ANTA Yaëlle

In513 – Real Time Operating Systems for Embedded Systems

Final Assignment

December 2023

Abstract

A customized scheduler has been implemented on FreeRTOSv202107 using the Posix_GCC demo files. The scheduler runs 4 different periodic tasks and is based on the main_blinky() function of the demo.

Introduction

This project constitutes the final assignment of the course. After understanding what periodic tasks and scheduling methods are, this work's objective is to design an RTOS scheduler.

The workspace is the Ubuntu 22.04 WSL on Windows 11, with FreeRTOSv202107.00 installed on it.

NB The RTOS is not the most recent because it avoids fatal errors present on new releases.

Method

Testing the environment

In the file `main.c`, `BLINKY_DEMO` is set to 1 because we don't need to work with the full demonstration version for our purpose.

```
#define    BLINKY_DEMO    1
#define    FULL_DEMO      0
```

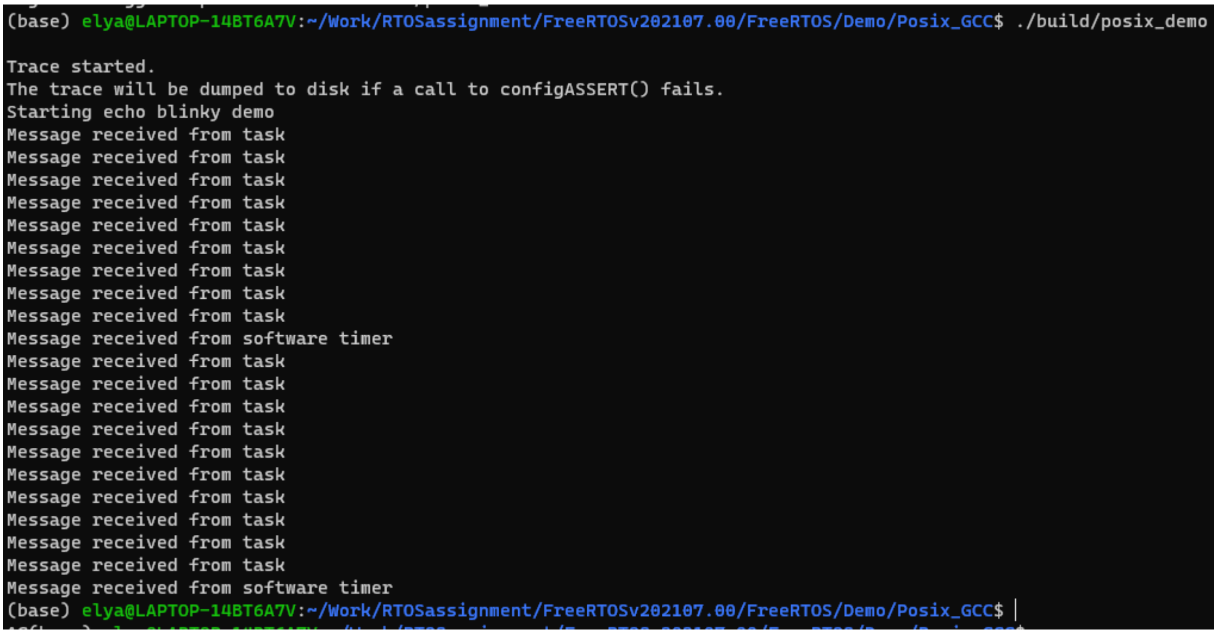
To build the project files, we must use the **`make`** command in the

```
FreeRTOSv202107.00/FreeRTOS/Demo/Posix_GCC
```

directory.

Then, we run the executable file using the following command:

```
./build/posix_demo
```



```
(base) elya@LAPTOP-14BT6A7V:~/Work/RTOSassignment/FreeRTOSv202107.00/FreeRTOS/Demo/Posix_GCC$ ./build/posix_demo

Trace started.
The trace will be dumped to disk if a call to configASSERT() fails.
Starting echo blinky demo
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from software timer
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from task
Message received from software timer
(base) elya@LAPTOP-14BT6A7V:~/Work/RTOSassignment/FreeRTOSv202107.00/FreeRTOS/Demo/Posix_GCC$ 
```

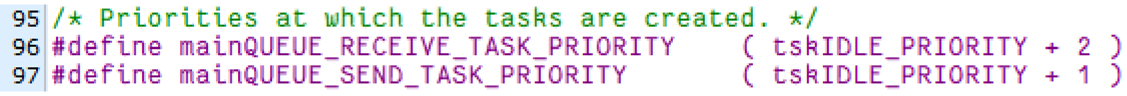
The environment works properly!

Create ipsa_sched.c

For grading, main_blinky.c and its associated function are both renamed ipsa_sched.

Now, to implement our own tasks, some steps must be followed:

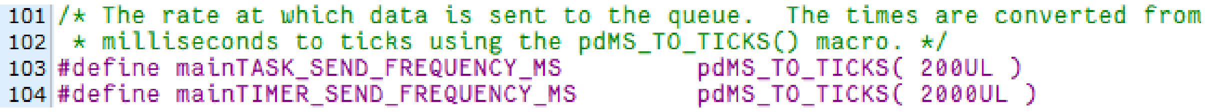
- Each new task must have a set priority here:



```
/* Priorities at which the tasks are created. */
#define mainQUEUE_RECEIVE_TASK_PRIORITY     ( tskIDLE_PRIORITY + 2 )
#define mainQUEUE_SEND_TASK_PRIORITY        ( tskIDLE_PRIORITY + 1 )
```

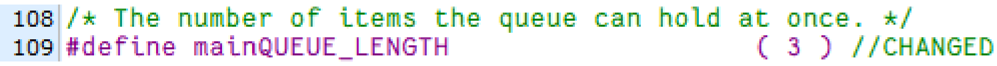
The first task here has the priority while the second one is next.

- Then, a frequency is added:



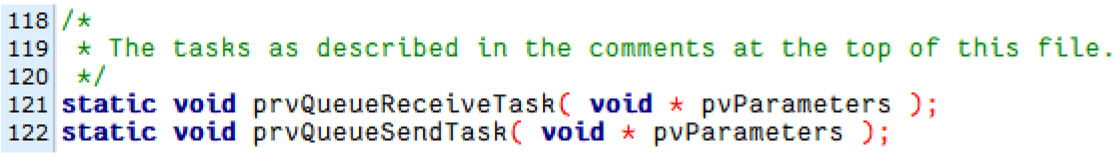
```
/* The rate at which data is sent to the queue.  The times are converted from
 * milliseconds to ticks using the pdMS_TO_TICKS() macro. */
#define mainTASK_SEND_FREQUENCY_MS          pdMS_TO_TICKS( 200UL )
#define mainTIMER_SEND_FREQUENCY_MS         pdMS_TO_TICKS( 2000UL )
```

- Here is the number of tasks in the scheduler queue:



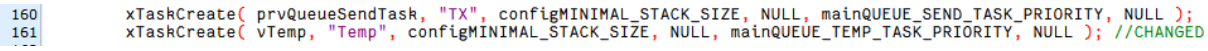
```
/* The number of items the queue can hold at once. */
#define mainQUEUE_LENGTH                    ( 3 ) //CHANGED
```

- Each new function that is implemented to be used by a task, must be declared here:



```
/*
 * The tasks as described in the comments at the top of this file.
 */
static void prvQueueReceiveTask( void * pvParameters );
static void prvQueueSendTask( void * pvParameters );
```

- Finally, the task is created in the scheduler `ipsa_sched()` as the demo tasks:



```
        xTaskCreate( prvQueueSendTask, "TX", configMINIMAL_STACK_SIZE, NULL, mainQUEUE_SEND_TASK_PRIORITY, NULL );
        xTaskCreate( vTemp, "Temp", configMINIMAL_STACK_SIZE, NULL, mainQUEUE_TEMP_TASK_PRIORITY, NULL ); //CHANGED
```

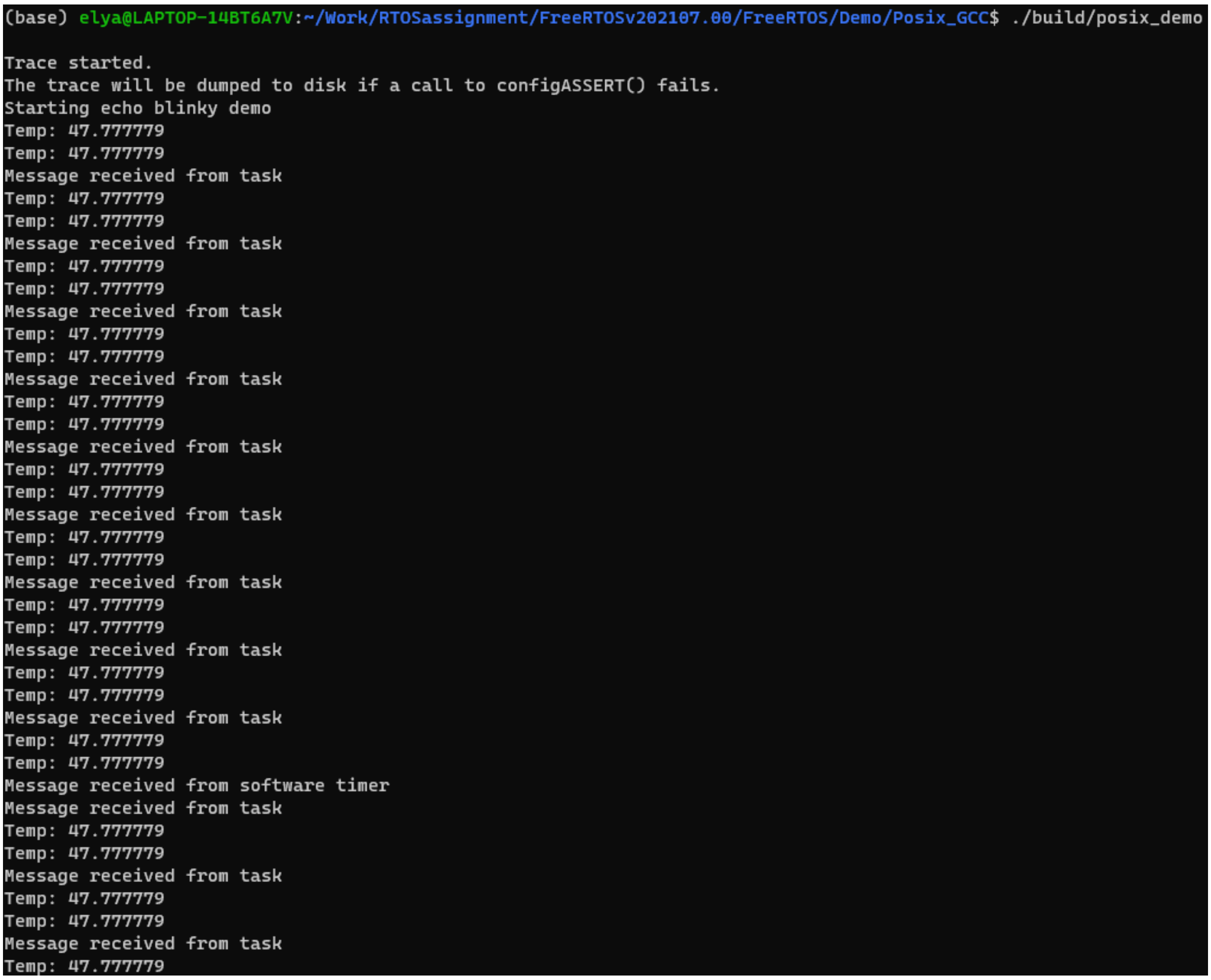
Results

Four tasks had to be implemented:

- Periodic Task 1: To print ”Working” or some other string which says everything is working as normal
- Periodic Task 2: Convert a fixed Fahrenheit temperature value to degree Celsius
- Periodic Task 3: Define any two long int big numbers and multiply them, print the result.
- Periodic Task 4: Binary search a list of 50 elements (fix the list and element to search)

I first tested to implement the task 2, scheduled with the two tasks present by default in the code.

Here are the results:

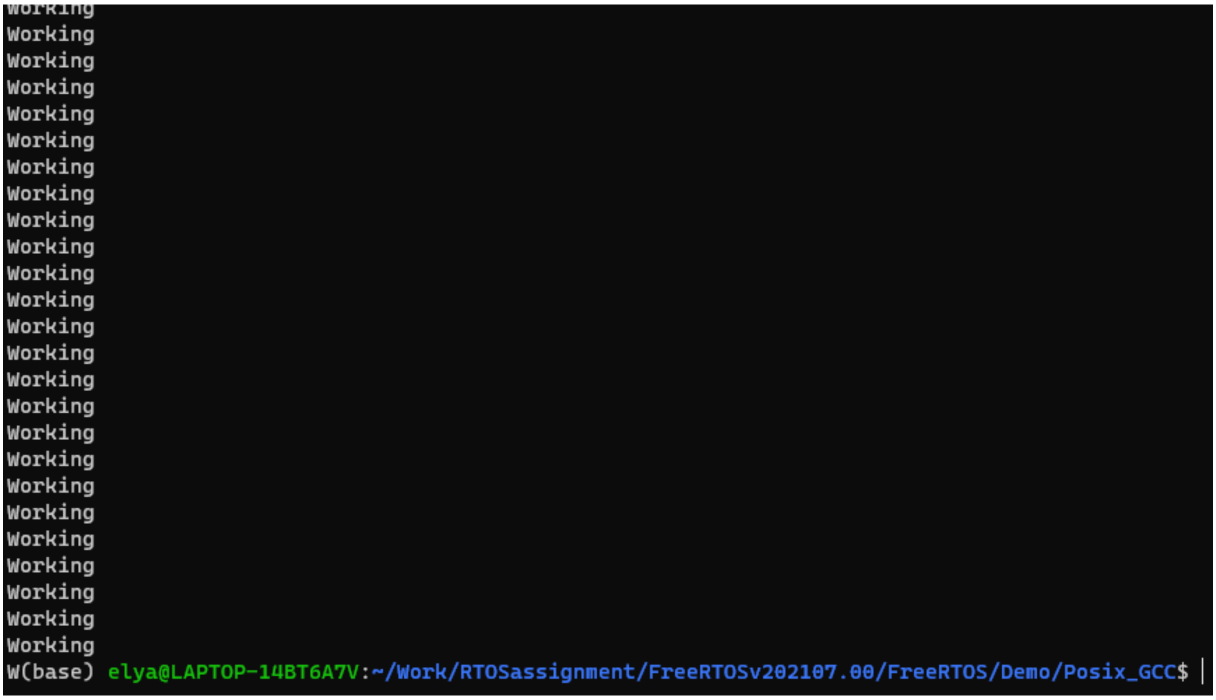


```
(base) elya@LAPTOP-14BT6A7V:~/Work/RTOSassignment/FreeRTOSv202107.00/FreeRTOS/Demo/Posix_GCC$ ./build/posix_demo

Trace started.
The trace will be dumped to disk if a call to configASSERT() fails.
Starting echo blinky demo
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from software timer
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
Temp: 47.777779
Message received from task
Temp: 47.777779
```

The scheduler is working as expected.

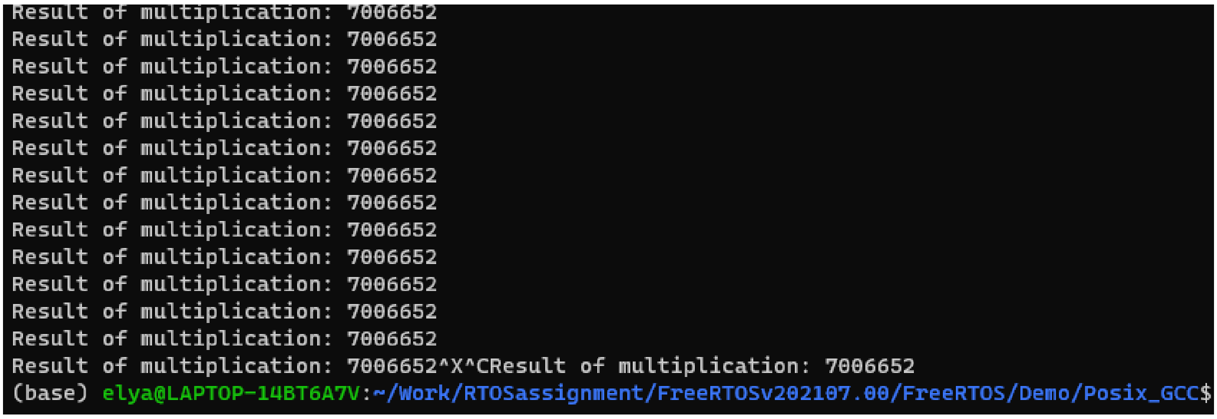
Then, I implemented one more task, the #1 that should print “Working” in the console:



```
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
Working
W(base) elya@LAPTOP-14BT6A7V:~/Work/RTOSassignment/FreeRTOSv202107.00/FreeRTOS/Demo/Posix_GCC$
```

The result is unexpected as it prints Working too fast!

If I remove it, another task does the same:



```
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652
Result of multiplication: 7006652^X^CResult of multiplication: 7006652
(base) elya@LAPTOP-14BT6A7V:~/Work/RTOSassignment/FreeRTOSv202107.00/FreeRTOS/Demo/Posix_GCC$
```