

Lab 1 for uC/OS-II: Periodic Task Emulation

Prof. Li-Pin Chang
ESSLab@NCTU

Objectives

- To implement periodic tasks
- To observe the scheduling behaviors
 - Fixed priority scheduling

Task Sets

- Two sets of periodic tasks
 - Task set 1 = { $t_1(1,3)$, $t_2(3,6)$ }
 - Task set 2 = { $t_1(1,3)$, $t_2(3,6)$, $t_3(4,9)$ }
 - Tasks all arrive at the same time
 - Show context switch behaviors
 - Show deadline violations if there is any

Issues

- How to create a task that executes exactly c units of time in every p units of time?
 - (c,p)
- Where in the kernel can we add code to observe context switches?
 - Voluntarily; complete
 - Involuntarily; preempted

Notices

- In actual real-time systems, task jobs are released by periodic interrupts
- Task computation time is determined by worst-case computation time analysis (WCET)
- In this project we emulate this behavior, and, more importantly, to get insights into how CPU time is allocated to tasks

Periodic tasks

- Call OSTaskCreate to create a task

```
208 static void TaskStartCreateTasks (void)
209 {
210     OSTaskCreate(Task1, (void *)0, &TaskStk[0][TASK_STK_SIZE - 1], 1);
211     OSTaskCreate(Task2, (void *)0, &TaskStk[1][TASK_STK_SIZE - 1], 2);
212 }
```

Task function Pointer

Extra Parameter

Stack Pointer

Priority

- A basic task template

while(1)

{

... do task computation ...

... up to C units of time ...

OSPendSem(); // posted by an ISR at a regular frequency

}

Periodic tasks

- A basic task template

```
while(1)
```

```
{
```

```
    ... do task computation ...
```

```
    ... up to C units of time ...
```

```
    OSPendSem();      // posted by an ISR at a fixed frequency
```

```
}
```

- A straightforward emulation of (c,p)

```
while(1)
```

```
{
```

```
    Start=OSTimeGet() ;
```

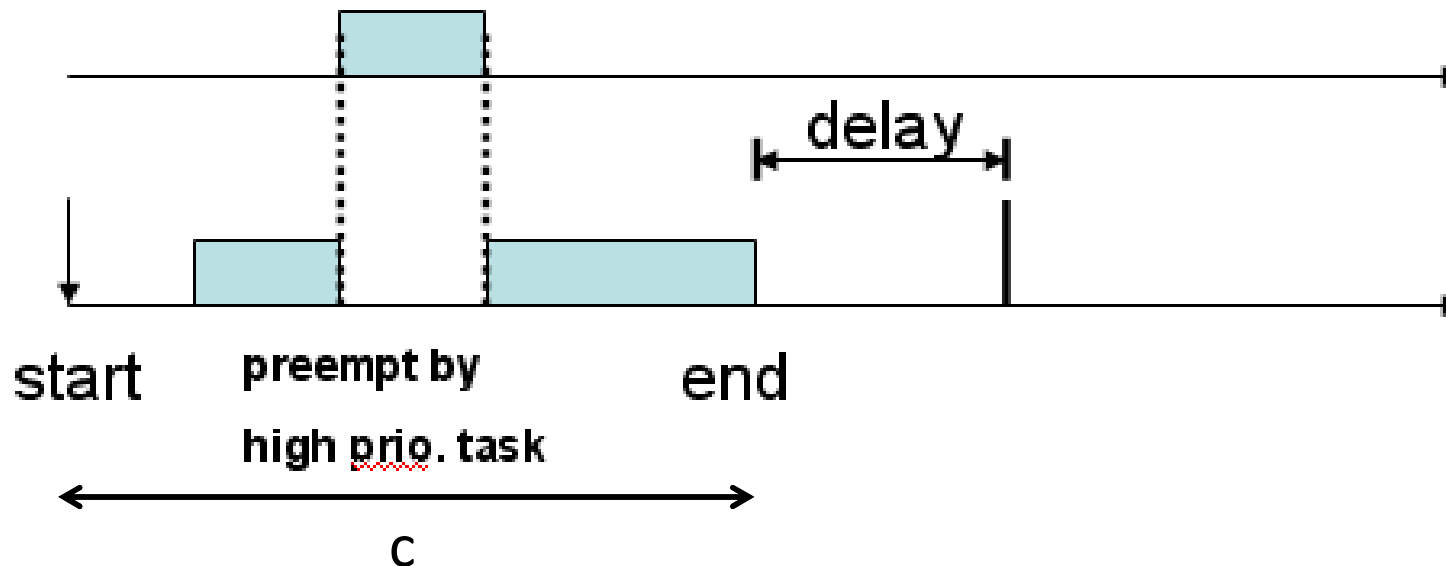
```
    While(OSTimeGet()-start < c) ;
```

```
    OSTimeDly (p-c) ;
```

```
}
```

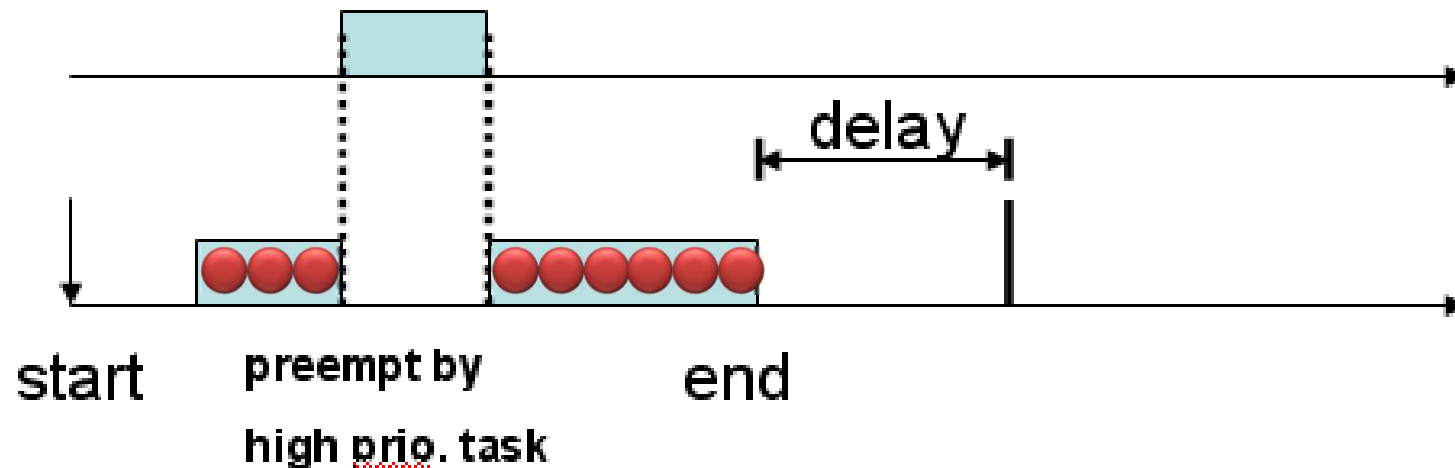
Periodic task

- Problem : the task does not receive “c” units of CPU time if it was preempted in [start,end]



Periodic task

- c = clock ticks spent on the task
- *delay* is still $p - (\text{end} - \text{start})$

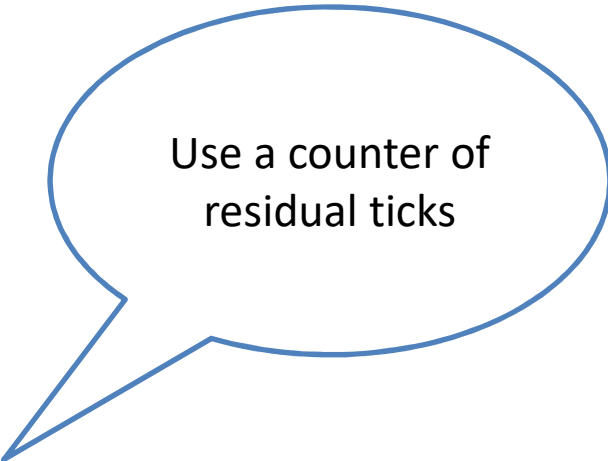


Idea

- Use an “execution counter”, just like the “delay counter”
 - Decrement when the corresponding task uses 1 tick of CPU time
- Struct OS_TCB
 - A per-task data structure, defined in uCOS-II.h
 - Add a variable compTime to store the residual clock ticks of a task
 - replenished to “c” at the beginning of every period
 - Add a variable of task period

Periodic task

```
void Task()
{
    int start ; //the start time
    int end ;   //the end time
    int toDelay;
    start=OSTimeGet();
    while(1)
    {
        while(OSTCBCur->compTime>0) //C ticks
        {
            // do nothing
        }
        end=OSTimeGet() ; // end time
        toDelay=(OSTCBCur->period)-(end-start) ;
        start=start+(OSTCBCur->period) ; // next start time
        OSTCBCur->compTime=C ;// reset the counter (c ticks for computation)
        OSTimeDly (toDelay); // delay and wait (P-C) times
    }
}
```



OS_ENTER_CRITICAL and OS_EXIT_CRITICAL
should be used to warp the access to
OSTCBCur->compTime

OSTimeTick

- OSTimeTick()
 - Defined in OS_CORE.C, called every time when a clock interrupt arrives
 - Add a piece of code in OSTimeTick to decrement the compTime counter in **the running task's** os_tcb
 - The current task has consumed 1 tick

OSInitExit

- OSIntExit()
 - Defined in OS_CORE.C
 - This function will manage the scheduling after the system has come back from the calling of ISR
 - We need to print out the “preempt” event here

OS_Sched

- OS_Sched()
 - Defined in OS_CORE.C
 - OS_Sched() is called when a task is voluntarily giving up its possession of the CPU
 - We need to print out the “complete” event here

Related Function

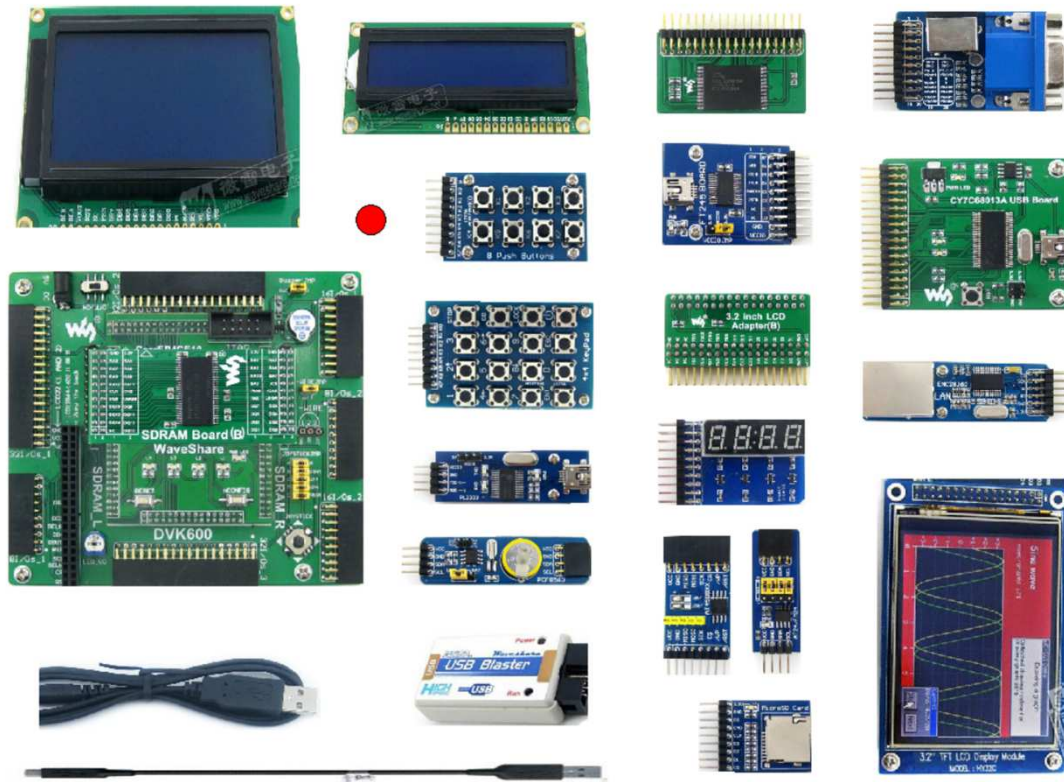
- OSStart():
 - This is function will try to find the task with the highest priority and schedule it to run.
 - Called only once when the system executing tasks for the very first time
 - This function is defined in OS_CORE.C

Printing messages

- Print messages
 - There's a printf that you can use
 - E.g., `printf("\n%10d Preempt ",timestamp);`

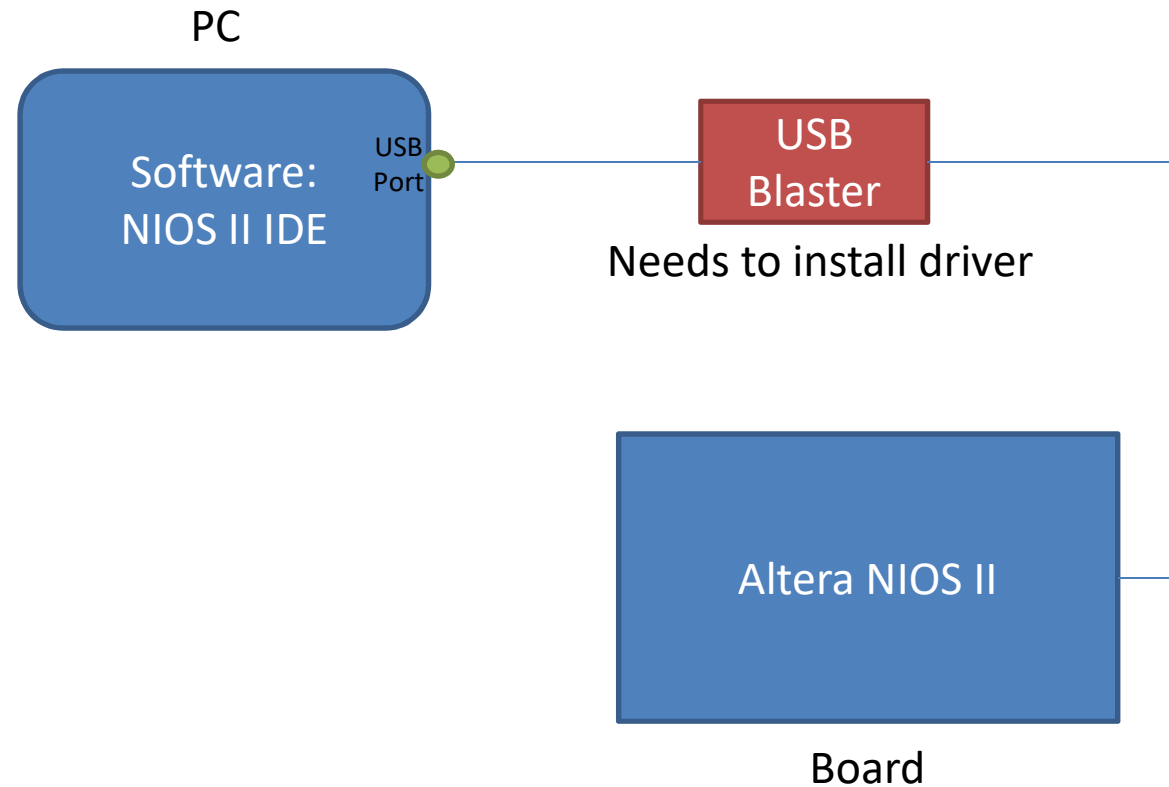
Evaluation boards

- Altera NiosII

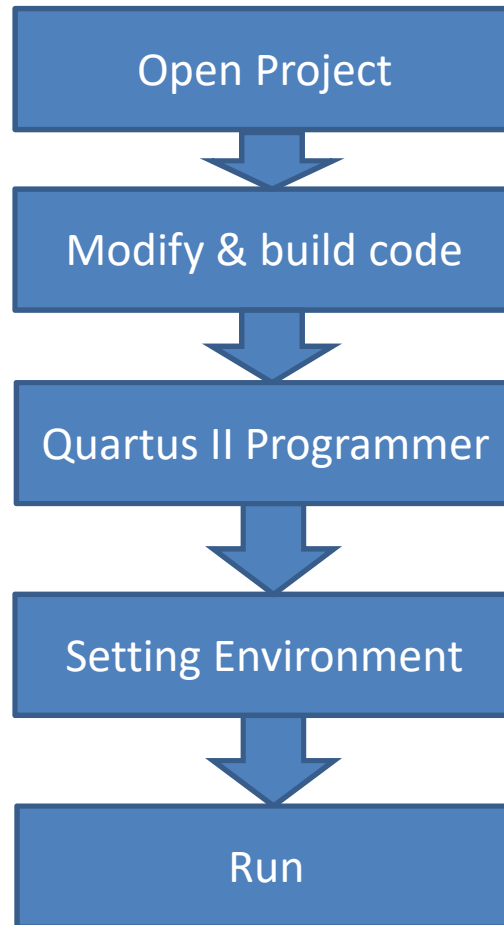


Architecture

Altera NIOS II



Board: Altera NIOS II




Open Project

- Open NIOS II -> set workspace: C:\cps\workspace
- Create Project
 - File -> New -> Nios II Application and BSP from Template
- New project setting
 - SOPC Information File name -> C:\cps\workshop\nios2ucosii\CORE_SOPC.sopcinfo
 - Select Project Template: Hello MicroC/OS-II
 - Finish

Porting: Quartus II Programmer

- NIOS II programmer
 - NiosII -> QuartusII Programmer
- Quartus II setting
 - Add File
 - Select “C:\cps\workshop\nios2ucosii\standard.sof”
- Hardware Setup -> USB-Blaster
- Start
- Close Quartus II

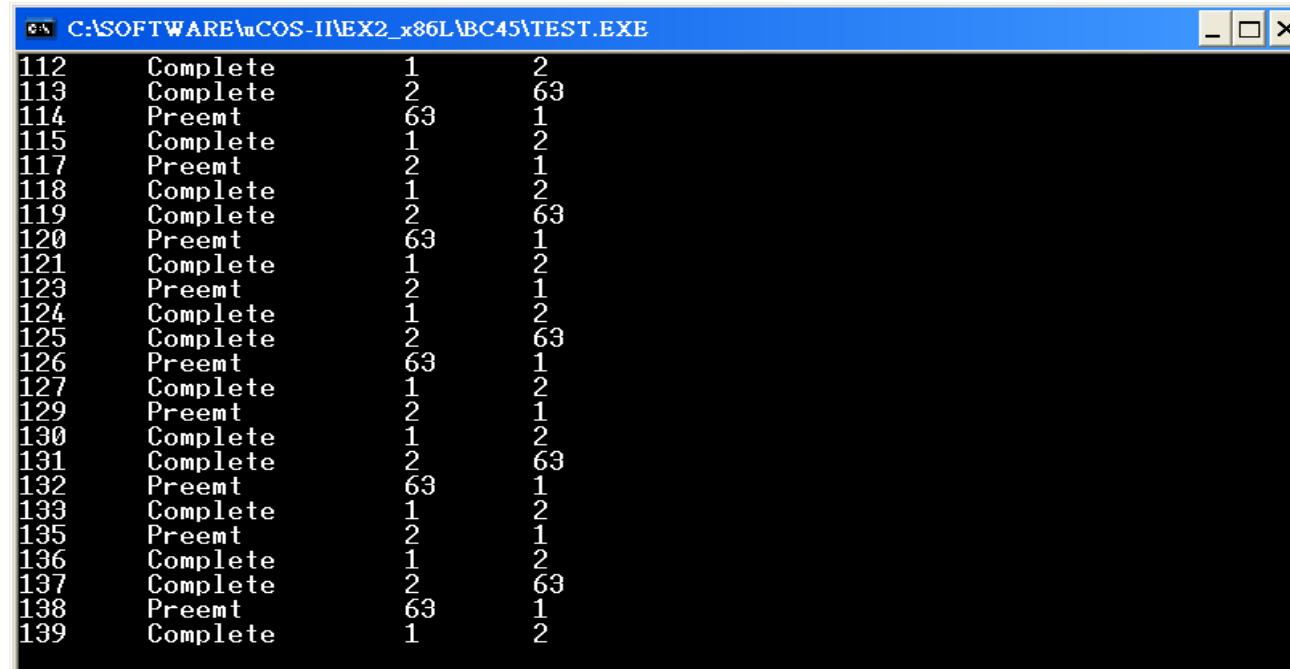
Run

- click Run 
 - Select NiosII Hardware
 - Target Connect -> Refresh connection -> Apply -> Run

Output Results

- expected output:

Current time	Event	[From Task ID]	[To Task ID]
Time tick	Preempt	TaskID(priority)	TaskID(priority)
Time tick	Complete	TaskID(priority)	TaskID(priority)



```
C:\SOFTWARE\mCOS-II\EX2_x86L\BC45\TEST.EXE
112 Complete 1 2
113 Complete 2 63
114 Preemt 63 1
115 Complete 1 2
117 Preemt 2 1
118 Complete 1 2
119 Complete 2 63
120 Preemt 63 1
121 Complete 1 2
123 Preemt 2 1
124 Complete 1 2
125 Complete 2 63
126 Preemt 63 1
127 Complete 1 2
129 Preemt 2 1
130 Complete 1 2
131 Complete 2 63
132 Preemt 63 1
133 Complete 1 2
135 Preemt 2 1
136 Complete 1 2
137 Complete 2 63
138 Preemt 63 1
139 Complete 1 2
```

Output Results

- Example Taskset = {t1(1,2), t2(2,4)}
 - Suppose program start at time tick 1
 - System time is the “OStime” global variable

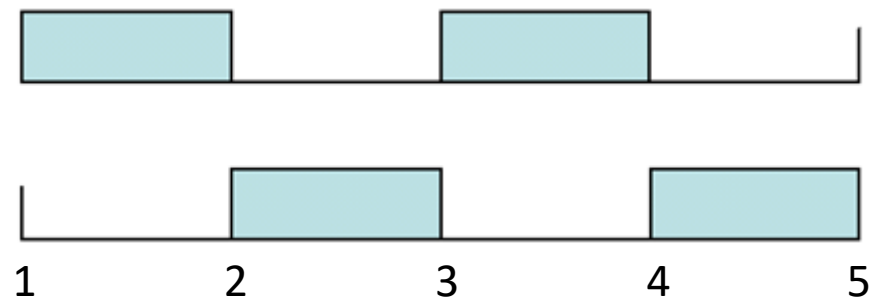
Time event from to

1 Preempt 63 1

2 Complete 1 2

3 Preempt 2 1

4 Complete 1 2



Output Results

altera NIOS II

Problems Console X Properties			
hello_ucosii_0 Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (12/12/07 2:09 PM)			
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)			
1	Complete	Task1(0)	Task2(1)
3	Preempt	Task2(1)	Task1(0)
4	Complete	Task1(0)	Task2(1)
5	Complete	Task2(1)	IdleTask(63)
6	Preempt	IdleTask(63)	Task1(0)
7	Complete	Task1(0)	Task2(1)
9	Preempt	Task2(1)	Task1(0)
10	Complete	Task1(0)	Task2(1)
11	Complete	Task2(1)	IdleTask(63)
12	Preempt	IdleTask(63)	Task1(0)
13	Complete	Task1(0)	Task2(1)
15	Preempt	Task2(1)	Task1(0)

More Information

- Remember to save your code for further use in Lab 1 and 2
- You can use `OSTimeSet(0)` to reset the tick counter if necessary
- Classroom: **EC222 (!)**

Grading

- Produce the correct schedules for the following tasks using RM
 - $\{ (1,3), (3,6) \}$
 - $\{ (1,3), (3,6), (4,9) \}$