Australian National University

Lab 8

Serial Communication of STM32 microcontroller: I2C LCD

ENGN4213/ENGN6213
Digital Systems and Microprocessors

Semester 1, 2023

ANU College of Engineering, Computing & Cybernetics

Table of Contents

1. Introduction

During this lab you will gain hands-on experience on using serial communication methods to transmit and receive data among computer, STM32 Nucleo board, and LCD display. You will understand the basics of UART (Universal Asynchronous Receiver-Transmitter) and I2C (Inter-Integrated Circuit) communication protocols and implement them on your STM32 Nucleo board. You will also practice initializing a peripheral based on the datasheet.

2.Pre-Lab Tasks

- Review the lecture slides on different communication protocols of STM32
- Read through section 4 and complete the comparison table in section 4.2

3. What You Need

Hardware:

- A STM32F411RE Nucleo board
- A USB type A to mini-B cable
- Jumper wires
- Computer
- A 16x2 LCD screen with I2C interface (PCF8574)

Software:

- A compiler (we use STM32CubeIDE as an example)
- A serial port terminal (we use Hercules SETUP utility as an example)

4. STM32 Nucleo Board Serial Communication

4.1. Serial Communication subsystems on STM32 Nucleo Board

Microcontrollers must often exchange data with other microcontrollers or peripheral devices. Data may be exchanged by using parallel or serial techniques. With parallel techniques, an entire byte of data is sent simultaneously from the transmitting device to the receiver device. While this is efficient from a time point of view, it requires eight separate lines for the data transfer together with address lines. In serial transmission, a byte of data is sent through a single bit at a time. Once eight bits have been received at the receiver, the data byte is reconstructed. While this is inefficient from a time point of view, it only requires a line (or two) to transmit the data.

STM32 Nucleo boards are equipped with 3 different serial communication subsystems – UART (Universal Asynchronous Receiver-Transmitter), SPI (Serial Peripheral interface), and I2C (Inter-

Integrated Circuit). In serial communication, the transmitting and receiving device must be synchronised to one another and use a common data rate and protocol. In asynchronous serial communication, as in the UART, a start and stop bits are used to notify the receiver to reset the sampling timer. In synchronous communication, such as SPI or I2C, a common clock between the two devices “syncs” the transmitter and receiver. Data transmission rates are typically specified as a Baud or bits per second rate. For example, 9600 Baud indicates the data is being transferred at 9600 bits per second. The UART and SPI are full-duplex systems which have two separate hardware for the transmission and reception. I2C, however, is a half-duplex system sharing a single data line, making the connection much simpler but with the extra cost of programming.

4.2 Based on your understanding compare UART and I2C [5 Marks]

	Duplex	Number of Devices	Speed	**Features in the data transmission**
UART	full	2	1-150k bits/s	asynchronous
I2C	half	7-bits=127	10k-3.4M bits/s	synchronous

5. Activity: Receive a message from computer (UART) and display a message on LCD screen (I2C) [25 Marks]

5.1 Hardware connection

1. According to the datasheets of 16x2 LCD, I2C interface module (PCF8574) and pinout diagram of Nucleo F411RE, use jumper wires to connect the LCD module to your microcontroller board to be ready for I2C communication. (Hint: Please check PCF8574 datasheet, LCD datasheet, and I2C section in stm32f411re datasheet to figure out the power supply voltage for the LCD module.)
2. From stm32F411RE datasheet and LCD module schematic, check whether pull-up resistors are available on SDA bus and SCL bus. If yes, check the resistance to make sure it is compatible with the I2C speed selected in our project. If not, add external resistors to achieve sufficient pull-up.
3. Power up the board. Adjust the backlight brightness according to the 'schematic of the LCD module' (Hint: Rotate the screw within the blue block on I2C interface module (PCF8574)).

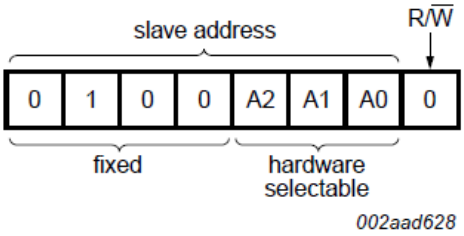
5.2 Configuration

Modify the Nucleo board in the "Pinout & Configuration" and "Clock Configuration" in the STM32CubeIDE to be ready for I2C communication.

1. Based on the hardware connections in Section 5.1, configure I2C SDA pin and I2C SCL pin in the 'Pinout & Configuration' tab.
2. Enable the selected 'I2C' from the left sidebar and configure the parameters, such as 'I2C speed'. The corresponding pins should turn green. (Hint: To configure 'I2C speed', double check the pull-up resistors.)
3. Clock configuration (optional):
   - In the 'system core' list, select RCC from the left sidebar of "Pinout & Configuration". Choose the High Speed Clock (HSE) as an external crystal resonator.
   - From the stm32f411re datasheet, check which bus (AHB1, AHB2, APB1, and APB2) is connected to the I2C. In "Clock Configuration" tab, choose HSE as the clock source and configure the corresponding clock speed.
4. Save the configuration and generate code.

5.3 I2C Slave device address

1. Based on the datasheet of I2C interface module (PCF8574) and the hardware selection on I2C interface module, obtain the slave address. (Hint: A0, A1, A2 are the nodes on the back of I2C interface PCF8574. Unsoldered = High, and soldered = Low.)



a. PCF8574

Figure 1 PCF8574 slave address (Source: PCF8574 datasheet)

5.4 I2C coding

5.4.1 Setup the I2C slave device and check whether the device is ready

1. Scan the I2C slave address and confirm the address we obtained in Section 5.3 (highly recommended, not compulsory) (Hint: The slave address is only 7 bits. If we scan the slave address from 0x00 to 0x7F, we should be able to find the correct address for LCD. You can also transmit the correct address to PC serial terminal through UART for better virtualization.)
2. Check whether the I2C device is ready. Please skip this step if you conducted 5.4.1. (Hint: Find the function in the HAL I2C section in the 'STM32F4 HAL' document.)

5.4.2 Complete the lcd i2c library

Import the 'i2c-lcd.c' and 'i2c-lcd.h' file into our project, you can find these two files in the 'Lab 8 Resources'. You need to complete this LCD library for initialization and data transmission.

1. Modify I2C handler 'I2C_HandleTypeDef' and Slave address 'SLAVE_ADDRESS_LCD' according to your I2C configuration and slave address.
2. Initializing by instruction: The initialization is conducted by sending commands to the LCD. In this lab, we provide an example of initialising the LCD into '4-bit mode', where each I2C frame only transmits 4 bits of commands. Please pay attention to how many frames (i2c_frame_size) we need to transmit for making the commands. For different commands, the frame size could be different. [5 marks]
    - Complete lcd_send_cmd() function to send command to LCD. (Hint: look up the hal document and find out HAL_I2C_Master_Transmit function for the 'master' transmitting data to a 'slave'.)

- Based on LCD datasheet (HD44780_LCD), complete function lcd_init() to initialise the LCD module into '4-bit-mode'. (Hint: lcd_send_cmd() function and HAL_Delay function can be used to achieve this goal.)

3. For data transmission: Complete function lcd_send_string() to display a string on LCD screen in "lcd_i2c.c". (Hint: You might call function lcd_send_data() in this library to send data one by one.) [5 marks]
    - Complete function lcd_send_data() to send data to LCD. (Hint: look up the hal document and find out HAL_I2C_Master_Transmit function for the 'master' transmitting data to a 'slave'.)
4. For LCD clearance: Complete function lcd_clear() to clear LCD. (Hint: You might call functions lcd_send_cmd() and lcd_send_data() in this library.)
5. In your header file "lcd_i2c.h", declare the functions you wrote in "lcd_i2c.c". Don't forget to include "stm32f4xx_hal.h" in your header file too.

5.4.3 transmit help message using COM port (UART) and display the same message on LCD (I2C)

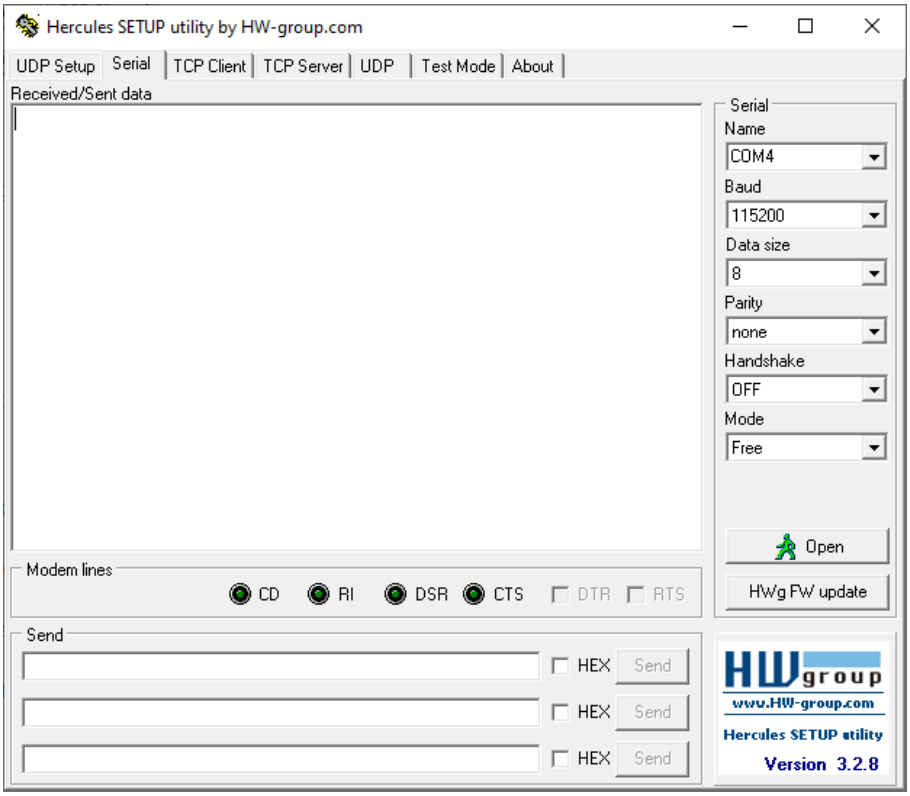
6. In "main.c", include header file "lcd_i2c.h".
7. Create a buffer to store the message to display.
8. Call functions in the lcd i2c library in an appropriate way to achieve the message displaying on the LCD screen.
9. Transmit "Please help, UnivID" message to serial port terminal of your computer (Hercules SETUP utility as an example). Please review the Lab 7 manual if you are unsure how to transmit data from the STM32 Nucleo board through UART.
10. Validate your work as the following process [10 marks]:
    a. Transmit help message from STM32 Nucleo board (through UART transmission) to COM Port of your computer (Similar to lab 7).
    b. Display the same help message on your LCD screen through I2C transmission. (Hint: If you find your message is not shown, please try to adjust the screw within the blue block on I2C interface module (PCF8574) firstly.)
11. Modify your code to display "Please help" and "UnivID" on your LCD in two lines. [5 marks]

Appendices

A. Hercules SETUP utility

Hercules SETUP utility is useful serial port terminal, UDP/IP terminal and TCP/IP Client Server terminal.

After you set up the UART transmission between your board and PC, check out which COM port your board is at (via Device Manager).



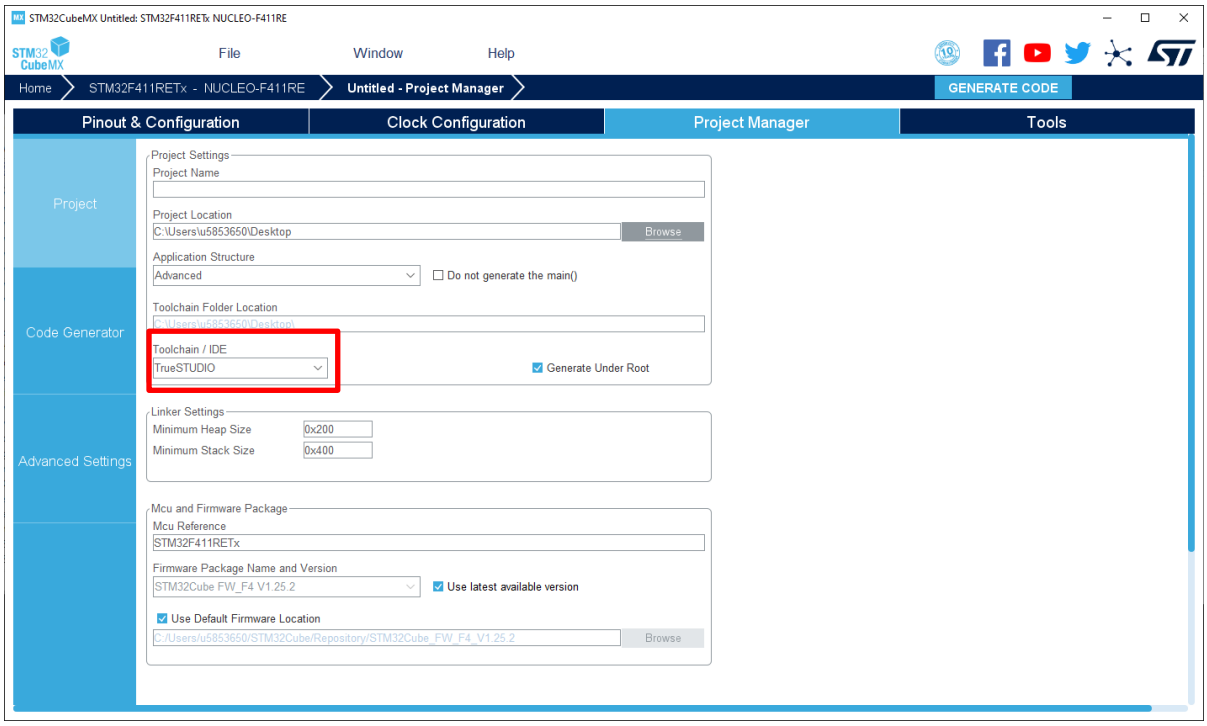
In the Hercules window, you should set the serial port to the exact COM port your board is at. You should also set the Baud rate the same to the number in your code.

Click **Open** to open the COM port and communication between the board and PC.

If you are transmitting data from the board to PC, you should see data in the big blank area. If you are receiving data, simply type some message in the input bar and click send. Either input bar among the three is fine.

B. CubeMX & TrueStudio

If you are not using STM32CubeIDE, but using a combination of CubeMX and some other compilers (for example TrueStudio), you can simply change the **Toolchain / IDE** in **Project Manager** tab (in your CubeMX).



Then, click on GENERATE CODE on the right top corner, you will have a project folder.

Open TrueStudio, go to File -> Open Projects from File System, import the folder from the location you just saved. Then you can do the same thing as using STM32CubeIDE.