

ZCFS - log-filesystem on STM32

Alberto Crosta

Matteo Zoia

October 8, 2021

Università degli Studi di Milano

- Introduction
- Hw components
- System Architecture
- Memory/Disk layout
- Broker and Protocols
- Demo
- Offline analysis
- Future works

Scenario:

- STM32 boards with low or no memory
- Want to log different actions in a structured way...
- ... off board

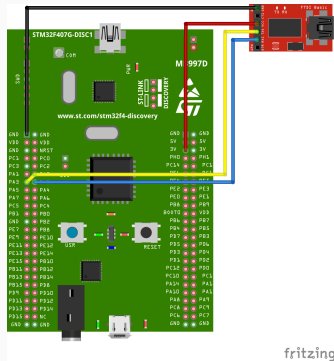
Our work:

- The implementation of a log-filesystem on the STM32 HAL
- The communication between the board and a host to store the data
- The organization of the memory to store files quickly

Hw components

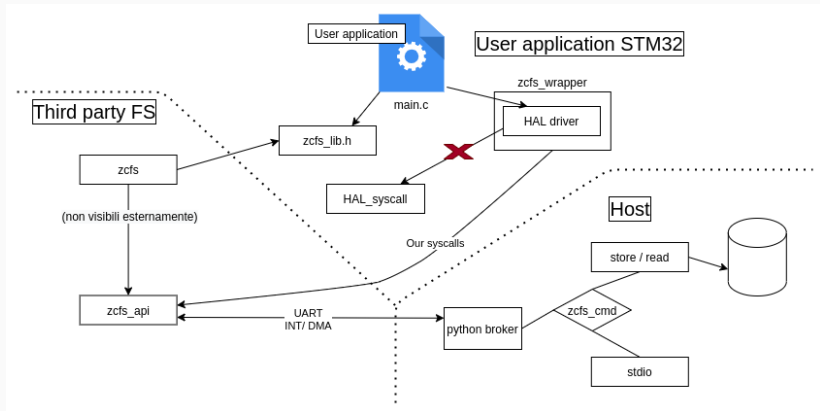
Hw components

- STM32F407 board
- FTDI/UART dongle



System Architecture

System Architecture

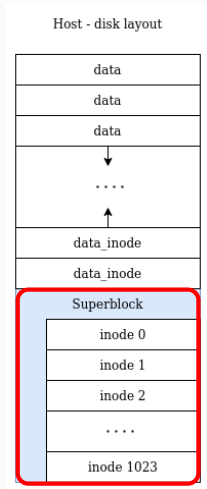


Memory/Disk structure

Memory/Disk structure - Superblock

The **superblock** keeps every information about the filesystem organization. The structure contains the following field:

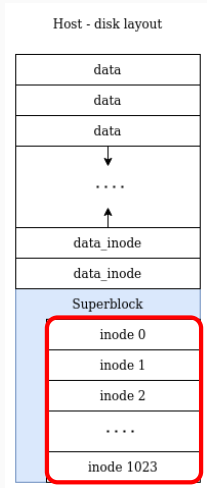
- address of the last data chunk written
- address of the last inode data chunk
- fd for the next request (incremental)
- list of inode



Memory/Disk structure - inode

Every **inode** is used to save information about a file. The inode structure contains:

- fd number
- file name
- time of last edit
- size of the file (summing each data chunk)
- is_open flag
- pointer to last data inode
- pointer to first data inode

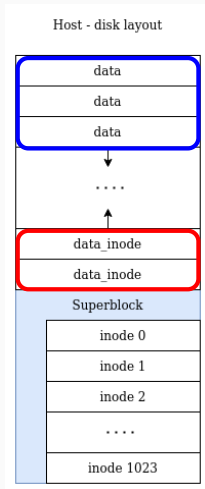


Memory/Disk structure - data-inode

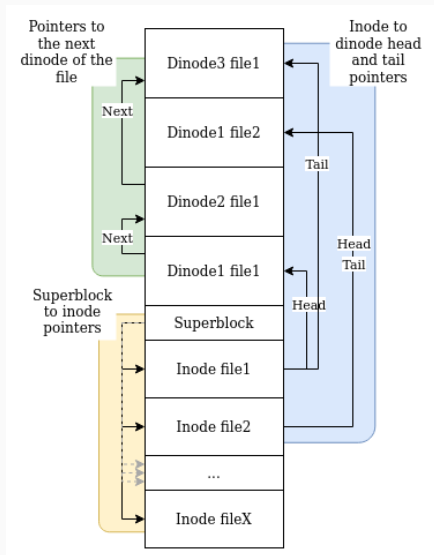
The **data-inode** is a structure that contains information about every chunk of data written to the disk. In particular, it contains:

- pointer to the data chunk
- size of the chunk
- pointer to the next dinode data chunk structure

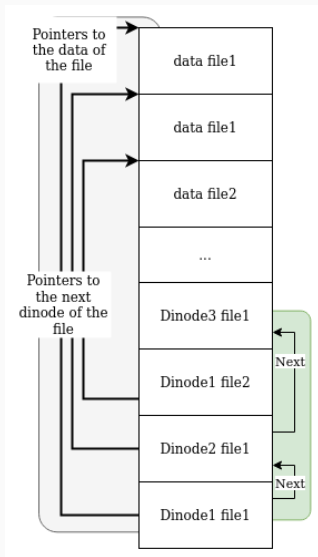
The **data chunk** are simply bytes append one above the other



Memory/Disk structure - inode and dinode



Memory/Disk structure - dinode and data chunk



Broker and Protocols

Broker and Protocols

The broker is the interface between the user program on the board and the host disk (USART protocol).

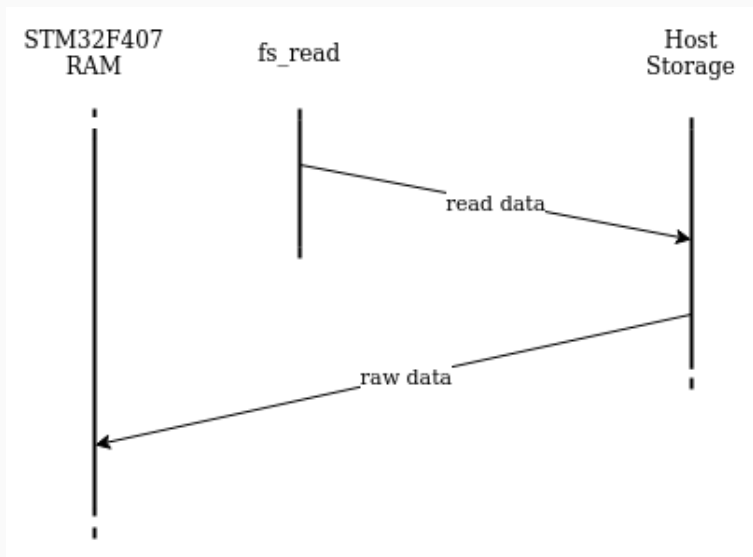
It performs all the operations with the disk:

- **reads** taking data from the board
- **write** updating the host disk with the new written data

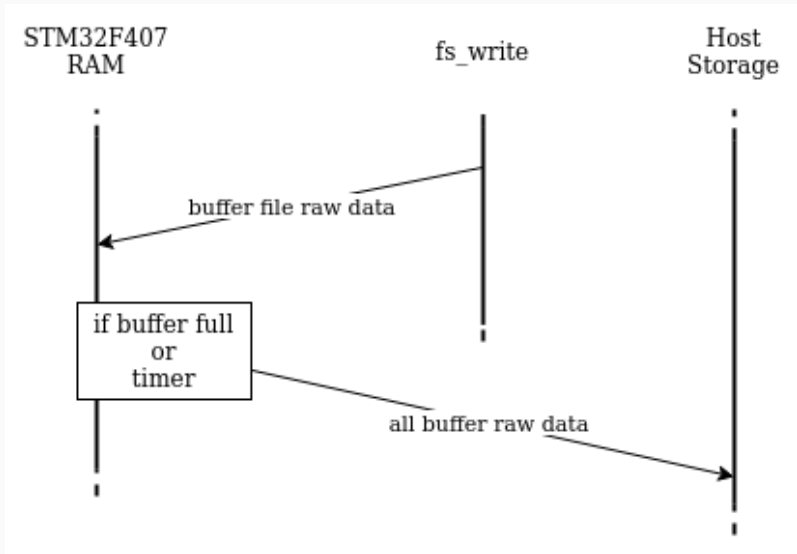
Broker and Protocols - API

- **zcfs_open(char* file_name)** opens a file, creating all structure on the disk and in memory.
- **zcfs_close(uint32_t fd)** closes the specific file, removing it from the opened ones but left the other structure so the user can have the possibility to reopen it later.
- **zcfs_write(int fd, char* ptr, int len)** writes the data in the file, only if the file is opened.
- **zcfs_read(uint32_t fd, char* data, int start, int stop)** reads the the file (entirely or not), only if the file is opened.

Broker and Protocols - protocol: read



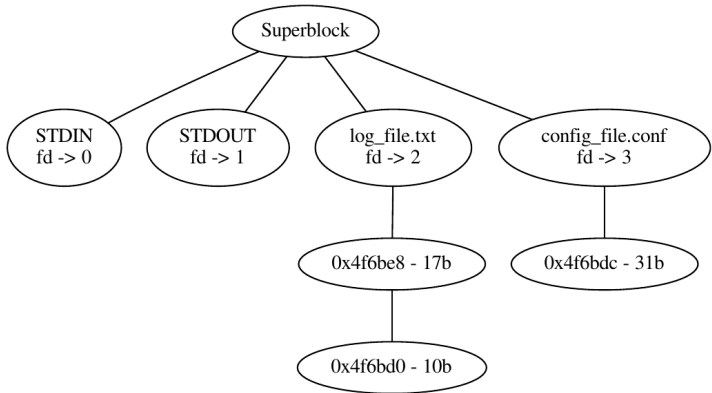
Broker and Protocols - protocol: write



Demo

Offline analysis

Offline analysis - inode and data inode (example)



Future works

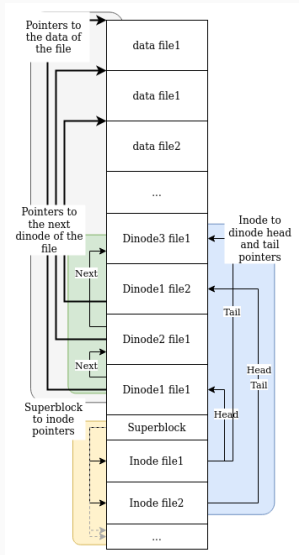
- Filesystem architecture
 1. fixed number of files
 2. directory supported but not implemented
- Broker setup
 1. configuration file
 2. disk reload
- Protocol configuration
 1. support different protocol (I2C, etc)
- Offline analysis
 1. tools for different type of analysis

- Questions?

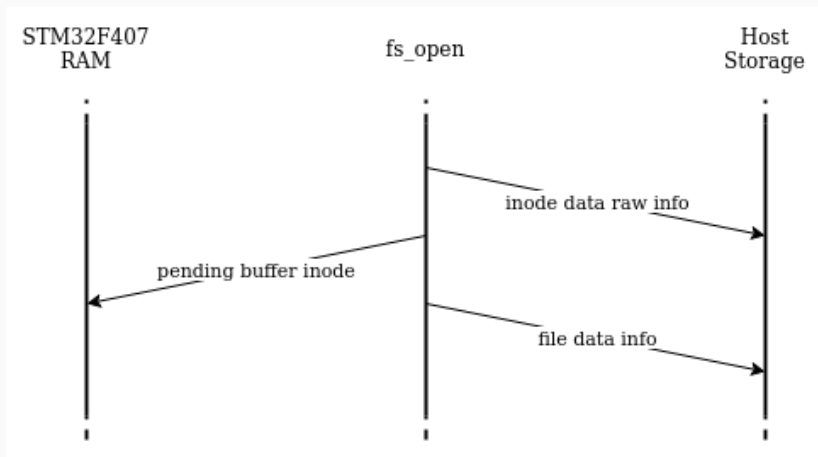
That's all folks

Extra

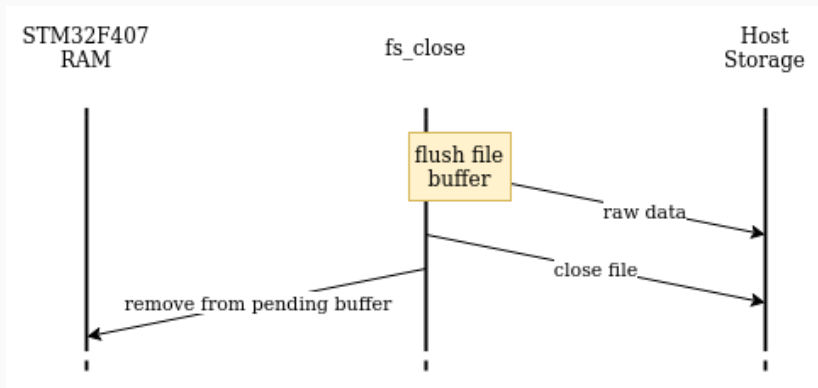
Extra - complete memory structure



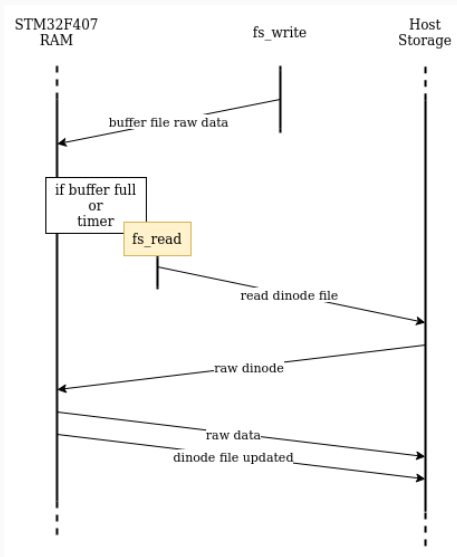
Extra - fs_open



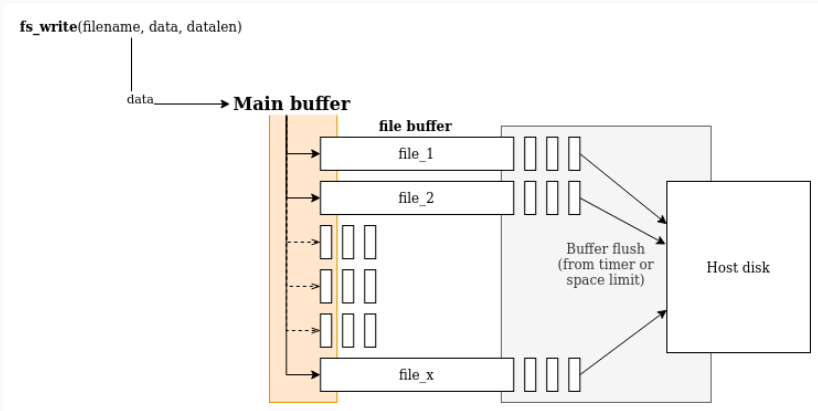
Extra - fs_close



Extra - fs_write_second_dinode



Extra - buffer implementation



Extra - NVIC configuration

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
DMA1 stream5 global interrupt	<input checked="" type="checkbox"/>	1	0
DMA1 stream6 global interrupt	<input checked="" type="checkbox"/>	1	0
USART2 global interrupt	<input checked="" type="checkbox"/>	0	0
TIM6 global interrupt, DAC1 and DAC2 underrun error in...	<input checked="" type="checkbox"/>	2	0
FPU global interrupt	<input type="checkbox"/>	0	0

Extra - USART configuration

Configuration

Reset Configuration

✓ NVIC Settings



✓ DMA Settings

✓ GPIO Settings

✓ Parameter Settings

✓ User Constants

Configure the below parameters :

▼ Basic Parameters

Baud Rate

Word Length

Parity

Stop Bits

38400 Bits/s

8 Bits (including Parity)

None

1

▼ Advanced Parameters

Data Direction

Over Sampling

Receive and Transmit

16 Samples