APPLICATION NOTES

4D SYSTEMS

TURNING TECHNOLOGY INTO ART

ViSi Read and Write to microSD

DOCUMENT DATE: **22nd 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 demonstrating to the user the basic of reading and writing to a micro SD with the use of FAT16 internal functions of the DIABLO16 embedded graphics processor. The following are needed for this application note.

- Any of the following 4D Picaso display modules:
  gen4-uLCD-24PT
  gen4-uLCD-28PT
  gen4-uLCD-32PT
  uLCD-24PTU
  uLCD-28PTU
  uVGA-III

and other superseded modules which support the ViSi Genie environment

- The target module can also be a Diablo16 display
  gen4-uLCD-24D series
  gen4-uLCD-28D series
  gen4-uLCD-32D series
  gen4-uLCD-35D series
  gen4-uLCD-43D series
  gen4-uLCD-50D series
  gen4-uLCD-70D series
  uLCD-35DT
  uLCD-43D Series
  uLCD-70DT

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

- 4D Programming Cable or µUSB-PA5
- 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

Application Overview

This application note is intended to demonstrating to the user the basic of reading and writing to a micro SD with the use of FAT16 internal functions of the DIABLO16 embedded graphics processor.

Setup Procedure

For instructions on how to launch Workshop 4, 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 Picaso and Diablo16

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

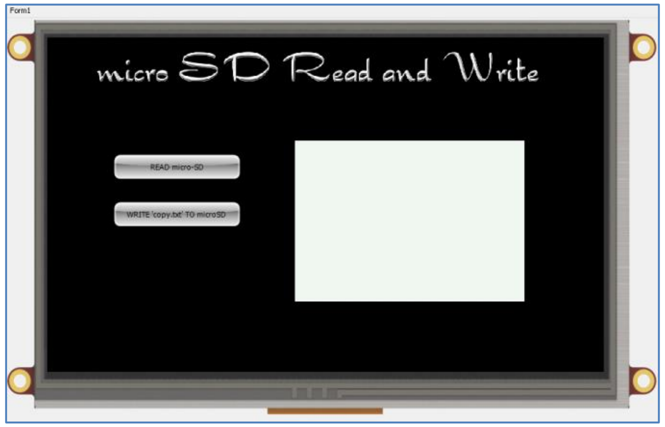
ViSi Getting Started - First Project for Picaso and Diablo16

Creating and copying of initial files on the micro SD

For this application project two files were copied to the micro-SD. The files were made using the Notepad text editor. Also a pair of 4DGL programs were copied to the micro-SD drive. These 4DGL files have a filename extension of .4XE. In addition, a couple of sound files are also copied to the micro-SD.

The contents of the micro SD shall be read by the display module and that all the contents are enlisted after reading the microSD drive. The initial files saved on the microSD will be used to demonstrate the read process.

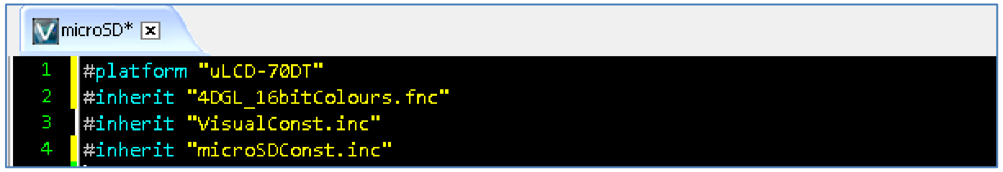
Design the Project



After all the objects have been laid-out, let's continue with the other half which involves the coding of the project. This will be presented in a sectional manner so as not to create confusion with the project. For an in-depth detail of the functions used in this application note please refer to the DIABLO16 Internal Functions Reference Manual.

The Include Section

This project starts with the identification of the platform being used as declared by the #platform function. For the program to be able to function properly files are included herein using the #inherit function. Three other files are included automatically during creation of the new project, namely: 4DGL_16bitColours.fnc, the VisualConst.inc, and user_imgConst.inc. Notice that the last include file is almost the same as the project filename. This file is automatically generated as soon as project saving is done.



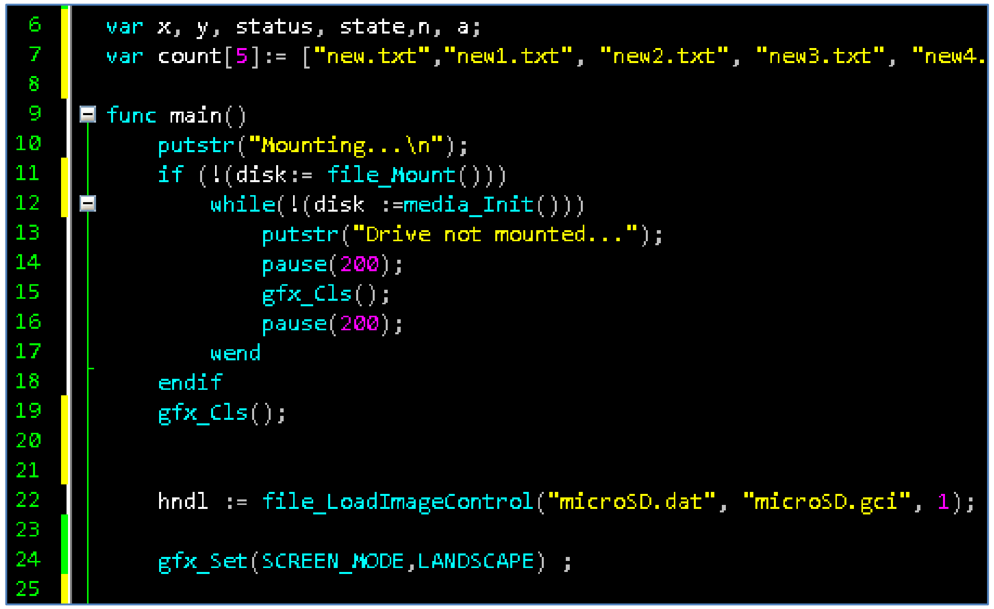
```
#platform "uLCD-70DT"
#inherit "4DGL_16bitColours.fnc"
#inherit "VisualConst.inc"
#inherit "microSDConst.inc"
```

The Main Program

The main program contains a simple program that first detects and initializes the micro SD card, the initial displaying of the objects used in the project and an endless loop that detects the buttons pressed by the user and subsequently executes other included statements.

The micro-SD Initialization

Let's start with the initialization of the uSD card. The uSD card contains all the image information about the objects used in the project. The object information and data are saved under a *.DAT and a *.GCI filename extension which is copied to the uSD during project compilation. Mounting of the disk in this application note was done using the following set of program statements.



```
    var x, y, status, state,n, a;
    var count[5]:= ["new.txt","new1.txt", "new2.txt", "new3.txt", "new4.

func main()
        putstr("Mounting...\n");
        if (!(disk:= file_Mount()))
            while(!(disk :=media_Init()))
                putstr("Drive not mounted...");
                pause(200);
                gfx_Cls();
                pause(200);
            wend
        endif
        gfx_Cls();


        hndl := file_LoadImageControl("microSD.dat", "microSD.gci", 1);

        gfx_Set(SCREEN_MODE,LANDSCAPE) ;

```

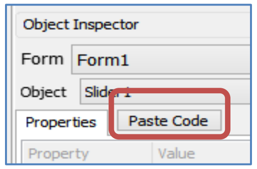
When starting a new project in the ViSi environment these set of statements are already included in the coding area. Notice a slight change has been made with while-condition. Instead of using the file_Mount() function, this was replaced with a media_Init() function. The media_Init() function checks the presence of the microSD. If a microSD is detect then the succeeding statements are executed. If not then a repetitive flashing of "Drive not mounted…" is displayed on the top left of the display module.

Another part of these set of statements uses a function file_LoadImageControl() to call on the object data/information files on the uSD drive. This initializes the data to be called in using the variable 'hndl'. In addition, the filenames for the dat and gci files are automatically changed to the project filename as soon as the project is saved.

Added to the statements above are the screen orientation setup function and the touch enable function. The screen orientation is set to portrait thereby providing an 800x480 pixel dimension. This setup is attained using the gfx_Set() function.

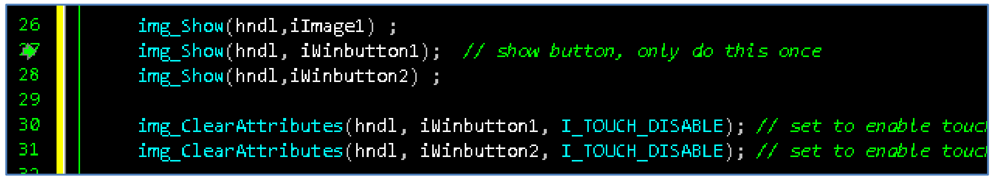
Having been able to load and initialize the uSD drive, the processor is now able to access the information stored therein. As mentioned from the previous section, the filenames with an extension of DAT and GCI has the image data and information. Therefore, the next part of the main program is to display all the objects that were placed on the Workshop IDE form viewer.

A special button from the Object Inspector can help reduce the time of coding of this part. The 'Paste Code" simply pastes object code.

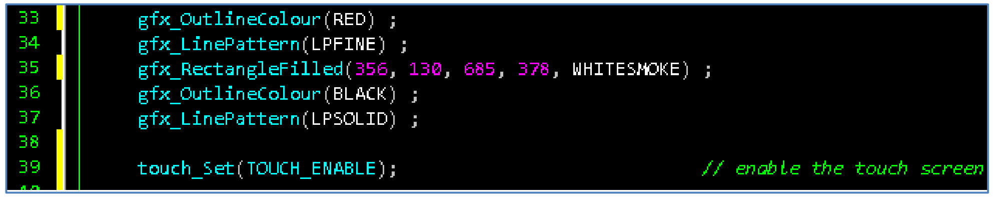


Displaying the Objects

In this part of the program, the img_Show() function simply calls out the object image and information found in the microSD drive. This set of statements call out the images from their handlers and are displayed on screen. The img_ClearAttributes() function is used to enable the touch feature on the winbutton objects.



```
        img_Show(hndl,iImage1) ;
        img_Show(hndl, iWinbutton1);  // show button, only do this once
        img_Show(hndl,iWinbutton2) ;

        img_ClearAttributes(hndl, iWinbutton1, I_TOUCH_DISABLE); // set to enable touc
        img_ClearAttributes(hndl, iWinbutton2, I_TOUCH_DISABLE); // set to enable touc
```

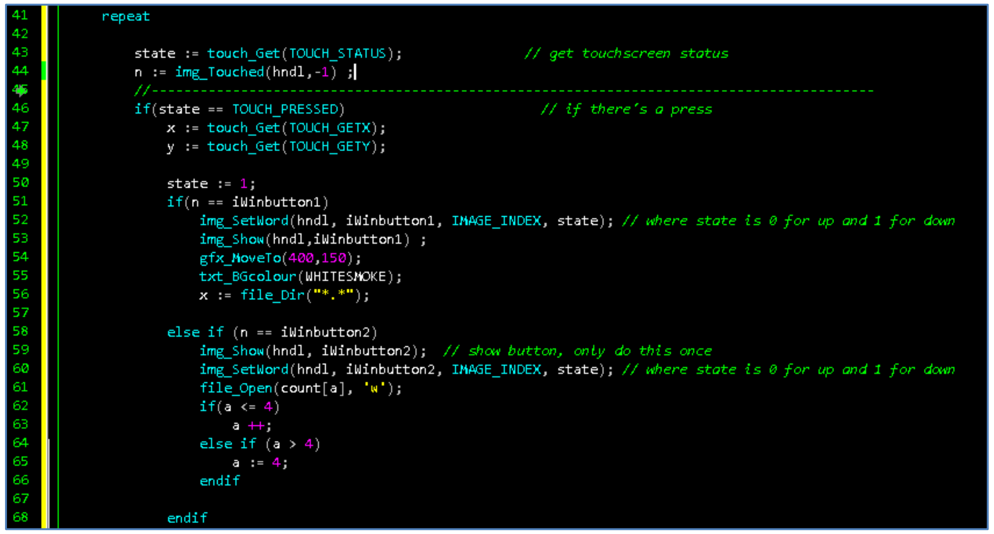


```
        gfx_OutlineColour(RED) ;
        gfx_LinePattern(LPFINE) ;
        gfx_RectangleFilled(356, 130, 685, 378, WHITESMOKE) ;
        gfx_OutlineColour(BLACK) ;
        gfx_LinePattern(LPSOLID) ;

        touch_Set(TOUCH_ENABLE);                            // enable the touch screen
```

The statements above is a continuation of the initial displaying the objects the statements above displays a colour filled rectangle. This rectangle will serve as the background for the displayed content of the microSD. At the end of the initial displaying of objects is the touch feature of the display module being enabled.

The Repeat-forever Loop

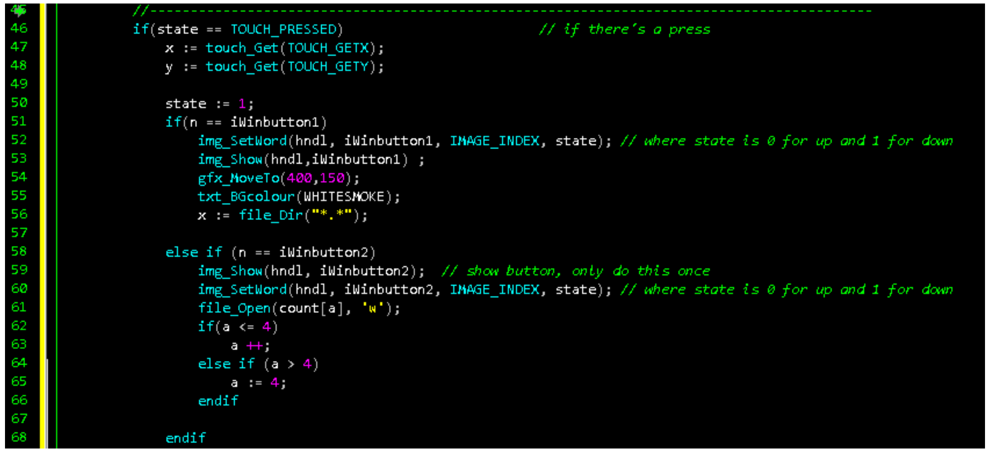
This section of the main program contains statements that are continually run by the processor. Referring to the image below the processor continually checks on the status of the touch panel.



```
    repeat

        state := touch_Get(TOUCH_STATUS);               // get touchscreen status
        n := img_Touched(hndl,-1) ;
        //-------------------------------------------------------------------------------------------
        if(state == TOUCH_PRESSED)                       // if there's a press
            x := touch_Get(TOUCH_GETX);
            y := touch_Get(TOUCH_GETY);

            state := 1;
            if(n == iWinbutton1)
                img_SetWord(hndl, iWinbutton1, IMAGE_INDEX, state); // where state is 0 for up and 1 for down
                img_Show(hndl,iWinbutton1) ;
                gfx_MoveTo(400,150);
                txt_BGcolour(WHITESMOKE);
                x := file_Dir("*.*");

            else if (n == iWinbutton2)
                img_Show(hndl, iWinbutton2);  // show button, only do this once
                img_SetWord(hndl, iWinbutton2, IMAGE_INDEX, state); // where state is 0 for up and 1 for down
                file_Open(count[a], 'w');
                if(a <= 4)
                    a ++;
                else if (a > 4)
                    a := 4;
                endif

            endif
```

The current condition of the touch panel is saved using the variable "state". Following the detection of the touch panel. Objects are also checked for a 'touched' condition. If an object is touched, the objects's index number is returned by the display module.

After checking for the status of the touch panel and touch status of objects, we now have a group of if-else-if conditions that are divided into three sections. The conditions are based on the status of the touch panel, these are – 'Touch Pressed', 'Touch Released' and 'Touch Moving'.

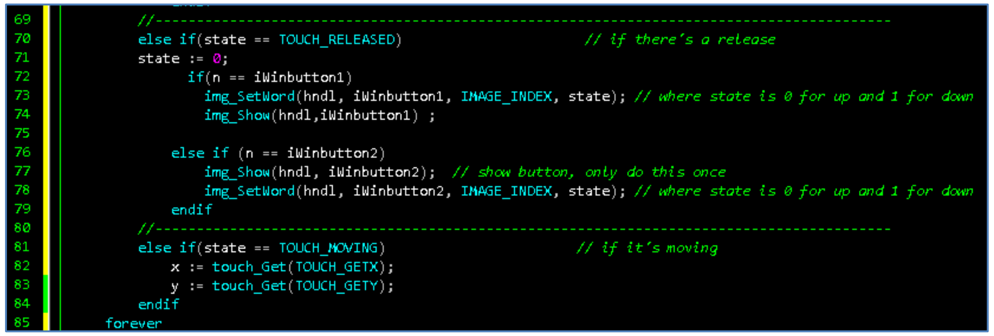


```
//----------------------------------------------------------------------------------------
if(state == TOUCH_PRESSED)                          // if there's a press
    x := touch_Get(TOUCH_GETX);
    y := touch_Get(TOUCH_GETY);

    state := 1;
    if(n == iWinbutton1)
        img_SetWord(hndl, iWinbutton1, IMAGE_INDEX, state); // where state is 0 for up and 1 for down
        img_Show(hndl,iWinbutton1) ;
        gfx_MoveTo(400,150);
        txt_BGcolour(WHITESMOKE);
        x := file_Dir("*.*");

    else if (n == iWinbutton2)
        img_Show(hndl, iWinbutton2);  // show button, only do this once
        img_SetWord(hndl, iWinbutton2, IMAGE_INDEX, state); // where state is 0 for up and 1 for down
        file_Open(count[a], 'w');
        if(a <= 4)
            a ++;
        else if (a > 4)
            a := 4;
        endif

    endif
```

Let us start with the Touch pressed touch condition. There are two Winbuttons in this project one button is used to read and display the content read from a micro SD. When a touch pressed on the iWinbutton1, the processor is directed to read the content of the microSD. To read the content of the microSD device, the file_Dir() function is used. To read a particular type of file the user may specify the file extension. For this application, wildcards (*.*) are used to detect and display each and every file inside the microSD.

On the other hand, files can also be written to the micro-SD drive. With the use of the file_Open() function, the user can create a new file in micro-SD drive. The file_Open function allows the user to read/write and append to an existing file. While on the other hand, using a file_Open function to a non-existent file will create a file according to the file name and extension specified. For this application, a new file is saved into the micro-SD each time the 'Write new.txt' file button is pressed. After adding the new file, a press on the read will update the list the display the new files created.

Last part of this simple application program is related to a touch_released condition. This condition simply returns the iWinbuttons to their frame 1. The transition of the frame 0 and 1 between a touch_pressed and a touch_released creates a simple animation during the transitions.



```
    //----------------------------------------------------------------------------------------
    else if(state == TOUCH_RELEASED)                        // if there's a release
    state := 0;
          if(n == iWinbutton1)
            img_SetWord(hndl, iWinbutton1, IMAGE_INDEX, state); // where state is 0 for up and 1 for down
            img_Show(hndl,iWinbutton1) ;

        else if (n == iWinbutton2)
            img_Show(hndl, iWinbutton2);  // show button, only do this once
            img_SetWord(hndl, iWinbutton2, IMAGE_INDEX, state); // where state is 0 for up and 1 for down
        endif
    //----------------------------------------------------------------------------------------
    else if(state == TOUCH_MOVING)                         // if it's moving
        x := touch_Get(TOUCH_GETX);
        y := touch_Get(TOUCH_GETY);
    endif
forever
```

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 (this option is not available for Goldelox displays), 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 Picaso and Diablo16

The uLCD-32PTU and uLCD-35DT display modules are commonly used as examples, but the procedure is the same for other displays.

Proprietary Information

Disclaimer of Warranties & Limitation of Liability