HOGESCHOOL ROTTERDAM

Real-Time Operating Systems

ROS01

Minor Embedded Systems

Week 4
Pre-emptive Scheduling

Planning ROS01

- Week 1: Introduction – Blinking leds
- Week 2: Super loop construct with an ISR
- Week 3: Cooperative Scheduling
- **Week 4: Pre-emptive Scheduling**
- Week 5: Using TI-RTOS
- Week 6: Schedulability Analyses, Priority Assignment
- Week 7: Response Time Analyses
- Week 8: Finalizing Final Assigment

Overview

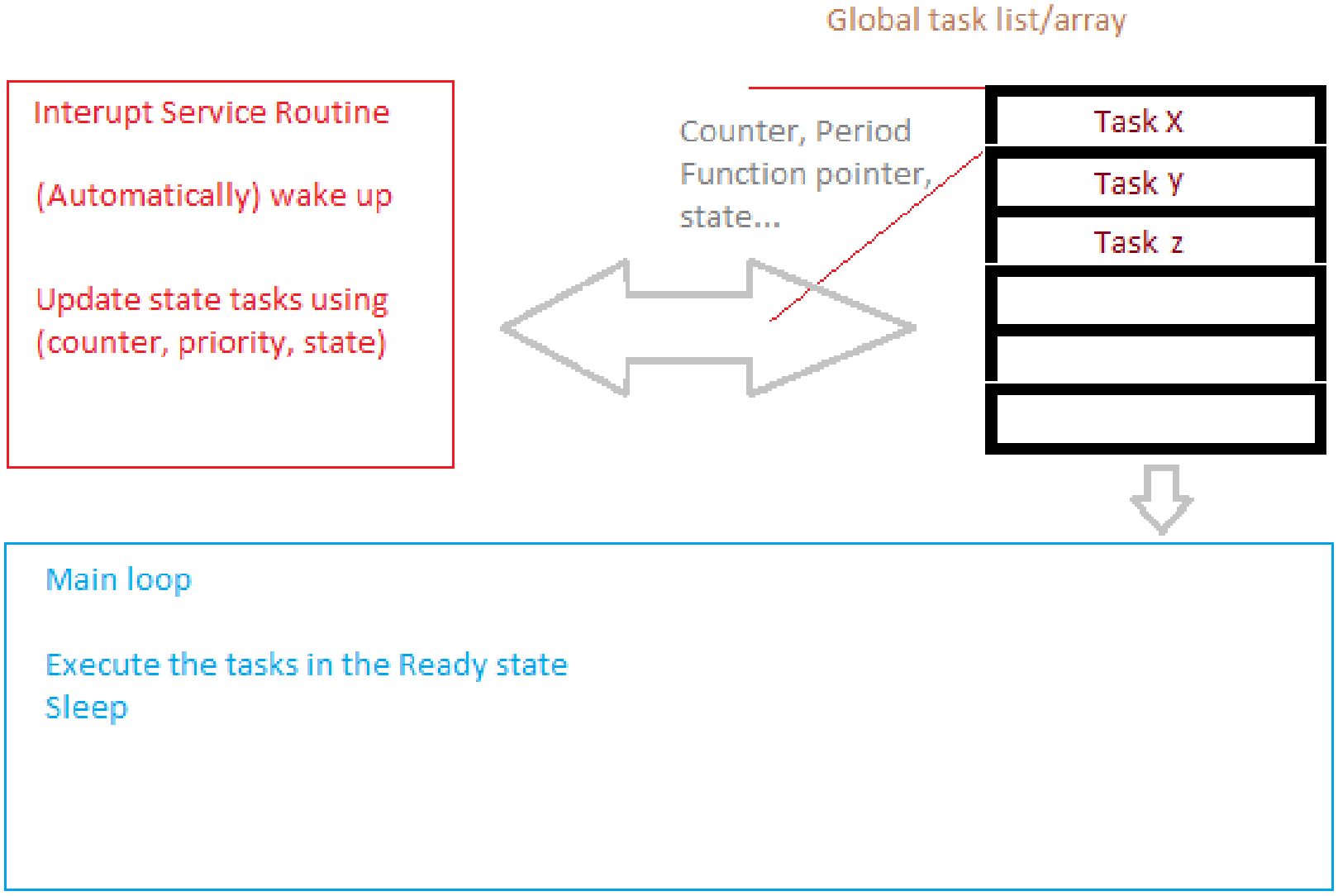
- Scheduling
  - Problem
  - Goal
  - Possible solution

Scheduling

- Problem
  - Multiple processes require CPU time
    - Some processes need it asap
    - Some processes just need to happen at some point in time
  - Multiple processes require bandwidth
    - USB, Serial, SPI ....
    - Prioritization?

- Goal
  - Create a framework that'll ease (CPU) time management
  - Easy to add new processes and to share resources

Review Cooperative Scheduler



Demo

- Demonstration of pre-emptive project

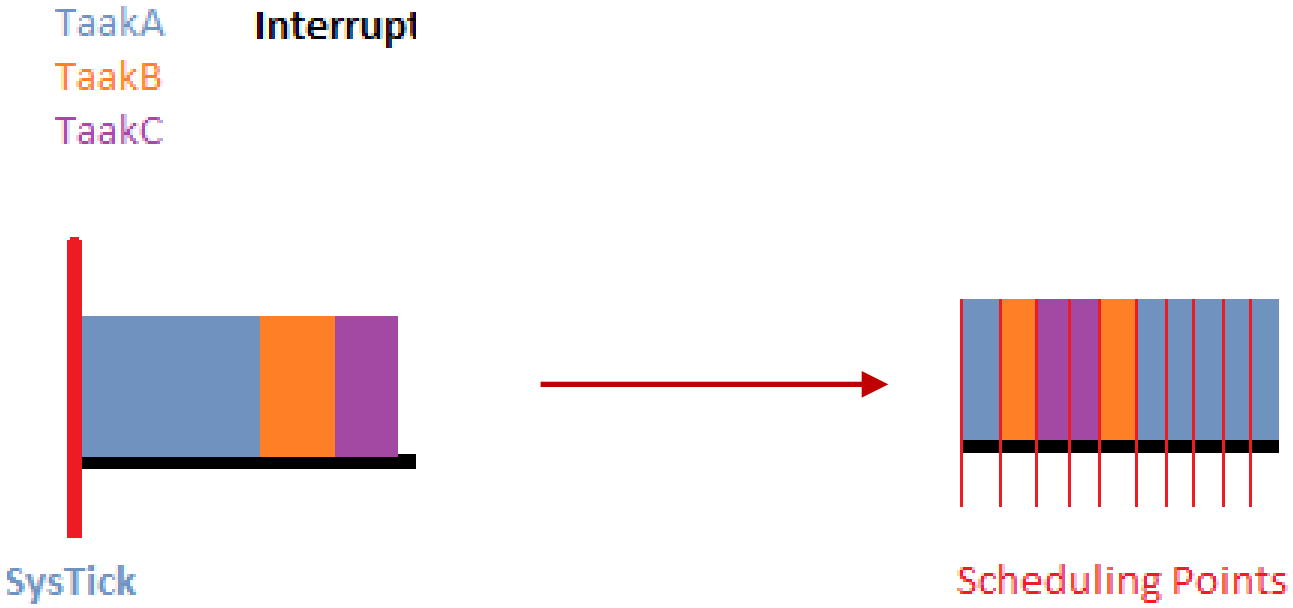
Cooperative versus Pre-emptive scheduling

- Cooperative
  - Tasks run sequentially
  - High priority tasks have to wait till last task finishes
  - Easy to set up
  - Low overhead scheduler

- Pre-emptive (Multi-tasking)
  - Important tasks always finish first
  - Danger of starvation and using hardware concurrently
  - More overhead on resources(RAM) and CPU time

Pre-emption

- Interrupting a task to execute a different task



- Scheduler decides next task
- Context switch switches the tasks
  - How does it work?

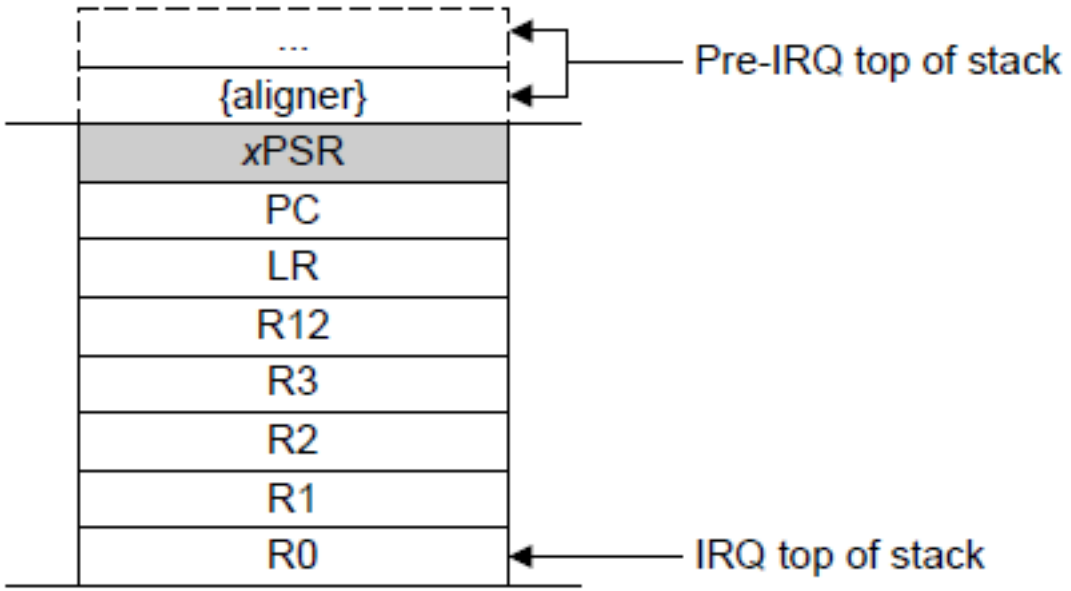
Switching context

```
while(1)
{
    //declare variables
    int a,b,c,d,e,f,g;

    //perform length calculation
    a=b*(c+d/e*f%g);

    //add to output queue
    addToQueue(a);
}
```

1. Enter exception (PendSV)
2. Save context (CPU registers) to stack
3. Switch stack to new task
4. Load context from stack to CPU
5. Leave exception using new stack

```
1  void task()
2  {
3      float samples[256];
4      float output[256];
5
6      while(1)
7      {
8          performFFT(samples);
9      }
10 }
```

Name	Value	Description
Core Registers		
PC	0x00000936	Program Cour
SP	0x20000640	General Purpo
LR	0x00000733	General Purpo
xPSR	0x21000000	Stores the stat
R0	0x00008000	General Purpo
R1	0xE000E100	General Purpo
R2	0x200002FC	General Purpo
R3	0x00000000	General Purpo
R4	0x00000002	General Purpo
R5	0x00000000	General Purpo
R6	0x00000000	General Purpo
R7	0x20000648	General Purpo
R8	0x00000000	General Purpo
R9	0x00000000	General Purpo
R10	0xA4420001	General Purpo
R11	0x400FD108	General Purpo
R12	0x400FD000	General Purpo
R13	0x20000640	General Purpose Register 13 [Core]
R14	0x00000733	General Purpose Register 14 [Core]



Exception frame without floating-point storage

Variables | Expressions | Registers | Breakpoints

	Value	Description
Registers		
C	0x00000BC6	Program Counter [Core]
P	0x20000204	General Purpose Register 13 - Stac...
R	0x00000BD7	General Purpose Register 14 - Link...
PSR	0x61000000	Stores the status of interrupt enab...
0	0x00000000	General Purpose Register 0 [Core]
1	0xFFFFFFFF	General Purpose Register 1 [Core]
2	0x00000002	General Purpose Register 2 [Core]
3	0x00000003	General Purpose Register 3 [Core]
4	0x00000002	General Purpose Register 4 [Core]
5	0x00000000	General Purpose Register 5 [Core]
6	0x00000000	General Purpose Register 6 [Core]
7	0x20000648	General Purpose Register 7 [Core]
8	0x00000000	General Purpose Register 8 [Core]
9	0x00000000	General Purpose Register 9 [Core]
10	0xA4420001	General Purpose Register 10 [Core]
11	0x400FD108	General Purpose Register 11 [Core]
12	0x0000000C	General Purpose Register 12 [Core]
R13	0x20000204	General Purpose Register 13 [Core]
R14	0x00000BD7	General Purpose Register 14 [Core]

Pre-emptive scheduling

- Priority based
  - Scheduler decides and update states of tasks
  - When high priority task comes alive, it interrupts lower priority tasks
  - When all tasks are suspended, the idle task can run
- Round robin
  - Every task gets equal CPU time
  - When all tasks are suspended, the idle task can run
- Demo
  - Instructor demonstrates algorithms

Problems with pre-emptive scheduling

- Starvation
  - Low priority tasks don't get cpu time
    - Possible solution: Aging
- Sharing resources
  - Tasks can't use hardware 'simultaneously'
  - Waiting for hardware to come available can cause deadlock or priority inversion
    - Next week

Assignment week 4/5

- Done with assignment 1-3?
  - Acquire VersdOS
    - https://bitbucket.org/HR_ELEKTRO/ros01_ccs_projecten
    - Import the project into the workspace
    - Possibly select the right compiler version under project settings
- Assignment 4
  - Modify scheduler from round-robin to priority based
  - Modify and optimize code as well as possible
  - Write report chapter about choices and modifications
    - Sunday week 6 23:59- Modulewijzer

Next Week

- TI-RTOS
  - What is it
  - Problems and challenges with
  - Threads and IPC (Inter Process Synchronization)
  - POSIX API overview

Read assignment 5 before next week's lesson!