



Serial uSD Card Access FAT16

DOCUMENT DATE: 20th May 2019
DOCUMENT REVISION: 1.1



Description

This application note explains how to access the uSD card mounted on a Picaso or a Diablo16 display module in the Serial environment. Here is a list of items required to replicate this application:

- Any of the following Picaso display modules:

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

and other superseded display modules which support the ViSi environment

- The target module can also be a Diablo16 display

gen4-uLCD-24D	gen4-uLCD-28D	gen4-uLCD-32D
Series	Series	Series
gen4-uLCD-35D	gen4-uLCD-43D	gen4-uLCD-50D
Series	Series	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 / \$\mu\$ USB-PA5/ \$\mu\$ USB-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 \(\$\mu\$ SD\)](#) 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

Description	2
Content	3
Application Overview	3
Setup Procedures	4
Control the Display	4
<i>Clear the Screen</i>	4
<i>Mount the uSD Card</i>	5
<i>Create a File</i>	5
<i>Write a String to the File</i>	7
<i>Close the File</i>	7
<i>Unmount the uSD Card</i>	8
<i>Edit the Contents of the Text File</i>	9
<i>Mount the uSD Card</i>	10
<i>Open a File for Reading</i>	10
<i>Read a String from the File</i>	11
Proprietary Information	12
Disclaimer of Warranties & Limitation of Liability	12

Application Overview

In this application note, the basic commands available for accessing a FAT16-formatted uSD card are discussed. The sample application uses the File Mount, File Open, File Write, File Read, and File Close commands to show the basics of writing to and reading from a uSD card.

This application note uses a Picaso display module as the sample model, but the procedures are also applicable to Diablo16 display modules. The main difference between a Picaso display and a Diablo16 display with respect to the Serial environment is the bytes used for the serial commands. The serial command bytes for clearing the screen and sending a string, for instance, are different for each graphics processor.

	Picaso	Diablo16
Clear screen	0xFFCD	0xFF82
Put string	0x0018	0x0018

A user with a Diablo16 display module must have to consider this fact when using this application note as a reference.

Setup Procedures

The display must be configured as a slave device first before it can be controlled by a host. For instructions on how to launch Workshop 4, how to connect the display module to the PC, and how to configure the display as a slave device, kindly refer to the section “**Setup Procedure**” of any of the application notes below. Choose according to your display module’s processor.

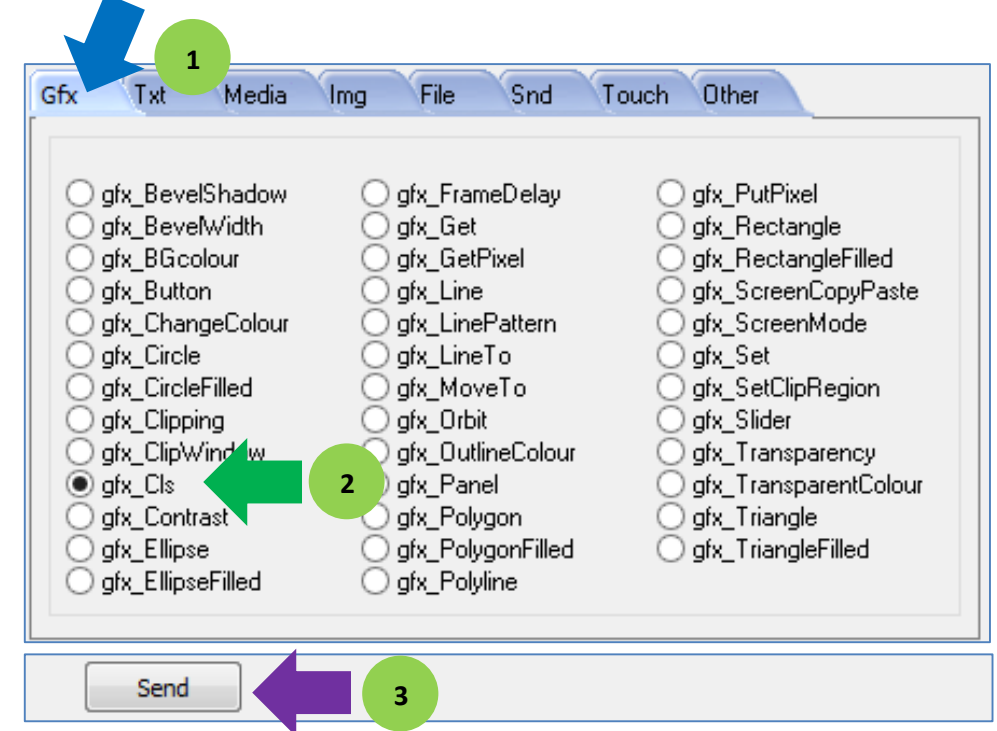
[Serial Picaso Getting Started - The SPE Application](#)

[Serial Diablo16 Getting Started - The SPE Application](#)

These application notes also introduce the user to the Serial Protocol through the use of the Serial Commander.

Control the Display

Clear the Screen



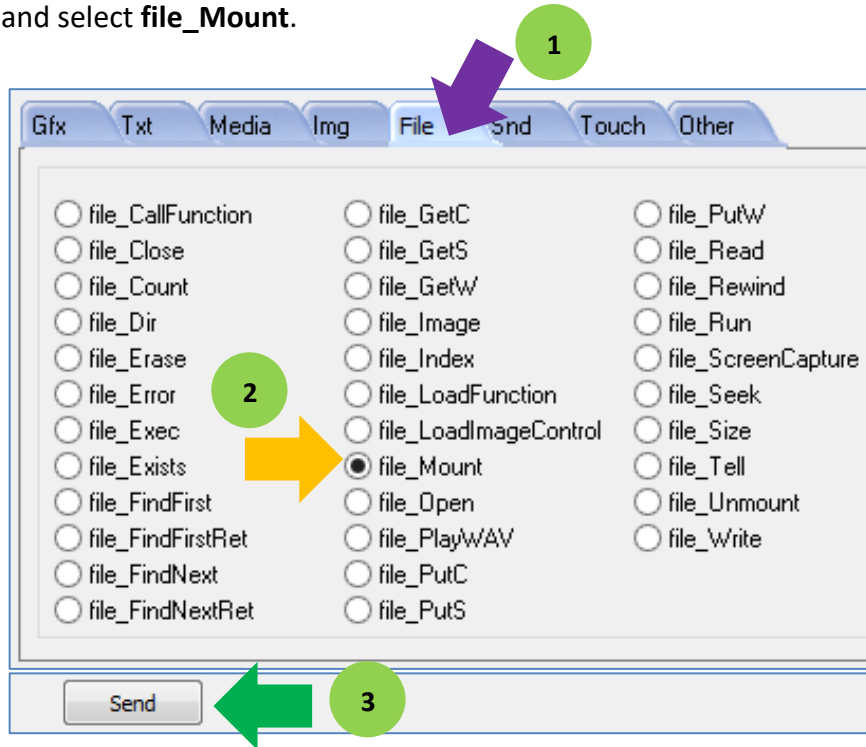
The bytes sent and received are:

```
gfx_Cls[FFCD ] 0.020 (ACK)
```

The screen should now be cleared.

Mount the uSD Card

Mount a FAT16-formatted uSD card on to the display module. Go to the **File** tab and select **file_Mount**.



The bytes sent and received are:

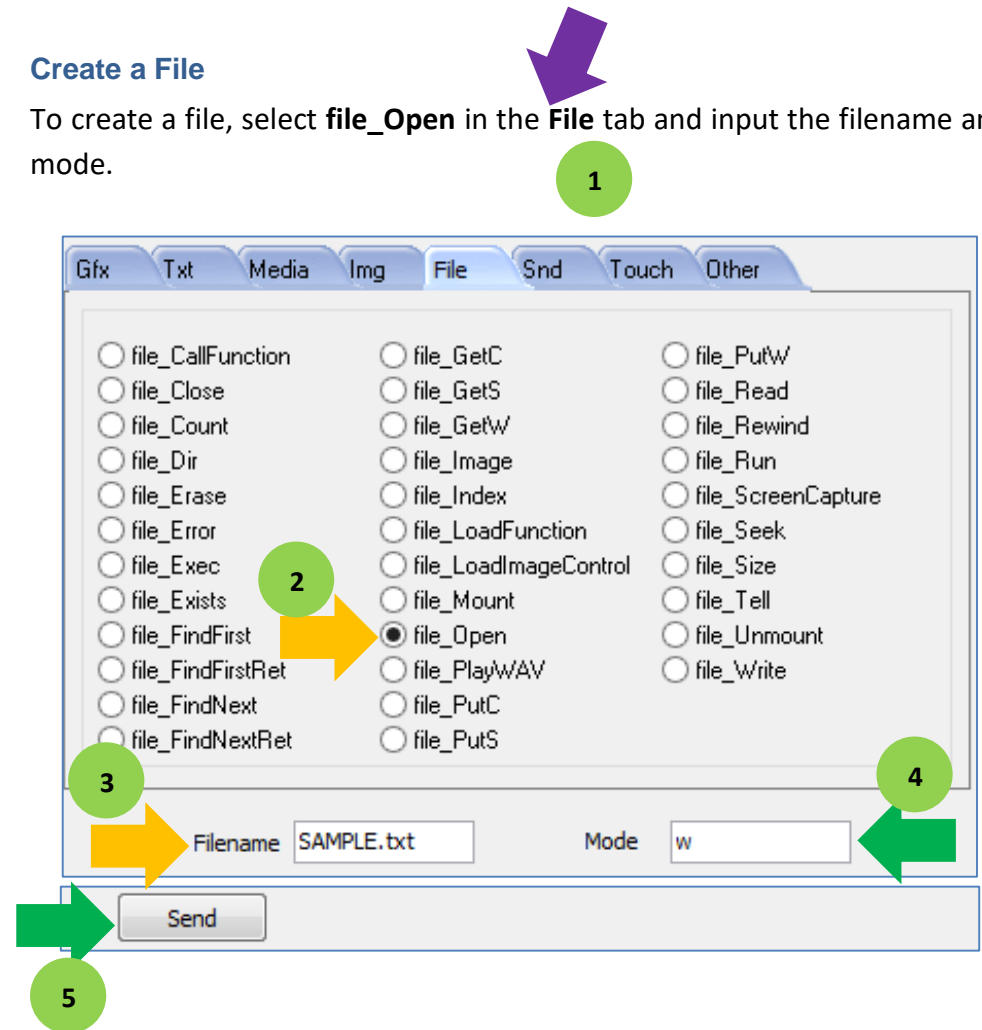
```
file_Mount[FF03 ] 0.110 (ACK 5447 0x1547)
```

This command starts up the FAT16 disk file services and allocates a small 20 byte control block for subsequent use. The **file_Mount** command must be sent before any other FAT16 file related functions can be used. The response will be an **ACK (0x06)** followed by a non-zero number (**5447 decimal or**

0x1547 hexadecimal in this example) if the command is successful, or zero (**0x00**) if unsuccessful.

Create a File

To create a file, select **file_Open** in the **File** tab and input the filename and mode.



The bytes sent and received are:

```
file_Open[000A "SAMPLE.txt" 'w'] 0.039 (ACK 5162 0x142A)
```

The returned value “5162” or “0x142A” in hexadecimal is the 16-bit file handle. **This value could be different in your case.**

The filename parameter is the name of the file to be opened (passed as a string). The mode parameter can be any of the following:

FILE_READ or 'r'
FILE_WRITE or 'w'
FILE_APPEND or 'a'

Note: If a file is opened for write mode 'w', and the file already exists, the operation will fail.

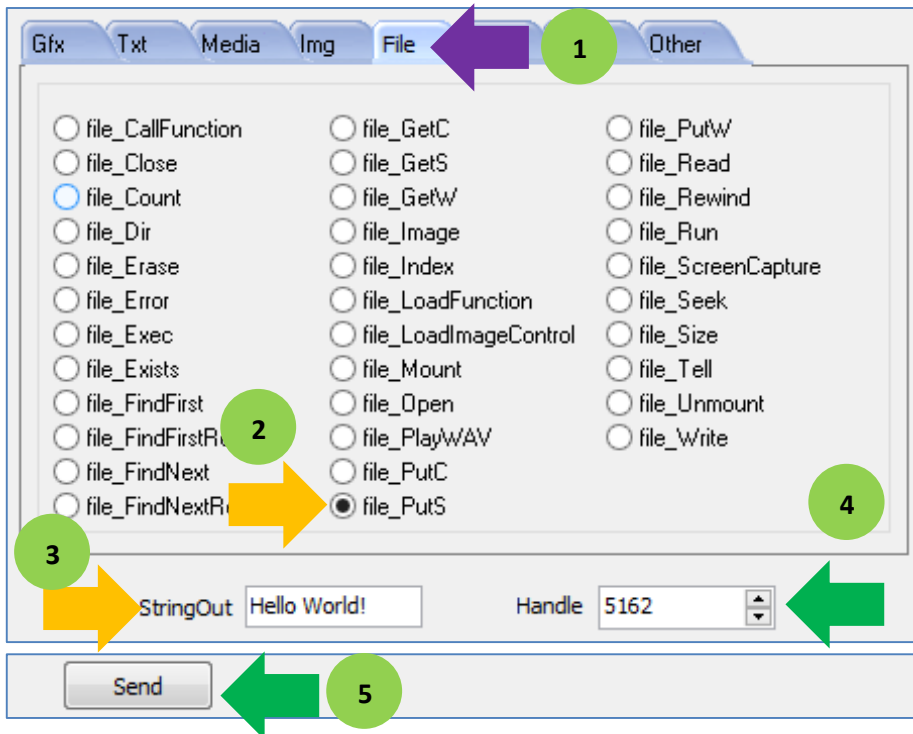
This command creates a new text file named “SAMPLE.txt” in the uSD card. The mode is 'w', which means that the file is opened for writing. **The display replies with the file handle if the file exists or if opened successfully. The file 'handle' that is created is now used as reference for further file commands such as “File Close”, etc.**

For File Write and File Append modes ('w' and 'a') the file is created if it does not exist. If the file is opened for append and it already exists, the file pointer is set to the end of the file ready for appending, else the file pointer will be set to the start of the newly created file.

If the file was opened successfully, the internal error number is set to 0 (i.e. no errors) and can be read with the “File Error” command. For File Read mode ('r') the file must exist else a null handle (0x00, 0x00) is returned and the 'file not found' error number is set which can be read with the “File Error” command.

Write a String to the File

To write a string to the file, select **file_PutS** in the **File** tab and input the string and file handle. Note that the file handle refers to "SAMPLE.txt", which was opened previously.



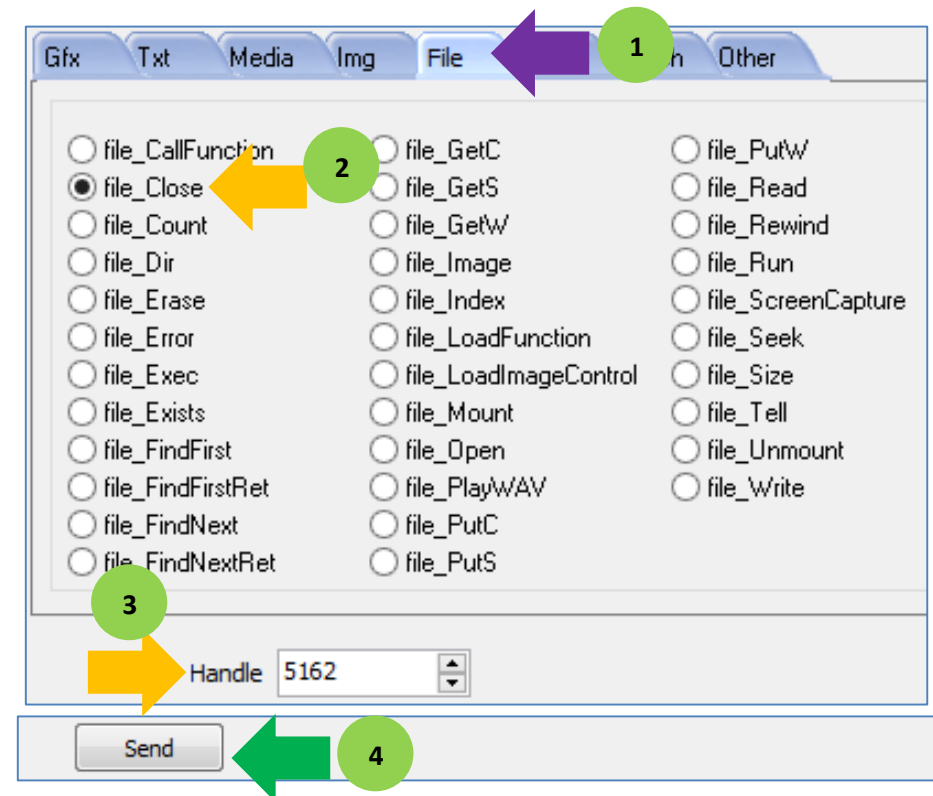
The bytes sent and received are:

```
file_PutS[0020 "Hello World!" 142A] 0.022 (ACK 12 0x000C)
```

This function writes a null terminated string to the file, at the position indicated by the associated file-position pointer and advances the pointer appropriately. The file must be previously opened with 'w' (write) or 'a' (append) modes. The display replies with an **ACK** and the **number of characters** written (excluding the null terminator).

Close the File

To close the file, select **file_Close** in the **File** tab and input the file handle. Note that the file handle refers to "SAMPLE.txt".



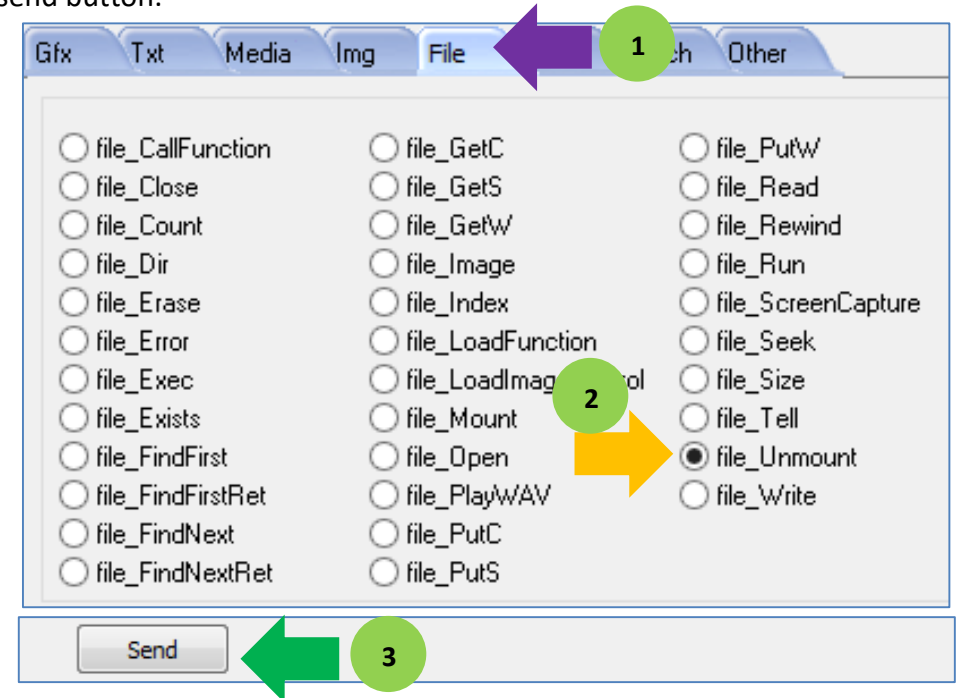
The bytes sent and received are:

```
file_Close[FF18 142A] 0.015 (ACK 1 0x0001)
```

The **file_Close** command will close a previously opened file. The display replies with an **ACK** and the **status**. Value of status can either be “1”, which means that the file was successfully **closed**, or “0”, which means that the file was **not closed**.

Unmount the uSD Card

To unmount the uSD card, select **file_Unmount** in the **File** tab and click on the send button.



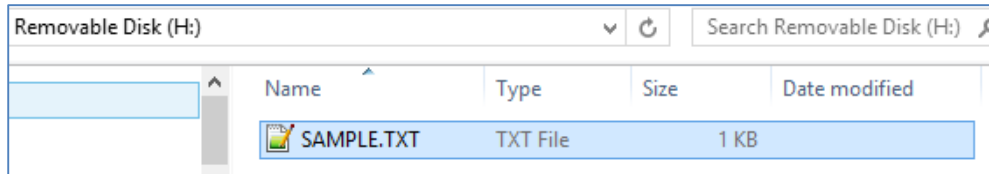
The bytes sent and received are:

```
file_Unmount[FF02 ] 0.004 (ACK)
```

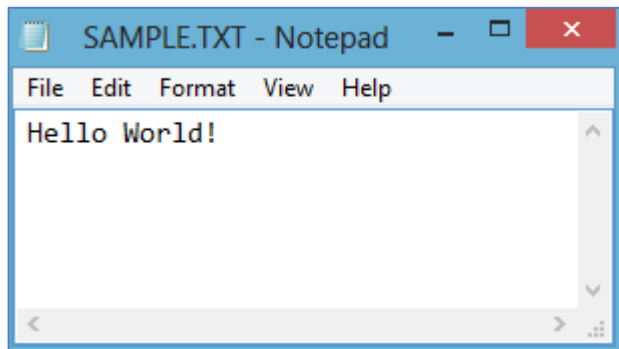
The **file_Unmount** command releases any buffers for FAT16 and unmounts the Disk File System. The display replies with an ACK.

Edit the Contents of the Text File

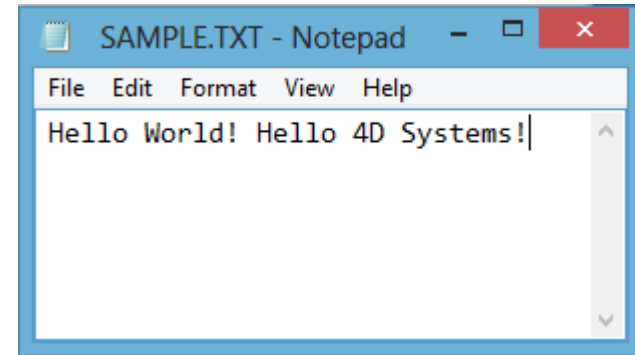
In the file explorer in Windows, the file "SAMPLE.txt" should be present in the uSD card.



The file should also contain the string "Hello World!".



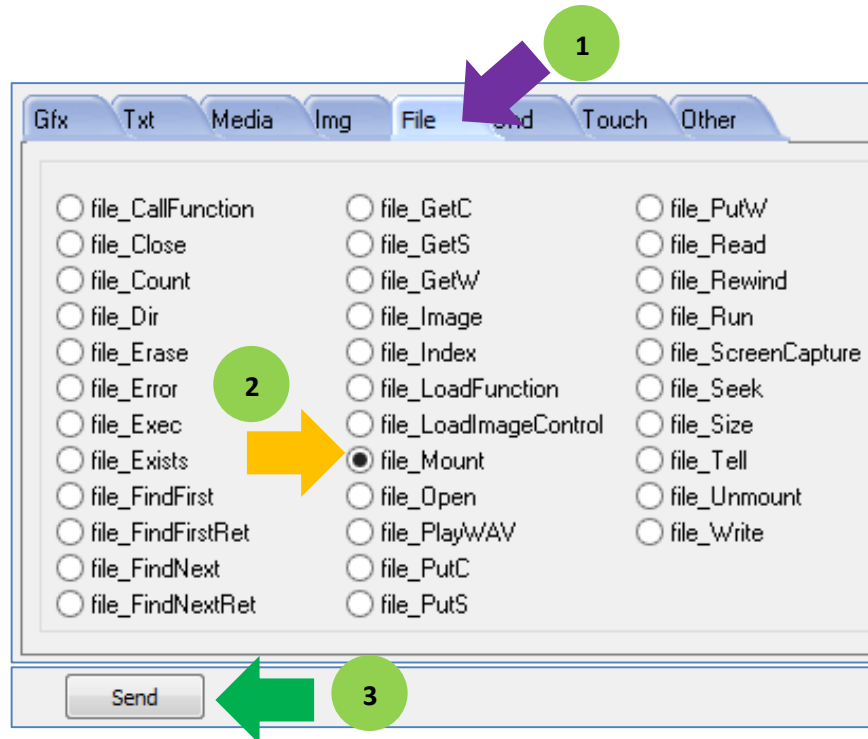
Append a string to the file. Here, the string "Hello 4D Systems!" is added.



Save the file, unmount the uSD card from the PC, and mount it back to the display module.

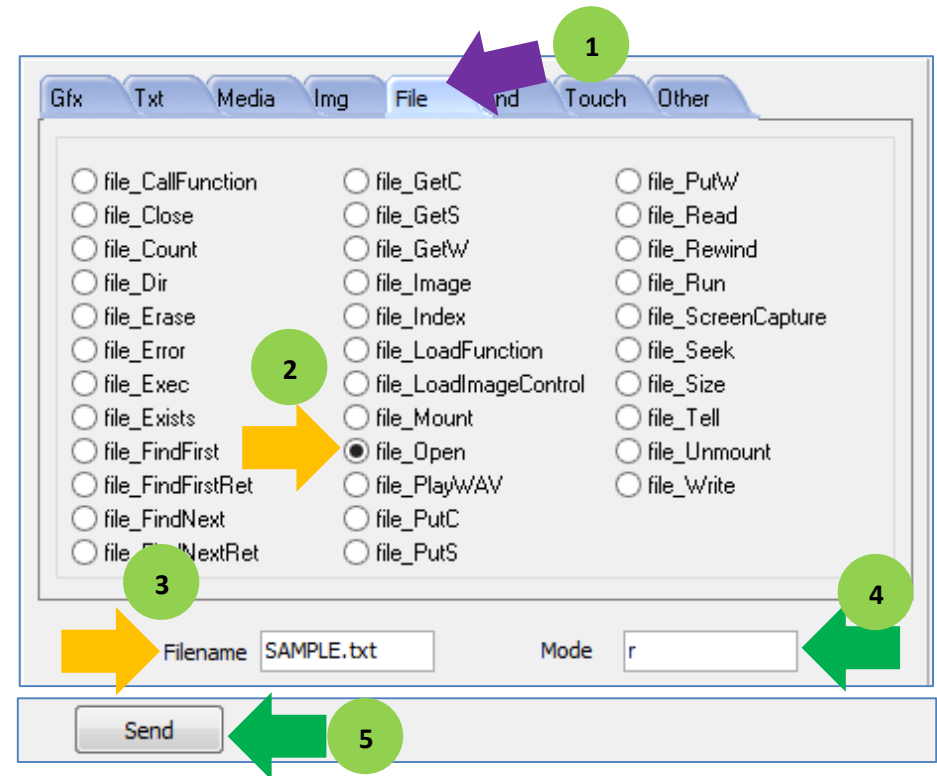
Mount the uSD Card

With the uSD card mounted onto the display module, it is necessary to perform a **file_Mount** first before sending any FAT-16 command, as was demonstrated earlier.



Open a File for Reading

To open an existing file for reading, select **file_Open** in the **File** tab and input the filename and mode.



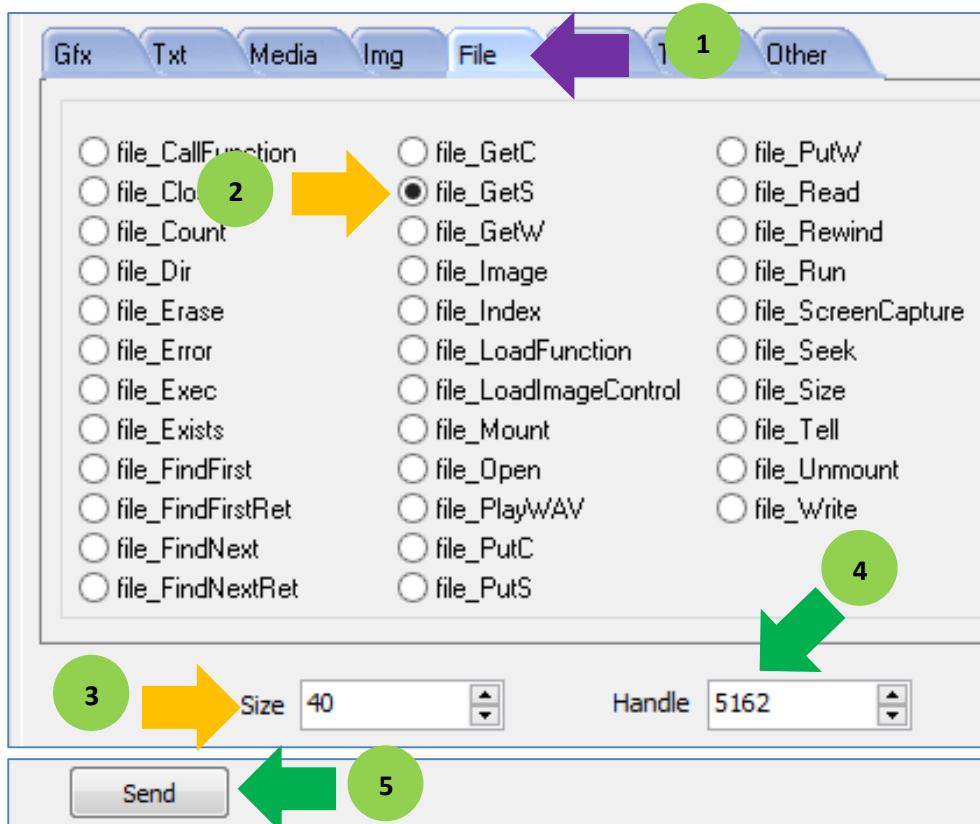
The bytes sent and received are:

```
file_Open[000A "SAMPLE.txt" r] 0.022 (ACK 5162 0x142A)
```

The returned value "5162" or "0x142A" in hexadecimal is the 16-bit file handle. **This value could be different in your case.** As discussed earlier, the mode 'r' means that the file is opened for reading.

Read a String from the File

To read a string from a file, select **file_GetS** in the **File** tab. Input the size (maximum number of bytes to be read) and handle. We are expecting 30 characters from the file "SAMPLE.txt".



The sent and received bytes are:

```
file_GetS[0007 0028 142A] 0.042 (ACK 30 0x001E, "Hello World! Hello 4D Systems!")
```

This function reads a line of text from a file at the current file position indicated by the associated file-position pointer (set by the "File Seek" or "File Index" commands) and advances the pointer appropriately. Characters are read until either a **newline** or an **EOF** is received or until the specified **maximum "size"** is reached. In all cases, the string is null terminated. The file must be previously opened with 'r' (read) mode.

The display replies with an **ACK**, the **number of characters** in the string, and the **string** itself.

Note: Before unmounting the uSD card from the display, it is necessary to close open files and perform a file_Unmount, as was demonstrated previously.

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.