Introduction

The STR912F has 544K of internal FLASH memory [Bank 0 has 512k and Bank 1 has 32k]. The demo project is designed to operate with the file system in Bank 0 – the main FLASH – and the parameter system in Bank 1 – the secondary bank.

The file system is set up to use 256k of the available FLASH space and is positioned at the end of the address space. The block size (granularity) of the STR912F in the main FLASH bank is 64k, which means that a sub-file system is activated which alls the use of files smaller than the FLASH block size. These file system blocks can then be individually written but not always individually erased – see more details about this below.

The smallest FLASH write to the STR91XF is a short word (16 bit) write to even addresses but this is invisible to the user.

Sub-file system

The sub-file system has some restrictions due to the fact that it is not always possible to delete individual files without also having to delete others in the same FLASH sector. Therefore the rules for writing to the sub-file system are a little different to the normal uFileSystem.

- Files at the start of a FLASH sector can be addressed by two different file names: by the file system name and also be the sub-file system name.
- Files which are not positioned at the start of a FLASH sector can only be addressed by the sub-file name.
- Files written to the file system using their sub-file name will not delete any FLASH to make space for themselves. This allows several different files to be loaded to a sector without influencing already programmed ones. However it is the user's responsibility to ensure that the space to be written is available – also when the file will take up space in more than one FLASH sector.
- Files written to the file system using their file system name (only the ones which can be addressed like this) will clear all sectors in the way – as the uFileSystem otherwise works. This is useful for example when uploading software since it 'bull-dozes' space without previous knowledge of whether space is available, as may be necessary with the sub-file method.
- Only the sub-file names are displayed when viewing loaded files by FTP.
- When deleting a file, all memory within its sector(s) is deleted, meaning that other sub-files may be deleted. The user thus needs more knowledge of the consequences of deleting and will often need to reload lost sub-files.
- In some circumstances it may be simplest for inexperienced users (or end users) to always perform a delete of all loaded files and then reload all files so that details of the sub-file system operation are not visible.

Note that the STR91XF has relatively long FLASH sector delete times in the order of 0,7s. This is thus noticeable to the user and can in certain applications have other negative consequences. For FTP work this doesn't disturb the protocol operation apart from the additional delays involved.

File System Position in Memory

The file system for parameters and web pages is made up as follows.

There are two configurations, depending on whether the device is booting from bank 0 or from bank 1.

Bank 0 Boot

The main FLASH memory starts at 0x00000 and the file system address starts at 256k above the FLASH start address (0x40000). The secondary FLASH memory bank starts at 0x80000 and there are two 8k parameter swap blocks beginning at 0x84000.

Bank 1 Boot

The main FLASH memory starts at 0x80000 and the file system address starts at 256k above the FLASH start address (0x840000). The secondary FLASH memory bank starts at 0x00000 and there are two 8k parameter swap blocks beginning at 0x4000.

Bank 0 layout in the demo project (4k sub-files)

0x40000 Block 4 File '0' (64k)
Sub-Files
- 0x40000 Block 4 File '4' (4k)
- 0x41000 Block 4 File '5' (4k)
- 0x42000 Block 4 File '6' (4k)
- 0x43000 Block 4 File '7' (4k)
- 0x44000 Block 4 File '8' (4k)
- 0x45000 Block 4 File '9' (4k)
- 0x46000 Block 4 File 'A' (4k)
- 0x47000 Block 4 File 'B' (4k)
- 0x48000 Block 4 File 'C' (4k)
- 0x49000 Block 4 File 'D' (4k)
- 0x4a000 Block 4 File 'E' (4k)
- 0x4b000 Block 4 File 'F' (4k)
- 0x4c000 Block 4 File 'G' (4k)
- 0x4d000 Block 4 File 'H' (4k)
- 0x4e000 Block 4 File 'I' (4k)
- 0x4f000 Block 4 File 'J' (4k)

0x50000 Block 5 File '1' (64k)
Sub-Files

0x50000 Block 5 File ‘K’ (4k)
0x51000 Block 5 File ‘L’ (4k)
0x52000 Block 5 File ‘M’ (4k)
0x53000 Block 5 File ‘N’ (4k)
0x54000 Block 5 File ‘O’ (4k)
0x55000 Block 5 File ‘P’ (4k)
0x56000 Block 5 File ‘Q’ (4k)
0x57000 Block 5 File ‘R’ (4k)
0x58000 Block 5 File ‘S’ (4k)
0x59000 Block 5 File ‘T’ (4k)
0x5a000 Block 5 File ‘U’ (4k)
0x5b000 Block 5 File ‘V’ (4k)
0x5c000 Block 5 File ‘W’ (4k)
0x5d000 Block 5 File ‘X’ (4k)
0x5e000 Block 5 File ‘Y’ (4k)
0x5f000 Block 5 File ‘Z’ (4k)

0x60000 Block 6 File ‘2’ (64k)
Sub-Files
0x60000 Block 6 File ‘a’ (4k)
0x61000 Block 6 File ‘b’ (4k)
0x62000 Block 6 File ‘c’ (4k)
0x63000 Block 6 File ‘d’ (4k)
0x64000 Block 6 File ‘e’ (4k)
0x65000 Block 6 File ‘f’ (4k)
0x66000 Block 6 File ‘g’ (4k)
0x67000 Block 6 File ‘h’ (4k)
0x68000 Block 6 File ‘i’ (4k)
0x69000 Block 6 File ‘j’ (4k)
0x6a000 Block 6 File ‘k’ (4k)
0x6b000 Block 6 File ‘l’ (4k)
0x6c000 Block 6 File ‘m’ (4k)
0x6d000 Block 6 File ‘n’ (4k)
0x6e000 Block 6 File ‘o’ (4k)
0x6f000 Block 6 File ‘p’ (4k)

0x70000 Block 7 File ‘3’ (64k)
Sub-Files
0x70000 Block 7 File ‘q’ (4k)
0x71000 Block 7 File ‘r’ (4k)
0x72000 Block 7 File ‘s’ (4k)
0x73000 Block 7 File ‘t’ (4k)
0x74000 Block 7 File ‘u’ (4k)
0x75000 Block 7 File ‘v’ (4k)
0x76000 Block 7 File ‘w’ (4k)
0x77000 Block 7 File ‘x’ (4k)
0x78000 Block 7 File ‘y’ (4k)
0x79000 Block 7 File ‘z’ (28k)
0x7a000 Block 7 -

0x7b000 Block 7 -
0x7c000 Block 7 -
0x7d000 Block 7 -
0x7e000 Block 7 -
0x7f000 Block 7 -

Eg.
H = Sub-File beginning at address 0x4d000
4 = Sub-File beginning at address 0x40000 (write will not disturb other sub-files in sector)
0 = File beginning at address 0x40000 (write will delete all files in shared sectors)

Reading files per FTP

The ftp server returns each file with its name, eg. “H” and length and a fixed time and date. It adds also the type *.HTM since this is generally the data type stored in the simple file system.

If the directory is empty this is displayed using a dummy file with this name.

Writing files per FTP.

The ftp server knows the address to set by the file name. eg. “2” is to be written to address 0x60000. It first deletes a block to make place for it, if it is not already erased. If the file is longer that a single block, subsequent blocks are deleted as required. The user must ensure that the file names are suitable to fit in the available blocks and that each file has adequate space.

After a complete file has been saved, its length is also saved at the first two locations (which were left free). This information is returned when a DIR is requested.

When a sub-file is saved – for example “a” rather than “2” – this will be written without first making space for it. The user should ensure that there the space to be written is already blank.

Demo web files

Address of block	Web page with available space	Sub-Block number and size of file in demo project
0x40000 Block 4 File '0' / sub. '4'	Menu page 8k	0
0x41000 Block 4 sub. '5'		1
0x42000 Block 4 sub. '6'	LAN interface config. 8k	2
0x43000 Block 4 sub. '7'		3
0x44000 Block 4 sub. '8'	Background Logo 8k	4
0x45000 Block 4 sub. '9'		5
0x46000 Block 4 sub. 'A'	I/O interface 8k	6
0x47000 Block 4 sub. 'B'		7
0x48000 Block 4 sub. 'C'	LAN Stats page 4k	8
0x49000 Block 4 sub. 'D'	Serial config. 8k	9
0x4a000 Block 4 sub. 'E'		10
0x4b000 Block 4 sub. 'F'	Admin page 8k	11
0x4c000 Block 4 sub. 'G'		12
0x4d000 Block 4 sub. 'H'	Help page 4k	13
0x4e000 Block 4 sub. 'I'	Email Setup 8k	14
0x4f000 Block 4 sub. 'J'		15
0x50000 Block 5 File '1' / sub. 'K'	Logo 64k	16
0x51000 Block 5 sub. 'L'		17
0x52000 Block 5 sub. 'M'		18
0x53000 Block 5 sub. 'N'		19
0x54000 Block 5 sub. 'O'		20
0x55000 Block 5 sub. 'P'		21
0x56000 Block 5 sub. 'Q'		22
0x57000 Block 5 sub. 'R'		23
0x58000 Block 5 sub. 'S'		24
0x59000 Block 5 sub. 'T'		25
0x5a000 Block 5 sub. 'U'		26
0x5b000 Block 5 sub. 'V'		27
0x5c000 Block 5 sub. 'W'		28
0x5d000 Block 5 sub. 'X'		29
0x5e000 Block 5 sub. 'Y'		30
0x5f000 Block 5 sub. 'Z'		31
0x60000 Block 6 File '2' / sub. 'a'		32
0x61000 Block 6 sub. 'b'		33
0x62000 Block 6 sub. 'c'		34
0x63000 Block 6 sub. 'd'		35
0x64000 Block 6 sub. 'e'		36
0x65000 Block 6 sub. 'f'		37
0x66000 Block 6 sub. 'g'		38
0x67000 Block 6 sub. 'h'		39
0x68000 Block 6 sub. 'i'		40
0x69000 Block 6 sub. 'j'		41
0x6a000 Block 6 sub. 'k'		42

0x6b000 Block 6 sub. 'l'		43
0x6c000 Block 6 sub. 'm'		44
0x6d000 Block 6 sub. 'n'		45
0x6e000 Block 6 sub. 'o'		46
0x6f000 Block 6 sub. 'p'		47
0x70000 Block 7 File '3' / sub. 'q'		48
0x71000 Block 7 sub. 'r'		49
0x72000 Block 7 sub. 's'		50
0x73000 Block 7 sub. 't'		51
0x74000 Block 7 sub. 'u'		52
0x75000 Block 7 sub. 'v'		53
0x76000 Block 7 sub. 'w'		54
0x77000 Block 7 sub. 'x'		55
0x78000 Block 7 sub. 'y'		56
0x79000 Block 7 sub. 'z'		57
0x7a000 Block 7		58
0x7b000 Block 7		59
0x7c000 Block 7		60
0x7d000 Block 7		61
0x7e000 Block 7		62
0x7f000 Block 7		63

Bank 1 boot layout in the demo project (4k sub-files)

If the µTasker demo project is build to boot from the secondary FLASH bank 1, the main FLASH bank 0 begins at 0x80000. It is I this case necessary to add 0x80000 to all addresses shown in the previous tables. For example the first file '0' is situated at the address 0xc0000 rather than 0x40000.