EGH456 Embedded Systems

Laboratory Exercise 5

Queues and Events

Learning Objectives

After completing this laboratory, and following up using documentation you should be able to:

1. Understand how queues are created and different methods for passing data between queues
2. Passing sensor data between tasks with a queue.
3. React to multiple events with an event group.
4. Update a display with text and simple graphs.
5. Handle sensor events and user button input alongside passing sensor data via queues.

Reference Documents

- FreeRTOS Reference Manual
- FreeRTOS on TM4C MCUs
- EK-TM4C1294XL User's Guide
- TM4C1294NCPDT Datasheet

Preparation Activities

Read the chapters on Multithreading and Shared Resources in the reference book by Lewis. Read Lecture 6, 7 and 8 notes. It will also be a good idea to check out technical documents (Getting Started and User Guides) related to FreeRTOS.

Laboratory Tasks

Task A - Understanding Queues (2 Marks)

1. Download the example project *queues* from the workshop page on canvas and export it to your workspace folder.
2. Examine the project files and open the *queue_task.c* file. Read it to get an understanding of what the program does.
3. Examine the source code. Relate to your understanding of queues and tasks.
4. Now run the project. Examine the output on the serial terminal.

Complete the following tasks and questions:

1. What is stored within each message object passed into the queue for this program? **(0.25 Marks)**
2. Why can the two receive tasks read from their queues without using a **mutex**? **(0.25 Marks)**
3. Where is the data stored in memory when we pass the message **by value** and where is it when we pass it by a **pointer**? **(0.25 Marks)**
4. Use `uxQueueMessagesWaiting()` inside one of the Receive tasks and print its value. What numbers do you see while the program runs, and what do they mean? **(0.25 Marks)**
5. Change the queue length from 4 to 1 in the example program. Write down what you observe change in the behaviour of the program and explain why. **(0.25 Marks)**
6. List two problems that can happen when we send a pointer instead of a copy, and give one way to stop each problem. **(0.25 Marks)**

Task B - Sensor Sampling via Queues (3 Marks)

For this task you will need to develop a program that shares data between two tasks: a sensor sampling task that is driven by a periodic timer and a sensor display task to display the result on the screen. The sensor we will use for this task is the OPT3001 light sensor used in the previous laboratory 4.

The task should read a sensor sample at 10Hz and calculate a moving average of the samples. To implement Task B, the following steps need to be taken:

1. Create a task that samples light values at a rate of 10Hz from the OPT3001 sensor. Your solution to lab 4 provides a good starting point for developing this program. This must be done with the use of a hardware timer or using the freeRTOS software timer module. You must configure the sensor to run at a sufficient conversion time. **(0.5 Marks)**
2. After sampling the sensor implement a noise filter of your choice (i.e. moving average filter) of sensor data. The filter should clearly demonstrate removal of some of the noise of the sensor readings. The filter should be implemented within the same task as the data reading. **(0.5 Marks)**
3. Create a queue and a custom message struct to transmit the sensor data between the sampling task and a second display task. Within the message struct store both the raw sensor data and the filtered sensor values and a timestamp to keep track of the readings. You can choose if you want to implement this via pass by value or pointer. Be careful where you create the message to put in the queue if you are sending a pointer to the data. **(1 Mark)**
4. Within the display task you should receive the data from the queue and then print both sensor values (raw and filtered) in a format that you can display two plots on the serial plotting tool used in previous labs. Demonstrate that you can see both plots and that the filtered sensor values are an improvement of the raw values. **(1 Mark)**

Task C - Plotting Data and Handling Events (3 Marks)

Show how **event groups** can help co-ordinate sensor events and user input while the light-sensor data keeps flowing. By using message queues to pass data between tasks and events as a signalling mechanism, the program can efficiently manage the flow of information and ensure that the display task always has the most up-to-date data from the light sensor. The use of events as a signalling mechanism allows the task handle other tasks efficiently. Complete the following:

1. Add the `grlib` graphics library to your project (see Lab the grlib_freertos project on canvas as an example) and use the library to draw a plot of the filtered sensor data received from the queue to the touchscreen. Make sure to plot the sample data in a graph over time using drawing functions. Use a time window of more than 10 seconds to display the data. See the grlib documentation for example functions that can assist in plotting data to the screen. (Hint: use `GrContextDrawLine()` or similar). **(1 Mark)**
2. Create one FreeRTOS **event group** and use three event bits to do the following: **(1 Mark)**
   - `EVENT_HIGH_THRESHOLD` – set by the sampling task if a lux value exceeds a defined high limit value
   - `EVENT_LOW_THRESHOLD` – set by the sampling task if a lux value is lower than a defined low limit value
   - `EVENT_BTN_TOGGLE` – set from a GPIO button ISR when the user taps the push-button or using a grlib button widget.
3. Update the *display task* to check for the different event bits. Make sure the sensor data is still received independently of these events using the queue (i.e. do not block on these events or vice versa):
   - On `EVENT_HIGH_THRESHOLD` Prints a warning message to the serial terminal to say a high threshold value was received.
   - On `EVENT_LOW_THRESHOLD` Prints a warning message to the serial terminal to say a low threshold value was received.
   - On `BIT_BTN_TOGGLE` it toggles whether the raw-data is plotted to the screen or the filtered values.

   (1 Mark)

Assessment

Show evidence of having performed the modifications and run the programs as required to the tutor. Answers to questions must also be written down and shown in class. You may work as a pair of 2. Each member must be present in the laboratory at the workstation and be participating in the exercise to receive credit.