MECHTREON 3TA4 Lab 3

Serial Communication

Introduction

This lab will help you to understand how the I2C serial communication protocol works. You will interface the STM32L476G Discovery board and STM32L476 MCU with an external FT24C64A EEPROM chip that communicates over I2C, and create the corresponding I2C bus circuit on your breadboard. You will also use the internal real-time clock (RTC) of the STM32L476.

Prelab [5 pts]

1. This lab assumes you are familiar with the material required for Lab1 and Lab2. In particular, you should feel comfortable structuring your programs using Finite State Machines.

2. Read Zhu, Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C, Chapter 18 Real-Time Clock (RTC) and Chapter 22 Serial Communication Protocols. The following slides provide an overview but are missing useful information
   Chapter_18_RTC
   Chapter_22_I2C

3. Review the datasheet of the FT24C64A datasheet for the I2C EEPROM chip. Focus on p.1-7 paying particular attention to the addressing info at the bottom of page 4.

4. Review the material in Chapter 39 of the MCU's Reference Manual and Chapter 30 of the HAL library user manual to learn about I2C. Some of these details are low level and you will not deal directly with them, but you must have an understanding of the subject matter.

5. Review the material in Chapter 38 of the MCU's Reference Manual, Chapter 48 of the HAL library user manual and the RTC Application Note to learn about the RTC in the STM32L476.

6. Read about BCD encoding. This will be needed for the RTC.

7. Draw a FSM that implements the behavior described in the Requirements section below and upload it to your folder on Avenue before the beginning of your (lab) session in the second week of Lab 3! (One per group)

Procedure

1. On your breadboard, create the ciruit shown below.

   - The Vcc power sources may be connected to the 3.3V output pin of the STM32L476G Discovery board. This is sufficient to power everything.
   - The GPIO pins of the STM32L476G Discovery board can be configured as internally pulled up or pulled down. In the library file provided in the starter project (`i2c_at32c64.c`), when the I2C is initiated, the SCL and SDA pins have been configured to be internally pulled up. Therefore the two external pull-up lines (the two wires connecting the SCL and SDA lines to

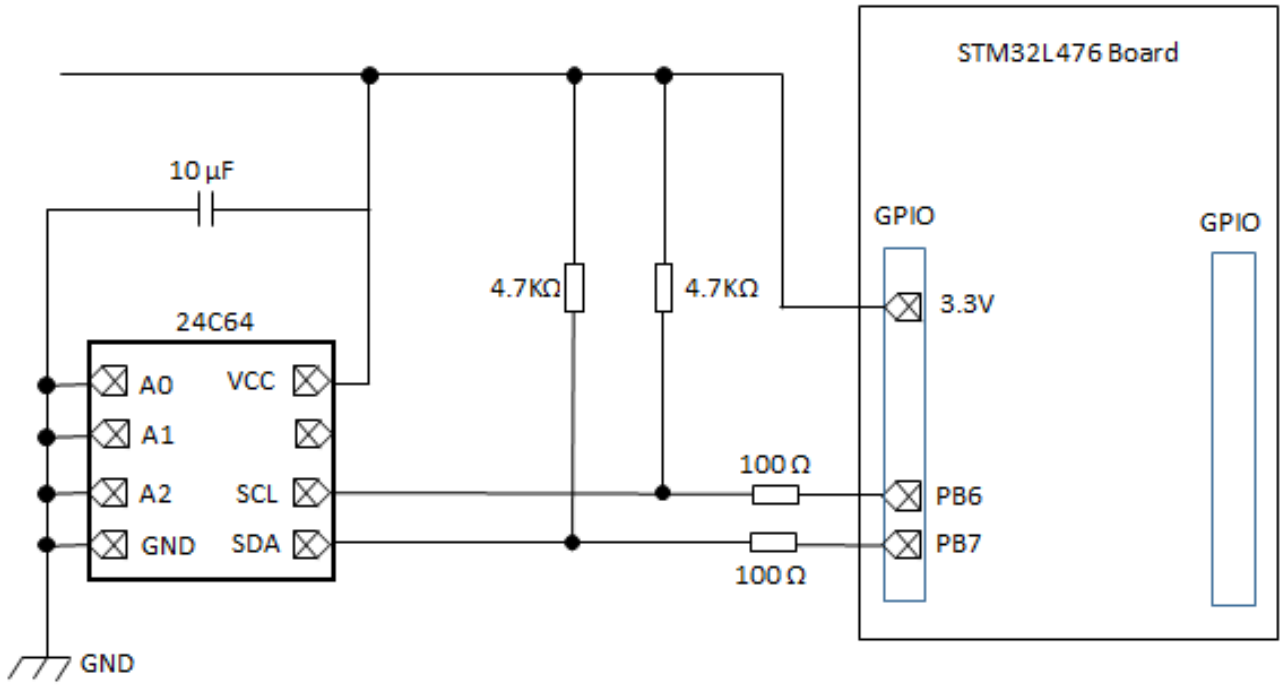

Figure 1: STM32L476G Discovery Board - EEPROM Connection Schematic Diagram

the 3.3V line through 4.7KΩ resistors) are not really necessary. You can choose to keep or omit these two wires in your breadboard circuit.

2. Download the `lab3starter.zip` starter project.

   - The starter project uses a library file `i2c_at32c64.c`, which defines functions for I2C and for wirting to and reading from FT24C64A/AT23C64 EEPROM.
   - Since FT24C64A/AT23C64 requires a 10ms delay after each byte write operation, the HAL_Delay() function is used in `i2c_at32c64.c`. HAL_Delay() function uses systick interrupt to measure time. In the starter project, the priority of the systick interrupt is set to the highest level. To avoid potential interrupt confiliction, you are recommended to set other interrupts' priorities to values other than 0 in your program.

     With proper configuration and consideration, you also can use HAL_Delay() function in your user application to let your program pause for certain length of time. like to let LCD display a string for a while.
   - The demo code writes and reads a byte to/from the FT24C64A/AT23C64 EEPROM memory using the library provided. To find the I2C port and I2C pins (SDA and SCL in the above diagram) configured by the library, please look in the "stm32l476g_discovery.h" file.

3. Make sure that the option `Use MicroLIB` is checked in the options for you Keil Project (in the `Target` tab of the `"Target option..."`). Otherwise, the size of the project image may exceed the maximum allowed for the free version of uVision.

4. Make sure to place the starter project in a folder structure similar to the one from Lab 1. Otherwise you need to modify the project's setting for including path.

5. Test your wiring with the C code provided in the starter package.

6. You will need to configure the internal RTC, and its Alarm interrupt. Please read the materials provided in the Prelab section. You should also look at the RTC-related examples. (`STM32Cube_FW_L4_V1.8.0\Projects \STM32L476RG_Nucleo\Example\RTC\RTC_Alarm`).

Requirements

1. Configure the RTC's alarm interrupt to make it activates every second. Each time the interrupt is activated, display the current time (HH:MM:SS) on the LCD. **NOTE:** Calling the fuction `HAL_RTC_GetTime()` will lock the values in the calendar shadow register. This will make the subsequent time reading get the unchanged results. You need to call the function `HAL_RTC_GetDate()` to unlock it.

2. Detect the time at which the Selection button of the joystick on the Discovery board is pressed. Get the time of the button push from the RTC and store it in the FT24C64A/AT24C64 EEPROM. You should be writing a log file in the FT24C64A/AT24C64 containing all the times that the button has been pressed.

3. When the Selection button of the joystick is **pressed and held**, display the current weekday, date, month, and year in sequence on the LCD.

4. When the Left direction of the joystick is pressed, display on the LCD the last two times the user button was pressed as recorded in the FT24C64A/AT24C64 EEPROM. The user should be able to exit this display mode by pressing the Left direction of joystick again.

5. When the Right direction of the joystick is pressed, enter into the date setting and time setting mode. Use the Left direstion of the joystick to choose the field (e.g., the week day, or the hour etc.) to be changed, use the Selection of the joystick to change the value of the selected field. Use the Right direction of the joystick to exit the date/time setting mode.

6. All values displayed on the screen should have labels to improve usability.

Assignment

Demo the program you wrote to your TA in lab and also defend your design and implementation decisions:

1. **[20 pts.]** The FSM you were required to do as prelab preparation. This should be uploaded to your folder on Avenue as `lab3_fsm.{png,pdf}`. Only png and pdf formats are allowed! (One per group)

2. **[65 pts.]** A functional program that implements all the requirements correctly.

3. **[10 pts.]**A heavily commented C code. Clear and concise comments are very important for others to understand your code and even for you later on when you need to remember what you did.

4. **[5 pts.]** Your folder on Avenue should be kept in a proper state from now on. This means:

    - There should only be one folder per lab (i.e., The only folders in your Avenue folder should be: lab1, lab2, lab3...).

- Your prelab submission for each lab should be in a "prelab" subfolder i.e. for lab 3: lab3/prelab/
- Only the three C source and header files (main, stm32l4xx_hal_map, stm32l4xx_it), as whell as other yourown source code, should be uploaded to Avenue. You do not need to upload library files.