ECEN 260 Lab 07
HAL Drivers

Lab Objectives

- Learn how to use the STM32 Cube IDE's provided IOC code generation tool.
- Learn how to use the built-in HAL Drivers that come with the STM32 microcontroller.

Required Parts (same as last lab)

- 1 Nucleo-L476RG development board
- 1 USB cable
- 1 breadboard (protoboard)
- 3 reed switches (each has two parts)
- 3-4 LEDs (you choose the colors)
- 3-4 resistors
- 1-2 push buttons
- jumper wires

Overview

For this lab, you will be learning how to use the HAL drivers developed by the chip manufacturer. HAL stands for Hardware Abstraction Layer. You will also learn how to use the I/O Configuration (IOC) interface to generate code that configures the I/O pins.

Lab Requirements

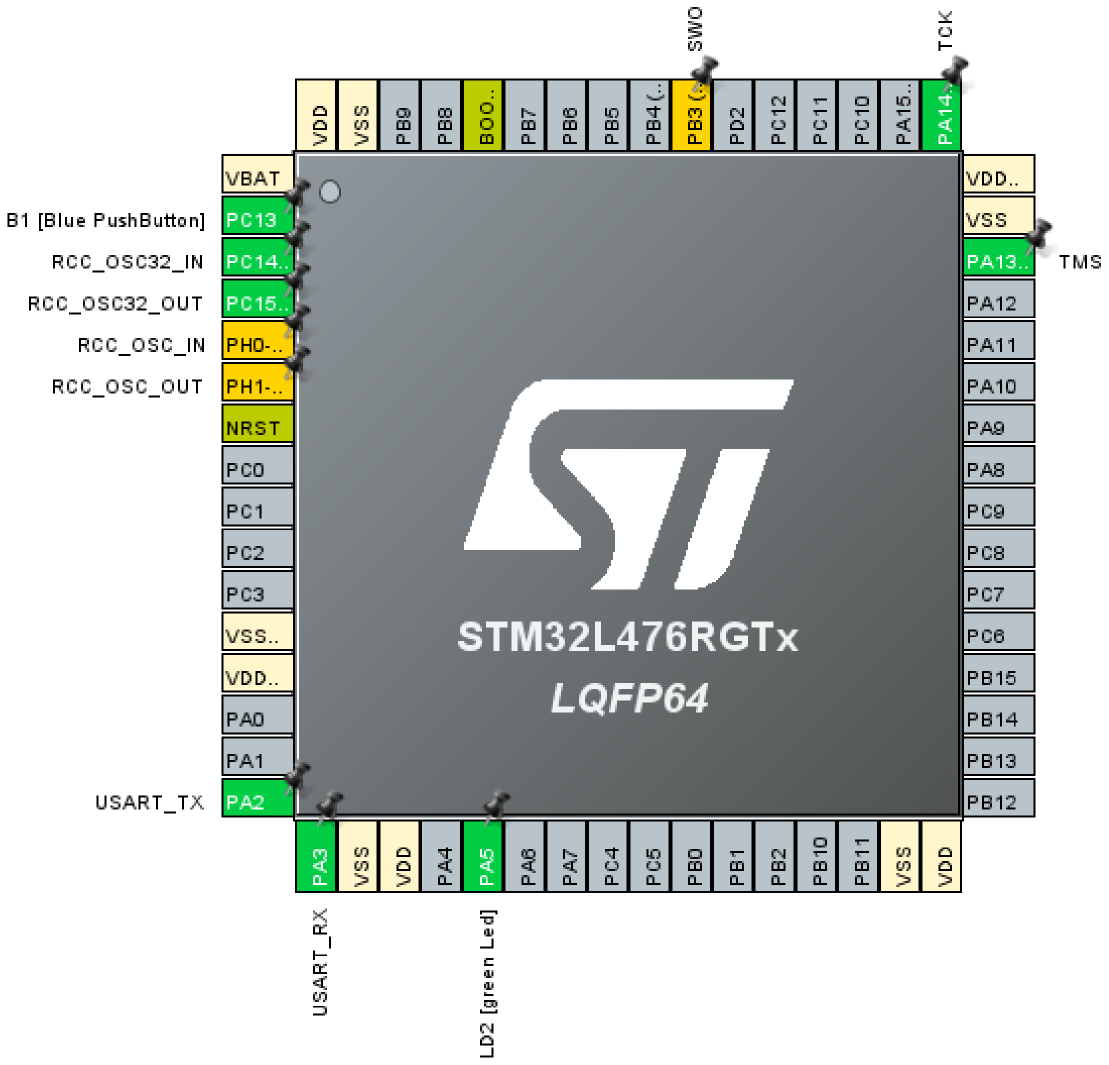
You will be creating a main.c file that operates the alarm code from the last two weeks but using the HAL drivers instead of your drivers. When you are finished, your project should work as an alarm system with the same functionality as last week.

Using the IOC Code Generation Tool

1. Before starting, make sure that you have a myST account. (If you already have a myST account, skip these steps.)
   a. Go to ST.com
   b. Click on the user profile icon in the top right corner.
   c. Click "Create Account."
   d. Follow all the steps to create an account.
2. Open the STM32 Cube IDE.
3. Log in to your myST account by clicking on the myST icon at the top next to the Help menu.
4. Create a new STM32 project: File -> New -> STM32 Project.

5. In the Target Selection Window, switch to the Board Selector tab and choose the NUCLEO-L476RG board. Then click Next.
6. Name the project, for example: Lab7. Do **NOT** select “Empty.” Instead, leave the targeted project type as “STM32Cube.” This will allow you to use the IOC and HAL driver functions that come with the board.
7. Click Finish.
8. When asked “Initialize all peripherals with their default mode?” select Yes.
9. If asked to switch perspectives, click Yes.

The *Device Configuration Tool*, aka the I/O Configuration (IOC) file, should now be open, showing a pinout image of the chip (see below). Notice that PA5 is already configured for LED 2 (LD2) and PC13 is already configured for Button 1 (B1). Other pins also have a default configuration, such as PA2 as the transmit (Tx) pin and PA3 as the receiver (Rx) pin. The Tx/Rx pins will be relevant later in the semester.

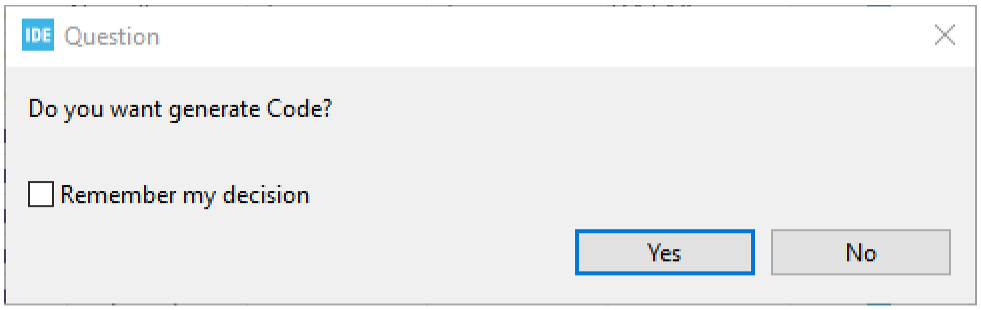


You will use this tool to name and configure the pins.

10. **Configure your input pins:** For each input pin, select the pin in the IOC and choose GPIO_Input.
11. **Configure your output pins:** For each output pin, select the pin and choose GPIO_Output.
12. **Name your pins:** For each pin, right-click the pin and select "Enter User Label." Then type a name for the pin. Use the same names from last week's lab, such as FRONT_SENSOR or ARM_BUTTON. (If it doesn't let you type the name, you may need to close and reopen your IOC file.)
13. **Enable pull resistors:**
    a. In the left pane, under "Categories," expand "System Core," and select "GPIO." That will open a new pane called "GPIO Mode and Configuration."
    b. In the "GPIO Mode and Configuration" page, select the "GPIO" tab under the "Configuration" header.
    c. For each input pin that needs an internal pull resistor:
        i. Select the relevant pin. That should display below some configuration options for that pin.
        ii. In the options below, find the "GPIO Pull-up/Pull-down" option and select the relevant pull resistor from the dropdown menu.

Once all the I/O pins are configured, you can generate code from the IOC that will use the HAL driver functions to correctly set up all the Special Function Registers for you.

14. **Generate Code:** To generate code, save the IOC file. It should ask you:



Do you want generate Code?

Click "Yes." If you don't see this window, you can manually tell it to generate code by going to Project -> Generate Code. (If asked to switch perspectives, always select "Yes.")

That should generate code in main.c that configures your I/O pins as you specified in the IOC file.

15. Take a minute to look through the generated main.c file. You don't need to understand everything yet, but pay attention to these parts:
    a. See that it includes a main.h file. If you control-click on that, it will open a main.h file that has your pin name macros.
    b. Go back to the main.c file and see that there is a `main()` function.
    c. See that inside the `main()` function it calls the function `MX_GPIO_Init()`.

d. See that there is an infinite while loop at the end of main.
e. Scroll down until you see the `MX_GPIO_Init()` function definition. Review the following code inside that function:
    i. See it enables the GPIO Port clocks.
    ii. See that it uses the `HAL_GPIO_WritePin()` function to start the output pins at an initial value: GPIO_PIN_RESET means 0 (LOW) and GPIO_PIN_SET means 1 (HIGH). Notice that it uses the Port and Pin macros from the main.h file. The
    iii. See that is uses the HAL_GPIO_Init() function, with a struct input parameter that includes options for making a pin input or output and enabling pull resistors.
f. Notice that throughout the file, there are comments marking certain areas as "USER CODE" areas with "USER CODE BEGIN …" and "USER CODE END …" comments. When you edit this generated file, you should only place your code between a pair of USER CODE BEGIN and USER CODE END comment tags. Code outside those areas will be overwritten if you regenerate from the IOC.

Using the HAL Driver Functions

Your code will be using these driver functions:

- `HAL_GPIO_Init()` – This function configures both the MODER and PUPDR. You don't need to write the code that does this, because it is generated for you in the `MX_GPIO_Init()` function from the IOC file.
- `HAL_GPIO_ReadPin()` – This function checks the IDR and returns a 1 or a 0 depending on if the pin has a HIGH or LOW voltage. To use this pin, pass the Port and Pin macros from your main.h file as input parameters to the function.
- `HAL_GPIO_WritePin()` – This function changes the ODR to the value you pass to. To use this pin, pass the Port and Pin macros from your main.h file as input parameters to the function. As the third parameter, pass a 1 or 0, choosing the value you want to write.

Review the main.c file from last week that programmed the alarm system to use the driver functions you wrote. Do something similar in the main.c for this lab, but instead using the HAL driver functions listed above. You will use the IOC file to generate all the code that names pins, configures them as input or output, and enables pull resistors. In the generated main.c file, declare an "int armed" variable before the main while loop. Then, inside of the main while loop, write the code that checks the ARM button, the DISARM button, and the reed switches and the code that changes the status of the LEDs like last week, but using the HAL driver functions instead of your driver functions. **Be sure that all the code you add to main.c is between a "USER CODE BEGIN …" tag and the corresponding "USER CODE END …" tag. Pay attention to curly brackets.**

Demonstration of Correct Operation

By Friday (6/7): Submit your main.c for a working alarm system (same as the last two weeks) but using the HAL drivers instead of your drivers. Also submit the generated main.h file and a demo video of your alarm system working while using the HAL drivers.

Lab Report Instructions

The Lab Report for this Lab will be combined with last week's Lab 6. Your test plan should include the results of your tests on both the Lab 6 code and the Lab 7 code (as separate columns in the table). Your code section should include some of your drivers.c file from last week as well as your main.c file from this week. Label each code section.

Starting with this lab report, your lab reports should also now always include a *schematic*. A schematic is a symbolic diagram showing how the components are connected.

For this report, your schematic should show each of the four LEDs as outputs with current-limiting resistors, each of the two push buttons (arm/disarm) as inputs with pull resistors, and each of the three reed switches (front/back/window) as inputs with pull resistors. You may use any schematic tool to create your schematic. If you'd like, you may use the partial schematic in Figure 1 on the next page as a starting point. It was created with www.circuit-diagram.org. You may upload this partial Lab6_schematic.cddx file from I-Learn to that website. Then, you can complete the schematic on that website. This partial schematic assumes you are using Button 1 (PC13) as your "Arm" button and LED 2 (PA5) as your "Armed Status" indicator.

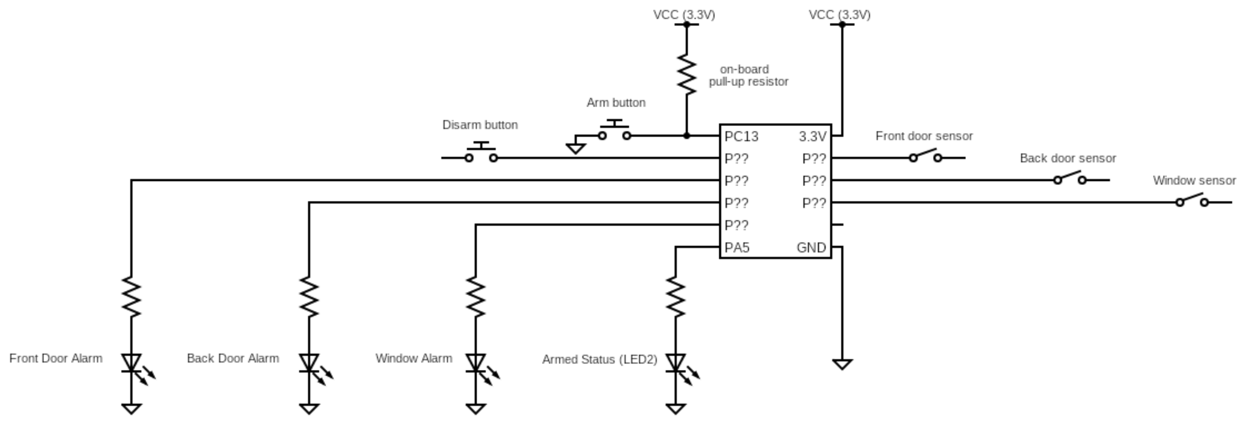


Figure 1: Partial Schematic (you must complete it)

Depending on if you choose to connect a particular button or switch to a high voltage VCC (3.3V) or to low voltage , that will determine if you need a pull-down resistor or pull-up resistor, respectively. You must add a "ground" or "VCC" connection to the other buttons and switches to indicate if they are wired as active-low or active-high, respectively, and then add a pull-up or pull-down resistor as appropriate for each. Also change the pin names on the chip icon from P?? to the appropriate names.

Include the schematic in your report using the Lab Report Template provided in the "Lab 06 & 07 Report" assignment in I-Learn, and then submit your report there.