APPLICATION NOTES

4D SYSTEMS
TURNING TECHNOLOGY INTO ART

ViSi-Genie Program Destination

DOCUMENT DATE:	**27th 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 differentiates between the Run RAM, Run flash, and uSD options under the project tab or menu in the ViSi-Genie environment. Before getting started, the following are required:

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and other superseded modules which support the ViSi Genie environment.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 / 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

In the processor of a display module (Picaso and Diablo16 displays), there are two destination options for a Designer or a ViSi program – the flash memory and the RAM. The difference between these two destinations is discussed in detail in the application note General Downloading an Application Program to RAM or Flash Memory.

A ViSi-Genie program, on the other hand, is downloaded (or uploaded) to the flash memory of the processor always. There is no option to download a ViSi-Genie program to the RAM section. Furthermore, when a program is downloaded to the flash memory, the user can choose whether to make it run from where it resides (run from flash) or from RAM (run from RAM). In the latter, the program will be copied to RAM and be executed from there. These options are made available to the user though the Run RAM and Run Flash buttons of the Destination group of the Project tab or menu.

A program can also be copied to the uSD card as a child program. In this case, a parent program, downloaded to the flash memory of the processor, is needed to access the child program from the uSD card and load it into RAM. This option is made available through the uSD button of the Destination group under the Project tab or menu.

The difference between the three options – Run RAM, Run Flash, and uSD – are discussed in the subsequent sections of this application note.

To better understand the differences between the Run RAM, Run flash, and uSD options, the reader should first know the components of the output of a ViSi-Genie project when it is built and compiled. These are discussed in the section "**The ViSi-Genie Program and the Supporting Files**".

Finally, the section "**Build and Upload the Project**" discusses the RAM and flash memory requirements displayed inside the message area when a project is compiled. The relevance of these information to the three program destination options is then explained.

Setup Procedure

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

ViSi Genie Getting Started – First Project for Picaso Displays (for Picaso)
or
ViSi Genie Getting Started – First Project for Diablo16 Displays (for Diablo16).

Create a New Project

Create a New Project

For instructions on how to create a new ViSi-Genie project, please refer to the section "**Create a New Project**" of the application note

ViSi Genie Getting Started – First Project for Picaso Displays (for Picaso)
or
ViSi Genie Getting Started – First Project for Diablo16 Displays (for Diablo16)

Design the Project

The project used in this application note is the same project used in the application notes

ViSi Genie Getting Started – First Project for Picaso Displays (for Picaso)
and
ViSi Genie Getting Started – First Project for Diablo16 Displays (for Diablo16).

The ViSi-Genie Program and the Supporting Files

For this application note, it is essential for the reader to be able to differentiate between the different components of the output of a ViSi-Genie project when it is built and compiled. Generally, the output files can be classified into two – the program file and the supporting (or support) files.

The program is the one that actually gets downloaded (or uploaded) to the RAM or flash memory of the processor of the display module. The program contains the instructions that the processor will execute during runtime.

The supporting files are files needed for the whole project to run. These may be graphics files, audio files, font files, string files, and others. Each of these may or may not be present in the output of a project, depending on the nature of the project. For example, a simple project with only a meter object in it will only have supporting graphics files since it does not make use of audio, fonts, or strings. Workshop generates and copies the supporting files of a project to the uSD card. Supporting audio (WAV) files however are not generated, they are copied to the uSD card as they are. When the uSD card is unmounted from the PC and mounted to the display module and if the correct program is uploaded to the display module processor, the application will now run. Below are descriptions and filename extensions of some of the more common supporting files of a ViSi-Genie project.

Supporting Graphics Files

The supporting graphics files contain the actual graphics of the project. If your project has a meter object with a minimum value of 0 and a maximum value of 100 in it for instance, Workshop will generate 101 image frames for the meter object – one frame for each state of the meter. Workshop will then put these meter object images into the supporting graphics file. Supporting graphics files actually come in pairs – a file that contains all the actual images (the .GCI file) and another file (the .DAT file) that contains a list of all the objects inside the .GCI file. Thus, with the meter object example, the .GCI file will contain the meter object images (and images of other objects if any), and the .DAT file will contain a list of all the objects in the .GCI file.

Supporting Files (Audio and Font)

The Picaso and Diablo16 processors can play WAV files. In a music player project for instance, Workshop will copy the necessary WAV files to the uSD card such that when the program runs on the processor, it will access the uSD card and look for the WAV files.

ViSi-Genie also allows the use of different fonts. Workshop generates the supporting font files in the background and copies them to the uSD card. Supporting font files have the extensions ".d***nn***" and "**.g*nn***", where nn is the index. The first font used in the project would have the supporting filename extensions ".d***01***" and ".g***01***", the second font ".d***02***" and ".g***02***", and so on.

Supporting Strings Files

In ViSi-Genie it is possible to create a string object with a predefined text in it. For example, if you want your project to display several paragraphs containing instructions during runtime, you could add a strings object to the project and put in it the instructions text. Since predefined strings or text are static, Workshop stores them in supporting files with the filename extension "**.txf**". The program will then access these files from the uSD card when they are needed during runtime.

Handling of Supporting Files

Workshop handles and manages all the supporting files of a ViSi-Genie project in the background. Thus, all that a ViSi-Genie developer has to do is let Workshop know the correct drive to which the supporting files will be copied. There is no need for the ViSi-Genie developer to know the contents of the supporting files. However, if you are interested to find out more about them and how the display processors deal with them at a lower level, you may refer to the following application notes. Note that these application notes are not for the ViSi-Genie environment.

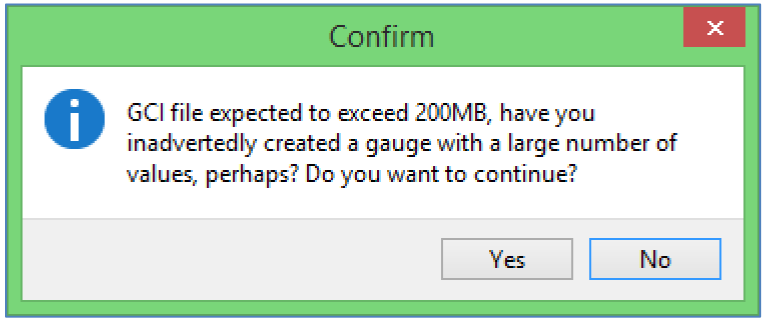
Designer Displaying Images from the uSD Card - GC FAT16
ViSi Displaying Third Party Fonts FAT16
Designer or ViSi Playing Sound

Troubleshooting Supporting Files

As mentioned previously, the ViSi-Genie developer does not need to worry about the supporting files. However, there are instances that the user may need to check the supporting files for troubleshooting purposes. For instance, a project would not run if the graphics files are not present on the uSD card. Hence, the user would just need to check and update the contents of the uSD card by clicking on the Build-Copy/Load button under the Home menu.

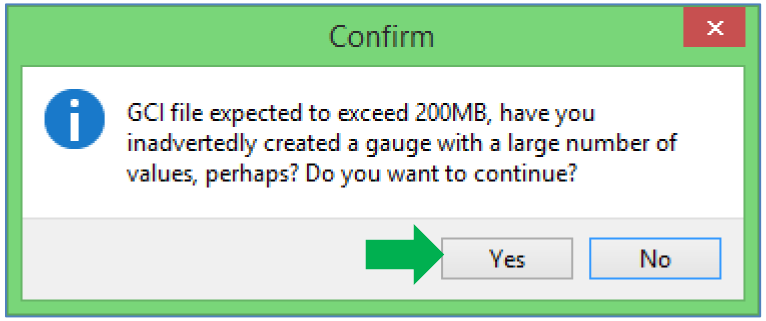
Size of Supporting Files

Now the total size of the supporting files may become quite significantly large depending on the project. If your project has ten forms with plenty of objects and uses lots of fonts, then expect that Workshop may take some time to build and generate the supporting graphics and font files and to copy them to the uSD card. By default, Workshop will warn you when the estimated supporting graphics file size exceeds 200 MB.


Confirm

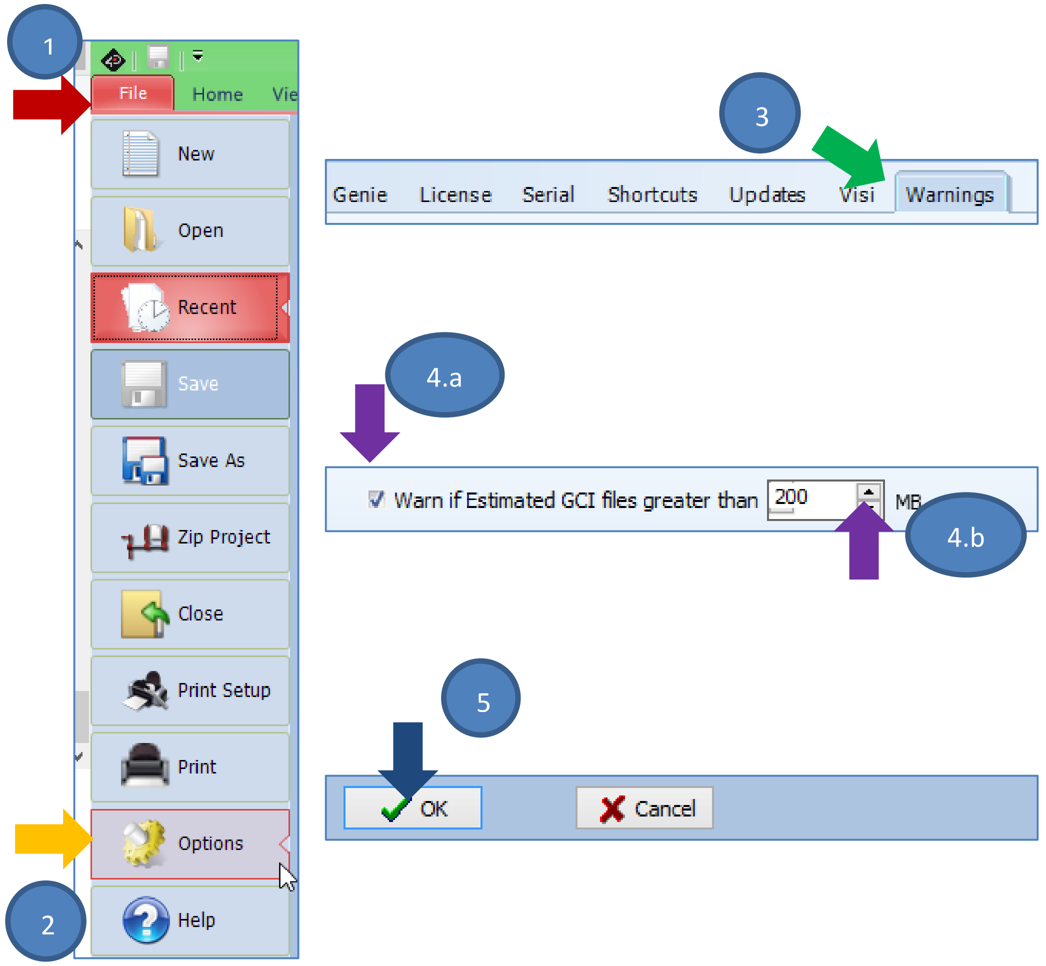
GCI file expected to exceed 200MB, have you inadvertedly created a gauge with a large number of values, perhaps? Do you want to continue?

A large gauge object with 5000 frames will most likely result to a very large GCI file. If you are sure that you need it or that if you really have a large project, click Yes to proceed.


Confirm

GCI file expected to exceed 200MB, have you inadvertedly created a gauge with a large number of values, perhaps? Do you want to continue?

To prevent Workshop from issuing this warning, uncheck the Warn-if-Estimated-GCI-file-size option as shown below. You can also increase the GCI file size warning threshold.



Warn if Estimated GCI files greater than 200 MB

Transfer of Supporting Files

Since the supporting files can become very large, Workshop always copies them directly to the uSD card. This means that Workshop will prompt you, the user, for the correct drive for the uSD card. This way, the transfer is much faster since transfer rate is around several megabytes per second when the uSD card is mounted to the PC.

It is also possible to transfer supporting files through the programming module (4D USB programming cable or uUSB-PA5) to a uSD card mounted on the display module. However, this would be very impractical for large files since the transfer rate is only in the range of several kilobytes per second. For more information on this, refer to the application note General Serial File Transfer from PC to Display Module uSD Card.

The Program

Again, the program is the collection of instructions to be executed by the processor. The program is downloaded to the display module processor through the programming module. Programs are relatively small in size (maximum of around 14KB for Picaso and 32KB for Diablo16) compared to the supporting files, so transferring them through the programming module should only take a moment.

The Child Program

The program can also be copied to the uSD card. In this case, another program would have to be downloaded to the processor. The program on the processor is the parent program. The program on the uSD card is the child program. When the parent program runs, it will access the uSD card, look for the child program, and load it to RAM. The child program therefore can be considered as a supporting file since it, like the other supporting files, is accessed from the uSD card by the parent program only when it is needed. The child program is a special supporting file however since it contains instructions like the parent program. Child programs have the filename extension "**.4xe**". The parent-child-programs scenario is discussed in more detail in the section "**The uSD Card as a Program Destination**".

Terminologies

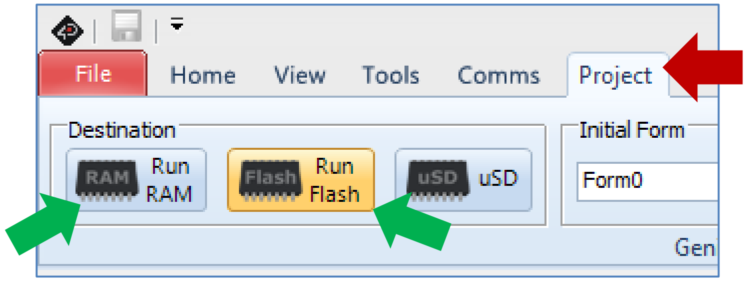
The main focus of this application note is the program and its relevance to the RAM and flash memory of the processor and the uSD card of the display module. However, again it is very important for the reader to know that the program is different from the supporting files, and that both are separate components of the whole project. From this point, we will use the terms "supporting files" and/or "graphics files" to refer to supporting files and the terms "program", "parent program", "child program", and/or "application" to refer to programs (i.e. those that contain actual instructions to be executed by the processor). Also, the terms "upload" and "download" are used interchangeably and mean the same thing in this application note.

Flash Memory vs. RAM

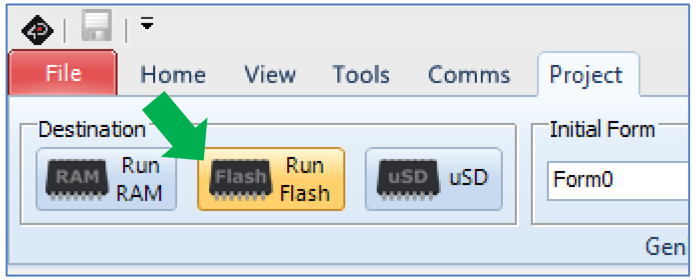
The Picaso processor has 14KB of flash memory and 14KB of RAM. The Diablo16 processor, on the other hand, has 32KB of RAM and six banks of flash memory – Bank 0 to Bank 5. Each bank of flash memory is 32 KB. A standard ViSi-Genie program is downloaded to the flash memory always. In Diablo16 processors, a standard ViSi-Genie program uses Bank 0 of the flash memory. The Goldelox processor has 10KB of flash memory and 510 bytes of RAM. The ViSi-Genie environment is not available for Goldelox display modules however, so Goldelox displays are not really relevant in this application note.

Run from Flash, Run from RAM, and uSD

After a ViSi-Genie program is downloaded to the flash memory section, the user has the option of making it run from flash or run from RAM. The Project menu shows these two options.

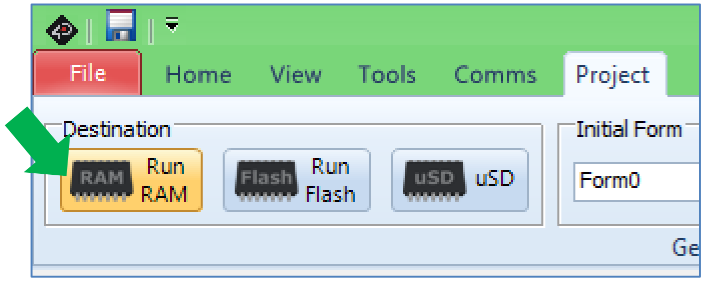


The Run-from-Flash Option



The run-from-flash or Run Flash option causes the program to be executed from the flash memory where it resides. This may cause the program to run slower, but more RAM space is available to the application.

The Run-from-RAM Option



The run-from-RAM or Run RAM option causes the program to be copied (from the flash memory where it resides) to the RAM section and executed from there. The program will run faster in this case, but less RAM space is available to the application since the application code itself occupies a part of the RAM space.

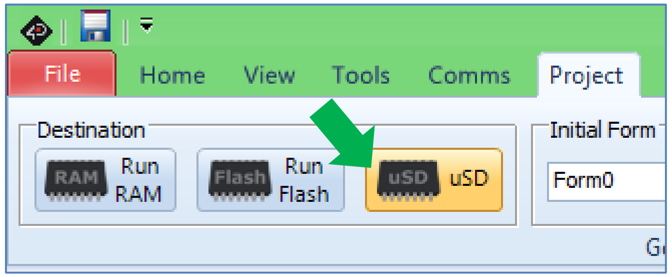
Run-from-Flash vs Run-from-RAM

Thus, it may be wise to choose Run-from-Flash or Run Flash when the program is large, such that all RAM space will be available for use by the application during runtime. Although the program may run slower in this case, more RAM space is available.

When the program is small, you can choose Run RAM. It will run faster, and, since the program code is small, there is still plenty of RAM space available. When a project is compiled, the message area at the bottom part of the Workshop 4 IDE shows the program code size and the initial approximate RAM size required by the program to run.

Note that for the Picaso processor, a program running from the flash memory is significantly slower compared to that which is running from RAM. For most uses however, this difference is not that noticeable. For the Diablo16 processor, a program running from RAM has about the same speed with that which is running from the flash memory.

The uSD Card as the Program Destination



The program can also be copied to the uSD card of the display module as a child program. Another program, the parent program, will then have to be downloaded to the flash memory of the processor. The parent program will execute from flash, access the uSD card, look for the child program, and load it to RAM. The child program will then execute.

Using the uSD card as the program destination may be useful in certain situations. For instance, there are projects wherein the display module is inside a permanent casing, such that only the uSD card slot is accessible and connecting/disconnecting a programming module is not possible. The solution would then be to download a parent program, using a programming module, to the flash memory of the processor prior to installing the display module inside the permanent casing. Updating the project would then be as simple as unmounting the uSD card from the display module, updating its contents (the child program and supporting files) using a PC, and mounting the uSD card back to the display module.

Build and Upload the Project

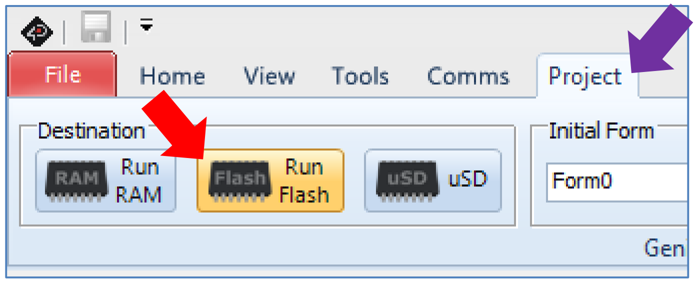
We will now put the theoretical discussions in the previous sections into practice. The project used in this application note is the same project used in the application notes

ViSi Genie Getting Started – First Project for Picaso Displays (for Picaso) and
ViSi Genie Getting Started – First Project for Diablo16 Displays (for Diablo16).

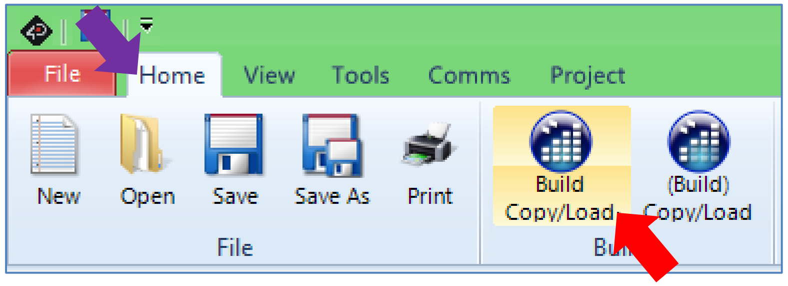
See the application notes above if you are not familiar with details such as how to change the target display module, how to save a project, proper connection of the programming modules, how to check if the display module is detected by Workshop, etc.

Run Flash

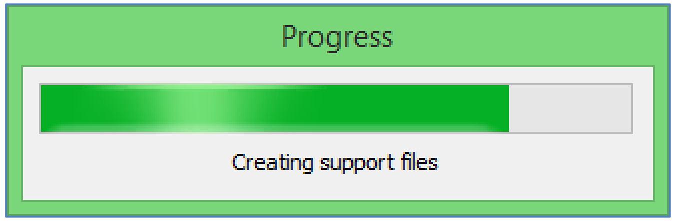
Step 1 – Choose Run Flash:



Step 2 – Click on the Build Copy/Load Button:

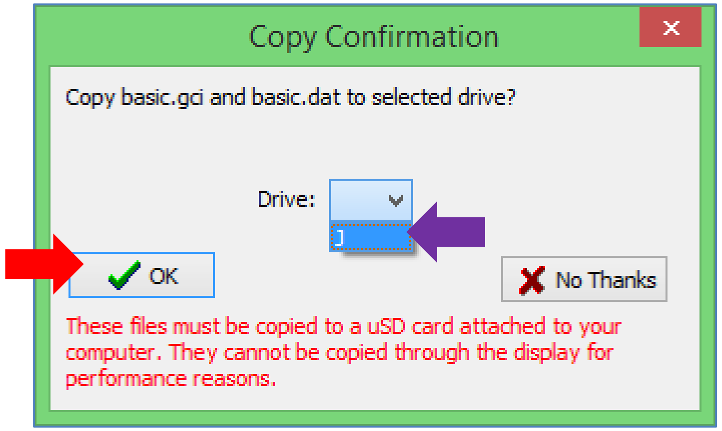


Workshop now generates the support or supporting files:



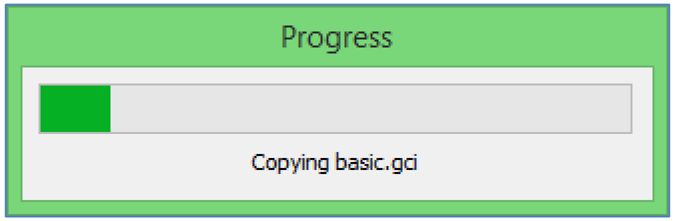
Step 3 – Select the Correct Drive:

Workshop asks for the correct drive for the uSD card. Here the uSD card is mounted as drive J. Click OK.

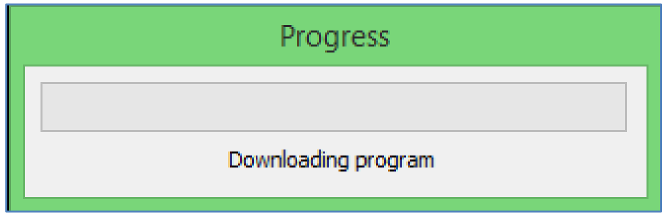


These files must be copied to a uSD card attached to your computer. They cannot be copied through the display for performance reasons.

Workshop now copies the supporting files to the uSD card.



Workshop now downloads the program to the flash memory of the display module processor.



The message area at the bottom part of the Workshop 4 IDE displays the errors, warnings, and notices for a project.

The Message Area for the uLCD-35DT (Diablo16)

```
0 errors
0 warnings
5 notices
No Errors, code size = 2977 bytes out of 32750 total
Approximate run RAM size = 1256 bytes out of 32768 total
Download to Flash successful.
```

	Flash	RAM
Occupied by the program code	2977 bytes	none
Required by the program to run (initial value)	none	1256 bytes
Total used	2977 bytes	1256 bytes
Total free	29773 bytes	31512 bytes
Total	32750 bytes	32768 bytes

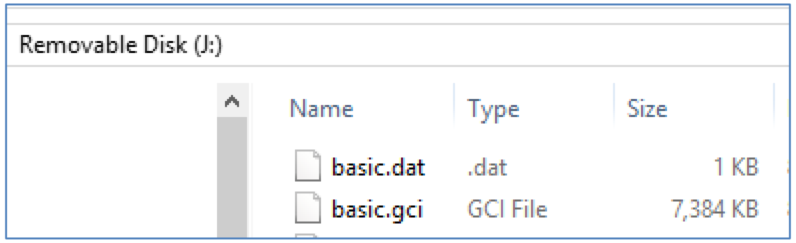
The Message Area for the uLCD-32PTU (Picaso)

```
Note: 0 Button Animation Timers in use.
0 errors
0 warnings
5 notices
No Errors, code size = 2977 bytes out of 14400 total
Approximate run RAM size = 1256 bytes out of 14400 total
Download to Flash successful.
```

	Flash	RAM
Occupied by the program code	2977 bytes	none
Required by the program to run (initial value)	none	1256 bytes
Total used	2977 bytes	1256 bytes
Total Free	11423 bytes	13144 bytes
Total	14400 bytes	14400 bytes

Step 4 – Check the Contents of the uSD Card:

Check the uSD card. It should contain the supporting graphics files.



Removable Disk (J:)

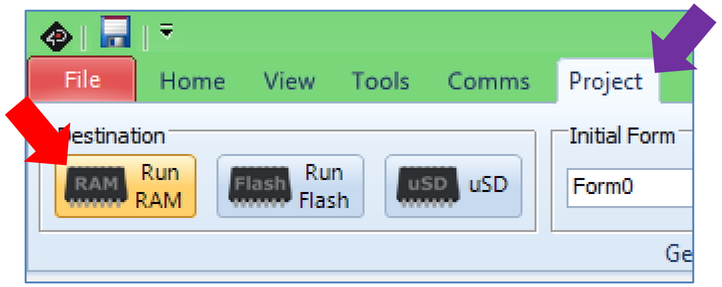
Name	Type	Size
basic.dat	.dat	1 KB
basic.gci	GCI File	7,384 KB

Step 5 – Run the Project:

Properly unmount the uSD card from the PC and mount it to the display module. The program should now run.

Run RAM

Step 1 – Choose Run RAM:



Repeat steps 2 to 5 of the previous section "**Run Flash**". Note however that the message area will now display a different information.

The Message Area for the uLCD-35DT (Diablo16)

```
Note: 0 Button Animation Timers in use.
0 errors
0 warnings
5 notices
No Errors, code size = 2977 bytes out of 32750 total
Approximate run RAM size = 1256 bytes out of 32768
total
Program will run from ram so total initial RAM size =
4233 bytes out of 32768 total
Download to Flash successful.
```

	Flash	RAM
Occupied by the program code	2977 bytes	2977 bytes
Required by the program to run (initial value)	none	1256 bytes
Total used	2977 bytes	4233 bytes
Total free	29773 bytes	28535 bytes
Total	32750 bytes	32768 bytes

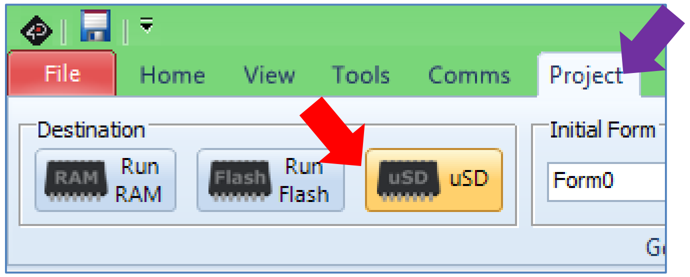
The Message Area for the uLCD-32PTU (Picaso)

```
Note: 0 Button Animation Timers in use.
0 errors
0 warnings
5 notices
No Errors, code size = 2977 bytes out of 14400 total
Approximate run RAM size = 1256 bytes out of 14400 total
Program will run from ram so total initial
               RAM size = 4233 bytes out of 14400 total
Download to Flash successful.
```

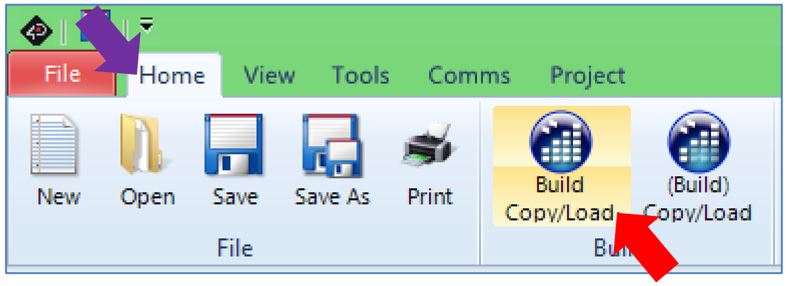
	Flash	RAM
Occupied by the program code	2977 bytes	2977 bytes
Required by the program to run (initial value)	none	1256 bytes
Total used	2977 bytes	4233 bytes
Total free	11423 bytes	10167 bytes
Total	14400 bytes	14400 bytes

uSD Card

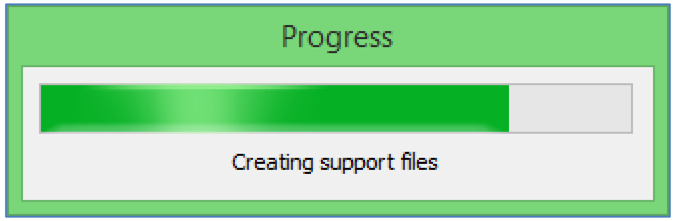
Step 1 – Choose uSD:



Step 2 – Copy the Child Program to the uSD Card:

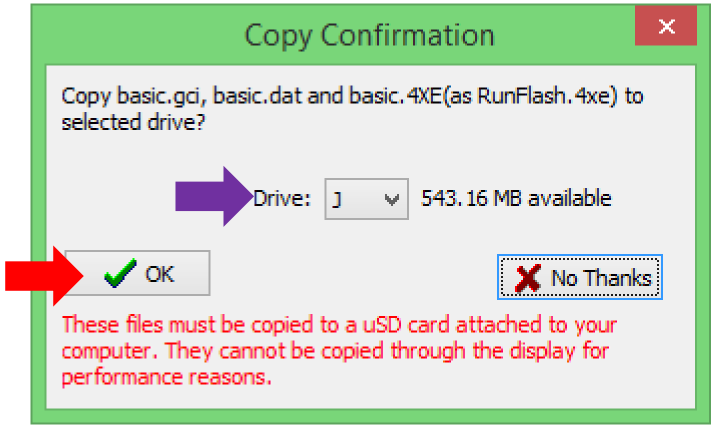


Workshop now generates the support files including the child program.



Step 3 – Select the Correct Drive:

Workshop asks for the correct drive for the uSD card. Here the uSD card is mounted as drive J. Click OK.



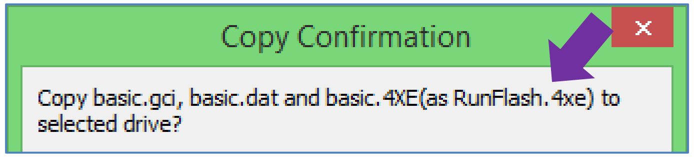
Copy Confirmation

Copy basic.gci, basic.dat and basic.4XE(as RunFlash.4xe) to selected drive?

Drive: J 543.16 MB available

These files must be copied to a uSD card attached to your computer. They cannot be copied through the display for performance reasons.

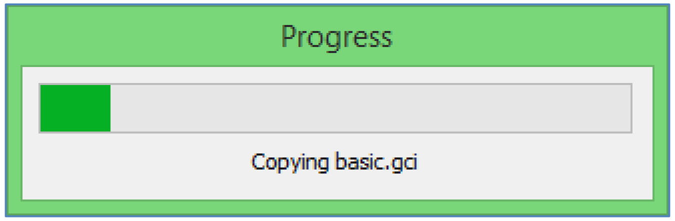
Note that the files include not only the supporting graphics files but also the child program (named as **RunFlash.4xe**).



Copy Confirmation

Copy basic.gci, basic.dat and basic.4XE(as RunFlash.4xe) to selected drive?

Workshop now copies the supporting files to the uSD card.



Progress

Copying basic.gci

At this point the message area will display the information as shown in the following screenshot images.

The Message Area for the uLCD-35DT (Diablo16), Child Program

```
0 errors
0 warnings
5 notices
No Errors, code size = 2977 bytes out of 32750
total1
Approximate run RAM size = 1256 bytes out of
32768 total
Program will run from ram so total initial RAM
size = 4233 bytes out of 32768 total
```

	Flash	RAM
Occupied by the program code	none[1]	2977 bytes
Required by the program to run (initial value)	none	1256 bytes
Total used	0 bytes	4233 bytes
Total free	32750 bytes	28535 bytes
Total	32750 bytes	32768 bytes

The Message Area for the uLCD-32PTU (Picaso), Child Program

```
0 errors
0 warnings
5 notices
No Errors, code size = 2977 bytes out of 14400
total[1]
Approximate run RAM size = 1256 bytes out of
14400 total
Program will run from ram so total initial RAM
size = 4233 bytes out of 14400 total
```

	Flash	RAM
Occupied by the program code	none[1]	2977 bytes
Required by the program to run (initial value)	none	1256 bytes
Total used	0 bytes	4233 bytes
Total free	14400 bytes	10167 bytes
Total	14400 bytes	14400 bytes

Note 1: For the Diablo16 processor, the line

```
No Errors, code size = 2977 bytes out of 32750
total
```

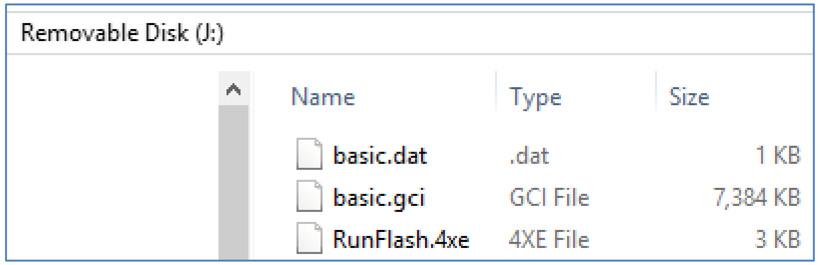
means that the child program would occupy 2977 bytes out of the total 32750 bytes of available flash memory, if it were placed in the flash memory. For the Picaso processor, the line

```
No Errors, code size = 2977 bytes out of 14400
total
```

means that the child program would occupy 2977 bytes out of the total 14400 bytes of available flash memory, if it were placed in the flash memory. The child program is copied to the uSD card however, so it does not actually need a space in the flash memory.

Step 4 – Check the Contents of the uSD Card:

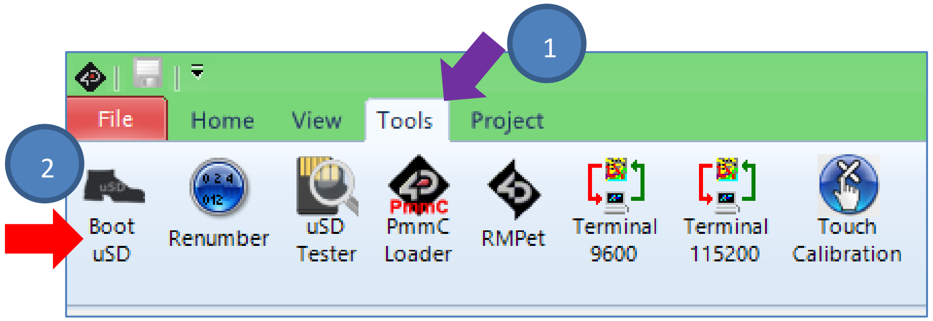
Check the uSD card. It should contain the supporting graphics files and the child program.


Removable Disk (J:)

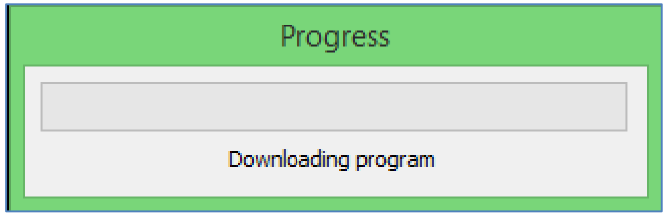
Name	Type	Size
basic.dat	.dat	1 KB
basic.gci	GCI File	7,384 KB
RunFlash.4xe	4XE File	3 KB

Step 5 – Download the Parent Program to Flash Memory:

As mentioned previously, we will need to download a parent program to the flash memory of the display. The parent program will run from RAM and there it will access the uSD card, look for the child program, and load it into RAM. To download the parent program to the flash memory of the display module processor, go to the Tools menu and click on the Boot uSD icon.



Workshop now downloads the parent program to the flash memory of the display module processor.



Workshop quickly displays the message area information for the parent program before it reverts back to that for the child program. If you are fast enough to read it, you will find the information shown below.

The Message Area for the uLCD-35DT (Diablo16), Parent Program

```
Note: 0 Button Animation Timers in use.
0 errors
0 warnings
0 notices
No Errors, code size = 204 bytes out of 32750 total
Approximate run RAM size = 200 bytes out of 32768
total
Download to Flash successful.
```

	Flash	RAM
Occupied by the program code	204 bytes	none
Required by the program to run (initial value)	none	200 bytes
Total used	204 bytes	200 bytes
Total free	32546 bytes	32568 bytes
Total	32750 bytes	32768 bytes

The Message Area for the uLCD-32PTU (Picaso), Parent Program

```
Note: 0 Button Animation Timers in use.
0 errors
0 warnings
0 notices
No Errors, code size = 204 bytes out of 14400 total
Approximate run RAM size = 200 bytes out of 14400
total
Download to Flash successful.
```

	Flash	RAM
Occupied by the program code	204 bytes	none
Required by the program to run (initial value)	none	200 bytes
Total used	204 bytes	200 bytes
Total Free	14196 bytes	14200 bytes
Total	14400 bytes	14400 bytes

Step 6 – Run the Project:

Properly unmount the uSD card from the PC and mount it to the display module. The program should now run.

Total RAM and Flash Requirements

To get the total RAM and flash requirements of the project, add the individual requirements of the child and parent programs.

	Flash	RAM
Occupied by the parent program code	204 bytes	none
Required by the parent program to run (initial value)	none	200 bytes
Occupied by the child program code	None (since it is on the uSD card)	2977 bytes
Required by the child program to run (initial value)	none	1256 bytes
Total used/required	**204 bytes**	**4433 bytes**

Updating or Changing the Project

Note that since Workshop saves any child program as "**RunFlash.4xe**", you can easily replace the contents of the uSD card with an updated version or an entirely new project. When the uSD card is mounted back to the display module, the parent program residing on the flash memory should be able to locate and load the new child program.

Files for the Parent Program

The files associated to the parent program can be found inside the folder "**C:\Users\YourUserID\AppData\Local\Temp**". The files will have the name "**temp4D.***". Workshop uses this folder and filename when downloading programs, including parent programs, to the display module processor.

Proprietary Information

Disclaimer of Warranties & Limitation of Liability