Peer-graded Assignment: Assignment 2

You passed!
Congratulations. You earned 170 / 200 points. Review the feedback below and continue the course when you are ready. You can also help more classmates by reviewing their submissions.

Review Classmates' Work

Instructions

My submission

Development of Real-Time Systems – Assignment 2

Discussions

Overview

In this assignment you will implement some functionalities learned from the theoretical lessons. We continue using FreeRTOS and some of its more advanced features. In this example we will **focus on task priorities.** We have one high intensity task which uses much of the CPU and one sparse intensity task which mainly uses I/O. The task here is to handle the priorities of the tasks to make sure that no tasks miss the deadline. You will also learn how to measure time in FreeRTOS to evaluate the feasibility of a real-time system in practice.

Help Center

After completing this assignment, you will be able to:

- Handle priorities explicitly in FreeRTOS
- Measure time in FreeRTOS
- Use call back hooks in FreeRTOS

Pre-requisite

It is recommended to use the free software Visual Studio Express for this assignment. The FreeRTOS system you are about to setup will execute in Visual Studio Express, and output from the system is available from its console. More help Here about setting up Visual Studio Express with the FreeRTOS project.

It is also possible to use the free software Eclipse for this assignment. Notes on setting up Eclipse with the FreeRTOS project Here.

Programming ssignment

-Download the FreeRTOS project Here

-Import the project into Visual Studio Express 2015. More information from our Documentation.

-In Visual Studio click the “build” menu and “Build solution”. This will compile the project. More information from our Documentation.

-Now run the project. More information from our Documentation

-Familiarize yourself with the FreeRTOS API and locate to the main() function in the FreeRTOS project in Visual Studio Express

-Create a task "matrixtask" containing the following functionality:

```
#define SIZE 10
#define ROW SIZE
#define COL SIZE
static void matrix_task()
{
  int i;
  double **a = (double **)pvPortMalloc(ROW * sizeof(double*));
  for (i = 0; i < ROW; i++) a[i] = (double *)pvPortMalloc(COL * sizeof(double));
  double **b = (double **)pvPortMalloc(ROW * sizeof(double*));
  for (i = 0; i < ROW; i++) b[i] = (double *)pvPortMalloc(COL * sizeof(double));
  double **c = (double **)pvPortMalloc(ROW * sizeof(double*));
  for (i = 0; i < ROW; i++) c[i] = (double *)pvPortMalloc(COL * sizeof(double));

  double sum = 0.0;
  int j, k, l;

  for (i = 0; i < SIZE; i++) {
    for (j = 0; j < SIZE; j++) {
      a[i][j] = 1.5;
      b[i][j] = 2.6;
    }
  }

  while (1) {
    /*
    * In an embedded systems, matrix multiplication would block the CPU for a
      long time
    * but since this is a PC simulator we must add one additional dummy delay.
    */
    long simulationdelay;
    for (simulationdelay = 0; simulationdelay<1000000000; simulationdelay++)
      ;
    for (i = 0; i < SIZE; i++) {
      for (j = 0; j < SIZE; j++) {
        c[i][j] = 0.0;
      }
    }

    for (i = 0; i < SIZE; i++) {
      for (j = 0; j < SIZE; j++) {
        sum = 0.0;
        for (k = 0; k < SIZE; k++) {
          for (l = 0; l<10; l++) {
```

```
            sum = sum + a[i][k] * b[k][j];
          }
        }
        c[i][j] = sum;
      }
    }
    vTaskDelay(100);
  }
}


```

-Create a task "communicationtask" containing the following functionality:

```
static void communication_task()
{
  while (1) {
    printf("Sending data...\n");
    fflush(stdout);
    vTaskDelay(100);
    printf("Data sent!\n");
    fflush(stdout);
    vTaskDelay(100);
  }
}

```

-Create the tasks in FreeRTOS with the task creation call:

```
xTaskCreate((pdTASK_CODE)matrix_task, (signed char *)"Matrix", 1000, NULL, 3,
  &matrix_handle);
xTaskCreate((pdTASK_CODE)communication_task, (signed char *)"Communication",
  configMINIMAL_STACK_SIZE, NULL, 1, &communication_handle);

```

-"communicationtask" must send a simulated data packet every 200ms but is often blocked by matrixtask, fix this problem without changing the functionality in the tasks.

-Create a new task "prioritysettask" which:

1. Sets the priority of "communicationtask" to 4 in case its execution time is more than 1000 milliseconds (Hint: look at vApplicationTickHook() to measure it)
2. Sets the priority of "communicationtask" to 2 in case its execution time is less than 200 milliseconds (Hint: look at vApplicationTickHook() to measure it)

-Provide a screenshot of the execution and answer the following questions in a report:

- Why is "matrixtask" using most of the CPU utilization?
- Why must the priority of "communicationtask" increase in order for it to work properly
- What happens to the completion time of "matrixtask" when the priority of "communicationtask" is increased?
- How many seconds is the period of "matrixtask"? (Hint: look at vApplicationTickHook() to measure it)

Review criteria less

Everyone enrolled in this course must review at least three other submissions to ensure everyone receives a grade.