Real-time Embedded systems

Lab6 Resource Management

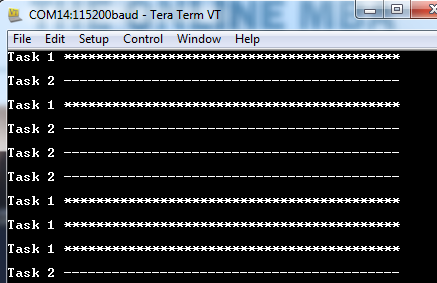
Example 15 Rewriting vPrintString() to use a semaphore

This example creates a new version of *vPrintString()* called *prvNewPrintString()*, then calls the new function from multiple tasks. *prvNewPrintString()* is functionally identical to *vPrintString()*, but uses a mutex to control access to standard out in place of the basic crtical section.（控制对标准输出的访问，而不是基本的标准部分。）

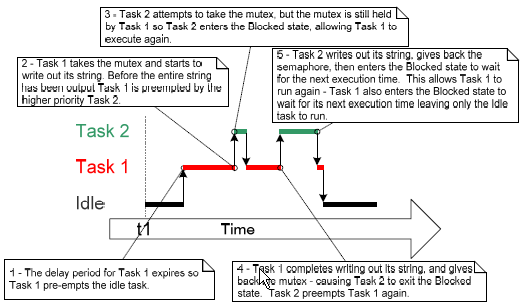
*prvNewPrintString()* iscalled repeatedly by two instances of a task implemented by *prvPrintTask*(). A random delay time is used between each call. The task paramter is used to pass a unique string into each instance of the task.

As normal, *main()* creates the mutex, two tasks, and then starts the scheduler. The two instances of *prvPrintTask*() are created at different priorities, so the lower priority task will sometimes be pre-empted by the higher priority task. A mutex is used to ensure each task gets mutally exclusive access to the terminal, even when pre-emption occurs, the strings that are displayed will be corrected and in no way corrupted. The frequency of the pre-emption can be increased by reducing the maximum time the tasks spend in the Blocked state, which is defaulted to 0x1ff ticks.

The output produced when Example 15 is executed is shown below.



A possible execution seuqnece is described in the following.



Example 16 Rewriting vPrintString() to use a gatekeeper task

Please edit the FreeRTOSConfig.h file.

Set the macro configUSE\_TICK\_HOOK to 1 as below.

#define configUSE\_TICK\_HOOK 1

This example provides an alternative implementation for vPrintString(). This time, a gatekeeper task is used to manage access to standard out. When a task wants to write a message to the terminal, it does not call a print function directly but, instead, sends the message to the gatekeeper.

The gatekeeper task uses a FreeRTOS queue to serialize access to the terminal. The internal implementation of the task does not have to consider mutual exclusion because it is the only task permitted to access the terminal directly.

The gatekeeper task spends most of its time in the blocked state, waiting for messages to arrive on the queue. When a message arrives, the gatekeeper writes the message to the standard out, before returning to the Blocked state to wait for the next message.