**Chapter 1:**

•What is BOOTLOADERS/ISP/IAP and how do we program our Arduino/STM32/SJ-ONE boards?

•What is Bootloader?

To download/upload binaries (IAP). Bootloader allows to reprogram the chip. It is a small piece of code stored in the MCU flash/ROM to act as an application loader as well as mechanism to update the application whenever required.

For Arduino (Atmega 328P):

MCU comes with on chip bootloader in Arduino IDE,

We first write the code into Arduino IDE and then click upload,

board goes to reset, and bootloader runs and waits for command from Arduino IDE. Bootloader receives a code (binary) and it then programs the flash of the MCU and again goes to reset and gives control to application you programmed. This is in application programming (IAP).

•What happens when Arduino undergoes reset?

Arduino bootloader runs first. It waits for command-if it does not receives any command it switches control to Application program.

For STM32f446RE nucleo64:

It has a Bootloader in its ROM(Programmed by STM).This Bootloader code cannot be altered(thanks to its ROM. We need to replace the ROM chip).

•Does it run whenever MCU undergoes reset?

No. We need to activate, changing the status of boot pin. (Refer the manual. We need to short two pins on board).

What is ICD/ICP(in circuit Debugger/Programmer) ?(ST\_link)

We use ICD to program ST32 board and debug. Its an external circuit connected to Flash. We can directly flash the chip without involving MCU. Hence it is called as ISP(in system Programming).ICD enables us to debug the code/Access the Rx values/Flash.

Arduino does not have ICD/ICP, Thus we cannot debug the code in Arduino(No breakpoints).we need IAP(in Application Programming) in such cases.

Imp Links:(READ IT FOR CLEARITY BTW ISP/IAP)

<http://www.esacademy.com/en/library/technical-articles-and-documents/8051-programming/using-flash-memory-in-embedded-applications.html>

* What is the objective of this project and which board to choose for this project?

Nucleo board has STM bootloader in its ROM. We can’t program it. We would program our own bootloader in one sector of Flash and Application Program in another sector. There would two application:

1: Host(Same as Arduino IDE) through which we can command our bootloader. This application is designed using Python and its pyserial lib. When you plug the board, its detected as virtual com port and communicates with the board via USB-UART over UART2.

2: Bootloader which would be present in sector 1 of flash.

STM32f446RE nucleo64 is chosen as it has ICD. If something goes wrong, we could erase the entire flash.

CubeMX facilitates inbuild API for Communication and to flash the Memory.

When installing OS into PC, we enter boot mode by pressing some keys (DEL/F2) during start-up. Similarly, we would press a user button to enter boot mode. If button pressed is detected, we would enter boot mode and board will wait for command from the host. If button is not pressed the program will jump to user application located in sector 2 of flash (Transferring control to user Application).

**Chapter 2: Memory Organisation of stm32f446xx**

It has internal flash/Embedded flash memory of 512kB.

internal SRAM1 of 112kB

internal SRAM2 of 16KB

System Memory (ROM) of 30kB --this is READ ONLY MEM. This Memory is used by ST for its native bootloader.

OTP memory of 528B -- it can be programmed only once and is used to store the product no, serial no, etc.

Option bytes Memory of 16B. --It has flag which control the access to flash memory.

Backup Ram of 4KB --When you remove the power, you could still hold/backup the contents of RAM by Battery power.

internal flash512KB --Used to store application code/read only data/vector table.

begins at 0x0800 0000

ends at 0x0807 FFFF

Its non-volatile.

memory is divided into 8 sectors:

sectors 0 16kB 0x0800 000

sectors 1 16KB

sectors 2 16KB

sectors 3 16KB

sectors 4 64KB

sectors 5 128KB

sectors 6 128KB

sectors 7 128KB 0x0807 FFFF

Internal SRAM1

begins at 0x2000 0000

ends at 0x2001 BFFF

used to store application global data/static variables.

used for stack and heap

volatile

execute code from this memory.

Internal SRAM2

size: 16kB

begins at 0x2001 C000

ends 0x2001 FFFF

used to store application in global data/static variables.

used for stack and heap.

Volatile.

execute code from this memory.

System internal memory(ROM)

size:30KB

begins: 0x1FFF 0000

ends: 0x1FFF 77FF

All ST MCU's store bootloader in this memory

this is read-only.

by default, MCU does not execute any code from this memory but you can configure MCU to boot from this memory.