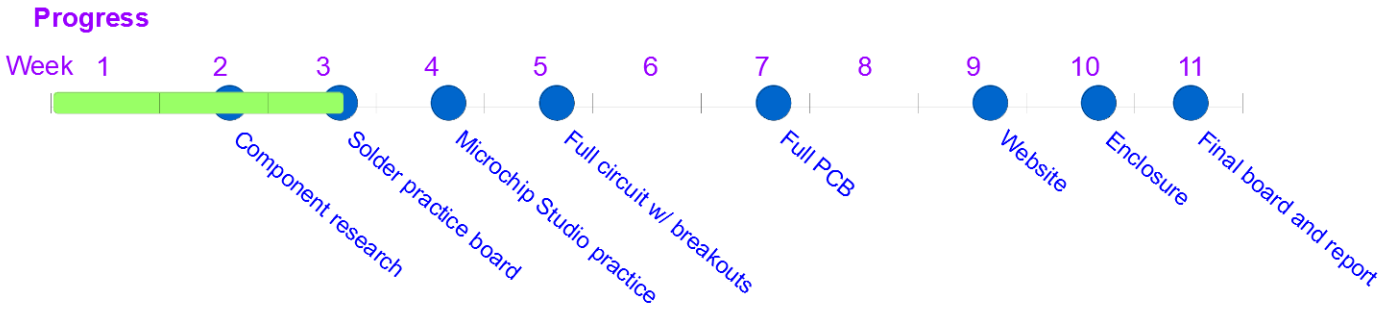
EE 326: Electronic System Design 1

Task 3: Microchip Studio Practice

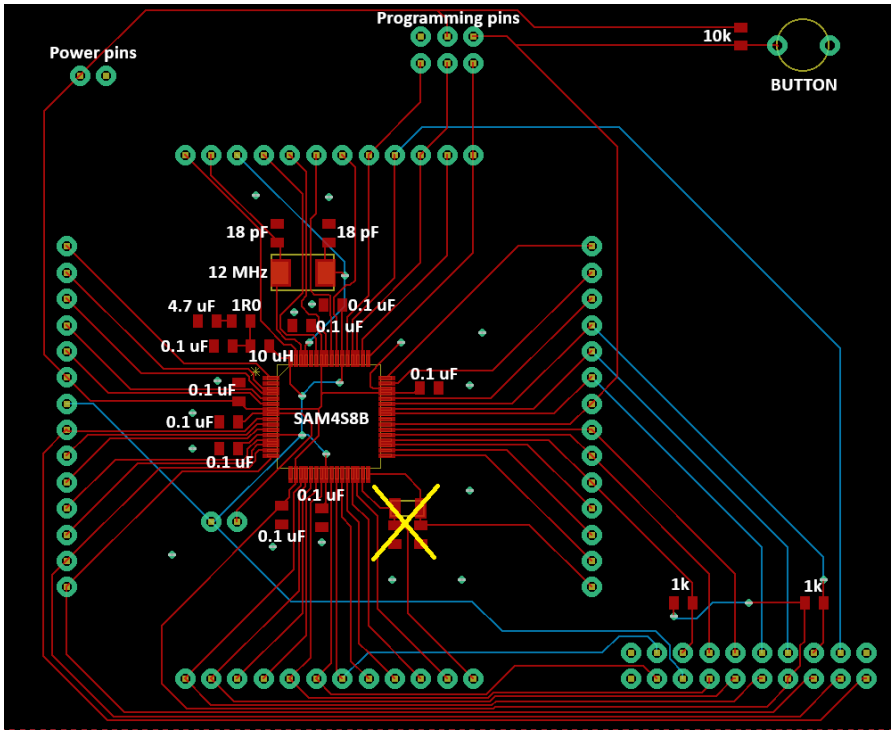
Due: Jan 30, 2025

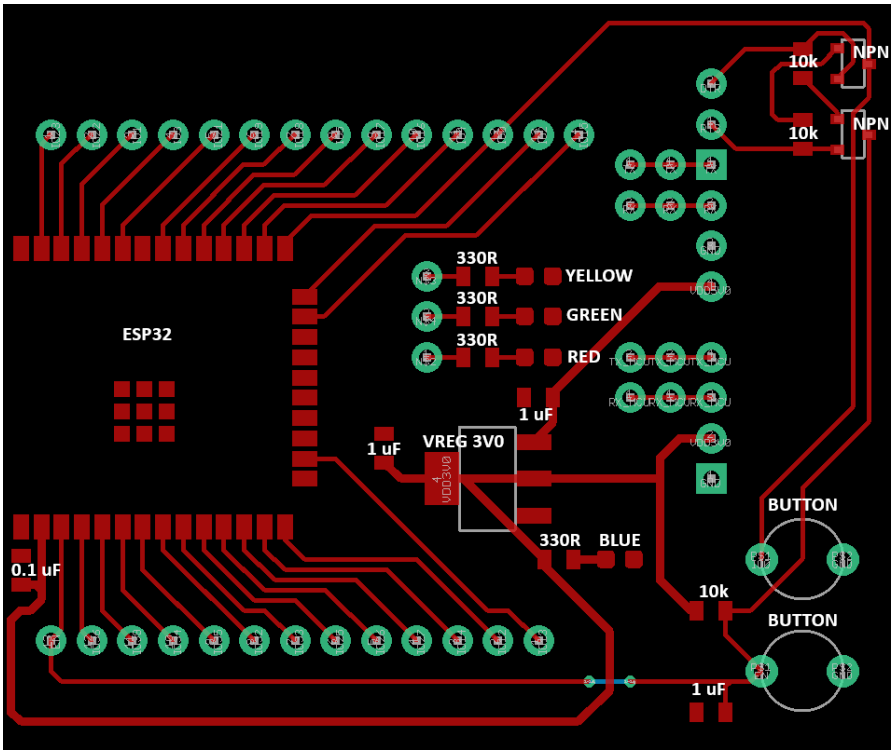


Please complete this assignment with your group. To get credit for your work, please complete the following 2 parts.
1) Upload your zipped project folder to Canvas. The project folder should contain the .atsln file, and a folder containing all the source and debug files.
2) Please come to either Ilya's, Yiming's, Enya's, or Shannon's office hours and show them your circuit with the interrupt-controlled button.

For this assignment, you will construct a circuit using your MCU breakout (as well as your WiFi breakout) and program it to light an LED when a button is pressed (and to turn it off when the button is released), using interrupts.

1. CONSTRUCTING THE BREAKOUTS
   For this step, you will construct the SAM4S8B (MCU) and ESP32 (WiFi) breakout boards using surface mount soldering.
   - Use the images to the right and below for component placement.
   - Use male headers for all header locations.
   - For the ESP32 board, soldering the NPN transistors and their corresponding 10k resistors is optional. If you include them, you will have an easier time programming your ESP32 board. If you don't it is also not a problem, especially because you will only program your ESP32 once or twice this quarter.

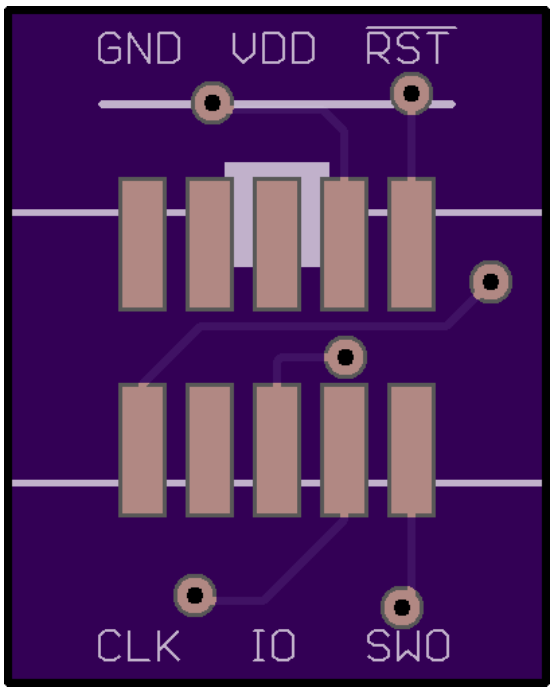




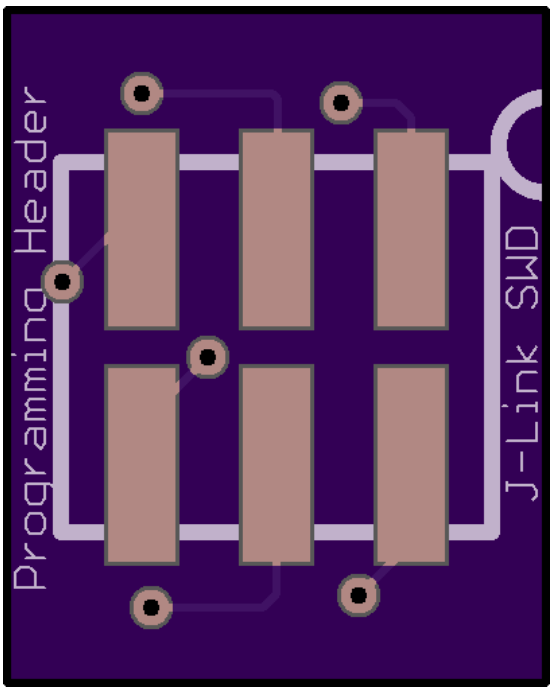
2. CONSTRUCTING THE PROGRAMMING HEADER

For this step, you will construct the programming interface header, which you will use to connect your J-Link programmer to the MCU breakout.

- Use the small interface PCB, the shrouded 10-pin connector, and the SMD 6-pin header.
- Make sure the opening in the shroud of the connector is facing toward the line on the PCB.
- *Do not use hot air aimed at the headers; this will melt them!*

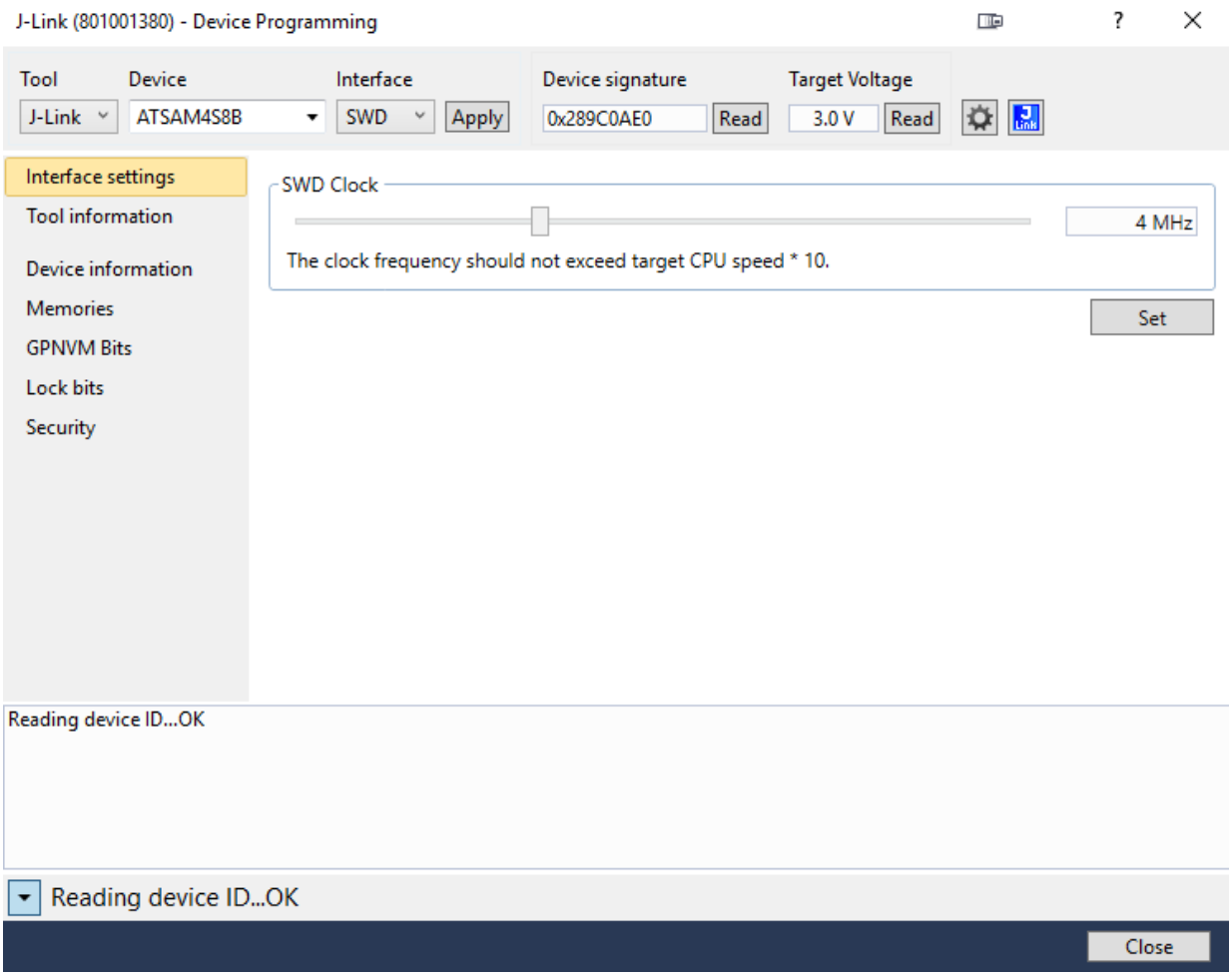


Top (10-pin, shrouded header)

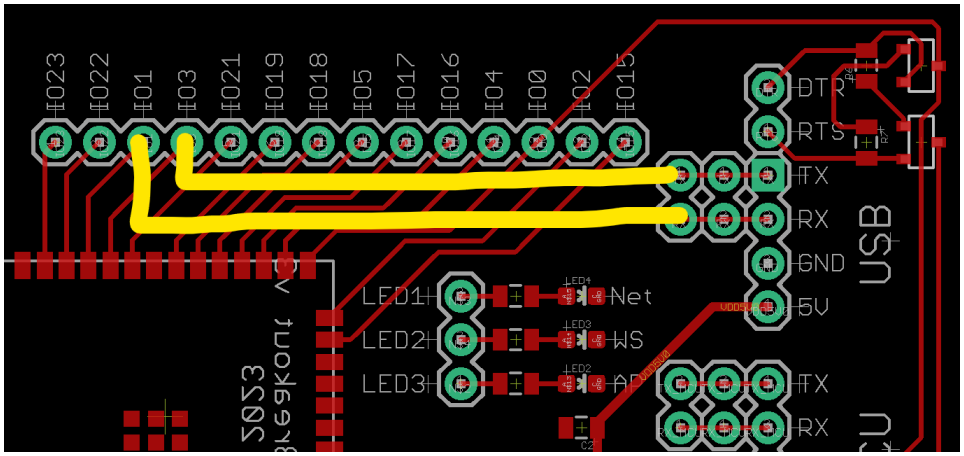


Bottom (6-pin, unshrouded header)

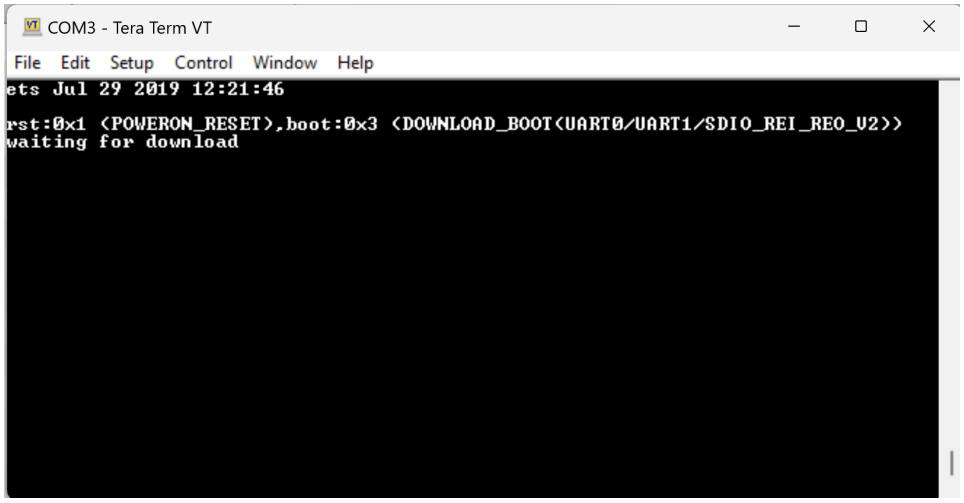
Before you can test the MCU board, you have to create a 3V supply. You will use the 3V regulator on the WiFi board. Using your USB-to-UART module, connect the 5V, GND, TX, and RX pins to the corresponding pins on the WiFi board labeled “USB”. Make sure that the “POWER” LED lights up. Now, connect the 3V and GND pins to the power pins on the MCU board. Also, on the MCU board, place a jumper on the two pins to the bottom left of the MCU. Now, connect the programming header. Be careful with orientation. Open Microchip Studio, and go to Tools > Device Programming. In the window that pops up, select “J-Link” as your Tool, “ATSAM4S8B” as your Device, and “SWD” as your Interface. Click “Apply”. Finally, under “Device signature”, click “Read”. If you get an ID and a voltage (see below for an example), then there is a good chance that you have soldered successfully, and you are ready to start programming. If not, there are probably some loose connections, or perhaps solder bridges causing short circuits. (Note: Just because you pass this test does not mean you have no soldering mistakes. If your code does not run, soldering may still be the issue.)



This is also a good time to test the WiFi board (to make sure the USB-to-UART connections are correct). To test your ESP32 board, make sure you still have your USB-to-UART module connected. Then, connect the pins right next to the TX and RX pins (which are connected directly to them), to pins IO1 and IO3 of the ESP32, where IO1 should go to your USB-to-UART module’s RX pin, and IO3 should go to its TX pin (see image below).



Then, set up a Terminal Application on your computer, such as Tera Term, CoolTerm, or PuTTY, and set the baud rate to 115200. On your ESP32 board, hold down the IO0 button and press and release the RESET button. Then, release the IO0 button. You should see "waiting for download" on the terminal (see below for an example). If that is what you see, then your connections are correct. (Note: Again, this is not a guarantee that all of your connections are correct, but rather that just the pins that communicate with your computer are connected correctly.)



3. PROGRAMMING THE MCU

For this part, you will use your completed breakout boards, along with a breadboard, an LED, and a pushbutton. The goal is to make the LED light up when the button is pressed, and to turn off when it is released. Both the press and release should be sensed by interrupts. You do not have to put your MCU to sleep between events. The steps below should get you started.

(a) Start a new project in Microchip Studio.
   i. Select "GCC C ASF Board Project".
   ii. Name your project.
   iii. Select the "ATSAM4S8B" device.

(b) After the project is created, the first screen you will see is the ASF Wizard. Include the following modules, which you should include with every project.
   - System Clock Control (service)
   - Delay routines (service)
   - GPIO - General purpose Input/Output (service)
   - IOPORT - General purpose I/O service (service)

(c) Open another instance of Microchip Studio, and click "New Example Project".
   i. Select the "SAM4S" device family, and press the dropdown to view all example projects.
   ii. Open the "Common IOPORT Service Example 3" project. This project shows good examples of simple GPIO routines, such as setting a GPIO high or reading a GPIO value, as well as initializing GPIOs to be either inputs or outputs.
   iii. Click File > New > Example Project. Open the "OV7740 CMOS image sensor example" project. This project shows an example of configuring a pin as an interrupt, and how to handle the interrupt.

(d) Explore the two sample projects.
   - Try to follow the logic of the examples by starting in the "main()" function, and making your way through.
   - Right click on variable names and functions, and click "Goto Implementation" to learn more and to see how the programs are organized.

(e) In your own project, create two new source files: button.h and button.c. You can follow the steps below to do this.
   i. Right click on "src", and select "Add > New Item".
   ii. Select "C File", and name it "button.c".
   iii. Right click on "src", and select "Add > New Item".
   iv. Select "Include File", and name it "button.h".

   These files should now appear next to your "main.c" file in the file tree.

(f) Based on the example projects you have open, copy and paste relevant pieces of code and modify them for our task as necessary.

Notes:
- Don't forget to include <asf.h> in every .h file.
- Don't forget to declare variables as "volatile" if they are changed in interrupt service routines.
- Try to spend as little time as possible in the interrupt service routines.
- It may be helpful to turn off compiler optimization for debugging.

FAQ

Q. Where do you turn off compiler optimizations?

A. Go to Project > Properties. On the left, select Toolchain. Under ARM/GNU C Compiler, select Optimization. In the Optimization Level dropdown, select None.

Q. What are GPIO pins in the board of MCU? It seems like MCU only has PIO pins. Is there any difference between these?

A. GPIO stands for "general purpose input/output". This basically means a pin can be used for general digital input or output. Some pins have reserved functionality, but most are available as GPIOs. For Microchip, they use "PIO" for "parallel input/output", because that is the name they gave to the hardware that handles these ports.

Q. What is the watchdog timer, and do I need to enable or disable it?

A. Use the "WDT" driver in the ASF Wizard and the function from the watchdog timer framework to disable it. The watchdog timer is a hardware level timer that can reset your MCU if it gets stuck in the

code for whatever reason. Basically, you must tell the timer that everything is running smoothly once per main loop. You set the timeout of the timer to be the worst-case execution time of your main loop. If it is not serviced in that time, that means your program is stuck somewhere for software or hardware reasons, and the watchdog timer resets the MCU. Alternatively, it can be configured to perform some other action in response to this condition. The watchdog timer is disabled during debugging automatically, since you can take your time while stepping through your program. However, when not debugging, it is enabled by default.

Q. Trying to program the MCU results in the error "Failed to launch program." What do I do?

A. Ensure power is provided to the MCU, the programming header is on correctly, and the jumper is on the MCU board. If all is correct, try restarting your computer.

Q. When I connect the power and ground of the ESP32 to the power and ground of the microcontroller breakout board, the power LED on the ESP32 breakout turns off. Any ideas as to why this would be happening?

A. It sounds like there is a short between power and ground on your MCU board. Check your connections carefully.

Q. I have connected my USB-to-UART module to the TX and RX pins of the MCU, and I have pressed the IO0 and RESET buttons as specified, but I don't see anything on the serial monitor. What may be wrong?

A. This is most commonly caused by the following issues:

- Your TX and RX lines are reversed.
- Your soldering is incomplete.
- Your serial terminal is not configured to the proper baud rate or COM port.