Operating System Lab Assignment 02: Design and Implement Boot Loader Program for DUOS

Dr. Mosaddek Tushar, Professor
Computer Science and Engineering, University of Dhaka,
Draft Version 1.0 – look for update
Duration: two weeks

October 29, 2024

Contents

1 Introduction

Maintaining and updating firmware is crucial for enhancing functionality, addressing bugs, and strengthening security in embedded systems. This assignment centers on developing a custom bootloader for the STM32F4xx microcontroller series, serving as the gateway for downloading and updating firmware for the DUOS operating system. The STM32F4xx microcontrollers feature advanced capabilities that support a variety of communication interfaces and memory management techniques. A bootloader is a compact software component that executes during startup, enabling the loading of application code from external sources. In this assignment, students will gain

experience creating a robust bootloader capable of managing firmware updates, ensuring that the DUOS application remains up-to-date and functional. Participants will explore key concepts such as bootloader architecture, flash memory management, and communication protocols. The assignment will require the implementation of functions for firmware reception, validation, and secure execution, providing a comprehensive understanding of the processes involved in embedded firmware management.

1.1 Learning Outcomes

By completing this assignment, students will gain hands-on experience in embedded systems development, understand the complexities of firmware updates, and learn how to implement reliable communication protocols. This project will enhance their skills in managing microcontroller firmware, preparing them for real-world applications in device maintenance and updates.

2 Objectives and Policies

2.1 General Objectives

The objectives of the Bootloader lab assignment are to gain hands-on experience in solving complex engineering challenges, particularly in version checking and uploading DUOS using customized frame transfer protocols. Additionally, this assignment aims to develop a bootloader for an ARM processor-based system. A key focus is utilizing the UART protocol to communicate between the server program and the bootloader. The assignment includes identifying practical solutions for significant issues related to the bootloader program and its associated protocols, such as:

- Develop a robust and error-free server program to facilitate communication and upload the DUOS to the STM32F4xx microcontroller.
- Create a bootloader client application for the STM32F4xx, enabling the seamless downloading of updated DUOS.
- Acquire a deeper understanding of the essential architecture and concepts involved in upgrading the microcontroller, as well as the operating system and firmware of microcomputers.
- The development process will empower system users to utilize various ICT communication technologies for firmware upgrades.
- Understand the memory mapping associated with the process section of each system program.
- Recognize the challenges of employing CRC algorithms for data integrity in communication.

2.2 Advanced Objective

The advanced objective is to leverage diverse wireless communication technologies, including wireless and GSM protocols, to download the latest version of the DUOS.

- Understanding the Transformative Impact of Communication Technologies in Modern Data Retrieval: This assignment examines how advanced communication technologies, such as wireless and GSM protocols, facilitate access to and updates for the DUOS.

- Challenges and Solutions in Implementing Wireless Communication Protocols: This assignment will explore the technical and practical challenges of integrating wireless communication protocols into existing systems. Furthermore, it will propose solutions and best practices for seamless integration, emphasizing real-world examples that resonate with students.

2.3 Assessment Policy

The assignment has three level objectives (i) primary objectives, (ii) advanced objectives, and (iii) optional boost objectives. Every student must complete the general objective; however, they can attempt advanced and optional Boost-up objectives. The advanced objective will be a primary objective in the subsequent assignment. Further, you can achieve five (5) marks for completing the optional objectives and add these marks at the end of the semester with your total lab marks. However, you cannot get more than 100% assigned for the labs.

3 Description

This assignment entails designing and implementing a custom bootloader tailored for the STM32F4xx microcontroller series, specifically to manage firmware updates for the DUOS operating system. This bootloader will facilitate secure and efficient downloading of updated firmware versions, thereby enhancing the embedded system's overall functionality and performance. The critical components of the assignment are:

- Bootloader Schematic
  - Analyze the foundational structure and primary function of a bootloader.
  - Investigate the built-in bootloader capabilities of the STM32F4xx and explore how to integrate them into a custom solution.
- System Serial Communication and Initialization
  - Configure the microcontroller's system clock and essential peripherals (e.g., USART for serial communication and WiFi modules).
  - Set up GPIO pins to serve as status indicators or communication interfaces.
- Communication Protocol for Binary file transfer
  - Design a robust server-side application using a suitable programming language to establish a reliable communication protocol between the bootloader and host systems (such as PCs or other devices).
  - Implement command structures for initiating firmware updates, transmitting firmware data in multi-frames, and acknowledging successful data transfers.
- Firmware Download and Validation
  - Develop functions for the reception of firmware data in packetized formats.
  - Implement error detection mechanisms, such as CRC or checksums, to ensure the integrity and accuracy of the received firmware.
- Flash Memory Management Access Control

  - Create routines for erasing specific flash memory sectors designated for DUOS firmware.
  - Implement safe programming routines to write the incoming firmware data to flash memory while adhering to STM32 flash programming guidelines to preempt corruption.

- DUOS Binary Transfer
  - Design a robust mechanism to transfer control from the bootloader to the newly updated DUOS Operating System.
  - Configure the application's system stack and vector table to ensure correct and seamless execution.

- Testing and Error Handling
  - Develop comprehensive test cases (with Major, Minor versions) to validate bootloader functionality and reliability across various scenarios, including power loss and communication interruptions.
  - Implement error-handling routines capable of recovering from failed updates or instances of data corruption.

- Documentation
  - Maintain thorough documentation of the bootloader's architecture and design, including well-commented code, system architecture diagrams, and a user manual for firmware flashing.
  - Document testing procedures and outcomes to showcase the bootloader's robustness and reliability through empirical data.

3.1 Bootloader Linker

This assignment involved understanding the mapping of linker memory (Flash and RAM) and various program sections, including the ISR vector table, text, data, bss, and others. The system's (MCU) default bootloader downloads your customized bootloader and executes the reset handler and the main function. The main function is to verify that the major and minor versions of the DUOS are present on the computer communicating with the terminal server program. The customized bootloader unlocks the flash memory to enable writing if the program detects outdated version numbers. Subsequently, the bootloader sends a request to the terminal server program to initiate the transfer. The terminal server first sends the size of the DUOS binary before starting frame transmission. Each frame must include the size and CRC-32 code in its header, followed by a payload. Once a frame is successfully received, the bootloader begins writing the payload to the designated location as assigned by the DUOS linker. After completing writing all the frames, the bootloader modifies the MSP, updates the ISR vector's start point in the DUOS flash memory, major and minor version numbers, and jumps (Function CALL) to the DUOS reset function. You can exploit and modify VTOR register with a $SRAM\_BASE|OFFSET$ or $FRASH\_BASE|OFFSET$ to redirect the ISR vector table in cm4.h.

3.2 Relocation of ISR Vector Table

```
typedef struct __scb_t
 {
        volatile uint32_t CPUID;         // CPUID Base Register 0x0
        volatile uint32_t ICSR;                // Interrupt Control and State Register 0x4
        volatile uint32_t VTOR;                // Vector Table Offset Register 0x8
        volatile uint32_t AIRCR;        // Application Interrupt and Reset Control Register 0xC
        volatile uint32_t SCR;                // System Control Register 0x10
        volatile uint32_t CCR;                // Configuration and Control Register 0x14
        volatile uint8_t SHP[12U];         // Exception priority setting for system exceptions
        volatile uint32_t SHCSR;        // System Handler Control and State Register 0x24
        volatile uint32_t CFSR;                // Configurable Fault Status Register combined of MemManage,
        // Fault Status Register, BusFault Status Register, UsageFault Status Register 0x28
        volatile uint32_t HFSR;                // HardFault Status Register 0x2C
        volatile uint32_t DFSR;                // Hint information for causes of debug events
        volatile uint32_t MMFAR;        // MemManage Fault Address Register 0x34
        volatile uint32_t BFAR;                // BusFault Address Register 0x38
        volatile uint32_t AFSR;                // Auxiliary Fault Status Register 0x3C: Information
        //for device-specific fault status
        volatile uint32_t PFR[2U];         // Read only information on available processor features
        volatile uint32_t DFR;                // Debug Feature: Read only information on available debug features
        volatile uint32_t AFR;                // Auxiliary Feature: Read only information on available auxiliary features
        volatile uint32_t MMFR[4U];         // Memory Model        Feature Registers: Read only information
        volatile uint32_t ISAR[5U];         // Instruction Set Attributes Register: Register Read only information
        uint32_t RESERVED1[5];                // Now it is reserved: Unknown
        volatile uint32_t CPACR;         // Coprocessor access control register 0x88
} SCB_TypeDef;
```

3.3 Writing Flash Memory

For flash memory writing please read chapter 3 of the STM32F446re reference manual. The registers in the FLASH memories are:

```
typedef struct __flash_t
{
        volatile uint32_t ACR;
        volatile uint32_t KEYR;
        volatile uint32_t OPTKEYR;
        volatile uint32_t SR;
        volatile uint32_t CR;
        volatile uint32_t OPTCR;
}FLASH_TypeDef;
```

To unlock the flash memory, insert the access code into the KEYR and OPTKEYR registers. The sequence for entering KEY1 and KEY2 will unlock the FLASH_CR register, enabling you to program or erase it (refer to Section 3.5). Similarly, OPTKEY1 and OPTKEY2 are used to unlock the FLASH_OPTCR for programming purposes. It is important to review the FLASH_CR and FLASH_OPTCR registers thoroughly. The flash and SRAM memory are organized into several sectors, as detailed in Table 4 of Chapter 3. Once programming is complete, be sure to lock the flash memory.

3.4 References – recommended videos

Videos	Link
STM32F7 (ARM Cortex M7) Bootloader Tutorial Part 1-7 - Writing Simple STM32 Bootloader	`https://www.youtube.com/playlist?list=PLArwqFvBIlwHRgPtsQAhgZavlp42qpkiG`
Blinky To Bootloader: Bare Metal	`https://www.youtube.com/playlist?list=PLP29wDx6QmW7HaCrRydOnxcy8QmW0SNdQ`
stm32 bootloader	`https://www.youtube.com/playlist?list=PLM7yYW7w7MWms-Um-dHxbftXRUFT74u9l`

4 What to do?

- Upload bootloder src to classroom
- Terminal Server Program
- Frame formate and communication protocol
- DUOS patches for test cases
- Documentation (PDF) – Generated from Latex src to pdf

5 Deliverables

- A complete and fully functional bootloader codebase developed in C/C++ and designed for compatibility with the STM32F4xx series.
- Comprehensive documentation outlining the bootloader's architecture, communication protocol, and user operational guidelines.
- A demonstration of the bootloader's capabilities through a series of test cases, illustrating its effectiveness in managing firmware updates for DUOS.
- The bootloader must store the recently downloaded DUOS Major and Minor version number.
- A function get˙bootloader() – returns current version number like major.minor like 1.02, for major 1 and minor 02 version numbers

6 Rubrics

The student achieves 60 points by completing the assignment with the following distribution.

- Terminal Server 10 Points
- Bootloader 13 points
- Test Cases – at least 4 for each version of DUOS – 12 points
- Team work – 15 points
- documenttation – 10 points

Alert: You can discuss with your classmates, however, do not copy code from others.