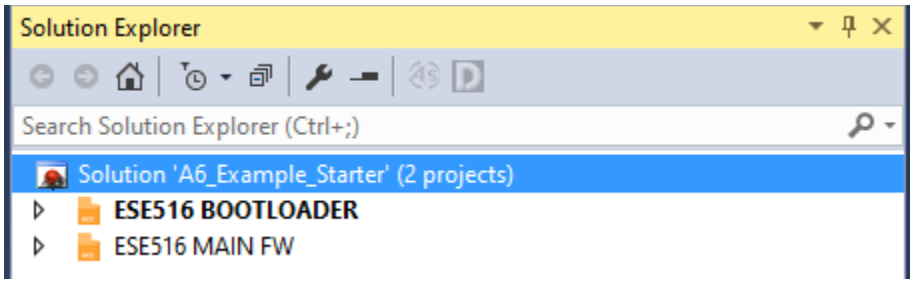
A9 Assignment – ESE516-SPRING 2022

DUE DATE: Monday April 1 2022 before 11:59pm EST (By almost midnight). To be submitted on Google Drive of Team Folder on a folder called A9. GROUP PROJECT

Remember: Please submit your complete Atmel Project on Google Drive. The project must be complete (must have everything so we can compile it!).

1. INTRODUCTION[0 points, but helps with understanding]

We will use the starter code through the rest of the semester. When you open the starter code workspace, you will be greeted by two projects : They are the bootloader and the main application:



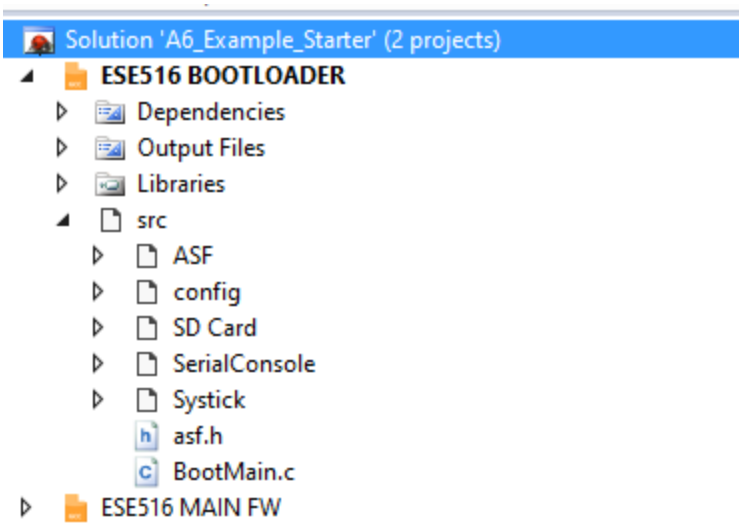
- ESE516 Bootloader: Bootloader code. It is a bare metal implementation and resides in address 0x0.

- ESE516 Main FW: Will be our main FW. We will not use this project for now (all your code will be on the bootloader).

To select the bootloader project, right click on it and select "Set as Startup Project". This will select the project and put the letters in bold, as seen above.

Expand the project. You will already have libraries that implement the following:

- SD Card: Initialization code for the SD Card
- SerialConsole: UART Implementation for CLI
- Systick: Method to measure time lapses
- Bootmain.c: Main bootloader code.



Open Bootmain.c. There, you will have an example code in the main() function that will do the following:

1. INIT SYSTEM PERIPHERALS INITIALIZATION
2. STARTS SIMPLE SD CARD MOUNTING AND TEST
3. STARTS BOOTLOADER HERE (Incomplete section – this is where you can do your code!)
4. DEINITIALIZE HW AND JUMP TO MAIN APPLICATION!

You should develop your code mainly in sections 3 and 4 of the code. Check for the comments for where these sections are!

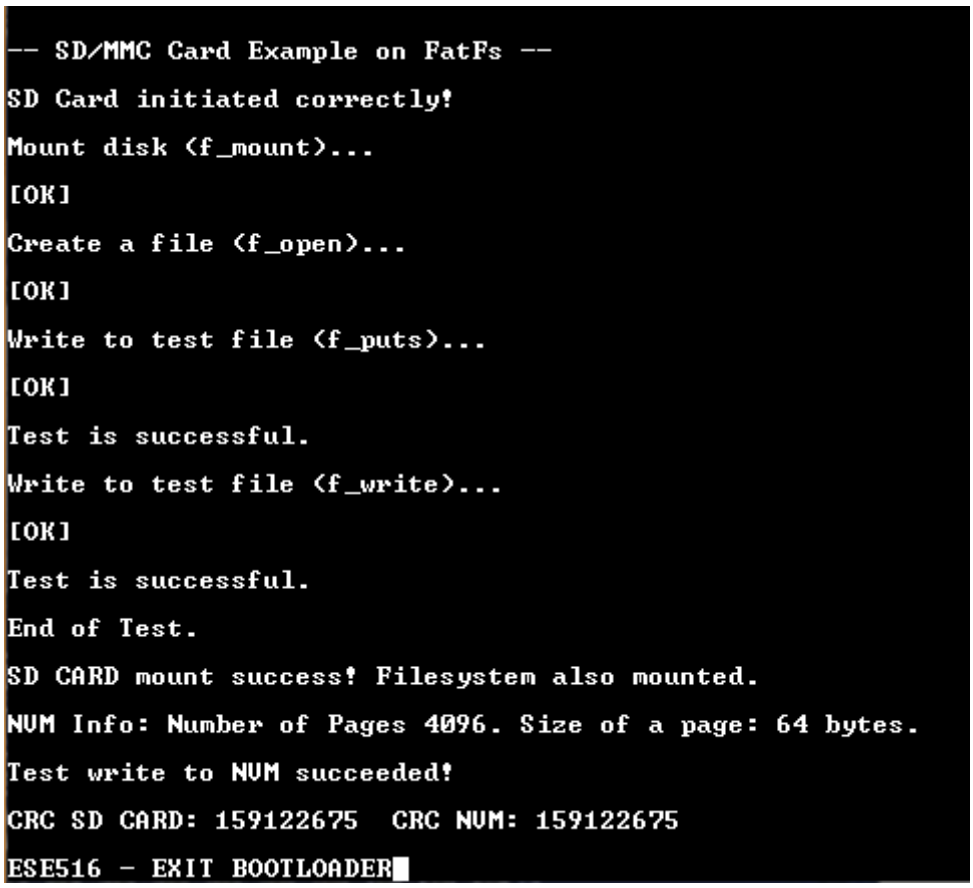
The main code has a toy example that showcases how to use the FATFS and NVM API. **This code is put on section 3. Please use this code as a basis for your work, but remember to erase it at the end! You can use the flag definition "MEM_EXAMPLE" to tell Atmel Studio to not compile this section: Just comment the line "#define MEM_EXAMPLE 1" in the Include Section.**

The toy example reads 256 bytes from the SD card, erases a row from the device, writes the 256 read bytes into NVM at address 0x12000, and compares them using the CRC.

Wiring the Device and Testing

Please follow the file "**ESE516_ONLINE_BUILD_SPRING2022_A9.pptx**" on the A9 folder on how to wire the system! Please be very careful following the directions as failure to do so can mean your components can be damaged!

Once you have the connection, feel free to try out the test program! Open Teraterm, add the micro sd card to the PMOD SD Card, program the program into your device, and read the output of the program on the terminal. You should get the following output if everything went ok!



2. BOOTLOADER[200 points]

Please submit a word document with the answers to these questions, and submit a zip file of your bootloader project to your group's Google Drive folder on a folder called "A8". Submit a word file with the answers to the following questions, which will aid you on designing your bootloader before you implement it.

1. Please provide a flow chart of your bootloader implementation. Try to be as detailed as possible!

2. Implement a bootloader as seen in class! To do so, please follow the following assumptions:

- To help you test, we will give you two binary files that we can use as our firmware to Test! The are called "TestA.bin" and "TestB.bin".
- You can add these two files to your SD CARD.
- What they will do:
    - Test A: Test A FW will flash the LED every 500ms. When the SWO button is pressed, it will write a flag to the SD card called "TestB.txt" and reset. This flag can be used by the bootloader as a flag to load the TEST B FW
    - Test B: Test B FW will flash the LED every 100ms. When the SWO button is pressed, it will write a flag to the SD card called "TestA.txt" and reset. This flag can be used by the bootloader as a flag to load the TEST A FW
- With this test binaries, make your design do the following:
    - If the bootloader sees the "TestA.txt" flag, order it to update the Firmware region with the Test A binary
    - If the bootloader sees the "TestB.txt" flag, order it to update the Firmware region with the Test B binary

A correctly implemented bootloader with there two programs loaded in the SD card would switch between the TestA FW and the TestB FW every time the button SWO is pressed!

Implement the bootloader as seen in class!

- **Before coding, organize your code.** Don't write a huge function inside main. Instead, consider dividing the tasks into subfunctions – possibly residing in another file from main. Think how the bootloader tasks can be divided into subtasks that benefit from being their own function.

- **Print debug information.** Use the terminal to print bootloader information along the way, printing what it is doing each step of the process. That way you can know the device is working as intended and can make debugging easier.

A9 RUBRIC

Points	Rubric
-200 points	If the project is incomplete or has missing files, it will incur this penalty.
50 points	Bootloader Design Document: Document has clear answers to the questions asked, clear diagrams for the questions that asked them, and fully explains the bootloader design to be implemented for a sound bootloader design.
100 points	Bootloader code implements a correct bootloader. If the binary test files TestA.bin and TestB.bin are loaded into the SD Card, your bootloader code effectively updates the binaries as described earlier at the touch of a button.
50 points	Code has sufficient comments (functions have comment headers in Doxygen format style). **Tip:** See how the functions on SerialConsole.c are commented at the start of the function.