# Task Execution

(slides are based on Prof. Dr. Jian-Jia Chen and http://www.freertos.org)

Dr. Kuan-Hsun Chen

**LS 12, TU Dortmund**

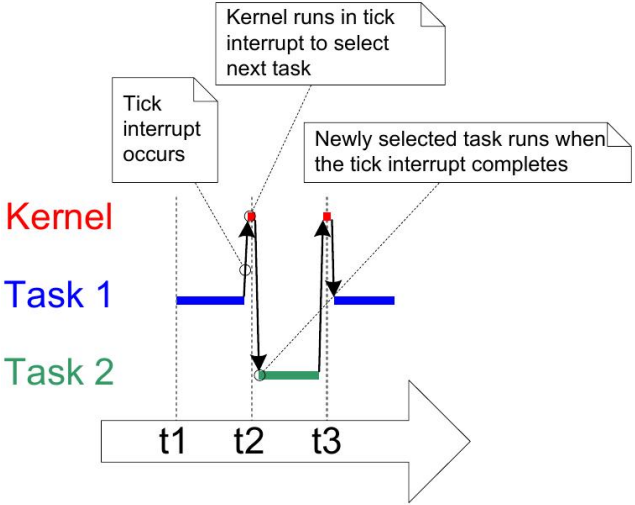
December 5, 2019

# Task Creation (Recall)

- All the related information of a task is stored in a task control block (TCB) so that the operating systems can use it for operating on a task.

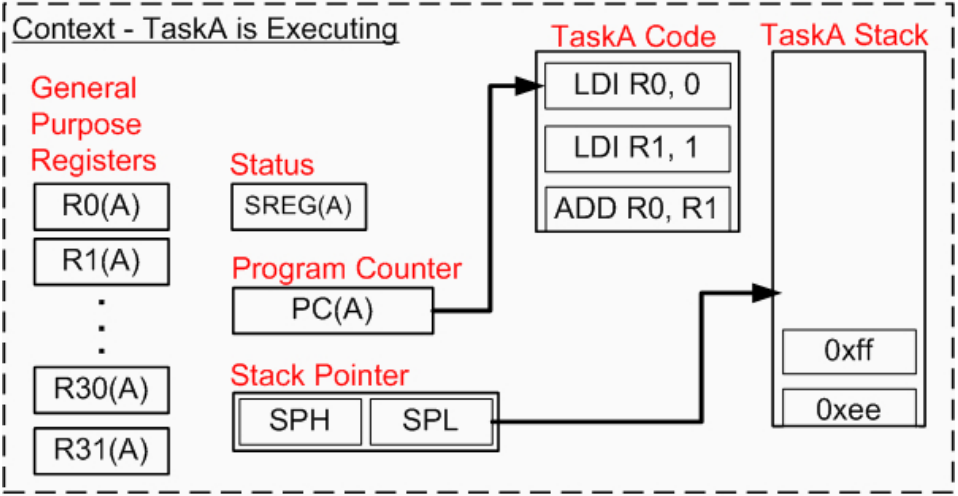
| | |
|---|---|
| Top of Stack | pointer to the last item place on the stack for the task |
| Task State | list item used to place the TCB in ready and blocked queues |
| Event List | list item used to place the TCB in the event lists |
| Priority | task priority (0=lowest) |
| Stack Start | pointer to the start of the process stack |
| Others | other information |

# Task Execution Sequence



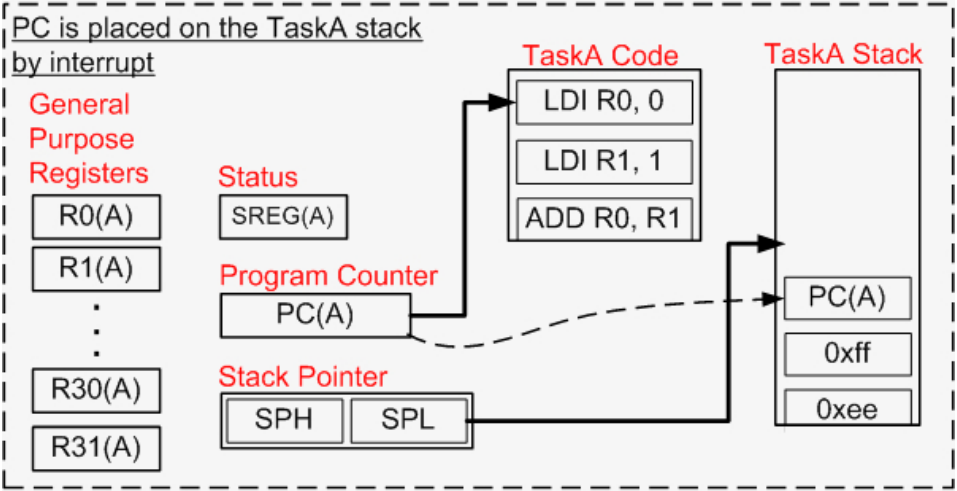

[http://www.freertos.org]

# Task Execution - Prior to the RTOS tick interrupt



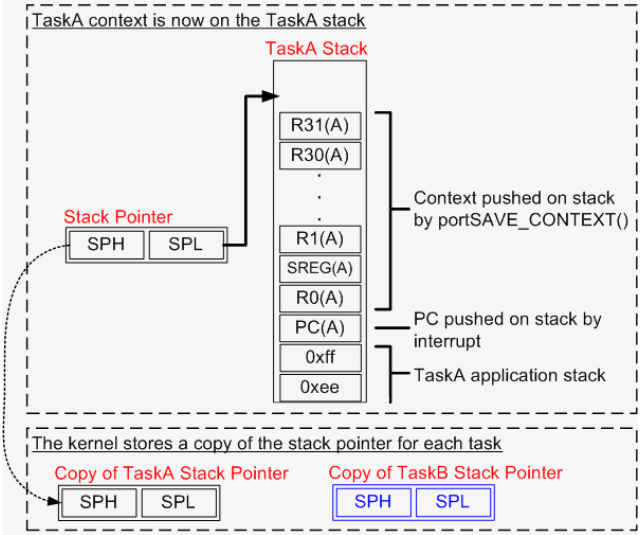

[http://www.freertos.org]

# Task Execution - RTOS tick interrupt occurs




[http://www.freertos.org]

# Task Execution - RTOS tick interrupt is executed

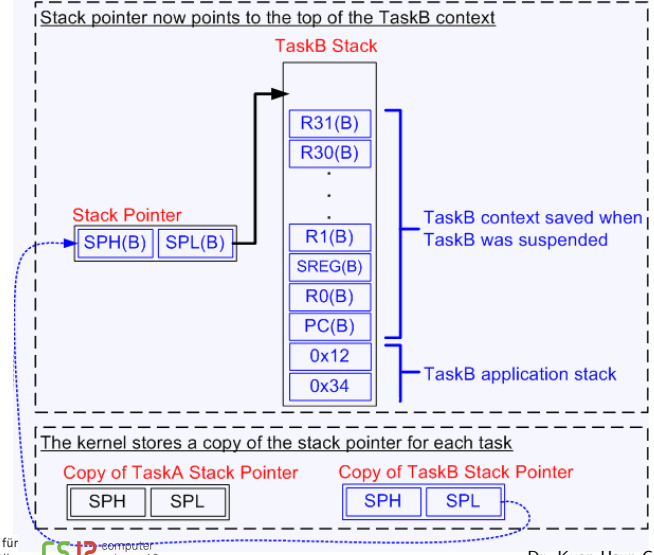



[http://www.freertos.org]

# Incrementing the Tick Count

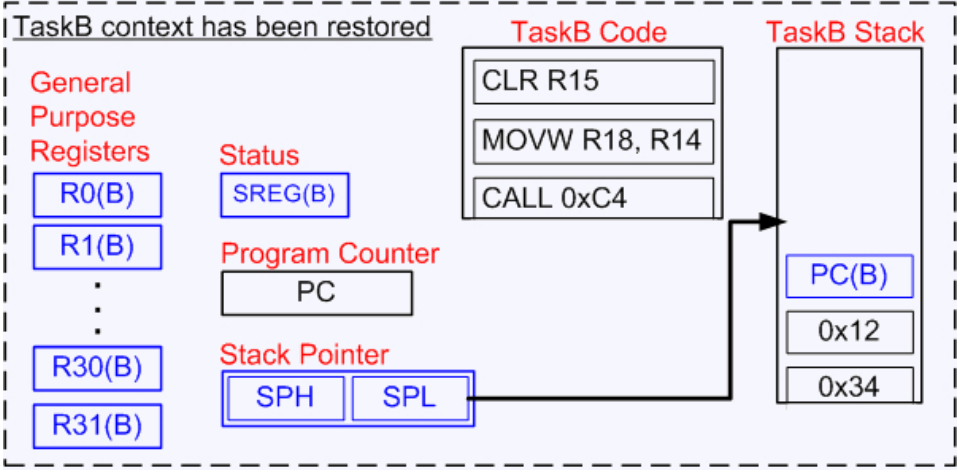
- *vTaskIncrementTick()* executes after the TaskA context has been saved.
- TaskB has higher priority than TaskA and is ready to run.
- *vTaskSwitchContext()* selects TaskB as the task to be given processing time when the ISR completes.

# Task Execution - TaskB stack pointer is retrieved



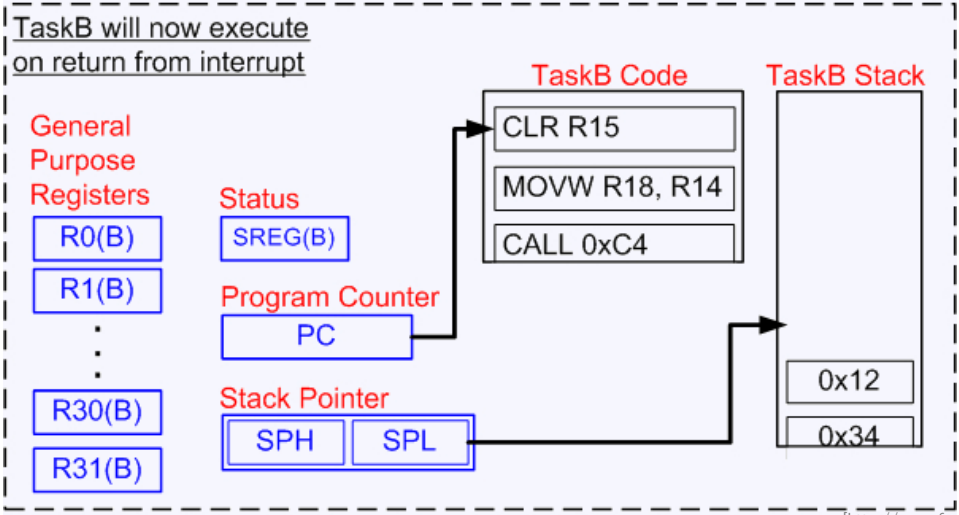

[http://www.freertos.org]

# Task Execution - Restore the context of TaskB




[http://www.freertos.org]

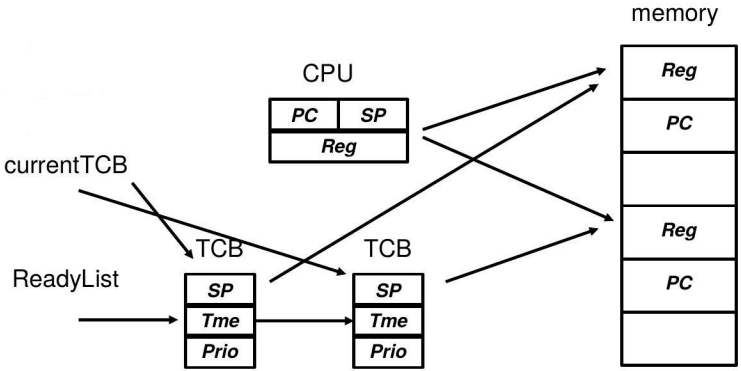
# Task Execution - The RTOS tick exits




[http://www.freertos.org]

# Task Control Block and Task Execution

- Central Processing Unit (CPU)
  - Program Counter (PC), Stack Pointer (SP), Registers (Reg)
- Task Control Block (TCB)
  - Stack pointer, Waiting time (Tme), priority (Prio)



# Task Control Block in FreeRTOS - struct tskTaskControlBlock

```
typedef struct tskTaskControlBlock{
    /*< Points to the location of the last item placed on the stack. */
    volatile portSTACK_TYPE *pxTopOfStack;
    ...
    /*< List item used to place the TCB in ready and blocked queues. */
    xListItem  xGenericListItem;
    /*< List item used to place the TCB in event lists. */
    xListItem  xEventListItem;
    /*< The priority of the task where 0 is the lowest priority. */
    unsigned portBASE_TYPE  uxPriority;
    /*< Points to the start of the stack. */
    portSTACK_TYPE *pxStack;
    /*< Descriptive name given to the task when created for debugging */
    signed char pcTaskName[ configMAX_TASK_NAME_LEN ];

    .....

} tskTCB;
```

# xTaskGenericCreate

```
signed portBASE_TYPE xTaskGenericCreate( ... ){
    ...
    tskTCB * pxNewTCB;
    configASSERT( pxTaskCode );
    configASSERT( ( ( uxPriority & ( ~portPRIVILEGE_BIT ) ) < configMAX_PRIORITIES ) );
    /* Allocate the memory required by the TCB and stack for the new task, checking that the allocation was successful. */
    pxNewTCB = prvAllocateTCBAndStack( usStackDepth, puxStackBuffer );
    if( pxNewTCB != NULL )  {
    ....
        /* Setup the newly allocated TCB with the initial state of the task. */
        prvInitialiseTCBVariables( pxNewTCB, pcName, uxPriority, xRegions, usStackDepth );
        ....
```

# xTaskGenericCreate (cont.)

```
/* We are going to manipulate the task queues to add this task to a ready list, so must make sure no interrupts occur. */
taskENTER_CRITICAL();        {
    uxCurrentNumberOfTasks++;
    if( pxCurrentTCB == NULL )          {
    /* There are no other tasks, or all the other tasks are in the suspended state - make this the current task. */
        pxCurrentTCB =  pxNewTCB;
    ....}
    else{
    /* If the scheduler is not already running, make this task the current task if it is the highest priority task to be created so far. */
        if( xSchedulerRunning == pdFALSE ){
            if( pxCurrentTCB->uxPriority <= uxPriority ){
                pxCurrentTCB = pxNewTCB;}
        }
    }
    ....
```

# xTaskGenericCreate (cont.)

```
            prvAddTaskToReadyQueue( pxNewTCB );
            xReturn = pdPASS;
            ...}
        taskEXIT_CRITICAL();
    }
    ...
    if( xReturn == pdPASS ){
        if( xSchedulerRunning != pdFALSE ){
            /* If the created task is of a higher priority than the current task then it should run now. */
            if( pxCurrentTCB->uxPriority < uxPriority ){
                portYIELD_WITHIN_API();
            }
        }
    }
    return xReturn;
}
```

Let's read the source code for a while.