APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

# Designer or ViSi Memory Management Picaso

DOCUMENT DATE: **13th April 2019**
DOCUMENT REVISION: **1.1**

## Description

This Application Note is intended to demonstrating and teaching the user how to test the interface between the 4D PICASO display modules with the MICROCHIP PIC microcontrollers. This application is intended for use in the 4D Workshop 4 – Designer environment. The tools needed includes the following;

Before getting started, the following are required:

- Any of the following 4D Picaso display modules:

  uLCD-24PTU uLCD-28PTU uVGA-III
  gen4-uLCD-24PT gen4-uLCD-28PT gen4-uLCD-32PT

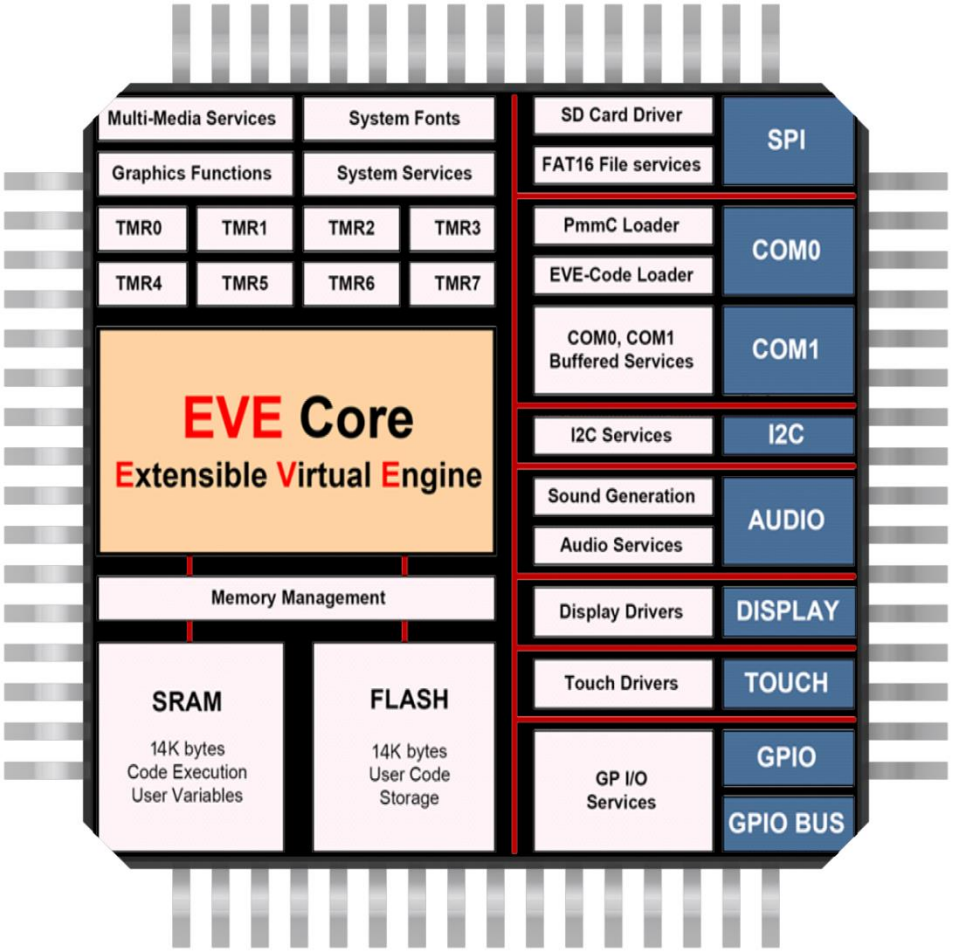
  and other superseded modules which support the Designer and/or ViSi environments.

- 4D Programming Cable / µUSB-PA5/µUSB-PA5-II
  for non-gen4 displays (uLCD-xxx)
- 4D Programming Cable & gen4-IB / gen4-PA / 4D-UPA,
  for gen-4 displays (gen4-uLCD-xxx)
- 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

When programming a Picaso display module with a 4DGL application, the user might come across a point where he needs more FLASH and/or RAM to execute his program.




A program that is set to be downloaded to “Flash”, actually executes from RAM if #MODE RUNFLASH is not added to the code. A 4DGL program that has #MODE RUNFLASH, executes from Flash, providing the entire RAM for the user variables, stack and other temporary data.

A program that executes from RAM has to allow for the Code execution space, user variable, stack and temporary data.

Also, code space within the FLASH might not be enough for some applications and the user might need to off load some part of the code to the uSD card in the form of child programs.

**Note:** Destination options in the Projects menu as Flash or RAM only define ‘where the code is downloaded to’. #MODE RUNFLASH defines ‘where the code is executed from’.

| Destination | #MODE RUNFLASH | Where Program Runs | Program available after power cycle |
|---|---|---|---|
| RAM | YES | RAM | No |
| RAM | NO | RAM | No |
| FLASH | YES | FLASH | Yes |
| FLASH | NO | RAM | Yes |

## Setup Procedure

For instructions on how to launch Workshop 4, how to open a **Designer** project, and how to change the target display, kindly refer to the section "**Setup Procedure**" of the application note

**Designer Getting Started - First Project**

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**

## Run the Program

### Setting up the RUNDEMO.4dg demo

Make sure the uSD card is FAT (aka Fat16) formatted.

Extract the files from **RUNDEMOS.zip** folder and you shall see these files/folders within,

**Demo Copy To Disk:**
There are a bunch of files in this folder that include 4XE, 4FN, GCI, DAT and WAV files that are needed for this demo. Copy all of these files to the uSD card.

**Demo Source Code:**
This folder has the source code for the 4XE and 4FN files that are in the "Demo Copy to Disk" folder. The source code could be used by the user for reference.

**RUNDEMOS.4DG**
It's the main program that gets compiled and downloaded to the module through the 4D Workshop4 IDE. This program resides inside the flash. Before downloading this program, make sure the destination is set to "Flash" from the Projects menu in the 4D Workshop4 IDE.

**RUNDEMOS_Readme.txt**
It's a readme file about the demo.

## Simulating the RUNDEMO.4dg demo

After copying the files to the uSD card as described above, insert the uSD card in to the display module. Connect the module with the PC through one of 4D's Programming adaptor, compile and download the RUNDEMOS.4dg file to the Display module.

Note: 4D's programming adaptors are, 4D Programming Cable, uUSB-CE5, uUSB-MB5 and uUSB-PA5.

After loading the program, you should see a menu on the screen; click one of the buttons to run the appropriate child program off the uSD card. All the programs in the menu are 4XE files. To check out a operation of a 4FN file, go to **Tetris** program and click Help or Reset or Exit button and you shall see a dialogue box which is basically a 4FN function file called.

## Conclusion

The demo program shows how large programs could made modular to fit the available memory resources.

Check out the following link that shows how this demo works on the display module.
https://www.youtube.com/watch?v=vWKWQcfNd3Y

## Setting up the callfn.4dg demo

This project is much smaller in size.

Make sure the uSD card is FAT (aka Fat16) formatted.

Extract the files from **memory management.zip** folder and you shall see these files/folders within,

**callfn.4dg:**
This file is the main program that calls the rest of the programs.

**callback.4dg:**
This file should be compiled to generate callback.4FN file which should be copied to the uSD card.

**testfn.4dg:**
This file should be compiled to generate testfn.4FN file which should be copied to the uSD card.

## Simulating the callfn.4dg demo

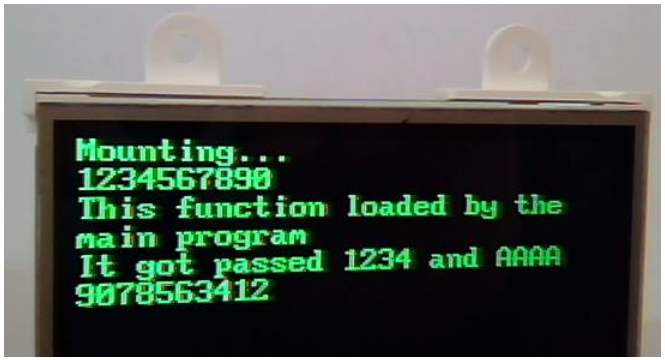
Compile callfn.4dg and load it to the display module. Compile testfn.4dg and callback.4dg files and copy the *.4FN files generate to a FAT (aka FAT16) formatted uSD card and insert it in to the module.

This is what the set of codes does, **callfn** loads **testfn** and **callback** in to the RAM returning back function pointers using **file_LoadFunction(..)**. **callfn** then prints the elements of the global array created in the main program. **testfn** is called (Note: rtn contains the function pointer to the **testfn**) with a pointer to the global array and the **callback** function pointer as arguments. Within **testfn**, global array elements which reside in the main program are swapped. Also, the **callback** function is called using the function pointer that was passed earlier. The handle is returned from **callback** to **testfn** and eventually to the main program printing the array elements again. Test the program to get a better understanding of this description.

**Note:** You can't call back to a function in main, but you can call a function loaded in main from any other loaded function.

## Conclusion

The demo program shows how functions could be loaded from the uSD card in to the RAM and then called from the main program or another child program. It also explains how global variable in the main program could be accessed from the child programs.




***This application note applies to the ViSi Environments as well. In case of ViSi Environment, the multimedia resources that are generated with the child program wold have to be copied to the uSD card manually which would have been copied to the uSD card via 4D Workshop IDE automatically***

## 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.