Cooperative Multitasking

If you disable Round-Robin Multitasking you must design and implement your tasks so that they work **cooperatively**. Specifically, you must call the system wait function like **os_dly_wait()** function or the **os_tsk_pass()** function somewhere in each task. These functions signal RTX Kernel to switch to another task.

The following example shows a simple RTX program that uses Cooperative Multitasking. RTX Kernel starts executing task 1. This function creates task 2. After the counter1 is incremented once, RTX Kernel switches to task 2. After the counter2 is incremented once, RTX Kernel switches back to task 1. This process is repeated indefinitely.

```
#include <RTL.h>

int counter1;
int counter2;

void task1 (void) __task;
void task2 (void) __task;

void task1 (void) __task {
   os_tsk_create (task2, 0); /* Create task 2 and mark it as ready */
   for (;;) {                /* loop forever */
      counter1++;            /* update the counter */
      os_tsk_pass ();        /* switch to 'task2' */
   }
}

void task2 (void) __task {
   for (;;) {                /* loop forever */
      counter2++;            /* update the counter */
      os_tsk_pass ();        /* switch to 'task1' */
   }
}

void main (void) {
   os_sys_init(task1);       /* Initialize RTX Kernel and start task 1 */
   for (;;);
}
```

The difference between system wait function and os_tsk_pass is that system wait allows your task to wait for an event, while os_tsk_pass switches to another ready task immediately.

Note

- The **os_tsk_pass** will not switch to the next ready task, if this one has **lower priority** than the currently running task.