STME32Fx Microcontroller Custom Bootloader Development

About this Upskilling Process

To learn about Bootloader working method and process inside MCU, to write my own custom bootloader for STM32Fx microcontroller and to test it's operations. To implementation Bootloader Communication. To know about Custom Bootloader command packets and different boot modes of the STM32 microcontroller.

And to know about Boot loader:

- Flash handling implementation (Sector Erase/Program/Mass erase)
- Options bytes(OB) Program handling implementation
- Flash sector protection status handling implementation
- In application programming implementation (IAP)
- Vector table relocation of ARM cortex Mx processor

The Logical Phase of the Whole Process

- **About Bootloader**
- **MCU memory, Reset Sequence and Boot configs**
- **Exploring STM32 Native Bootloader**
- **Custom Bootloader Communication with Host**
- **Boot-Loader :**
  - Project Creation
  - UART Testing
  - Jumping to User Code
  - Read Commands from Host
- Implementing :
  - BL_GET_VER_CMD
  - BL_GET-HELP_CMD

    - BL_GET_CID_CMD
    - BL_GET_RDP_LEVEL_CMD
    - BL_GET_GO_TO_ADDR_CMD
    - BL_FLASH_ERASE_CMD
    - BL_MEM_WRITE_CMD
- **Options Bytes and Flash Sector protection**
- **Exploring Host Application**

> ***Let's Begin the Journey***

Bootloader and Development Keywords

- In-Circuit Debugger/Programmer (ICDP)
- In Application Programming (IAP)
- In System Programming (ISP)

About Bootloader

> ***Bootloader***
> A Bootloader is nothing but a small piece of code stored in the MCU flash or ROM to act as an application loader as well as a mechanism to update the applications whenever required.

The Application of the bootloader is to download and upload binaries in to MUC.

Though STM32 Nucleo has ICDP with in its frame work, still we can use its bootloader sitting inside the MCU itself to download and upload the application binary into the flash memeory to execute the instructions.

MCU memory, Reset Sequence and Boot configs

STM32F446xx Memory Organixation

- Internal Flash memory also called as Embedded Flash memory of 512KB (on-chip Flash)

- Internal SRAM1 of 112KB
- Internal SRAM2 of 16KB
- System Memory (ROM) of 30KB (Read only memory where bootloader get stored)
- OTP memory of 528 bytes (One Time Programmer)
- Option bytes memory of 16bytes
- Backup RAM of 4KB (can hold the data using battery)

- Internal Flash Memory

- Size is 512KB
- Begins @ 0x0800 0000
- Ends @ 0x0807 FFFF
- Used to store your application code and read only data of the program, vector table.
- Non volatile

- Internal SRAM1

- Size is 112KB
- Begins @ 0x2000_0000
- Ends @ 0x2001_BFFF
- Used to store your application global data, static variables
- Also used for Stack and Heap purpose
- Volatile
- You can also execute code from this memory

- Inteernal SRAM2

- Size is 16KB
- Begins @ 0x2001_C000
- Ends @ 0x2001_FFFF
- Used to store your application global data, static Variables
- Also can be used for Stack and Heap Purpose
- Volatile
- You can also execute code from this memory

- System Memory (ROM)

- Size is 30KB
- Begins @ 0x1FFF_0000
- Ends @ 0x1FFF_77FF
- All the ST MCUs store bootloader in this memeory

- This memory is Read Only
- By default MCU will not execute MCU to boot or execute boolader from this memeory.

- Flash module Organization

- Main Memory (On Chip Flash)
  - Sector 0 => 16 Kbytes
  - Sector 1 => 16 Kbytes
  - Sector 2 => 16 Kbytes
  - Sector 3 => 16 Kbytes
  - Sector 4 => 64 Kbytes
  - Sector 5 => 128 Kbytes
  - Sector 6 => 128 Kbytes
  - Sector 7 => 128 Kbytes
- System memory (30 Kbytes)
- OTP area (528 bytes)
- Option bytes (16 bytes)

Understanding: Reset Sequence and memory Aliasing of the MCU

- When we reset the MCU, the PC of the processor is loaded with the value 0x0000_0000
- Then processor reads the value @ memory location 0X0000_0000 in to MSP (Main Stack Pointer)
  - MSP = value@0x0000_0000
  - MSP is a Main Stack Poniter register
  - That means, processor first initialize the stack pointer register.
- After that, processor reads the value @ memory location 0x0000_0004 in PC.