Communication between a computer and a STM32F401RTB chip through serial port com

The repository is separated in 3 subfolders:

- Python scripts to control the chip through serial com
- STMCubeIDE project files to compile bin file to flash stm32 chip (CubeIDE version 1.12.1)
- Openocd flashing configuration files

Look into the Python files to get an overview of the commands and how they work.

To compile the bin file yourself, you need STM32CubeIDE or STM32CubeCLT.

STM32 chip program

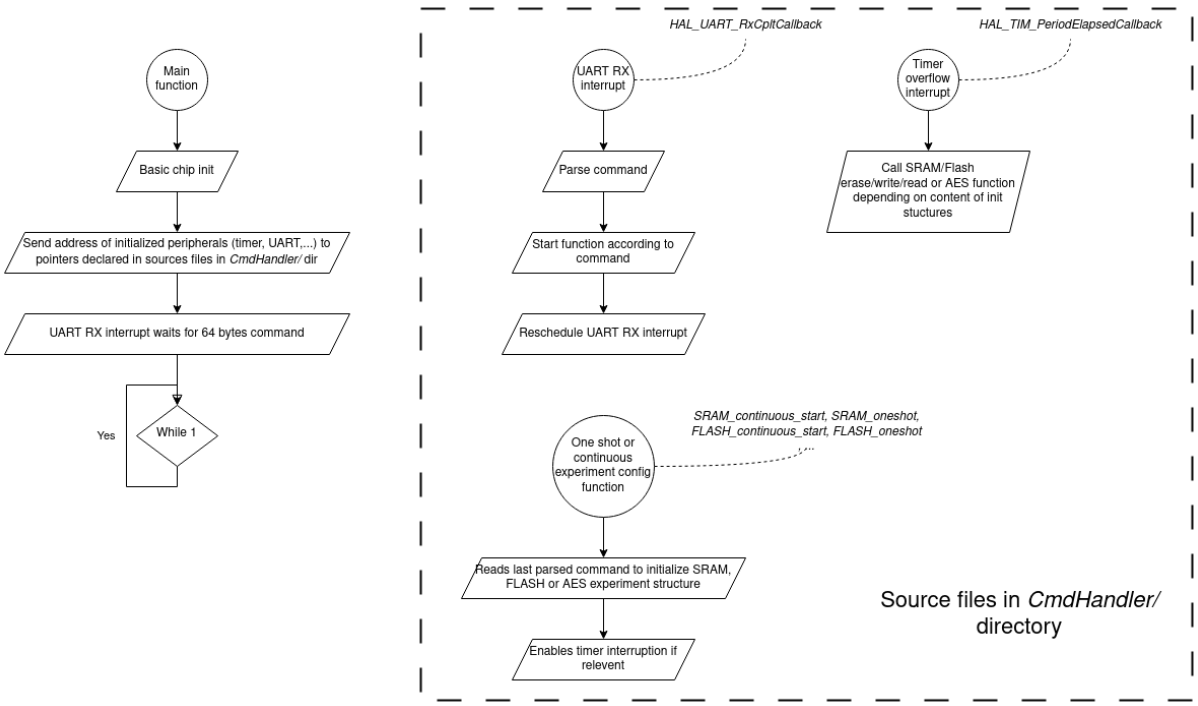
The organisation of the program is as follows:

- Application source code are in *Core/Src/CmdHandler* and *Core/Inc/CmdHandler*.
- In the *main.h* file, *cmd_handling.h* header is included to link application source code.
- In the *main.c* file, *Init_Cmd_Handling()* and *Start_Cmd_Reception()* are the only user code added. They enable the rest of the user code to use the right reference to the uart, dma and timer. This way, it is possible to decide which peripheral is used without changing the user files.

Once *Start_Cmd_Reception()* is called, the UART waits for 64 bytes before triggering a callback function that parses the received command and calls the corresponing functions.

When one-shot commands are issued, the fonction is directly called by this callback. When a continuous mode command is issued, the callback setups a timer and enables it's IT so that the corresponding function is called repeatedly by the timer's interruption in parallel to the rest of the program.

Three structures are defined to store the parameters of Flash read/write/erase, SRAM read/write and AES operation. It is mostly useful for continuous mode as the timer callback just needs to read those structs to know what functions to start with what parameters.



Porting the chip program to an other STM32 chip

The structure of the code should make porting as easy as possible. The *configuration.h* header file should centralize all the parameters that change from one chip to another.

Regarding FLASH erase operation, porting might require more work if the new chip's FLASH contains a different number of sectors of if it is organized in pages instead of sectors. In this situation the file *FPEC_interface.c* would also have to be tweaked.

Serial communication

See *NucleoF4.py* in *python_scripts/* directory to see what the commands are and what they look like.

They are basic string commands padded with '\0' to reach 64 bytes and sent to the chip through a serial com port.

The *main.py* Python script provides examples for every feature.