STM32-Bootloader

Bootloader, External Flash Loader, OTA Firmwareupdate

Discription

This is my own little wiki on how to build a custom Bootloader for STM32 F446 Series of Microcontroller. This repo contains usefull links to other smart distributors on git-hub and youtube.

External Loader

How to write code to read and write an external Flash connected via SPI (later for advanced speed with Quad-SPI in single dual or quad mode). The ext. flash is supposed to store new firmware which the bootloader has to load to internal flash. There is a possibility to map external flash to STM32's internal flash address space. Maybe this could become handy. Refer to reference manual section 12.1 "memory-mapped mode: the external Flash memory is mapped to the device address space and is seen by the system as if it was an internal memory".

Mauro De Vecchi - External loaders for Standard SPI flash memories

ControllersTech - W25Q FLASH Memory Part 8 How to create an External Loader

ControllersTech - W25Q FLASH Memory Part9 External Loader in SPI Mode

ControllersTech - W25Q FLASH Memory Series

ControllersTech - YT W25Q Flash using SPI

STM32 MOOC - QSPI External Loader Series

Bootloader

My very first attempt on Bootloader. Its basically straight forward to what Vikto Vano discrips on youtube (link below). There are three small stand allone projects

- Bootloader
- App1
- App2

All three projects have to be build on its own and flashed to the MCU. Study the xxx_flash.ld file which gives Info on where the code will be placed on the internal flash. After Reset (Black Nucleo Button) the Bootloader prints info on the terminal on UART2 (Nucleo USB). Make a selection on which partition to boot from and press the Blue button.

Pin PC0 (Nucleo A5) and Pin PC1 (Nucleo A4) are configured as GPIO Input with Pull Down. Use a Jumper Wire to simulate switches for these two pins. The bootloader will loop until one presses the blue button (Nucleo B1). According to state of switches the bootloader decides where to jump to. TIM6 is not used it is initialized just for demo to make sure it is working on Application after bootloader finished its execution.

On application Project, the Interrupt Vector Table has to be configured in system_stm32f4xx.c at line 94 and 108. Vector Table Offset must match Linker file!

Good easy going Tutorial

Viktor Vano - STM32 Bootloader using Blue Pill Board

Viktor Vano's github

General Understanding of STM32 Bootloader Theory

STM32 MOOC - How to Create a Super Simple Bootloader

Still to watch these ones

Worlds Simplest Bootloader :: Bare Metal Programming Series 4

STM32F7 (Cortex M7) Bootloader Tutorial Part 1 & 2 - Bootloader Introduction and Design for STM32

Good To Know

How Microcontroller Memory Works | Embedded System Project Series #16

STM32 MOOC - Boot and Startup

Firmware-Update OTA

Firmware Update over the air combines all the features discused so far. The binaries have to be transmitted throught e.g. wifi and stored in the flash. After, the device has to force reboot and bootloader does its job.

Git Commands

how to use multiple git accounts

how to show git config

Regular workflow using command line tool

`git add .`

`git commit -m "whats new - my notes on this commit"`

`git push`

Show basic informations

`git status`

`gitk`

Merge branches

`git checkout destination-branch`

`git pull`

`git merge branch-that-has-newes-features`

`git push`

`git checkout back-to-working-branch`

Undo changes

`git restore`