



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:
 - o Pixxi28/44
 - o Diablo16
 - o 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

Description 2

Content..... 2

Application Overview 3

Setup Procedure 3

Create a New Project..... 3

Mother Program.....4

Design the Project..... 4

Mother Program Code 4

Build and Upload the Mother Program 5

Child Program for Addition..... 6

Child Program Code..... 6

Build Child Program Files 6

Child Program for Subtraction..... 6

Child Program Code..... 6

Build Child Program Files 6

Run the Program..... 7

Proprietary Information 8

Disclaimer of Warranties & Limitation of Liability 8

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

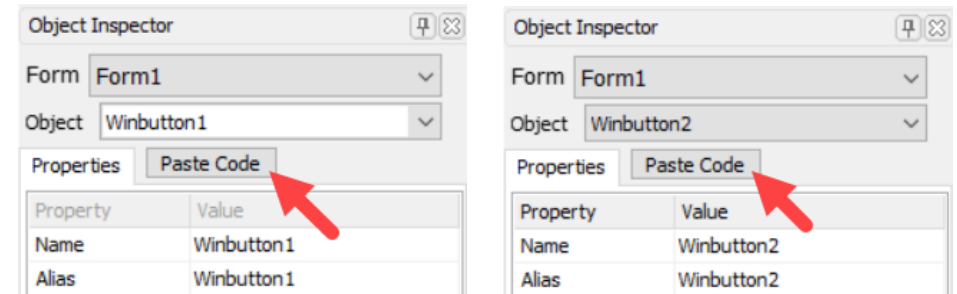
The mother program is the one residing in the processor's main Flashbank.

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.



```

36 // Winbutton1 1.0 generated 03/04/2020 8:35:51 am
37 img_ClearAttributes(hndl, iWinbutton1, I_TOUCH_DISABLE);
38 img_Show(hndl, iWinbutton1); // show button, only do this
39 img_SetWord(hndl, iWinbutton1, IMAGE_INDEX, state); // wh
40 img_Show(hndl, iWinbutton1);
41
42 // Winbutton2 1.0 generated 03/04/2020 8:35:51 am
43 img_ClearAttributes(hndl, iWinbutton2, I_TOUCH_DISABLE);
44 img_Show(hndl, iWinbutton2); // show button, only do this
45 img_SetWord(hndl, iWinbutton2, IMAGE_INDEX, state); // wh
46 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.

```

13 if (!(file_Mount()))
14     while(!(file_Mount()))
15         putstr("Drive not mounted...");
16         pause(200);
17         gfx_Cls();
18         pause(200);
19     wend
20 endif
21 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.

```

33     var values[3];    // Argument Array
34
35     values[0] := 2;    // Arg count
36     values[1] := 5;    // First Argument
37     values[2] := 3;    // Second Argument
38

```

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

```

39     repeat
40         touchStatus := touch_Get(TOUCH_STATUS);
41         widget := img_Touched(hndl,ALL);
42
43         switch (touchStatus)
44             case TOUCH_PRESSED:
45                 if (widget == iWinbutton1)
46                     img_SetWord(hndl, iWinbutton1, IMAGE_INDEX, 1); // where
47                     img_Show(hndl,iWinbutton1) ;
48                 else if (widget == iWinbutton2)
49                     img_SetWord(hndl, iWinbutton2, IMAGE_INDEX, 1); // where
50                     img_Show(hndl,iWinbutton2) ;
51                 endif
52             break;
53

```

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.

```

54         if (widget == iWinbutton1)
55             img_SetWord(hndl, iWinbutton1, IMAGE_INDEX, 1);
56             img_Show(hndl,iWinbutton1) ;
57             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.

```

58         gfx_MoveTo(172, 60) ;
59         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.

```

60         else if (widget == iWinbutton2)
61             img_SetWord(hndl, iWinbutton2, IMAGE_INDEX, 1);
62             img_Show(hndl,iWinbutton2) ;
63             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.

```

64         gfx_MoveTo(172, 136) ;
65         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.

```
8 func main(var value1, var value2)
9
10     return value1 + value2;
11
12 endfunc
13
```

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.

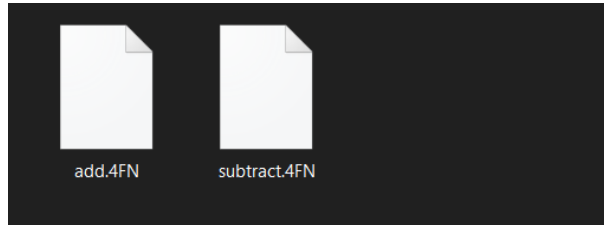
```
8 func main(var value1, var value2)
9
10     return value1 - value2;
11
12 endfunc
13
```

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

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.