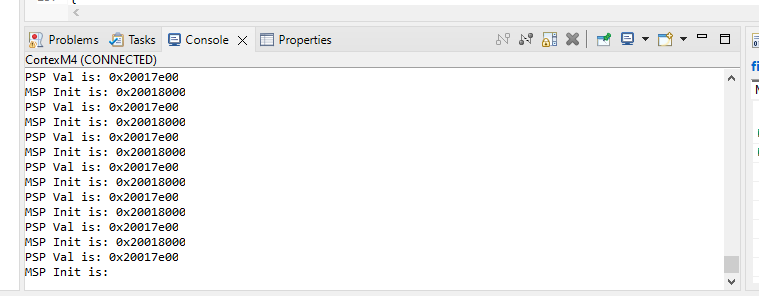
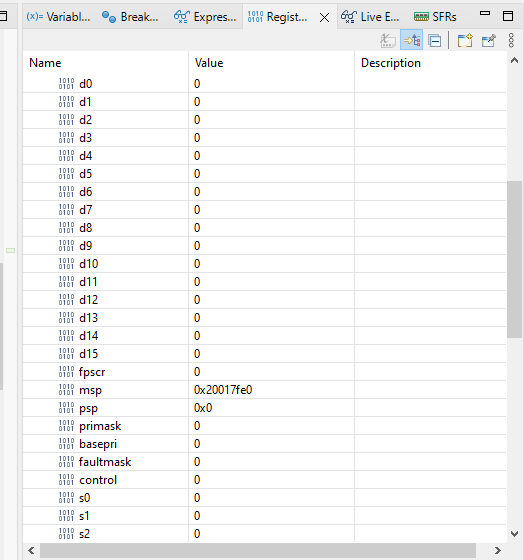
Lab 1 Reflection

# Proof



# Question 1

In your own words, explain what the HAL is and why it is used in a general embedded project.

## Answer:

HAL is the Hardware Abstraction Layer that has been written by ST Microelectronics which is a software library provided to simplify the interaction with the hardware peripherals. It becomes easier for the developers not to deal with the low-level hardware details by providing this high-level interface. IDE will generate or include the HAL code whenever we set up a project for the microcontroller simplifying the initial setup for hardware interactions.

# Question 2

In your own words, explain what the \_\_io\_putchar function does and why it is needed.

## Answer:

\_\_io\_putchar is a wrapper function with the purpose of writing a single character to the appropriate IO channel (in this case, UART2). This function is needed in embedded systems so that you can use standard C library functions like printf for printing strings easily. It is a crucial part of setting up and configuring serial communication for microcontroller-based applications.

# Question 3

Why did we have to use the debugger in this lab? Why didn’t we just use printf to instrument our code and make sure it worked?

## Answer:

The debugger allows us to examine the program’s execution as it runs, specifically the registers, which printf can’t do. We can set a breakpoint and inspect variables at runtime. With this real-time debugging, it will be useful for diagnosing complex problems and understanding program behavior.

# Question 4

Explain how the stack allocation method we used works. Start by assuming that we have obtained the value for MSP\_init\_val. How did we get a new value for PSP\_val? Why did we need to know the initial location of MSP for this to work?

## Answer:

This stack allocation works by taking the initial value of the Main Stack Pointer (MSP\_INIT\_VAL), subtracting an offset (0x200) to reserve space for the Process Stack Pointer (PSP\_val). Knowing the initial location of the MSP is important because it allows the code to determine where the MSP starts and allocate space below it for the PSP, ensuring that the two stacks do not overlap and that they have separate stack spaces.

* To get the MSP value:

As we know that address 0x0 stores the initial MSP value, we wanted to know what is exactly stored at the 0x0 and that’s why we created a double pointer where we set up the MSP\_INIT\_VAL as a pointer to a uint32\_t. 0x0 is now a pointer to that pointer. Then we dereference 0x0 once to get the address of the start of the MSP stack.

* To calculate the PSP value:

We calculated a ‘PSP\_val’ which represents the initial value for the PSP, the code subtracts ‘0x200’ from ‘MSP\_INIT\_VAL’. ‘MSP\_INIT\_VAL’ represents the initial address of the Main stack pointer. By subtracting ‘0x200’ from it, the code is effectively reserving 512 bytes at the bottom of the stack for the process stack. Separating them ensures that the main program can have its own stack space without interference from exceptions.

# Question 5

Say that we wanted to use the same stack allocation method to allocate a number of new stacks for use in multiple threads. Say this number is known at compile-time. What would we need to change in our code to make this work? Do not worry about actually using multiple stacks. This question is purely asking about how you would know where each stack starts in memory.

## Answer:

We start by calculating unique PSP values for each thread based on the initial MSP value and desired stack sizes. Then, reserve memory regions for each stack, configure the PSP and Control Register for each thread and implement a thread management system to handle the execution of multiple threads. Each thread should have its own dedicated stack space to ensure thread isolation and proper functioning of the multi-threaded application.

# Grading

Each question is evaluated based on the following scale.

**Meets expectations:** The answer is entirely correct. The answer is sufficiently detailed to include the major ideas necessary to prove that the student understands the concept. The answer is clear.

**Approaching expectations:** The answer contains no incorrect statements, but requires more detail or is too vague to convince the grader that the student understands the concept. The answer is clear.

**Does not meet expectations:** The answer either contains some incorrect statements or lacks detail to the degree that major portions of the question are left unanswered. The answer is not clear.

**Not observed:** the answer is largely incorrect or absent.

To get a grade for a question, you must obtain a rating of “meets expectations” on that question. You may resubmit as many times you like until you achieve that rating.