EGH456 Embedded Systems

Laboratory Exercise 4

Serial Communication

Learning Objectives

After completing this laboratory you should be able to:

1. Use serial communication in an embedded system.
2. Understand the role and function of serial communication drivers.
3. Implement interrupt-driven input and output techniques.
4. Configure and utilise TivaWare I2C driver libraries effectively.
5. Communicate with an external sensor via the I2C protocol.
6. Visualise and interpret sensor data in real-time using serial plotting tools.

Reference Documents

- OPT3001 Ambient Light Sensor Datasheet
- BOOSTXL-SENSORS Sensor BoosterPack User's Guide
- FreeRTOS Reference Manual
- FreeRTOS on TM4C MCUs
- EK-TM4C1294XL User's Guide
- TM4C1294NCPDT Datasheet

Preparation Activities

Before starting this lab:

1. Carefully read the OPT3001 Ambient Light Sensor datasheet, focusing on sections 7.5 and 7.6 regarding I2C communication.

2. Review and familiarise yourself with the TivaWare I2C driver library API documentation (Chapter 16).

3. Download and familiarise yourself with the Serial Plot tool - Serial Plot.

Laboratory Tasks

Task A - Initial Sensor Communication (1 Mark)

In this task, you will verify communication between the TM4C1294 microcontroller and the OPT3001 sensor using I2C. Download and run the provided `opt3001_demo` project. Observe the serial terminal output and confirm correct communication with the sensor. If the initialisation is failing try resetting the microcontroller or flagging this with the teaching team as it could be a faulty sensor.

Read through the *opt3001.c* and *i2cOptDriver.c* files and familiarise yourself with how these drivers work and answer the following questions. It will also be helpful to refer to Figure 20 and 21 within the OPT3001 Datasheet

Answer the following questions:

1. Which I2C port (device number), GPIO port and pins of the microcontroller are used to connect and communicate with the OPT3001 sensor? **(0.25 marks)**

2. When enabling the sensor (`SensorOpt3001Enable(true)`), which bits and what is the value set for the Range Number Field in the configuration register and what is the configured conversion time of the sensor? **(0.25 marks)**

3. In a 16-bit I2C write transmission to the OPT3001, which byte (high or low) is received first, and why is this important to know? **(0.25 marks)**

4. What is the method and what sensor registers are used in this demo to verify that the i2c communication and sensor is correctly setup? **(0.25 marks)**

Task B - FreeRTOS and I2C Interrupt Integration (3 Marks)

Convert the previous Tivaware Peripheral Driver library implementation of the opt3001_demo project into an implementation that is compatible to use within a FreeRTOS project. Specif-

ically, change the I2C driver which performs a blocking read and write, to an interrupt driven (ISR) and task thread-safe version for reading and writing from the sensor. You will need to do the following:

- Create a new FreeRTOS project. We recommend making a copy of the semaphore example provided in Computer Lab 3. Clearly structure your project, separating tasks, and driver implementations into different folders/files. Integrate the provided I2C driver files (`i2cOptDriver.c, i2cOptDriver.h, opt3001.c, opt3001.h`) from the opt3001 demo carefully into your project, ensuring that all necessary configurations and include paths are correctly set up. **(0.5 marks)**
- Add a sensor read task to your project using a freeRTOS task thread and call the sensor read function within this task. Confirm this compiles and that a sensor can be read within the task thread. Ensure to initialise the sensor at the beginning of the task thread. **(0.5 marks)**
- Modify the provided I2C driver functions (`writeI2C` and `readI2C`) by implementing an interrupt-driven method. Replace the existing polling method by removing all the while loops around the I2C master busy functions with a semaphore-based synchronisation method. It is recommended to use a semaphore within the ISR to signal to the read/write functions in the task thread that a read/write has completed. Clearly explain your modifications and justify how interrupt-driven processing improves system responsiveness and efficiency. **(1.5 mark)**
- Modify the sensor read task thread to use a push button to toggle starting and stopping the reading of the light sensor value and transmit the data clearly formatted through UART to the serial terminal. **(0.5 marks)**

Task C - Real-Time Serial Plotting (1 Mark)

For this task you will utilise a PC-based Serial Plotting tool to help debug sensor systems in embedded environments. Complete the following steps to plot light readings via the UART:

- Research, Download and familiriase yourself with a Serial Plotting tool. We recommend using this tool Serial Plot.

- Modify your FreeRTOS application to continuously output sensor lux values via UART at the fastest rate the sensor can measure.
- Modify the formatting of your data sent via UART to work with the Serial Plotting tool and demonstrate light values are plotted to the screen from live sensor measurements. **(1 mark)**

Task D - Light Threshold Triggering (3 Marks)

This task involves using the features of the OPT3001 light sensor to configure automatic event triggering based on light thresholds. You will explore how to change the configuration of the the sensor's window-style comparison mode, which continuously monitors ambient light levels and generates a digital I/O trigger when values fall outside predefined bounds. This feature is implemented through the The sensor's interrupt digital output (INT pin) that is connected to a GPIO pin on the TM4C1294 microcontroller. This GPIO pin can then be configured to trigger an internal GPIO interrupt. Complete the following steps to implement light threshold triggering events:

- Using I2C communication, setup the OPT3001 sensor configuration register with the following settings **(0.5 marks)**:
  - The sensor has a conversion time of 800ms
  - The sensor is running in continuous mode
  - The sensor is in Latched Window-Style Comparison Mode
  - The polarity field of the OPT3001 is set so the interrupt (INT) pin is active low
- Setup the sensor high and low limit registers, which are used to generate the change in the OPT3001 INT pin when light values are outside these values **(1 marks)**
  - Set the Low-Limit register to 40.95 lux
  - Set the High-Limit register to somewhere between 2000 and 3000 lux (you get to choose the exact value, be prepared to explain how that value was calculated)
- Setup a GPIO interrupt which is triggered on the falling edge of the INT pin from the OPT3001 (Consult Sensors boosterpack guide and launchpad user guide to determine GPIO pin). **Hint:** You will need to find the GPIO pin associated with the INT pin for

the Optical sensor. Also, you will need to setup the pull up or pull down resistors for the interrupt pin and justify your choice. Note: We have desoldered the resistor on the sensor boosterpack so that it doesn't interfere with the touchscreen when used in your group assignment. See the following links for diagrams that show the pin mapping for the launchpad (Launchpad Pin Map) and the boost-xl sensor board. (Sensors Boosterpack Pin Map) **(1 marks)**

- When a high or low limit interrupt has occurred, signal to a task thread (don't print from the ISR) to print "Low Light Event: *xx.xx* Lux" (where *xx.xx* is the current lux level) if the interrupt was caused by the light dropping below the level in the low-limit register and print "High Light Event: *xx.xx* Lux" for the opposite case. Otherwise, the program should continue printing the light level as normal: "*xx.xx* Lux". Demonstrate this is done via an ISR and the OPT3001 INT pin and not just through polling the light values manually. **(0.5 marks)**

Assessment

Demonstrate completed tasks and provide written answers to the questions and document your approach. You may work with a lab partner to complete these tasks.