Object Inspector, choose ILedDigits1 and click **Paste Code**



This will paste the useful lines of code that can be used for handling the widget.

```
// ILedDigits1 1.0 generated 11/23/2020 10:40:34 AM
gfx_LedDigits(value, vILedDigits1RAM, IILedDigits1) ;
```

The **first** line is simply a comment of containing the widget name and the date and time the piece of code was generated.

The **second** line can be used during setup and when updating and should be set at least once.

The parameters and variables required by the function are also automatically inserted near the start of the code.

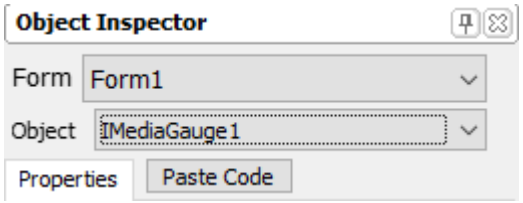
```
// ILedDigits1 Data Start
word IILedDigits1 56, 12, 65, 23, 5, 3, 0, 1, WHEAT, 0x18C3, 0x0
// ILedDigits1 Data End
```

Notice the fixed integers and colour included in the data block and the lines of code for the LedDigits.

These parameters in the data block and the useful lines of code discussed above are based on the values in the Object Inspector and may have to be regenerated by doing a new paste code action if the widget parameters are changed.

Paste the Inherent Media Gauge Code

Place the blinking cursor in the place where you want the code to be pasted. Then go to the Object Inspector, choose IMediaGauge1 and click **Paste Code**



This will paste the function for rendering the Gauge widget.

```
// IMediaGauge1 1.0 generated 11/23/2020 10:48:47 AM
vIMediaGauge1RAM[WIDGET_TAG] := gradientRAM ;            // uncomment gradientRAM variable
img_FunctionCall(hndl, igfx_MediaGauge, value, vIMediaGauge1RAM, IIMediaGauge1, 31, 0x0e) ;
```

The **first** line is simply a comment of containing the widget name and the date and time the piece of code was generated.

The **second** line should be set at least once during setup. This is unique to all 'Media' widgets. This allows the widget to draw its gradient parts.

Be sure to uncomment the variable gradientRAM and replace xxx as shown:

```
// uncomment and replace xxx with maximum of all inherent 'media' widgets
var gradientRAM[29+175*2] := [-1,-1,-9999,0,0,175] ;
```

The **last** line can be used to display the widget or update the widget value displayed on the screen by assigning a new value.

The parameters and variables required by the function are also automatically inserted near the start of the code.

```
// IMediaGauge1 Data Start
word IIMediaGauge1 0, 104, 175, 116, BLACK, BLACK, BLACK, WHITE, 0x1781, 0,
    0, 0, RED, BLACK, BLACK, RED, 8, 4, 0, 100, 0, 235, 305, 50, 25, 0, 1,
    0x33A7, 4, 20, 0
// IMediaGauge1 Data End
```

Notice the fixed integers and colour included in the data block and the lines of code for the gauge.

These parameters in the data block and the useful lines of code discussed above are based on the values in the Object Inspector and may have to be regenerated by doing a new paste code action if the widget parameters are changed.

If you check the pasted code, you'll find a comment regarding STACK requirement.

```
//#MODE RUNFLASH
#STACK 360
```

Follow this comment and raise STACK.

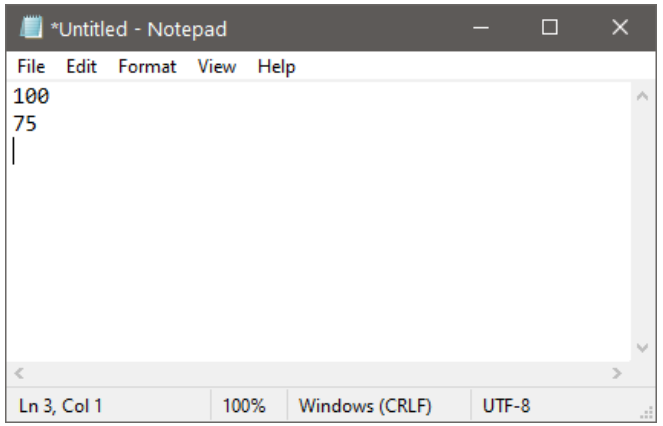
Adding Necessary Variable(s)

Some variables included in the generated code from Workshop4's Paste Code feature needs to be added to the project.

```
var value;
```

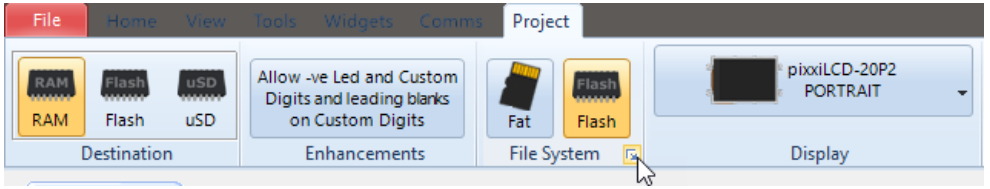
Adding Text Files to Flash

Open Notepad or any text editor and insert these 100 and 75 in separate lines

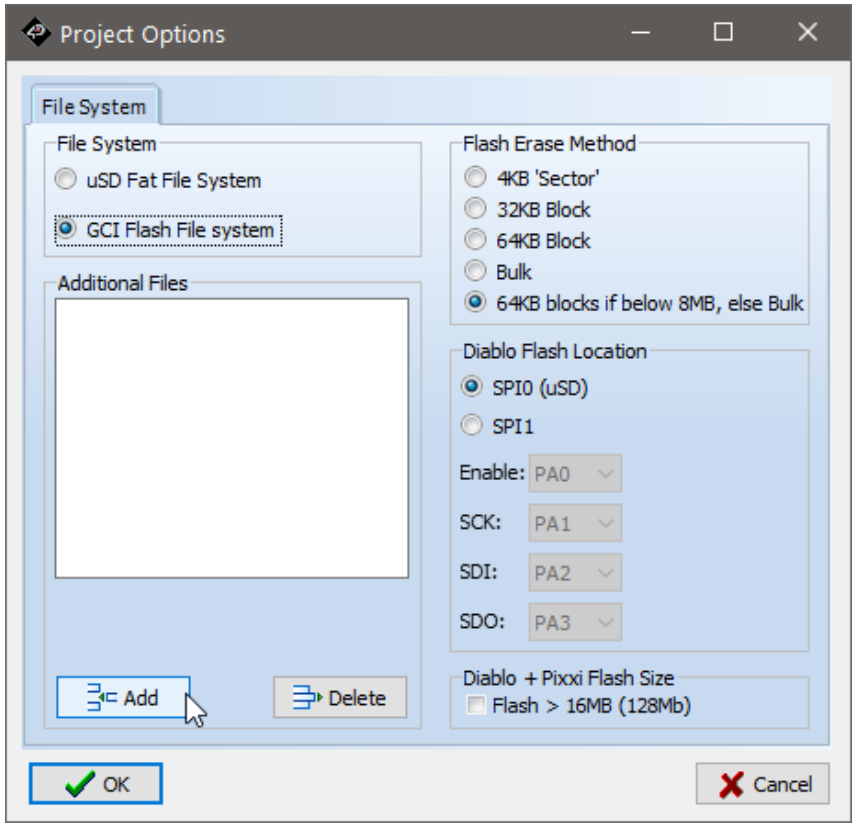


Save this file as *config.txt*

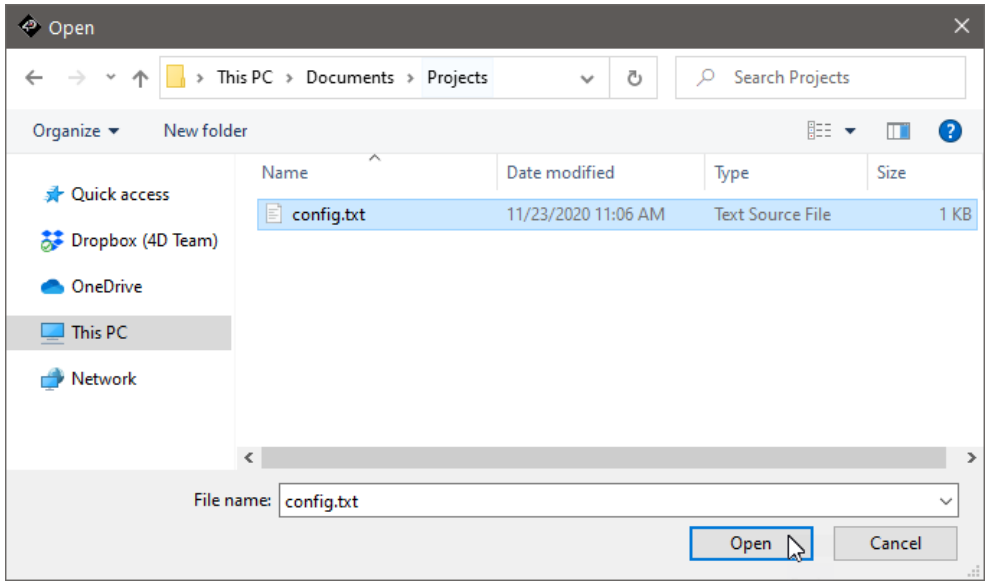
To add this file to Flash, go to ***Project Menu*** tab and press the button under File System as shown:



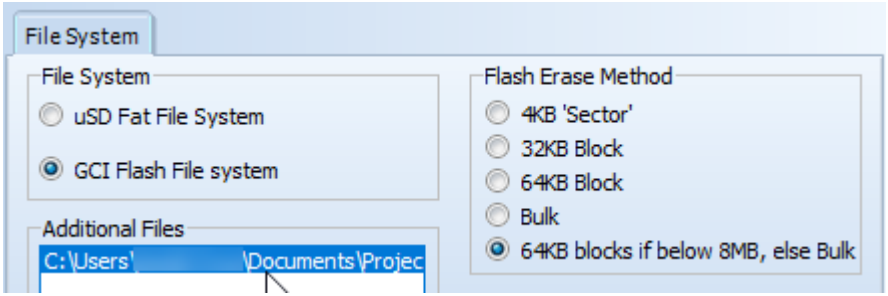
This will open a separate window:



Add the file by pressing the **Add** button and selecting the config file.



The file should appear in Additional Files as shown below.



Press **OK** to close the window and save your changes.

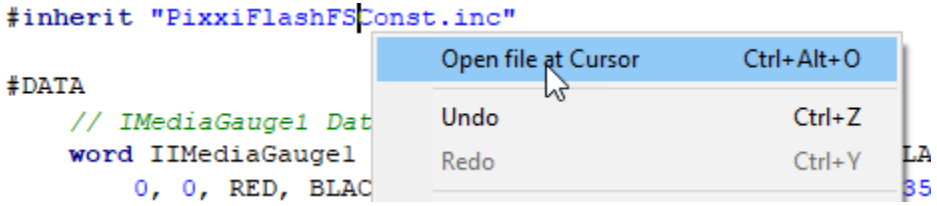
Accessing Text Files from Flash

Before proceeding with this step, try compiling your project to make sure you do not currently have any compilation issues. This will require you to save project if you have not already.

This will also generate the project's *Consts.inc file which contains some helpful constants.

```
#inherit "PixxiFlashFSConst.inc"
```

Put the cursor on this line and right click to bring a context menu.



From here, you can open the generated file.

If you have been following this application note, you should see something like this:

```
// File generated 11/23/2020 11:51:08 AM
// Warning! This is a generated file, any manual changes will be
// lost during the next generation.

#constant  IFONT_OFFSET     0
// object indexes into ImageControl
#CONST
    igfx_MediaGauge
    iExtraFiles1
#END


#constant  ffsEndUnits4k    0x0002  // First unused 4k Sector on Flash
#constant  ffsEnd4kH        0x0000  // Address of first unused 4k Sector on Flash, High
#constant  ffsEnd4kL        0x2000  // Address of first unused 4k Sector on Flash, Low
#constant  ffsEndUnits32k   0x0001  // First unused 32k Block on Flash
#constant  ffsEnd32kH       0x0000  // Address of first unused 32k Block on Flash, High
#constant  ffsEnd32kL       0x8000  // Address of first unused 32k Block on Flash, Low
#constant  ffsEndUnits64k   0x0001  // First unused 64k Block on Flash
#constant  ffsEnd64kH       0x0001  // Address of first unused 64k Block on Flash, High
#constant  ffsEnd64kL       0x0000  // Address of first unused 64k Block on Flash, Low

#IFNOT EXISTS NOGLOBALS
var hndl ;
#ENDIF
```

Notice the section with the comment: *object indexes into ImageControl*

This contains a list of keywords/constants that represent the widgets and additional files stored in the flash chip.

In this project, we have:

- Internal LedDigits – internal widgets do not require the Flash chip
- Inherent MediaGauge – stored as *igfx_MediaGauge*
- Additional text file – stored as *iExtraFiles1*

Read the file using the code below:

```
var temp[5], ptr;

ptr := str_Ptr(temp);
img_FileGetS(temp, 10, hndl, iExtraFiles1);   // Read the first value
str_GetW(&ptr, &value);                       // Convert String to 16-bit value

// ILedDigits1 1.0 generated 11/23/2020 10:40:34 AM
gfx_LedDigits(value, vILedDigits1RAM, IILedDigits1) ;

ptr := str_Ptr(temp);
img_FileGetS(temp, 10, hndl, iExtraFiles1);   // Read the next value
str_GetW(&ptr, &value);                       // Convert String to 16-bit value

// IMediaGauge1 1.0 generated 11/23/2020 12:20:10 PM
vIMediaGauge1RAM[WIDGET_TAG] := gradientRAM ;            // uncomment gradientRAM variable
img_FunctionCall(hndl, igfx_MediaGauge, value, vIMediaGauge1RAM, IIMediaGauge1, 31, 0x0e) ;
```

This will set the value of LedDigits and MediaGauge to the value stored in the text file.

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, 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 Pixxi Display Modules**

Proprietary Information

Disclaimer of Warranties & Limitation of Liability