Chapter 11

Real Time Operating System

Lesson 06

Case Study of Traffic Light for use of RTOS 51 in Design

Assumptions

- When a vehicle coming from north, Left turn (north to west) allowed directly
- Left-lane driving by the vehicles
- Two set of traffic lights positioned to control north to south (NS for straight path to south) and north to east (NE for right turn) traffic flows. Each set has one green and one red signal.

Assumptions

- Even numbered 8051 port 1 pins P1.0 and P1.2 controls and is used to permit the traffic flow along NS and NE by sending green signals NSG and NEG.
- ODD numbered 8051 port 1 pins P1.1 and P1.3 control and prevent the traffic flows along NS and NE and are sent red signals NSR and NER.

Multitasking system Task init

- Task init: First task to initialise the NS-NE traffic light controller system (NTCS)
- Task initialises a serial interface program serial.C
- Creates other application tasks and then deleted—Deactivated from scheduling by the kernel

Multitasking system Tasks 1, 2

- Task 1 clock: Controls the time clock
- Task 2 NSG: Controls the Green ON-OFF along NS using port P1.0

Multitasking system Tasks 3, 4 and 5

- Task 3 NSR: Controls the RED ON-OFF along NS using port P1.1
- Task 4 NEG: Controls the Green ON-OFF along NE using port P1.2
- Task 5 NEG: Controls the RED ON-OFF along NE using port P1.3

Case Study Example Program for RTOS RTX51functions for system of traffic flows

- From north to south and north to east using 8051
- Each task among 2, 3, 4 and 5 runs for 120000 ms and therefore signal is switched ON for 120000 ms
- Assume that RTX166 timer programmed for timeout after 40000 ms
- When ticks = 5 then wait is for 2 m
- When ticks = 20 then wait is for 8 m

Preprocessor Commands

#include <rtxt51full.h>

include <rtx166t.h>

main function

```
void main ( ) { while (1)
{os_start ( ); }
}
```

task init creation of other tasks

```
void init (void) _task_ init {
serial_init.c;
os_create_task (1) /* task 1 ready */
os_create_task (2); /* task 2 ready */
os_create_task (3); /* task 3 ready */
os_create_task (4); /* task 4 ready */
os_create_task (5); /* task 5 ready */
```

Traffic Light System

task init

priority 0

task time clock

priority 1

Task 2 NSG ON

priority 2

Task 3 NSG OFF

priority 3

Task 4 NER ON

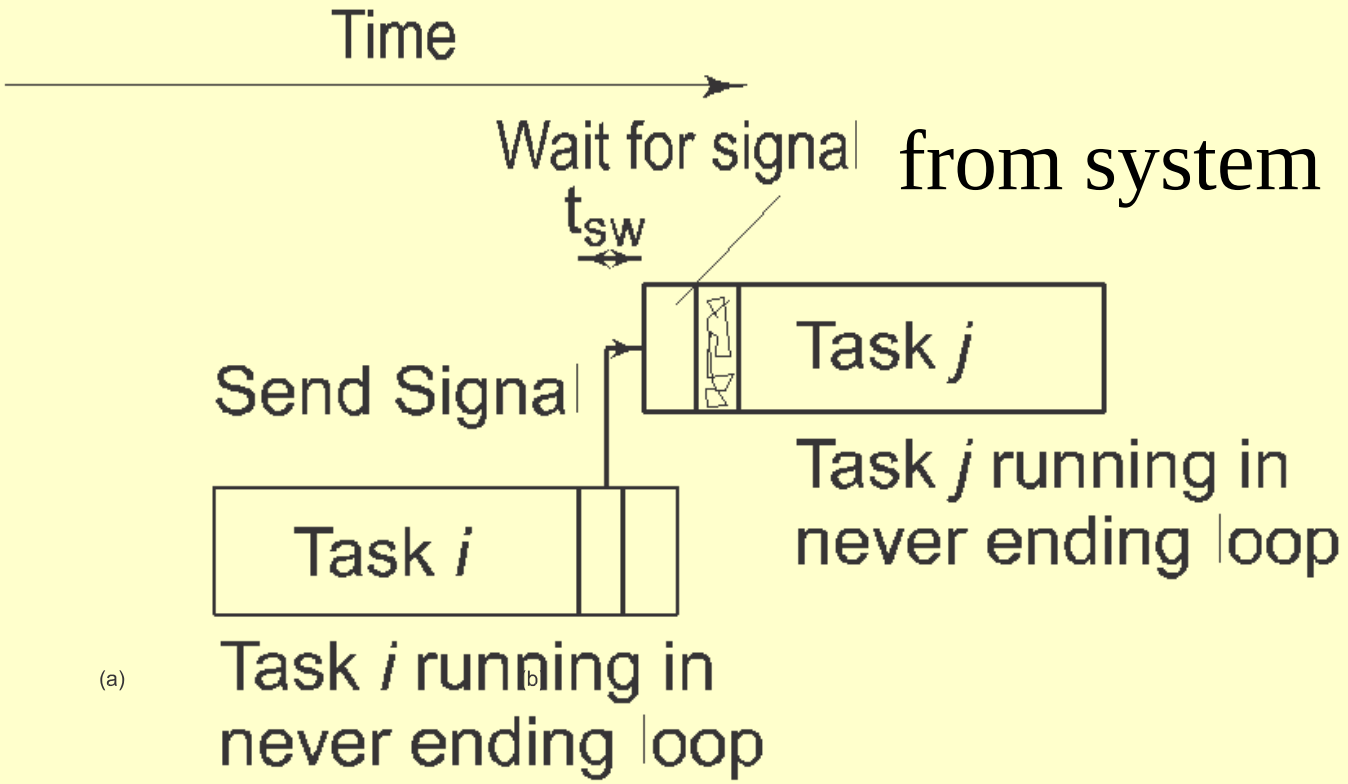
priority 4

Task 5 NER OFF

priority 5

task init starting task 2 and deleting itself

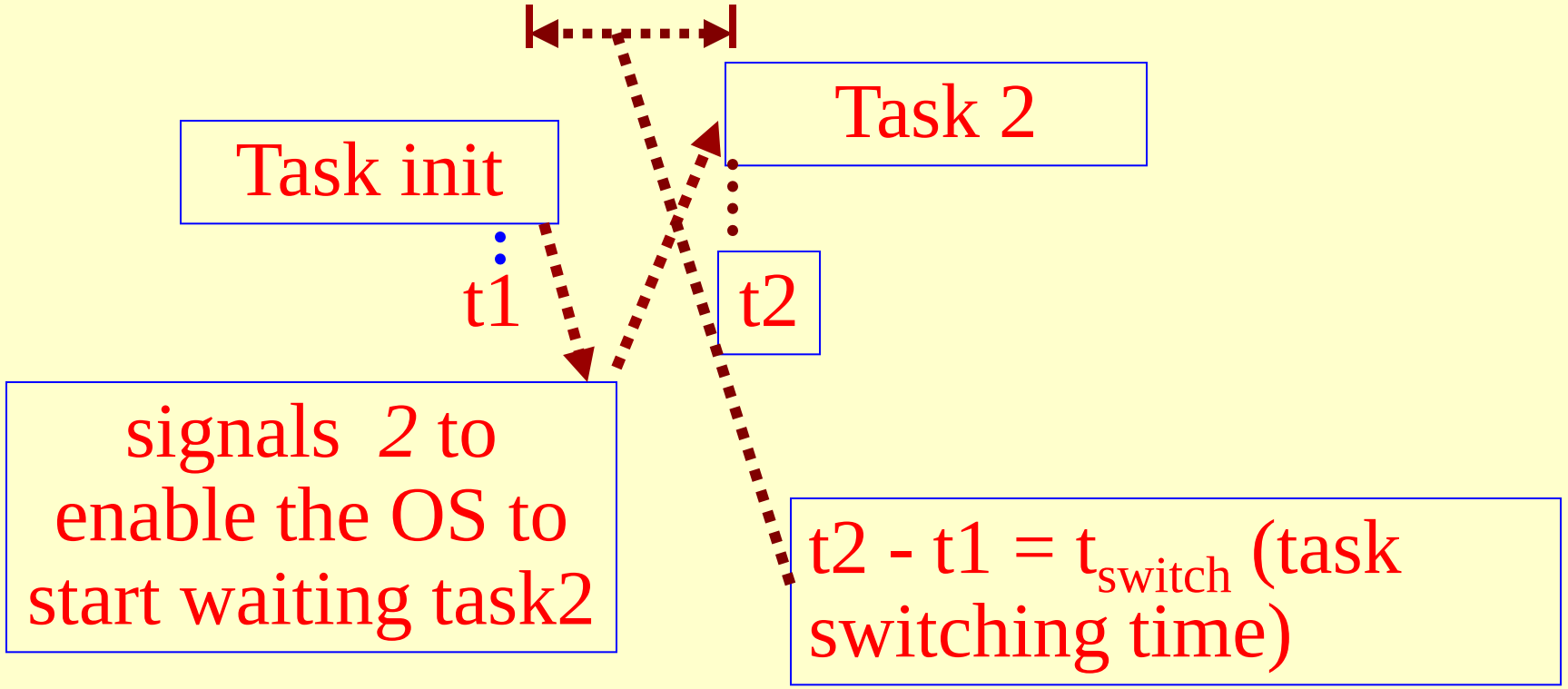
- while (1) { os_send_signal (2);/* Send signal to task 2*/
- os_delete_task (init); /* task_init deleted and RTOS does not take notice of it for ever*/
- }
- }


Time
Wait for signal from system
t_{sw}
Task *j*
Send Signal
Task *j* running in never ending loop
Task *i*
(a)
Task *i* running in never ending loop

task 2 wait for signal to NS Green ON

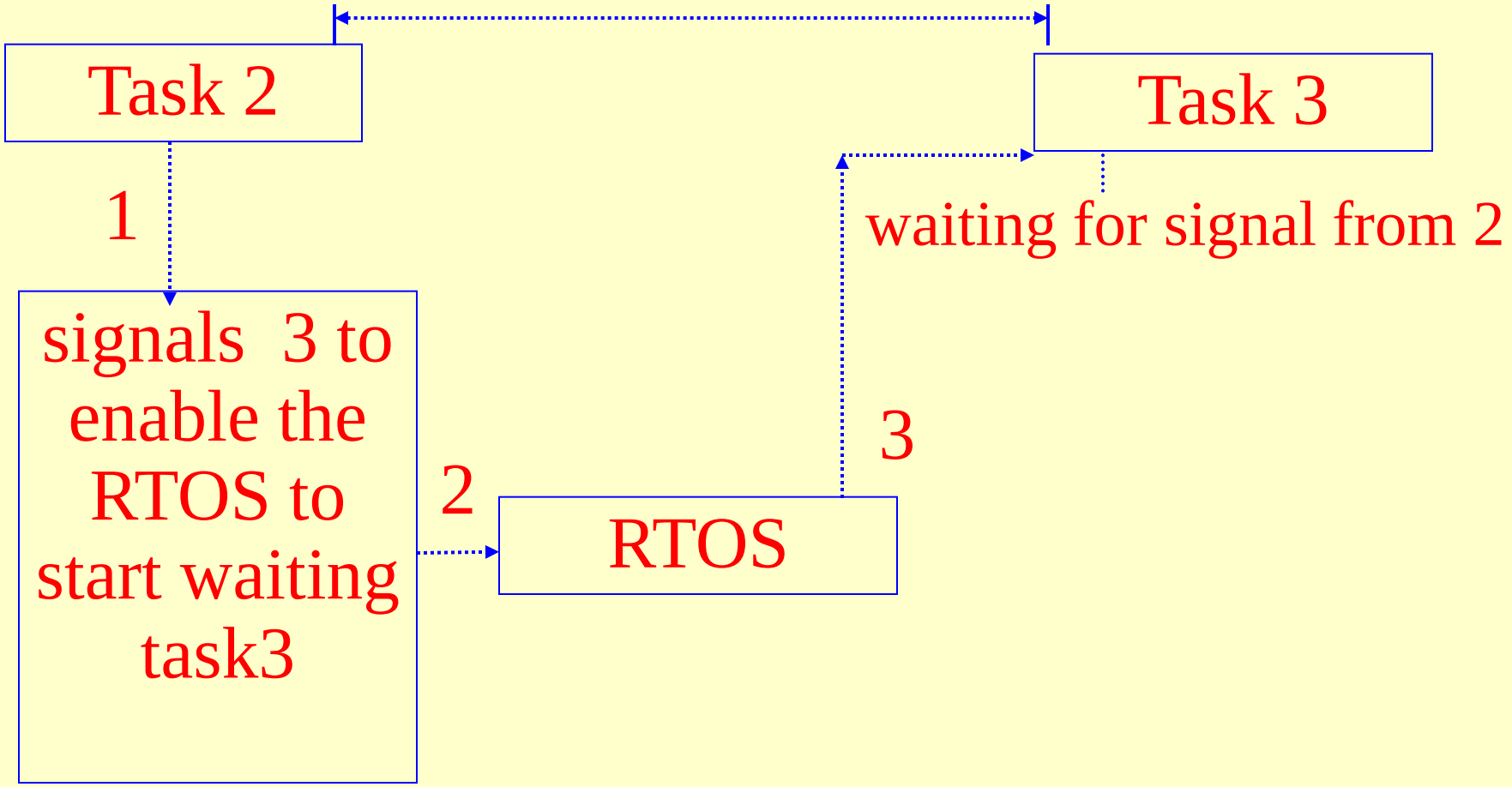
```
job2  ( ) _ task_2 {os_wait (K_SIG, 0, 0); /*
      Wait for signal first from init  */
   os_send_signal (3); /* Send signal to task 2*/
    while (1) { os_wait (K_SIG, 0, 0); /* Wait for
     signal */}
;/* Code for NS light green ON for 120000 ms (
      5 clock timeouts) and then OFF */
P1^0 = 1; /* NSG ON */
```

Signal between Two Tasks



Example of 3 tasks

Example - Task 1 and 3 Synchronisation



Traffic Light System

Priority Scheduling

task init

priority 0

task time clock

priority 1

Task 2 NSG ON

priority 2

Task 3 NSG OFF

priority 3

Task 4 NER ON

priority 4

Task 5 NER OFF

priority 5

When high priority task activates, then that preempts the low priority task running.

task 2 wait for timeout of ON period

```
os_wait (K_TMO, 5, 0); /* wait for 2 m */
P1^0 = 0; /* NSG OFF */
os_send_signal (3); /* Send signal to task 3*/
    };
```

task 3 wait for signal and NS Red ON

```
job3 ( ) _ task_3 {
    while (1) {
/* Code for NS light red ON for 240000 ms (10
   clock timeouts) and then OFF */
os_wait (K_SIG, 0, 0); /* Wait for signal from 2
   */
     P1^1 = 1; /* NSR ON */
```

task 3 wait for timeout of ON period NS Red

- os_wait (K_TMO, 20, 0); /* wait for 8 m */
- P1^1 =0; /* NSR OFF */
- os_send_signal (4); /* Send signal to task 4*/
- 11. };

task 4 wait for signal and NE Green ON

```
job4 ( ) _ task_4 {
    while (1) {
  /* Code for NE light green ON for 120000 ms
 (5 clock timeouts) and then OFF */
  os_wait (K_SIG, 0, 0); /* Wait for signal
 from 3 */
P1^2 = 1; /* NEG ON */
```

task 4 wait for time out of NEG

```
os_wait (K_TMO, 5, 0); /* wait for 2 m */
P1^2 =0; /* NEG OFF */
os_send_signal (5);  /* Send signal to task 5*/
        };
```

task 5 wait for signal and NER ON

```
job5 ( ) _ task_5 {
        while (1) {
        /* Code for NS light red ON for 240000 ms
    (10 clock timeouts) and then OFF */
os_wait (K_SIG, 0, 0); /* Wait for signal from 4
    */
      P1^3 = 1; /* NER ON */
```

task 5 wait for timeout of NER

```
os_wait (K_TMO, 20, 0); /* wait for 8 m */
    P1^3 =0; /* NER OFF */
os_send_signal (2);  /* Send signal to task 2*/
};
```

Summary

We learnt

- Traffic sight system
- Use of preemptive scheduling
- Task synchronized using signals
- Task wait for timeouts

End of Lesson 06 on

Case Study of Traffic Light for use of RTOS 51 in Design