Talaria TWO EVK has a 2MB SPI flash for storing user data. This application note describes using the APIs to show case the filesystem read, write, and remove functionalities on the Talaria TWO EVK.

# Sample Code Walkthrough

## Using\_filesystem

The using\_filesystem application illustrates performing the following basic filesystem operations. To mount the file system, utils\_mount\_rootfs() is called.

|  |
| --- |
| /\* Mount FS \*/  os\_printf("Mounting file system\n");  ret\_code = utils\_mount\_rootfs();  if(ret\_code != 0){  os\_printf("Failed to mount file system\n");  return 0; }  else  { os\_printf("File system mounted\n"); } |

## Read and Write

To read and write, the application uses the C library APIs - fputs() and fgets().

In this example, a new file named /data/sample\_text.txt is written in the DATA partition of the filesystem.

|  |
| --- |
| /\* Write file \*/  sample\_file = fopen(configfile, "w");  fputs("This is testing for file system...\n", sample\_file);  fclose(sample\_file);  os\_printf("Write Done.\n\n");  /\* Read file \*/  sample\_file = fopen(configfile, "r");  fgets(str, 50, sample\_file);  os\_printf("file data = %s \n", str);  fclose(sample\_file);  os\_printf("Read Done.\n\n"); |

## Fetching the File into a Buffer

The utils\_file\_get() API is used to get the content of a file into a buffer.

|  |
| --- |
| char \*file\_buf = utils\_file\_get(configfile, &size);  os\_printf("File Data = %s \n", file\_buf); |

# Flashing & Running the Application

In Talaria TWO module, there are two ways of writing into the file system:

1. Manually for factory setup
2. Programmatically

## Generate & Flash a Filesystem Manually for Factory Setup

To create a file system manually and write the file image using the scripts available in SDK, execute the following steps:

1. Generate an image file using the mklittlefs tool.

Go to *freertos\_sdk\_x.y/examples/using\_filesystem* folder.

**Note**: x and y refer to the SDK release version. For example: freertos\_sdk\_2.5/doc.

|  |
| --- |
| ./mklittlefs -s 0x40000 -c ./data ./flash.img |

**Note**: The flash.img will contain the file(s) present in the data folder of current directory where this binary (mklittlefs) is executed.

**Note**: Provide execute permission to mklittlefs (right click on: *Properties → Permissions → Allow executing file as a program*)

1. From the SDK folder, execute the remaining commands:

Ensure to have a partition table containing allocation for DATA as a prerequisite. To manually flash an image file to a specific location, execute the following steps:

* 1. Load gordon.elf onto the Talaria TWO module.

|  |
| --- |
| ./script/boot.py --device /dev/ttyUSB2 --reset=evk42\_bl ./apps/gordon.elf |



Figure 1: Loading gordon.elf

1. Flash the image file to the desired location on the filesystem.

|  |
| --- |
| ./script/flash.py --device /dev/ttyUSB2 part\_write DATA ./examples/using\_filesystem/flash.img |

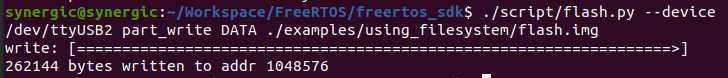


Figure 2: Flashing the image

1. Fetch the file(s) which was flashed.

|  |
| --- |
| ./script/storage.py ls /data/ |

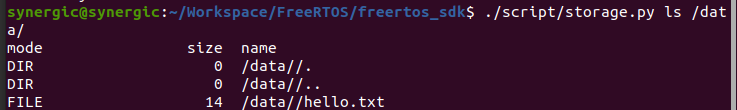


Figure 3: Fetching files from data partition

## Writing into the Filesystem Programmatically

### Running the Application

Program using\_fs.elf (*freertos\_sdk\_x.y\examples\using\_filesystem\bin*) using the Download tool (*freertos\_sdk\_x.y\pc\_tools\Download\_Tool\bin*):

1. Launch the Download tool provided with InnoPhase Talaria TWO SDK.
2. In the GUI window:
   1. Boot Target: Select the appropriate EVK from the drop-down.
   2. ELF Input: Load the using\_filesystem.elf by clicking on Select ELF File.
3. Programming: Prog RAM or Prog Flash as per requirement.

As mentioned in section 4.2, a new file named \data\sample\_text.txt is written in the DATA partition of the filesystem.

### Expected Output

|  |
| --- |
| UART:SNWWWWAE  4 DWT comparators, range 0x8000  Build $Id: git-ef87896f9 $  hio.baudrate=921600  flash: Gordon ready!  Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PNWWWAE  Build $Id: git-df9b9ef $  Flash detected. flash.hw.uuid: 39483937-3207-00b0-0064-ffffffffffff  $App:git-6600fea  SDK Ver: FREERTOS\_SDK\_1.0  Using File System Demo App  Mounting file system  File system mounted  Write Done.  File Data = This is testing for file system...  Read Done.  File Data = This is testing for file system...  Before deleting (unlink) file exist  Deleting (unlink) the file = /data/sample\_text.txt  Delete (unlink API) returns = 0  After deleting (unlink) file does not exist  ----------------Program Exit -------------------- |

After execution, for evaluation purposes, use the download tool Show File System Contents option. Click on Show File System Contents. The pop-up window displays the file written by the application.

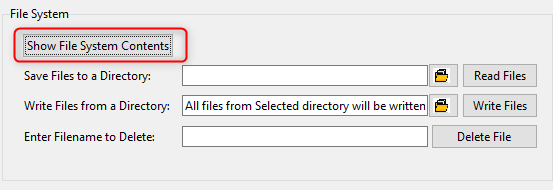


Figure 4: Download Tool - Show File System Contents

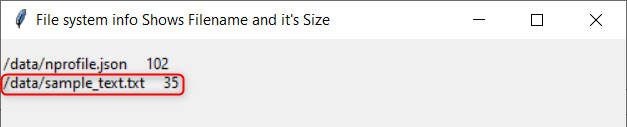


Figure 5: Filename and size

The file size is also equal to the read value from the application.

Using the Read Files option available in the Download Tool to retrieve the files and check the contents.

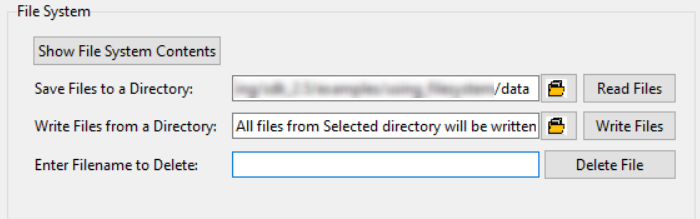


Figure 6: Download Tool - File System: Read files

Contents are extracted into the folder specified in the Download Tool.

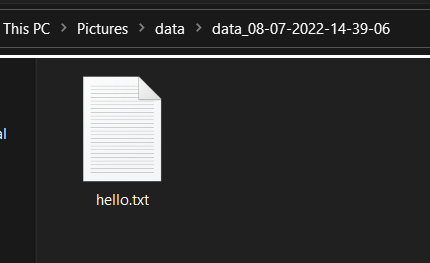


Figure 7: Extracted files