EE458 – Embedded Systems Programming
Project 3: Using Semaphores and Message Queues

Create an application with two tasks. There is one I/O task and one computation tasks. Both tasks should be created and started in the RTEMS Init() task. The RTEMS Init() task should delete itself after completing all application initialization.

The computation task should calculate an approximation to π based on the following infinite series:

$$\pi = 4\left\{1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} + \cdots\right\}$$

The I/O task should display information about the application and then enter an infinite loop where it prompts the user to enter the number of terms (n) to be using in the series approximation to π. The number of terms to be used in the approximation should be passed to the computation task via a message queue. The computation task should complete the calculation and release a semaphore to signal the I/O task that it has finished. The I/O task should display the approximate value and then prompt the user to repeat the calculation using a new value of n. The approximate value of π should be stored in a global variable so that it can be accessed by both tasks.

A sample session might look something like this:

```
This program will calculate an approximate value of pi using a
certain mathematical series.

Enter the number of terms to use in the series: 1
The approximate value of pi using 1 term is 4
Enter the number of terms to use in the series: 10
The approximate value of pi using 1 term is 3.04184
Enter the number of terms to use in the series: 0
Invalid input, the number of terms must be an integer greater
than 0.
Enter the number of terms to use in the series: 10000
The approximate value of pi using 1 term is 3.14149
```

Submit **a2ps** printout of your source code. Submit your source code and your RTEMS executable in a zip or tar archive by email to richardson.tony@gmail.com. In the email subject line use "EE458 Project 3 – Your Name".

Notes:
1. This series converges fairly slowly. You will need to use a large number of terms to get an approximation that is accurate to several digits.
2. The I/O task uses the floating point processor to display the number via printf(). As long as the application is structured properly (so that the computation is never preempted during a computation and the I/O task is never preempted while using printf(), there is no need to save the FPU registers during a task switch (although it doesn't hurt either).