South Dakota School of Mines & Technology
CENG 448/548 — Real-Time Operating Systems
Laboratory Assignment Two

Work individually. The objectives of this assignment are to:

1. Convert the hello world main program function into an RTOS task and modify it to print "Hello World" once per second.

2. Add a task that periodically reports CPU usage.

Before you begin, make sure that your previous lab code works.

Part 1: Converting to FreeRTOS

1. Copy your previous project directory into a new directory, or use `tar` to archive your previous project directory and then re-name it.

2. Either copy FreeRTOS into your new project directory, or (much better) create a symbolic link to a FreeRTOS directory that is stored outside of your project directory.

3. Copy the example `FreeRTOSconfig.h` file from `/opt/packages` into your project `include` directory.

4. Copy `src/main.c` to `src/hello_task.c`. In `src/hello.c`, convert the `main` function into a FreeRTOS task function named `hello_task`, and define the static TCB and stack. The task should not initialize the UART, so delete that part of the code. Modify the task function so that it prints "Hello World" once per second as long as the system is running.

5. Create a matching `include/hello_task.h` containing the function prototype and declarations for the TCB and stack.

6. Edit the original `src/main.c`. The main function should initialize the UART, then create the hello task, and start the scheduler.

7. Edit the `CMakeLists.txt` file and add/uncomment the appropriate entries to add the appropriate FreeRTOS files and directories into the build. Also make sure you add `hello.c` to the source file list.

8. Edit `startup_ARMCM3.S` and set the vector for the system timer (systick).

Your directory and file structure are shown in Figure 1. However, when you run `cmake`, it will create some additional files and directories that are not shown in the figure.

```
FreeRTOS/*
CMSIS/*
CMakeLists.txt
include/
    FreeRTOSconfig.h
    device_addrs.h
    hello_task.h
    uart.h
src/
    startup_ARMCM3.S
    main.c
    hello_task.c
    uart.c
    gcc_arm.ld
    system_ARMCM3.c
```

Figure 1: Directory structure after Part 1 is completed.

Part 2: Adding a Task to Report Statistics

FreeRTOS provides some facilities to support reporting run-time statistics. run-time statistics.

1. Create a new file named `src/stats_task.c` and define the statistics task, its TCB, and its stack.
2. Create a new file named `include/stats_task.h` and declare the statistics task, its TCB, and its stack. The new task should report CPU usage every ten seonds for as long as the system is running.
3. Edit `src/main.c` so that it includes the new header file, and creates the new task.
4. Edit `CMakeLists.txt` as appropriate to add the new C source file.
5. Edit `FreeRTOSconfig.h` as needed to enable the statistics reporting.

Your directory and file structure are shown in Figure 2. However, when you run `cmake`, it will create some additional files and directories that are not shown in the figure.

```
FreeRTOS/*
CMSIS/*
CMakeLists.txt
include/
    FreeRTOSconfig.h
    device_addrs.h
    hello_task.h
    stats_task.h
    uart.h
src/
    startup_ARMCM3.S
    main.c
    hello_task.c
    stats_task.c
    uart.c
    gcc_arm.ld
    system_ARMCM3.c
```

Figure 2: Directory structure after Part 2 is completed.