ECED4406 – Lab #1

For Lab #1, you will analyze a hex file that I have provided. You will need to submit the answers to the questions in the "Questions to Answer" section.

Step 1: Ghidra Setup & Testing

1. Follow the Ghidra Installation instructions:
   a. At https://www.youtube.com/watch?v=sNPFzVOS52Y (Windows)
   b. For Mac or Linux, see Ghidra installation instructions at https://ghidra-sre.org/

2. Follow along from the video to load the example .hex file & confirm your install is working. See https://github.com/colinoflynn/eced4406/blob/master/GhidraSetupIntro/Ghidra%20Setup.pdf for slides showing each step as well

If your screen looks (roughly) like the video, you are good!

Close that project (we will use the same file, but with additional steps before we analyze it).

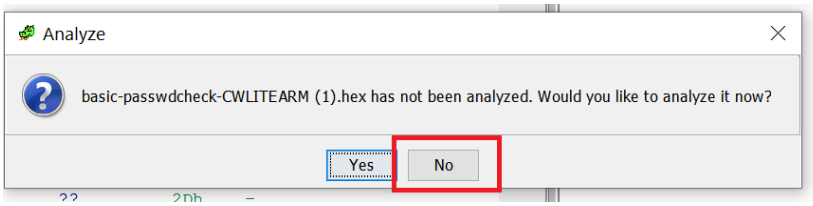
Step 2: Ghidra SVD Loader Setup

We will use Thomas Roth's SVD Loader, detailed at https://leveldown.de/blog/svd-loader/

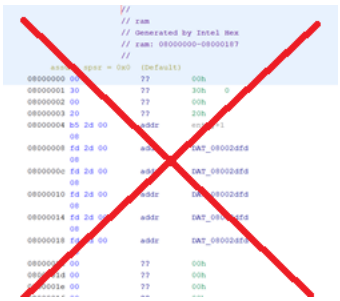
To install/use this:

Video version of these instructions: https://www.youtube.com/watch?v=J25HxGlBvSE

1. Download https://github.com/leveldown-security/SVD-Loader-Ghidra/archive/master.zip & extract somewhere.
2. Open the .hex file to analyze as a new project, but **do not click analyze yet**.

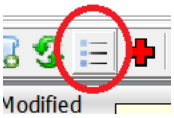


If your file already has annotations on lines 0800008 etc, you have analyzed it! Ensure the file does not look like this:

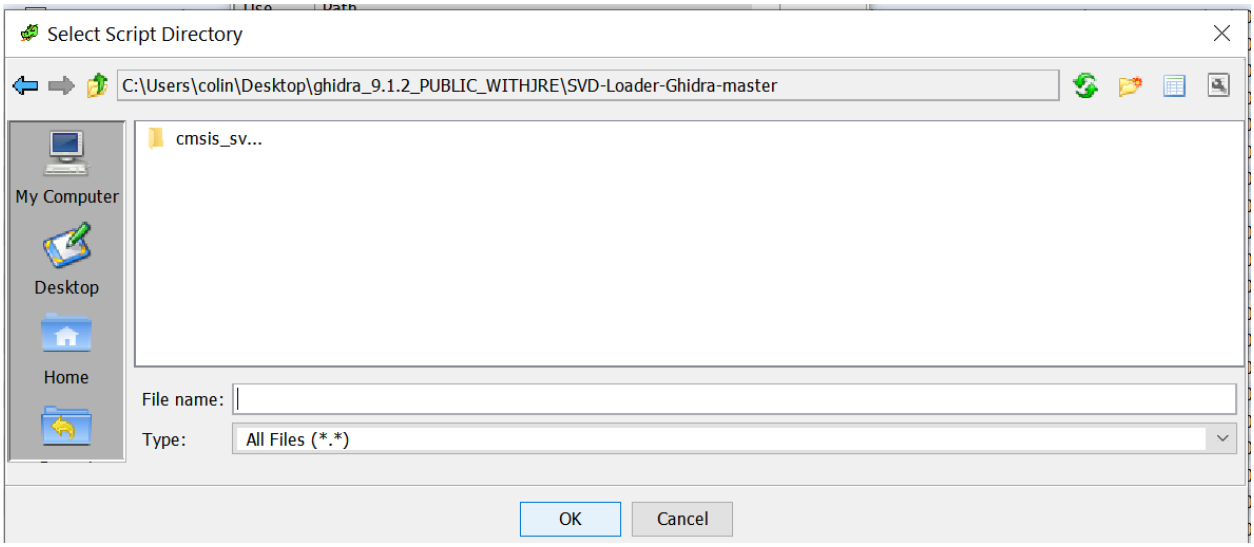


3. Press the “Display Script Manager” button:

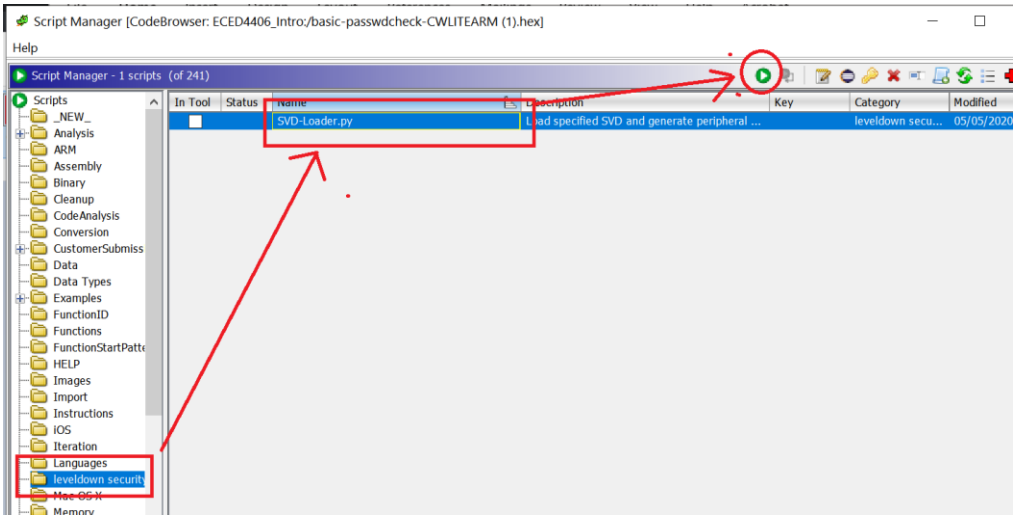
4. Press the “Script Directories” button:



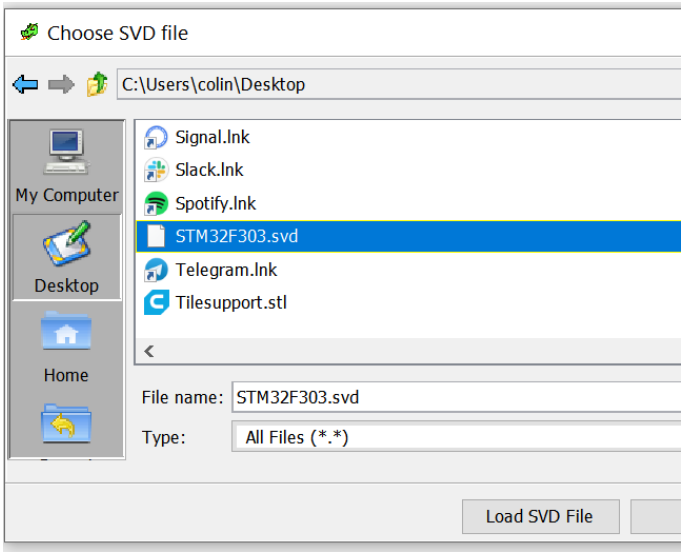
5. Press “Add”, then select the “SVD-Loader-Ghidra-Master” directory you extracted, and hit OK:



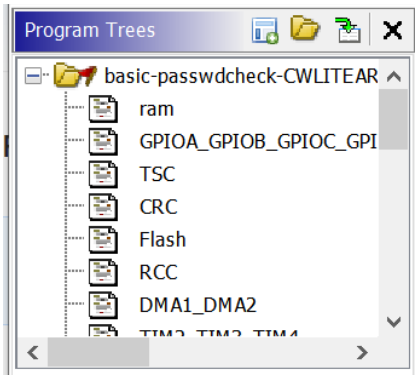
6. Close the directory list.
7. You should find “Leveldown Security” now – run the SVD loader script.



8. Download the SVD file from https://github.com/colinoflynn/eced4406/blob/master/GhidraSetupIntro/STM32F303.svd by Right-clicking the download button and hitting “Save File As” (also on Brightspace).
9. Load the SVD file



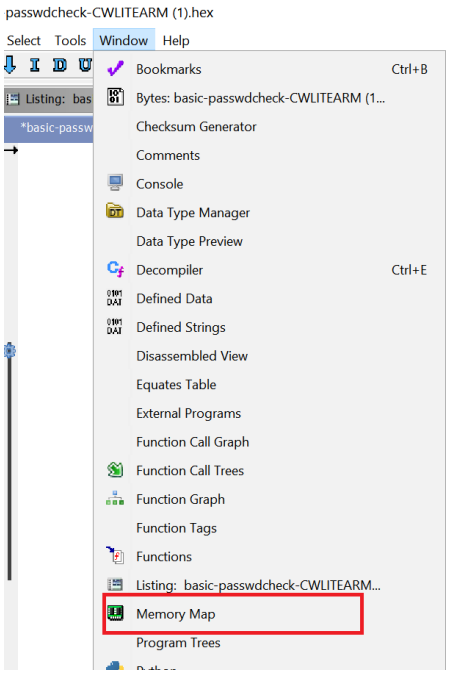
10. Close the script viewer. Once the script runs you should see the following on the “Program Tree”



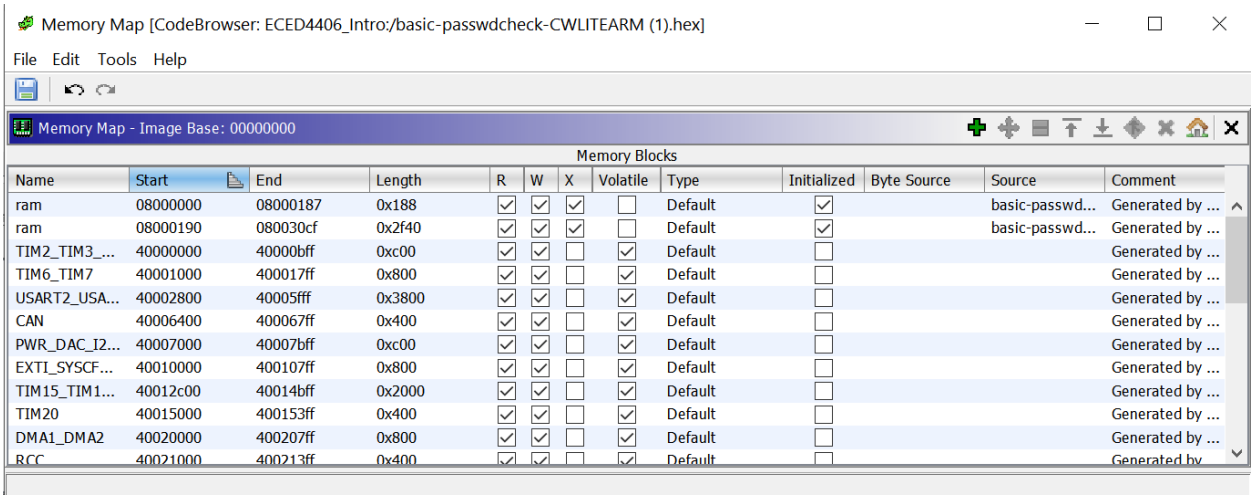
Keep this window open – we will continue to add data to our Ghidra install.

Step 3: Ghidra Memory Map

1. Under “Window” select “Memory Map”:



2. You should see the various peripherals already defined:



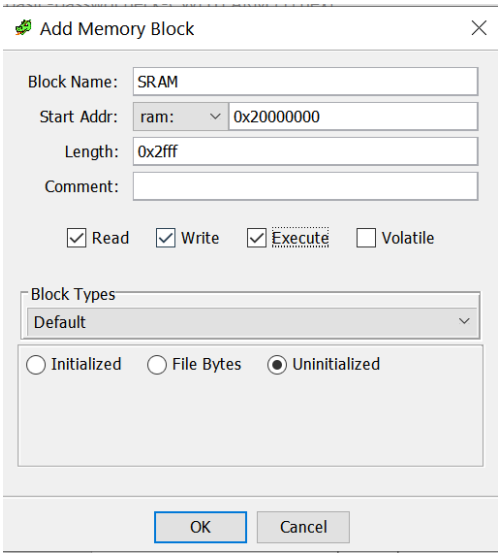
Name	Start	End	Length	R	W	X	Volatile	Type	Initialized	Byte Source	Source	Comment
ram	08000000	08000187	0x188	☑	☑	☑	☐	Default	☑		basic-passwd...	Generated by ...
ram	08000190	080030cf	0x2f40	☑	☑	☑	☐	Default	☑		basic-passwd...	Generated by ...
TIM2_TIM3_...	40000000	40000bff	0xc00	☑	☑	☐	☑	Default	☐			Generated by ...
TIM6_TIM7	40001000	400017ff	0x800	☑	☑	☐	☑	Default	☐			Generated by ...
USART2_USA...	40002800	40005fff	0x3800	☑	☑	☐	☑	Default	☐			Generated by ...
CAN	40006400	400067ff	0x400	☑	☑	☐	☑	Default	☐			Generated by ...
PWR_DAC_I2...	40007000	40007bff	0xc00	☑	☑	☐	☑	Default	☐			Generated by ...
EXTI_SYSCF...	40010000	400107ff	0x800	☑	☑	☐	☑	Default	☐			Generated by ...
TIM15_TIM1...	40012c00	40014bff	0x2000	☑	☑	☐	☑	Default	☐			Generated by ...
TIM20	40015000	400153ff	0x400	☑	☑	☐	☑	Default	☐			Generated by ...
DMA1_DMA2	40020000	400207ff	0x800	☑	☑	☐	☑	Default	☐			Generated by ...
RCC	40021000	400213ff	0x400	☑	☑	☐	☑	Default	☐			Generated by ...

We need to add the SRAM segments onto this.

3. Flip to page 59 of the reference manual to find this part of the memory map table:

	0x4000 0000 - 0x4000 03FF	1 K	TIM2	-
	0x2000 3000 - 3FFF FFFF	~512 M	Reserved	
	0x2000 0000 - 0x2000 2FFF	12 K	SRAM	-
	0x1FFF F800 - 0x1FFF FFFF	2 K	Option bytes	-
	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory	-
	0x1000 1000 - 0x1FFF D7FF	~256 M	Reserved	
	0x1000 0000 - 0x1000 0FFF	4 K	CCM SRAM	-
	0x0804 0000 - 0x0FFF FFFF	~128 M	Reserved	
	0x0800 0000 - 0x0800 FFFF	64 K	Main Flash memory	-

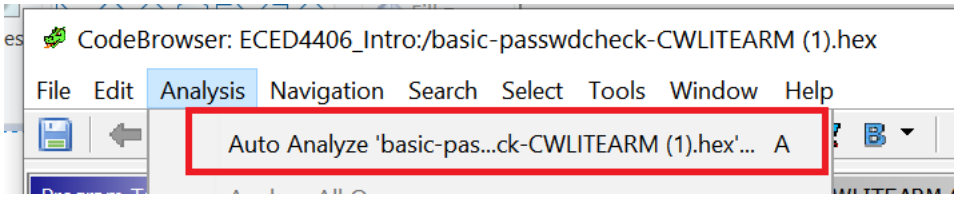
4. Add a new memory segment at address 0x2000000 that is of length 0x2FFF, with "Read, Write, Execute" permission. Call it SRAM (watch the correct number of 0's in the hex!):



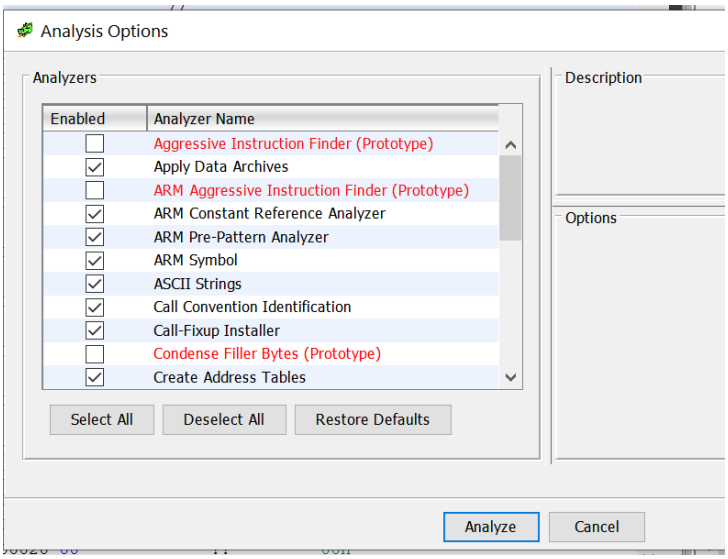
You are now ready to analyze the code.

Step 4: Auto Analyze

1. From the Analysis menu, select "Auto Analyze"



2. Accept the defaults, press "Analyze":



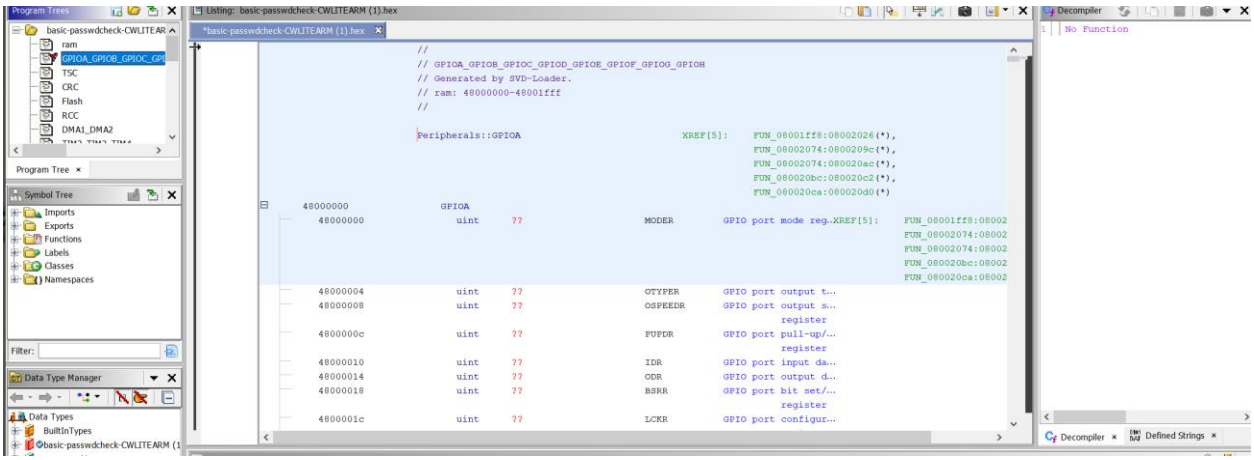
3. You should now be looking at a ready to roll system!

Step 5: Exploration

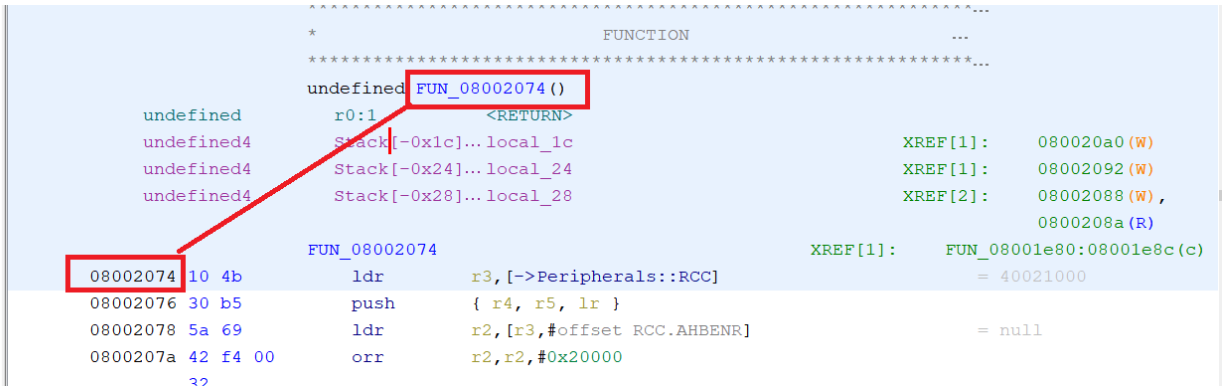
You can now use Ghidra to explore this binary. Some features to work with (see video initially):

- String view
- Use cross-references to find where functions are used
- Reference peripherals

Because you have the peripheral file loaded, you can do stuff like click on GPIOA on the right-side, and see all references to the functions:



In the following questions, we ask what the address of certain functions is. The address is shown to the left of the disassembly, and is shown in the function name normally too:



Questions to Answer [30 pts]

1. What function (i.e., what address) appears to configure GPIOA for use? In your lab report, include a snipped of the function and annotate what it is doing. Note there are multiple configuration locations, so you may have different answers for this. [10 pts]

2. List all the defined strings in this program [5 pts]

3. Does any peripheral in the code use the ADC1 or ADC2 peripheral? If so what is the address [2 pts]

4. What does the function at address **08001e1c** do? Include a short annotation of the decompiled C code. [3 pts]

5. For the password comparison logic: [10 pts]

    a. Where is the password that is entered compared with the stored password? Include a short annotation of the password comparison logic. [8 pts]
    b. What is the password? [1 pts]
    c. Where is the password stored? [1 pts]
    HINT: The password is stored as a string – but may not be identified as one!