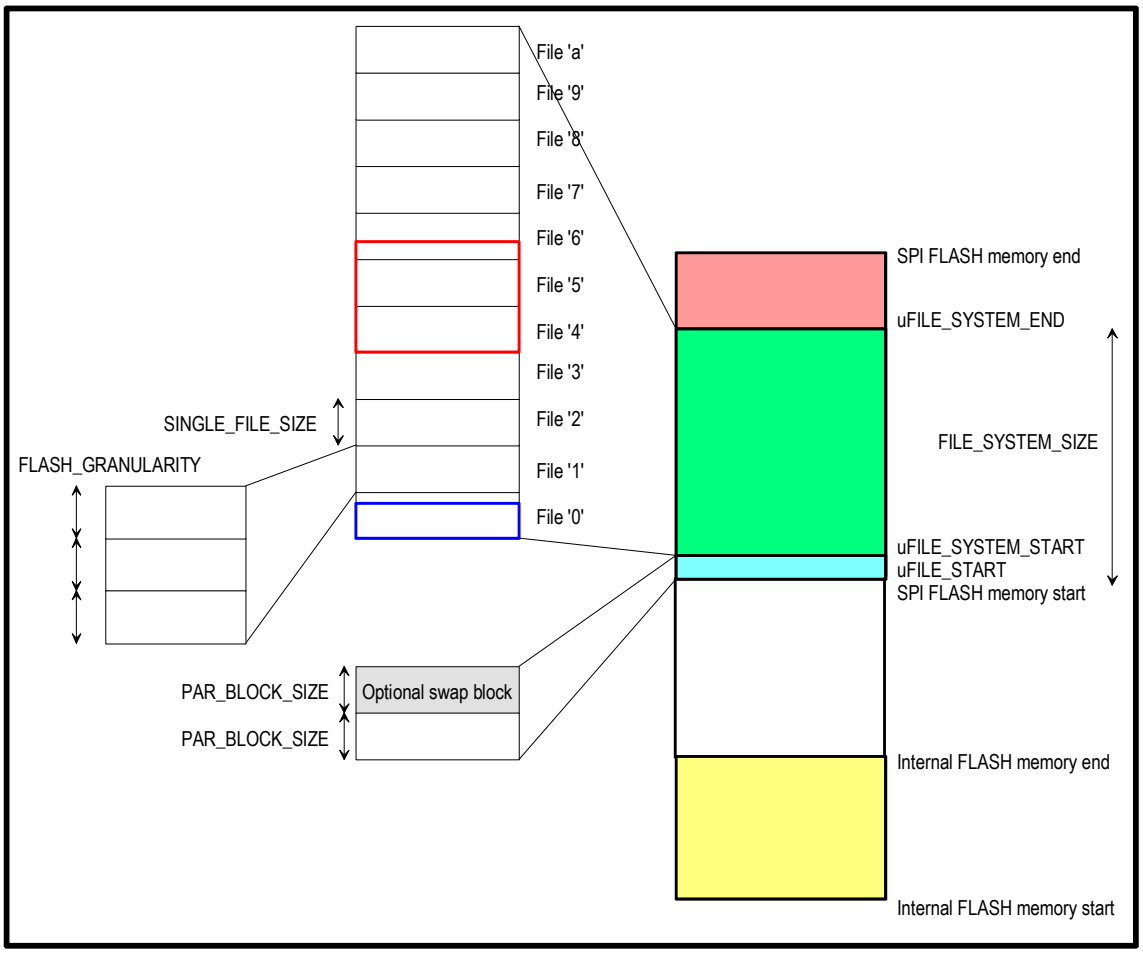
Introduction

This document describes the operation and use of the uFileSystem together with external SPI FLASH. This configuration enables larger memory sizes to be used but maintains compatibility with the µFileSystem as used in internal FLASH.

The implementation enables the parameter blocks to still be positioned in internal FLASH if required. It allows multiple SPI FLASH devices to be used, where they can be viewed as one larger device spanning a contiguous range of memory.

Please read the document "***µTaskerFileSystem***" to learn about the details of the µFileSystem itself.

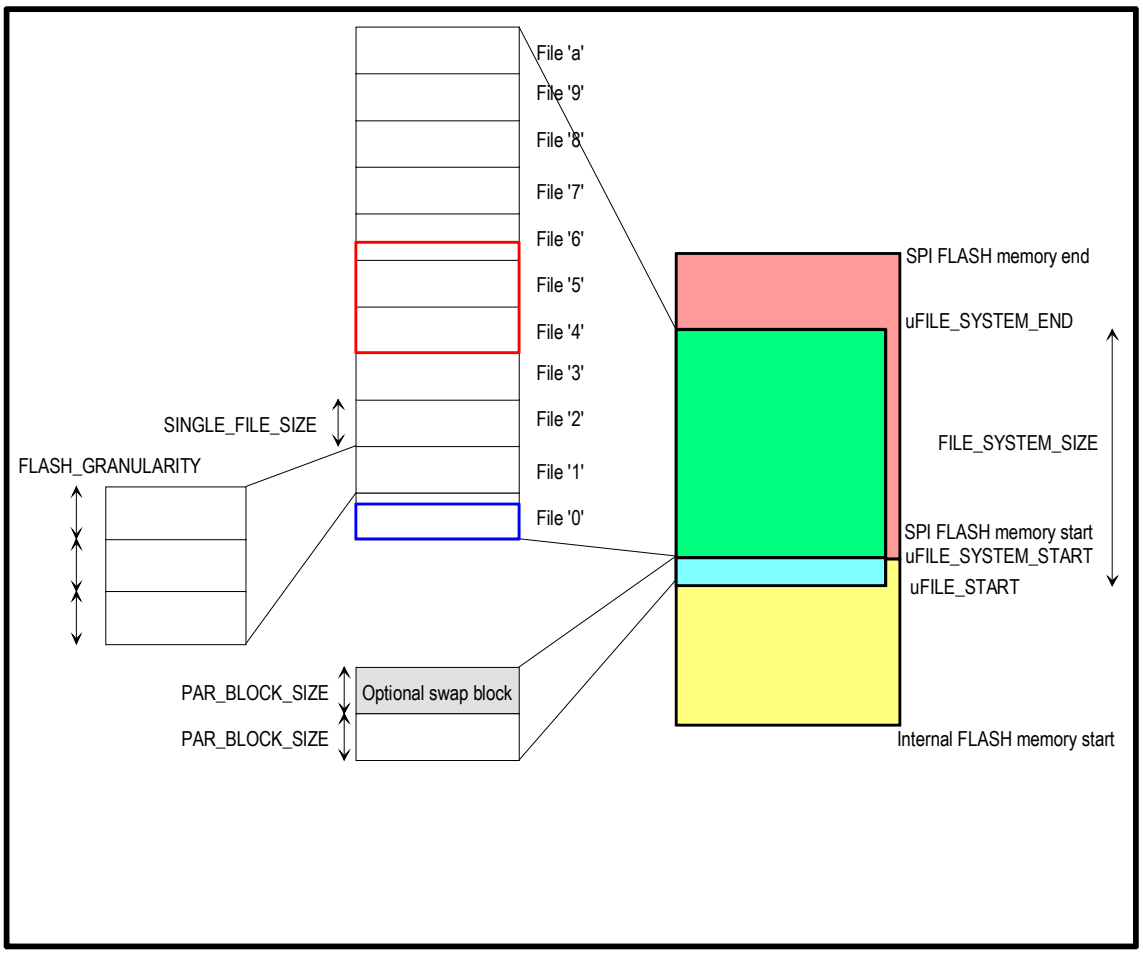


Location of µFileSystem in external SPI FLASH – example 1

The first example illustrates the SPI FLASH being located in the memory map at a higher location that the internal FLASH (internal FLASH often begins at location 0x00000000) and the SPI FLASH should always be located at a higher address in the memory map.

A space is illustrated between the end of the internal FLASH and the start of the external SPI FLASH. This space could also be 0 (the SPI FLASH start address is the location after the last internal FLASH address).

The µFileSystem is configured to be exclusively in external SPI FLASH and the parameter block is located at the beginning of SPI FLASH.



Location of µFileSystem in external SPI FLASH – example 2

The second example shows the external SPI FLASH located immediately after the internal FLASH in the memory map. In this case the µFileSystem is defined to start in the internal FLASH and to continue in the external SPI FLASH. This enables the parameter block to be situated in the internal FLASH and the files to be situated in external FLASH. This is possible due to the fact that the size of the parameter FLASH blocks can be specified to be different to the size of the FLASH blocks in the external SPI FLASH. These block sizes have to correspond to the capabilities of each FLASH type, which are often different.

Due to the typical difference in FLASH block sizes between the FLASH devices (internal and external SPI FLASH) the files themselves can not usually be spread between the two mediums. When more that one external SPI FLASH device is connected all should be of the same type. This results in a larger memory range without the need for the user to know the details involved in moving between the individual devices.

Configuration example

The following example shows a detailed configuration of a system with 1MByte external SPI FLASH and 256k internal FLASH starting at the location 0x00000000.

The SPI FLASH is configured to start immediately after the end of the internal FLASH, where the parameter FLASH is located and the FLASH parameter block is to be set in internal FLASH - this corresponds to example 2 above.

A suitable SPI FLASH device is the ***AT45DB081B*** which has 8MBit of FLASH (1 MByte). It has FLASH pages of *264* bytes in size and FLASH blocks of 8 x 264 byte (2112 bytes). The internal FLASH in the example has FLASH block sizes of 2k.

- app_hw_xxx.h configuration:

```
#define QSPI_CS0_LINE  PORT_QS_BIT3 // hardware specific for chip select control
#define FLASH_START_ADDRESS        (0x00000000)         // internal FLASH start
#define SIZE_OF_FLASH              (256*1024)           // internal FLASH size
#define SPI_FLASH_PAGE_LENGTH      264
#define SPI_FLASH_BLOCK_LENGTH     (8*SPI_FLASH_PAGE_LENGTH)
#define SPI_FLASH_PAGES            4*1024                 // 1M part expected
#define SPI_DATA_FLASH_SIZE        (SPI_FLASH_PAGES*SPI_FLASH_PAGE_LENGTH)
#define SPI_FLASH_START            (FLASH_START_ADDRESS + SIZE_OF_FLASH)
                                   // SPI FLASH starts immediately after FLASH

#define uFILE_START                0x3f000
         // FLASH location at 2 parameter blocks short of end of internal FLASH
#define SINGLE_FILE_SIZE     (FILE_GRANULARITY) // each file a multiple of 16k
#define FILE_SYSTEM_SIZE     (64*SINGLE_FILE_SIZE + PAR_BLOCK_SIZE)
                       // 1Meg reserved for file system (plus parameter blocks)

#define PARAMETER_BLOCK_SIZE       FLASH_GRANULARITY
#define PAR_BLOCK_SIZE             (2*PARAMETER_BLOCK_SIZE)
```

- config.h configuration:

```
#define SPI_FILE_SYSTEM            // we have an external file system via SPI
interface, rather than internal in FLASH
#define FLASH_FILE_SYSTEM          // we have an internal file system in FLASH

#define FILE_GRANULARITY       (8*SPI_FLASH_BLOCK_LENGTH)  // (2112 byte blocks)
                               // file granularity is equal to a multiple of the
                               // FLASH granularity (as defined by the device)
```

FLASH_GRANULARITY is defined as 2k in the processor's hardware header file.

When *SPI_FILE_SYSTEM* is set alone, it activates support for only an external SPI EEPROM – this is described in the document "***uTasker_SPI_EEPROM***". When set together with *FLASH_FILE_SYSTEM* it activates the SPI FLASH support together with internal FLASH support as required for the described configuration.

This example configuration defines the parameter swap block at the end of the internal FLASH and then the file system, consisting of (approx.) 1MByte made up of file segments of each 16'896 bytes in size. The µFileSystem can thus store up to 64 different files of up to 16'896 bytes in size (including file header according to µFileSystem use) each or less of multiples of the basic size.

By adding a second AT45DB081B with its chip select line attached to CS1 (the first is connected to CS0) the size of the SPI FLASH can be doubled.

- app_hw_xxx.h could be simply extended/modified to respect this as follows:

```
#define SPI_FLASH_MULTIPLE_CHIPS     // enable multiple SPI FLASH chip support
#define SPI_FLASH_DEVICE_COUNT   2   // 2 chips available
```

```
#define QSPI_CS1_LINE  PORT_QS_BIT4  // second chip select line
#define SPI_DATA_FLASH_SIZE          (SPI_FLASH_PAGES*SPI_FLASH_PAGE_LENGTH*
                                                    SPI_FLASH_DEVICE_COUNT)
```

- config.h:

```
#define FILE_GRANULARITY       (16*SPI_FLASH_BLOCK_LENGTH)  // (2112 byte blocks)
                               // file granularity is equal to a multiple of the
                               // FLASH granularity (as defined by the device)
```

This configuration has doubled the size of the data FLASH whereby the size of the FILE_GRANULARITY has been increased to 33'792 bytes per file block to compensate. Other mixtures are possible, depending on the exact project requirements. The basic principle behind the setup of the number and granularity of files is the same as in the µFileSystem for internal FLASH – the only difference is the fact that the size of memory available can be greatly increased and so it often makes sense to set up larger file blocks accordingly.

Simulating the set up in the µTasker simulator

When configuring the system as described above, the simulator will keep a copy of the contents of the internal FLASH (the parameter system in this case) plus a copy of the 1MByte SPI FLASH device(s). Each time the simulator is normally terminated the contents will be saved to the hard disc and re-loaded the next time it is started.

It is advised to delete these files each time the configuration is modified to ensure that the contents are suitable to the present mode. You will find these in the simulator directory:
FLASH_M5223X.ini – example when working with the M5223X processor [4k in size due to the content as explained above].
AT45DBXXX.ini – the SPI FLASH device [1'081'344 bytes in size as explained in the next section]. For each such device added, the size increases accordingly.

Why do the FLASH devices have 264 byte pages rather that 256 byte pages?

The fact is that the software to control 256 byte blocks is generally easier than the strange 264 byte blocks. The size seems to be historic since older SPI FLASH types can only do this size – whereas newer ones can be programmed (usually a one-time programming with no going back…) to alternatively do 256 byte pages.

To avoid problems where only old-types are available the µTasker solution is generally set up for 264 byte pages (although it can be modified to 256 byte pages by changing a define if needed - ***#define SPI_FLASH_PAGE_LENGTH 264*** to ***SPI_FLASH_PAGE_LENGTH 256***). Larger chips can also have pages of 528/512 bytes or even 1056/1024.

When the larger page size is specified, all available bytes are used and so the memory is actually larger – the AT45DB081B has in fact 1'081'344 bytes rather than 1'048'576 bytes. The extra bytes are lost when in the smaller page length mode.

It is possibly that some systems use the extra bytes for writing status information rather than actual using it for data. This is not the case for the µTasker solution as detailed here.

Modifications:
V0.01 17.11.2007: First draft