Real-time Embedded systems

**Lab1 Task Management in RTOS – Part 1**

Example 1: Creating tasks

This example demonstrates the steps of creating two simple tasks and then starting tasks executing. The tasks simply print out a string periodically, using a null loop to create a period delay. Both tasks are created at the same priority and are identical except for the string that they print.

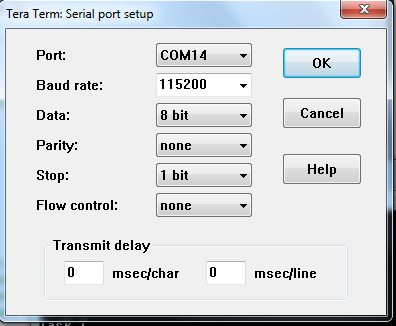
The C file main\_ex1.c includes detailed comments which are very helpful for understanding this code.

1. Open the Atmel Studio project created in the Getting started document.
2. Replace all the code in the file *main1.c* with the code in the *main\_ex1.c* file.
3. First, go through three functions *main(), vTask1()* and *vTask2()* in the file main.c, and pay attention to following points.

* In the task function vTask1 (or vTask2), there is an infinite *for(;;)* loop. All computations performed by a task should be implemented within this loop.
* The main() function creates two tasks before starting the scheduler.
* Study the usage of xTaskCreate() function which creates new task instances to be scheduled. This library function requires six input parameters.

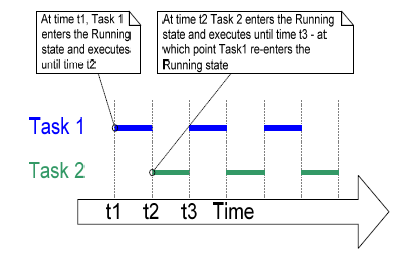
Explain the meaning of each of six arguments passed to the this function when creating two task instances *vTask1* and *vTask2* in the *main()* function.

1. Select F7 to build the project; then, select Alt+F5 (or click icon  on the tool bar) to start debugging session.
2. Open the *Tera Term* window and configure the serial port by selecting Setup>> Serial port ….. . The setup window as shown below pops up. Configure the parameters as below.



1. Run the project by selecting F5 (or clicking icon **).

Both vTask1 and vTask2 are running at the same priority, and so share CPU time on the single processor. Their actual execution pattern should be as the following.



The arrow along the bottom of the above figure shows the passing of time from time t1 onwards. The colored lines show which task is running at each point in time.

It is also possible to create a task within another task. You could have created Task 1 from main(), and then created Task 2 from within Task 1.

* Move the statement of creating Task 2 instance in main() into Task 1 immediately before the infinite *for(;;)* loop.
* Rebuild and execute the project.

Thus, Task 2 would not get created until the scheduler had been started. Please compare the output produced by the modified example with the one from the original example and check if they would be same.

Example 2: Using the task parameter

The two tasks *vTask1* and *vTask2* created in Example 1 are almost identical, except the text string they print out. We will create a single task implementation and use two instances of this same function to remove the duplication. The parameter *pvParameters* of a task function is used to pass into each task instance the string that it should print out.

1. Define two global (i.e. outside any function) string variables that will be passed in as the task parameters. They are created as *const char \** to ensure they are not allocated in the stack and will remain valid when the tasks are executing.
2. Create a new task function named as *vTaskFunction*. The easiest way is to modify the original definition of *vTask1* instead of coding from scratch.
3. Change the type of variable *pcTaskName* from *const char \** to *char \** without the value initialization.
4. Add a statement to assign the function input parameter *pvParameters* to *pcTaskName*. Remember cast the type from *void \** to *char \**.
5. In the *main* function, modify the two *xTaskCreate()* function calls.

* Replace the first parameter vTask1 (or vTask2) with the newly created function name *vTaskFunction* from step 2;
* Replace the 4th parameter NULL with the corresponding string variables defined earlier.

1. Same as the steps 4 – 6 in Example 1.

Example 3: Experimenting with priorities

In the example 1 and 2, two tasks have been created at the same priority, so both entered and exited the Running state in turn. This example looks at what happens when we change the priority of one of the two tasks created in Example 2.

* In the main() function, assign the priority of the first task instance at 1,and the priority of the second task instance at 2.

The single function vTaskFunction() that implements both tasks remain the same: it still simply prints out a string periodically using a null *for* loop to create a delay.

* Rebuild and execute the project.

The following figure shows the expected execution sequence.

