

HexFlashLib 1.0

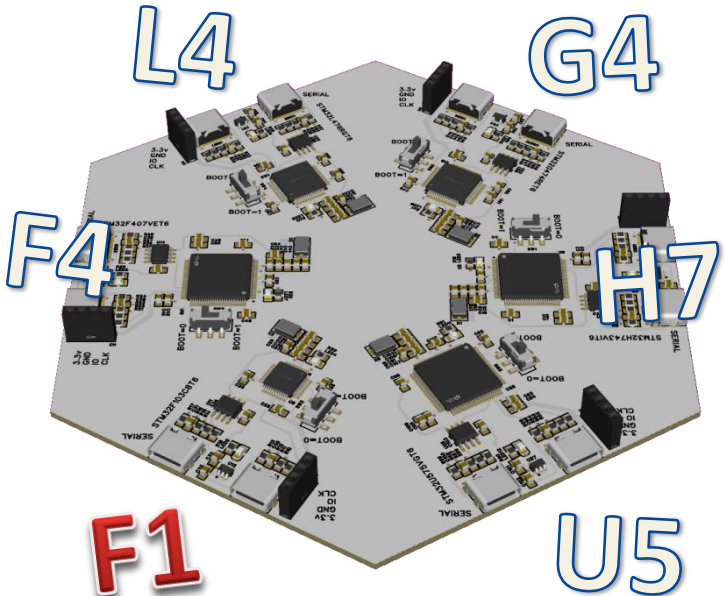
- Use a single flashing approach for any STM32
- Bootloaders that work across all STM32 families
- Learn multiple STM32 chips the easy way
- Stop wasting time



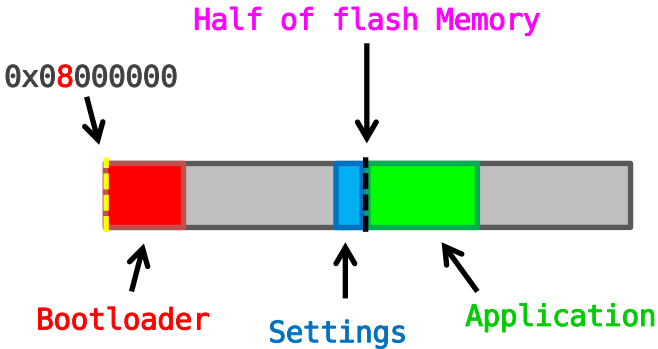
U5
F4 F1 L4
G4 H7

The Board

- I built this board to prove the concept
- Six different STM32 chips — same approach
- How to make portable STM32 apps

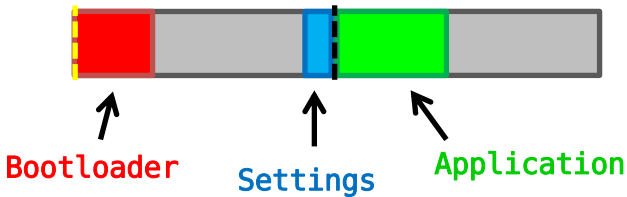


Simple Memory Layout



You no longer need to
handle addresses – the
library manages everything

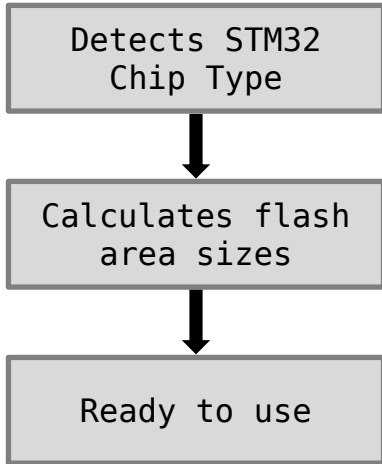
Simple Memory Layout



```
// Super easy functions
flash_region_erase_and_select();
flash_region_read();
flash_region_write();
flash_region_run();

// For example
flash_region_run(REGION_APPLICATION);
```

How so Simple?



Adding the HexFlashLib

- Copy **HexFlashLib** folder to workspace directory
- Copy **HexFlashApp** folder to workspace dir.
- Right-click on project name
- New ⇒ Folder ⇒ Advanced ⇒ Linked Folder
- Browse to add **HexFlashLib** and **HexFlashApp**
- Right-click on project name
- Properties ⇒ C/C++ General ⇒ Paths & Symbols
- Source location tab
- Add folder... ⇒ **HexFlashLib** & **HexFlashApp**
- Includes tab
- Add... ⇒ **../HexFlashLib** & **../HexFlashApp**

Adding the HexFlashLib

- Project Explorer ⇒ Open **main.c**
- Include application code

```
/* USER CODE BEGIN Includes */  
#include "app.h"  
/* USER CODE END Includes */
```

- You can select application like this

```
#define SELECT_APP_1 // Read/write/erase  
// #define SELECT_APP_2 // Flash info  
// #define SELECT_APP_3 // Bootloader  
// #define SELECT_APP_4 // LED blink
```

- No need to copy-paste between projects
- No need to modify **main.c** of each project
- Single **app_x.c** source file for all STM32 projects

Adding the HexFlashLib

- Add **app_boot()** into USER CODE **1**

```
/* USER CODE BEGIN 1 */  
app_boot();  
/* USER CODE END 1 */
```

- Add **app_init()** into USER CODE **2**

```
/* USER CODE BEGIN 2 */  
app_init(&huart1);  
/* USER CODE END 2 */
```

- Add **app_loop()** into USER CODE **WHILE**

```
/* USER CODE BEGIN WHILE */  
while (1)  
{  
    app_loop();  
/* USER CODE END WHILE */
```


Insanely Simple

- Three flash areas – no need for addresses!

```
#define REGION_BOOTLOADER    0
#define REGION_SETTINGS      1
#define REGION_APPLICATION    2
```

- Three functions to work with flash:

```
flash_region_erase_and_select(REGION_...);
flash_region_read(...);
flash_region_write(...);
```

- One function to run bootloader or application

```
flash_region_run(REGION...);
```

Yes, that's all you need!

Flash Read

- Single-byte command receiver

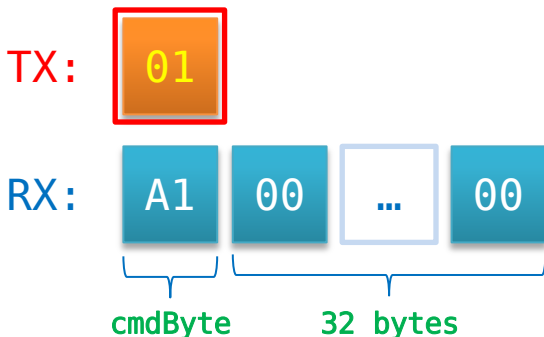
```
if (COM_ReadFast(&cmdByte, 1))
switch(cmdByte)
{
    // parse commands here
}
```

- Implement Read Flash command

```
case 0x01: // Command 0x01 - Read Flash
{
    uint8_t bytes[32];
    flash_region_select(REGION_SETTINGS);
    flash_region_read(bytes);

    // Send response to serial port
    cmdByte=0xA1;
    COM_Write(&cmdByte, 1);
    COM_Write(&bytes, 32);
    break;
}
```

Flash Read



```
case 0x01: // Command 0x01 - Read Flash
{
    uint8_t bytes[32];
    flash_region_select(REGION_SETTINGS);
    flash_region_read(bytes);

    // Send response to serial port
    cmdByte=0xA1;
    COM_Write(&cmdByte, 1);
    COM_Write(&bytes, 32);
    break;
}
```

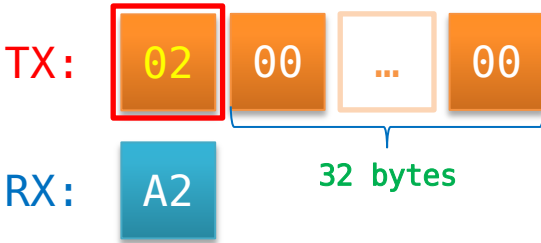
Flash Write

- Write Flash serial command

```
case 0x02: // Command 0x02 - Write Flash
{
    uint8_t bytes[32];
    if (COM_Read(&bytes,32))
    {
        flash_region_select(REGION_SETTINGS);
        flash_region_write((uint8_t*)&bytes);

        // Send response to serial port
        cmdByte=0xA2;
        COM_Write(&cmdByte, 1);
    }
    break;
}
```

Flash Write



```
case 0x02: // Command 0x02 - Write Flash
{
    uint8_t bytes[32];
    if (COM_Read(&bytes, 32))
    {
        flash_region_write((uint8_t*)&bytes);

        // Send response to serial port
        cmdByte=0xA2;
        COM_Write(&cmdByte, 1);
    }
    break;
}
```

Flash Erase

- Erase Flash serial command

```
case 0x03: // Command 0x03 - Erase Flash
{
    flash_region_erase_and_select(
                                REGION_SETTINGS);

    // Send response to serial port
    cmdByte=0xA3;
    COM_Write(&cmdByte, 1);
    break;
}
```

TX:

03

RX:

A3

Portability

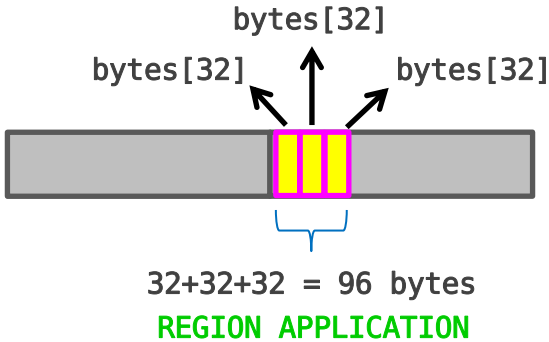
- Compiles and works on STM32F4, STM32H7, ...
- Easy to extend to other series
- Shared application code

Flash Read

- Select region and start reading

```
uint8_t bytes[32];  
flash_region_select(REGION_APPLICATION);  
flash_region_read((uint8_t*)&bytes); // 32  
flash_region_read((uint8_t*)&bytes); // 32  
flash_region_read((uint8_t*)&bytes); // 32
```

- Only select and read – no erase

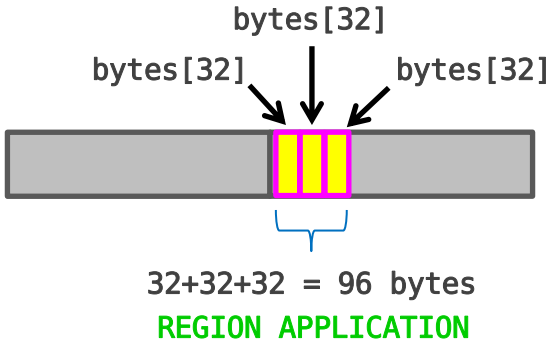


Flash Write

- Erase + Select region and start writing to it

```
uint8_t bytes[32];  
flash_region_erase_and_select(REGION_APP...);  
flash_region_write((uint8_t*)&bytes); // 32  
flash_region_write((uint8_t*)&bytes); // 32  
flash_region_write((uint8_t*)&bytes); // 32
```

- Flash address is incremented automatically



Flash Erase

- Flash erase is simple

```
flash_region_erase_and_select(REGION_...);
```

- Just select one of these three regions

```
#define REGION_BOOTLOADER    0  
#define REGION_SETTINGS     1  
#define REGION_APPLICATION   2
```

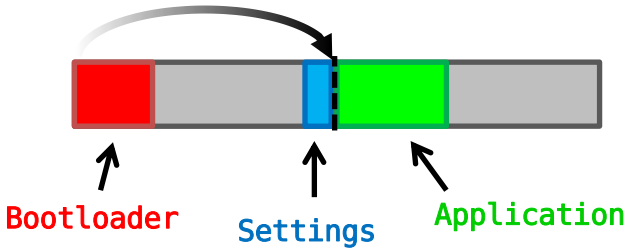
- You can customize regions – source code is tiny!
- You can select region without erasing it

```
// Erase and select  
flash_region_erase_and_select(REGION_...);  
  
// Select only (no erase)  
flash_region_and_select(REGION_...);
```

Flash Run

- Run Application region

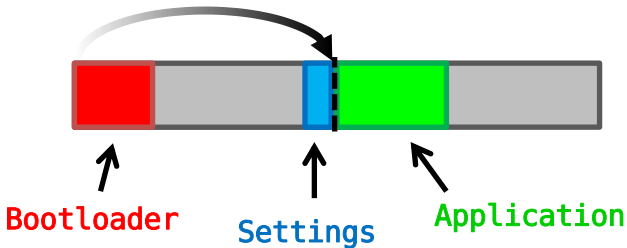
```
flash_region_run(REGION_APPLICATION);
```



Flash Run

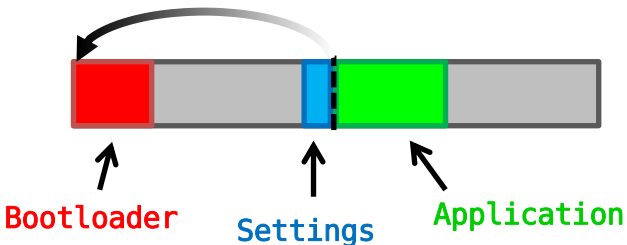
- Run Application region

```
flash_region_run(REGION_APPLICATION);
```



- Return to Bootloader

```
flash_region_run(REGION_BOOTLOADER);
```



Stay Updated

- Use the library today
- Learn tricks like shared folders
- Subscribe for update videos