APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

ViSi: Receiving Return Value from Child Program

DOCUMENT DATE:	**3rd APRIL 2020**
DOCUMENT REVISION:	**1.0**

Description

This Application note shows how to call and run child programs stored in the uSD card.

Before getting started, the following are required:

Hardware

- Any 4D Systems display module powered by any of the following processors:
    - Pixxi28/44
    - Diablo16
    - Picaso
- Programming Adaptor for target display module
- uSD Card

Software

- Workshop4

This application note comes with one (1) ViSi project and a zip file containing the child program files to be copied to the uSD card:

- main.4DViSi
- uSD_Files.zip

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

Content

Application Overview

This application note demonstrates on how to run child programs residing outside the processor's internal flash banks. This practice is useful for large applications that could not fit in the processors internal flash memory. The child program in this application note will execute its coded tasks using the arguments passed from the main program then return the values back to the mother program.

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 Picaso and Diablo16
- ViSi Getting Started – First Project for Pixxi Displays

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

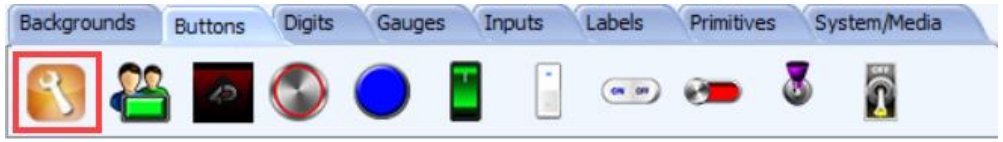
- ViSi Getting Started – First Project for Picaso and Diablo16
- ViSi Getting Started – First Project for Pixxi Displays

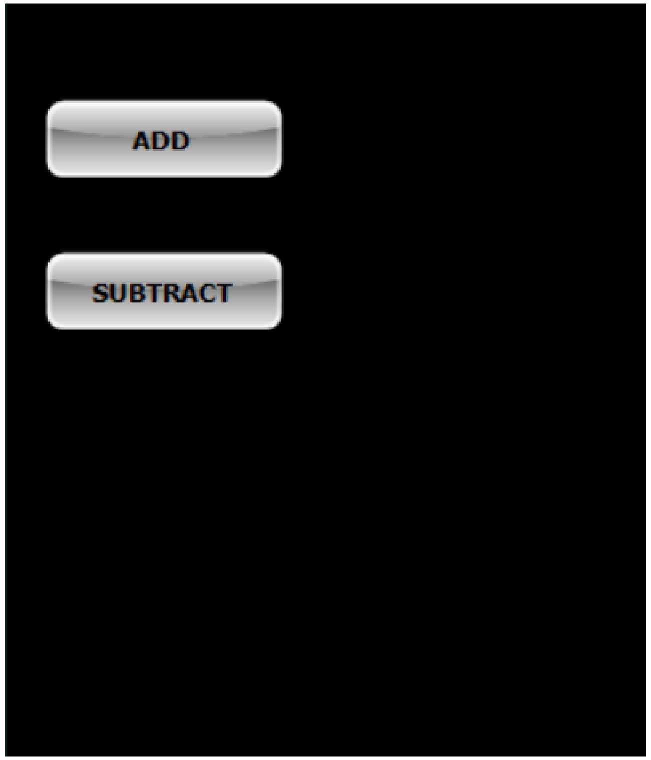
Mother Program

The mother program is the one residing in the processor's main Flashbank.

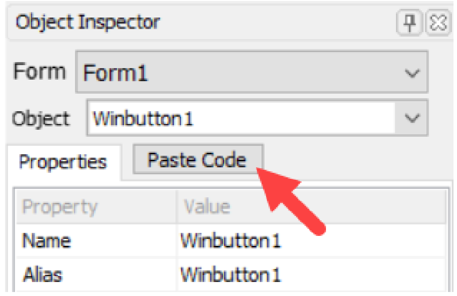
Design the Project

Add two buttons to Form0.





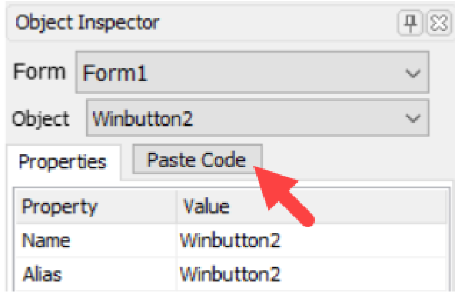
Paste the code onto the Code editor by clicking on widget and then pressing the **Paste Code** button in the object inspector.



Object Inspector

Form: Form1
Object: Winbutton1
Properties | Paste Code

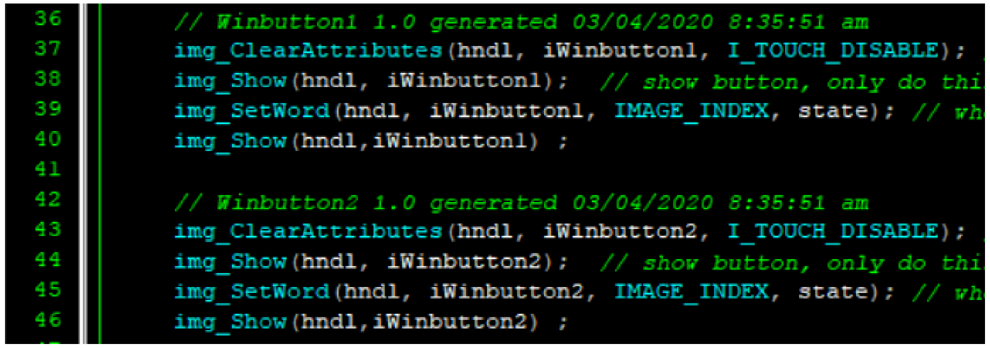
Property	Value
Name	Winbutton1
Alias	Winbutton1



Object Inspector

Form: Form1
Object: Winbutton2
Properties | Paste Code

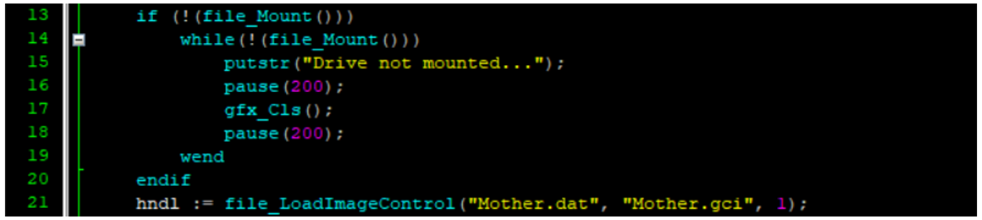
Property	Value
Name	Winbutton2
Alias	Winbutton2



```
        // Winbutton1 1.0 generated 03/04/2020 8:35:51 am
        img_ClearAttributes(hndl, iWinbutton1, I_TOUCH_DISABLE);
        img_Show(hndl, iWinbutton1);  // show button, only do thi
        img_SetWord(hndl, iWinbutton1, IMAGE_INDEX, state); // wh
        img_Show(hndl,iWinbutton1) ;

        // Winbutton2 1.0 generated 03/04/2020 8:35:51 am
        img_ClearAttributes(hndl, iWinbutton2, I_TOUCH_DISABLE);
        img_Show(hndl, iWinbutton2);  // show button, only do thi
        img_SetWord(hndl, iWinbutton2, IMAGE_INDEX, state); // wh
        img_Show(hndl,iWinbutton2) ;
```

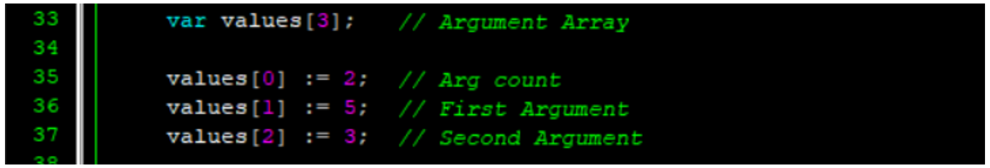
Mother Program Code

The program starts by mounting the media containing the child program. This is provided in the ViSi template by simply uncommenting these code snippets in the code editor.



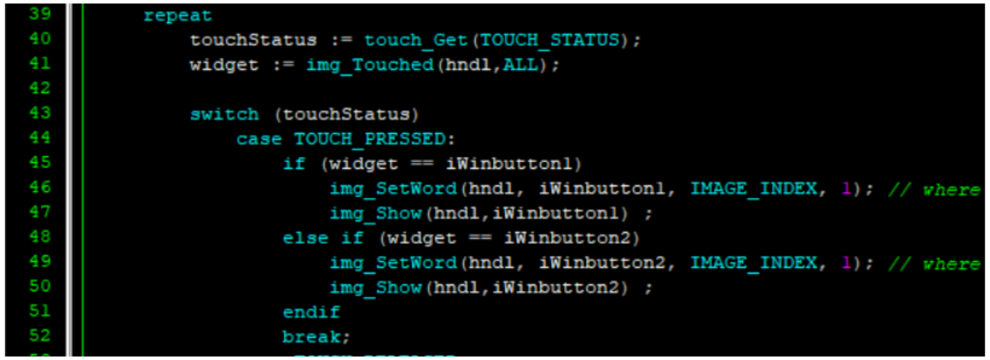
```
    if (!(file_Mount()))
        while(!(file_Mount()))
            putstr("Drive not mounted...");
            pause(200);
            gfx_Cls();
            pause(200);
        wend
    endif
    hndl := file_LoadImageControl("Mother.dat", "Mother.gci", 1);
```

The argument array containing the values for sending to the Child programs are prepared in this section for use with the child programs. The first element contains the count for the number of arguments the Child program will receive, in this case there are two values as arguments.



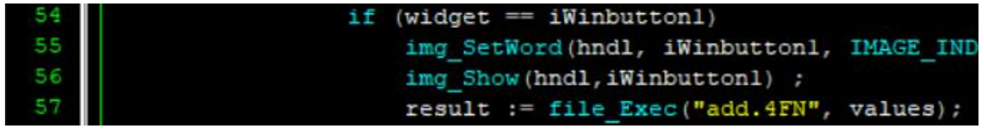
```
var values[3];    // Argument Array

values[0] := 2;   // Arg count
values[1] := 5;   // First Argument
values[2] := 3;   // Second Argument
```

In the Main Loop, the program will constantly check for button action in this section.



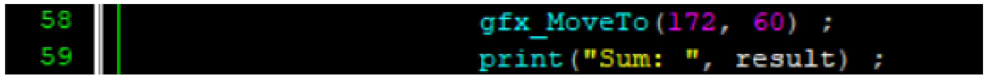
```
repeat
    touchStatus := touch_Get(TOUCH_STATUS);
    widget := img_Touched(hndl,ALL);

    switch (touchStatus)
        case TOUCH_PRESSED:
            if (widget == iWinbutton1)
                img_SetWord(hndl, iWinbutton1, IMAGE_INDEX, 1); // where
                img_Show(hndl,iWinbutton1) ;
            else if (widget == iWinbutton2)
                img_SetWord(hndl, iWinbutton2, IMAGE_INDEX, 1); // where
                img_Show(hndl,iWinbutton2) ;
            endif
            break;
```

If the button for ADD (Winbutton1) is toggled by touch, the child program with the **filename** “add.4FN” residing in the external media will be called through the **file_Exec(filename, arglistptr)** function. The **arglistptr** contains the argument array **values** containing the values to be calculated by the child program.



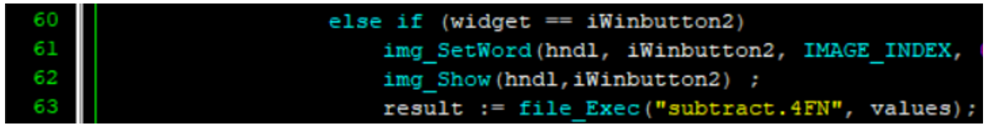
```
if (widget == iWinbutton1)
    img_SetWord(hndl, iWinbutton1, IMAGE_IND
    img_Show(hndl,iWinbutton1) ;
    result := file_Exec("add.4FN", values);
```

The return value from the child program is then stored in the variable **result** and then printed beside the button.



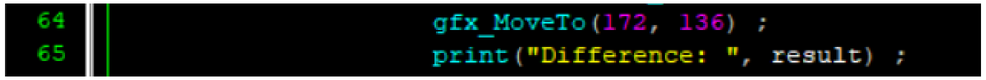
```
gfx_MoveTo(172, 60) ;
print("Sum: ", result) ;
```

If the button for SUBTRACT (Winbutton2) is toggled by touch, the child program with the **filename** “subtract.4FN” residing in the external media will be called through the **file_Exec(filename, arglistptr)** function. The **arglistptr** contains the argument array **values** containing the values to be calculated by the child program.



```
else if (widget == iWinbutton2)
    img_SetWord(hndl, iWinbutton2, IMAGE_INDEX,
    img_Show(hndl,iWinbutton2) ;
    result := file_Exec("subtract.4FN", values);
```

The return value from the child program is then stored in the variable **result** and then printed beside the button.



```
gfx_MoveTo(172, 136) ;
print("Difference: ", result) ;
```

Build and Upload the Mother Program

The mother program is uploaded to the display module just like any normal ViSi project. For instructions on how to build and upload a **Designer/ViSi** project to the target display, please refer to the section “**Build and Upload the Project**” of the application note

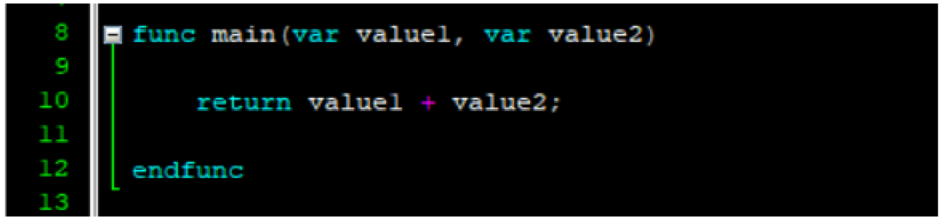
- ViSi Getting Started – First Project for Picaso and Diablo16
- ViSi Getting Started – First Project for Pixxi Displays

Child Program for Addition

This child program will reside in the uSD card. This program will be executed by the mother program to add two values. For this application note, a Designer project is used for creating this child program.

Child Program Code

The Main function parameters will receive the values from the mother program, the values are added before being sent back to the mother program through the **return** function.



```
func main(var value1, var value2)

    return value1 + value2;

endfunc
```

Build Child Program Files

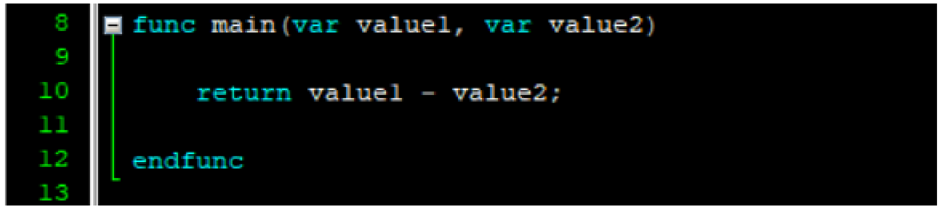
The child program is built by pressing the **Compile** button in the Home Tab without uploading it to the display module. This will compile the program and generate the compiled program code with the "**.4FN**" file extension.

Child Program for Subtraction

This child program will reside in the uSD card. This program will be executed by the mother program to subtract two values. For this application note, a Designer project is used for creating this child program.

Child Program Code

The Main function parameters will receive the values from the mother program, the values are subtracted before being sent back to the mother program through the **return** function.



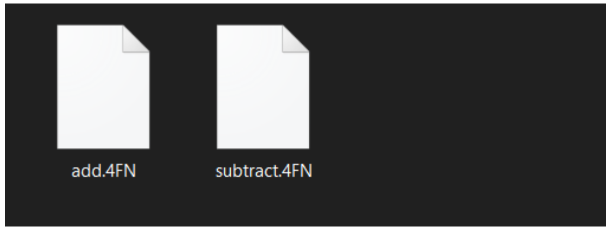
```
func main(var value1, var value2)

    return value1 - value2;

endfunc
```

Build Child Program Files

The child program is built by pressing the **Compile** button in the Home Tab without uploading it to the display module. This will compile the program and generate the compiled program code with the "**.4FN**" file extension.

Run the Program

Copy the child programs into the uSD Card before inserting it into the display modules.



With the mother program downloaded into the processor's main Flashbank, it will always run first. The child program will only run depending on which button is pressed.

Proprietary Information

Disclaimer of Warranties & Limitation of Liability