Chapter 11

Real Time Operating System

Lesson 05

Exemplary Use of RTOS in System Design for of two LEDs ON-OFF program

RTOS

- RTX51 Tiny
- Let us set counts and write simple code without use of Timeout of the timer

Preprocessor Statements

- #include <rtxt51tiny.h>
- int counter0;   .
- int counter1;

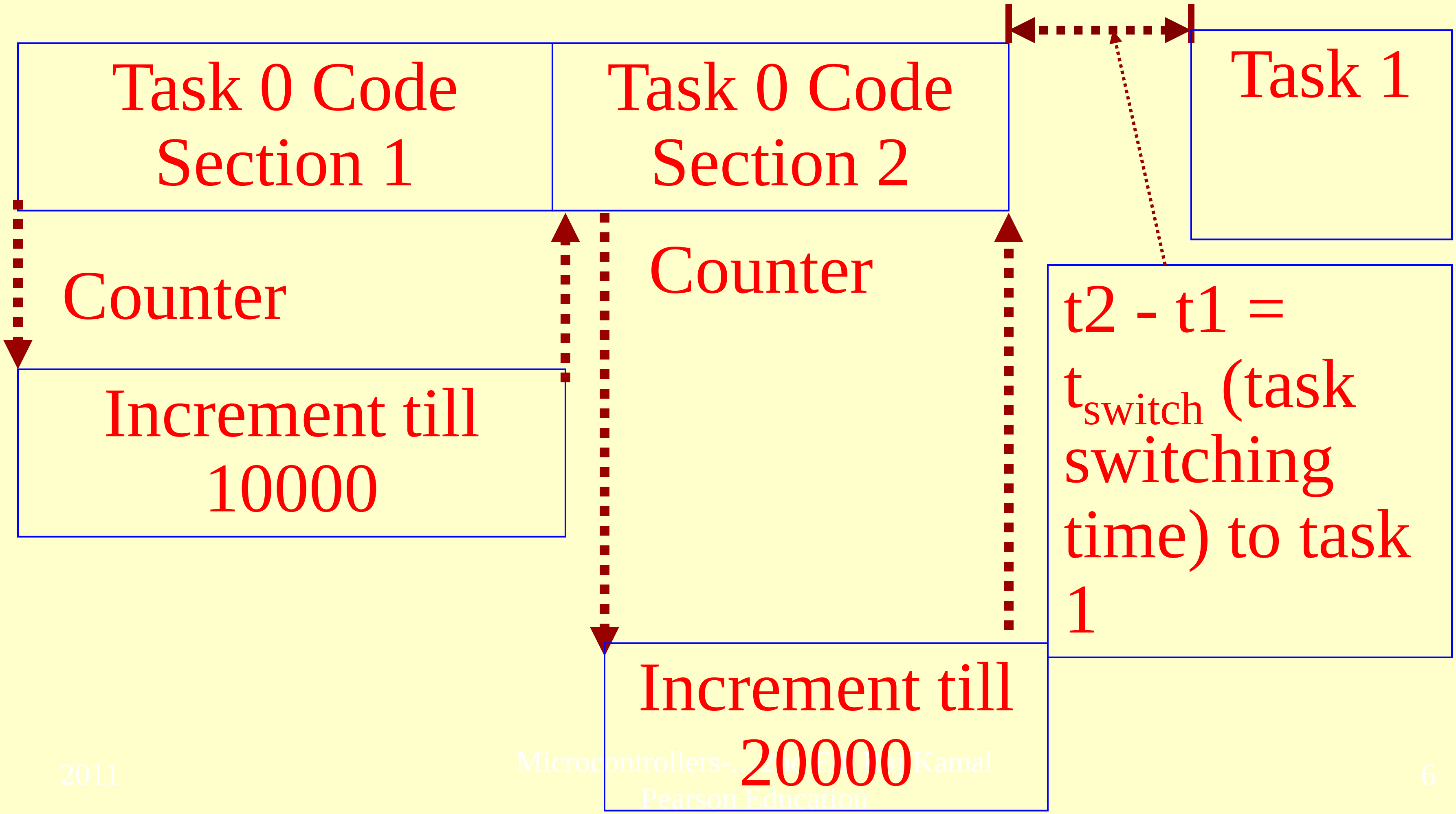
Task 1 Create and infinite in Code for task 0

2. job0 ( ) _task_ 0 { os_create_task (task 1); /* task 1 ready = 0*/

/* Code for LED at port P1.0 OFF for counts < 10000 and ON for counts<20000 */

```
while (1) {
counter0 =0; P1^0 = 0;
while { counter0 < = 10000} {count0++;};
P1^0 = 1;
while { counter0 < = 20000} {count0++;};
 }
```

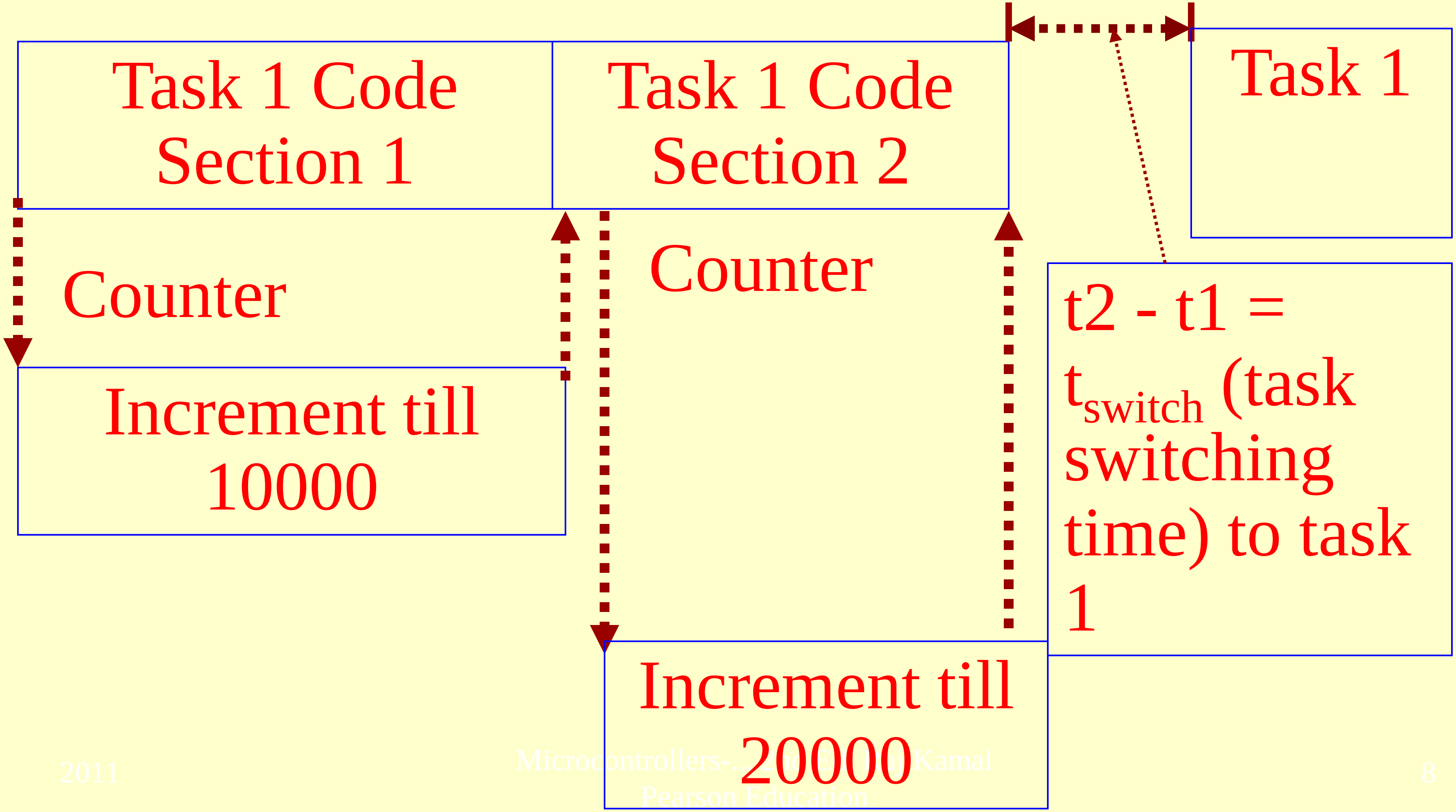
While counting action Actions between Two Sections in Task 0



Infinite loop in Code for task 1

```
job1 ( ) _task_ 1 {
 /* Code for LED at port P1.1 OFF for counts < =10000
     and ON for counts <=20000  */
 while (1) {
counter1 =0; P1^1 = 0;
while { counter1 < = 10000} {count0++;};
P1^1 = 1;
while { counter1 < = 20000} {count0++;};
 }
};
```

While counting action Actions between Two Sections in Task 1



Task 1 Code Section 1

Task 1 Code Section 2

Task 1

Counter

Counter

Increment till 10000

Increment till 20000

t2 - t1 = t_{switch} (task switching time) to task 1

Disadvantage of Using counter loop

- CPU is busy all the time and cannot run another task

RTOS

- Let us use timeout of timer in RTX51 tiny and set timeout for RTX51 tiny system clock
- After timeout, the RTX51 tiny interrupts task 1 and context switches to task 0

Use of the RTOS RTX166 tiny timer and RTX51 Tiny wait functions

- Two LEDs connect to port 1 pins P1.0 and P1.1
- A task 0 for port pin 1.0 switches OFF for 100 ms and ON for 100 ms
- A task 1for port pin 1.1 switches OFF for 150 ms and ON for 150 ms
- RTX51 tiny does an interrupt of each job after a timeout period

Use of the RTOS RTX166 tiny timer and RTX51 Tiny wait functions

- The timeout period is predefined
- However use of the RTOS RTX166 tiny timer and RTX51 Tiny wait functions in a program for this system design— an efficient method.

Preprocessor Statements

1. #include <rtx51full.h>

   #include <rtx166t.h>

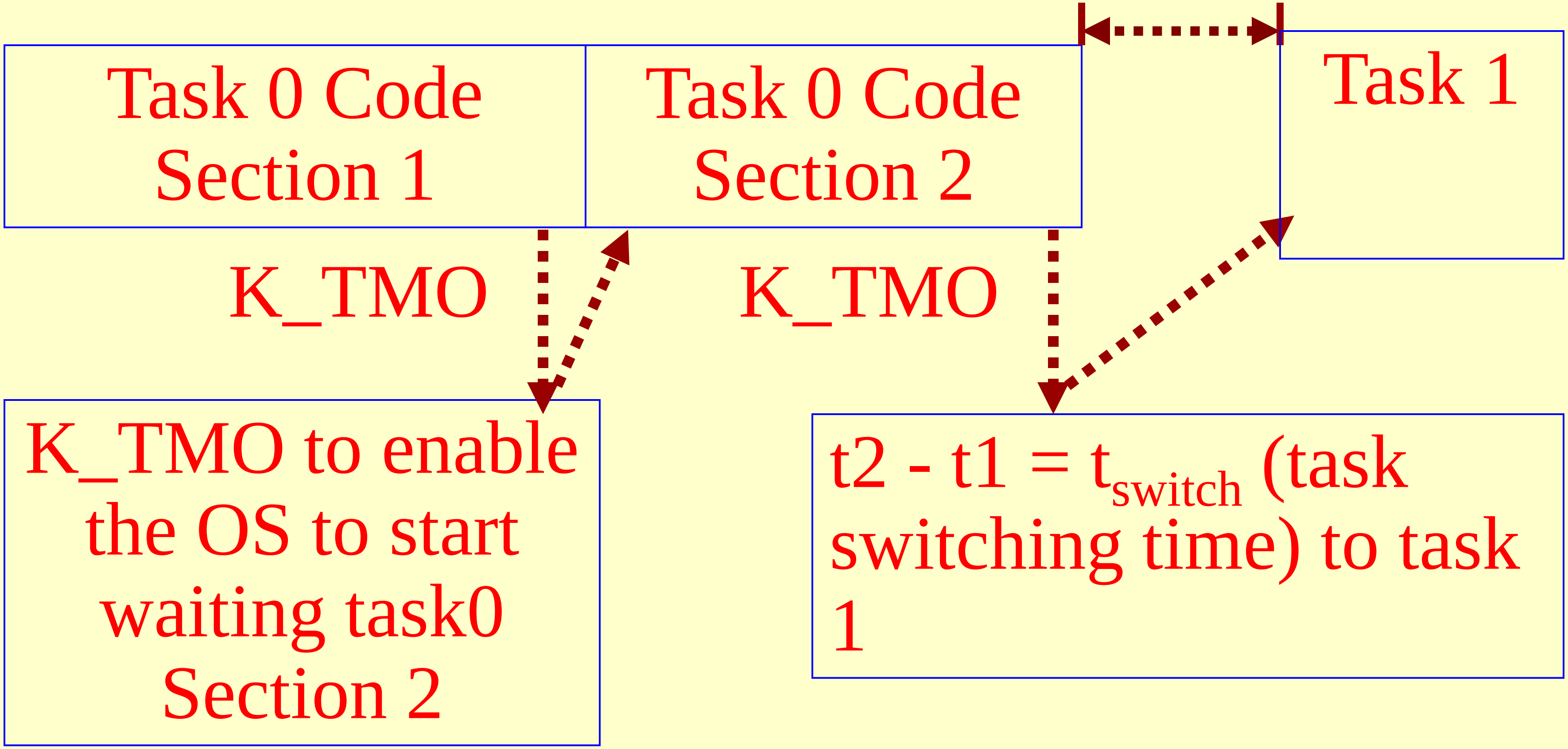
Task 1 Create in Code for task 0

- 2. job0 ( ) _task_ 0 { os_create_task (task 1); /* task 1 ready = 0*/
- /* Code for LED at port P1.0 OFF for 100 ms and ON for 100 ms */

Infinite in Code for task 0

- while (1) {
- P1^0 = 0;
- *os_wait* (K_TMO, 100, 0); /* Wait for signal K_TMO after the number of system timer ticks (overflow interrupts) increment by 100 */
- P1^0 = 1;
- *os_wait* (K_TMO, 100, 0); /* Wait for signal K_TMO after 100 ticks*/
- }

Timeout Actions between Two Sections in Task 0



Task 0 Code Section 1

Task 0 Code Section 2

Task 1

K_TMO

K_TMO

K_TMO to enable the OS to start waiting task0 Section 2

$t2 - t1 = t_{switch}$ (task switching time) to task 1

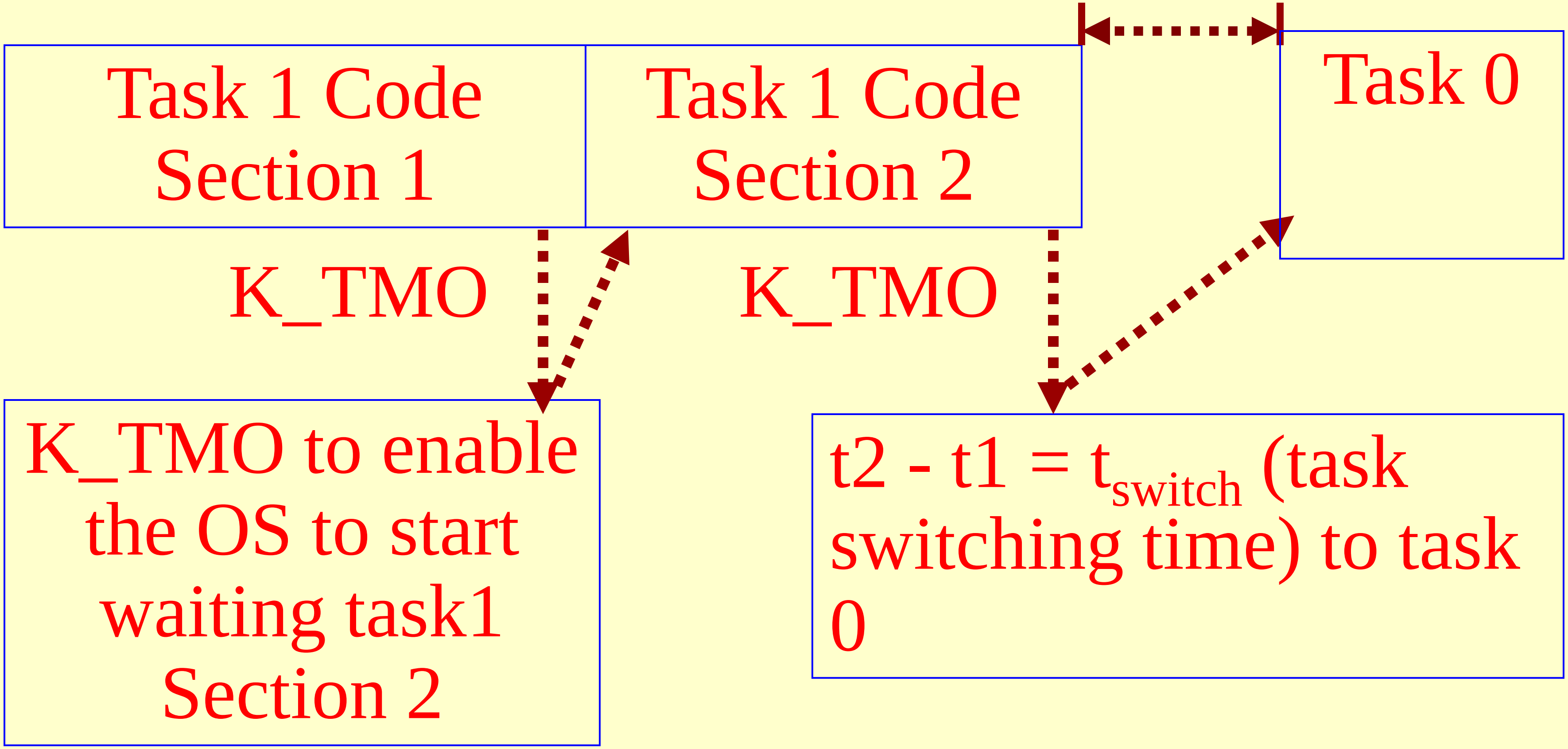
Code for task 1

- job1 ( ) _task_ 1 {
- /* Code for LED at port P1.1 OFF for 150 ms and ON for 150 ms */

Infinite loop in Code for task 1

- while (1) {
- P1^1 = 1;
- *os_wait* (K_TMO, 150, 0);
- P1^1 = 0;
- *os_wait* (K_TMO, 150, 0);
- }

Timeout Actions between Two Sections in Task 1



Task 1 Code Section 1

Task 1 Code Section 2

Task 0

K_TMO

K_TMO

K_TMO to enable the OS to start waiting task1 Section 2

t2 - t1 = t_{switch} (task switching time) to task 0

Advantage of Using RTOS Timeout Functtion

- During the period timer is running the system can run other tasks, task2, task 3, ….

Summary

We learnt

- Exemplary application of two LEDs ON-OFF
- Using variable *counter*
- Using timer and time-out function in RTOS
- Advantage of using RTOS timer function

End of Lesson 05 on

Exemplary Use of RTOS in System Design for of two LEDs ON-OFF program