POLITECNICO DI TORINO

OSES LAB #1 – task and alarms

Department of CONTROL AND COMPUTER ENGINEERING (DAUIN)
Master Degree in Mechatronic Engineering
2023/2024
Operating systems for embedded systems– Prof. Violante
Laboratory 01
Wednesday, 18 October, 2023

Students :
s320296 DI MARTINO GIULIO

Exercise #1

In the exercise 1 we shall create a program that give in the output a value; this value depend on the task.

- Task A, with period 500 msec and priority 2; task A shall execute upon starting the operating system; The task manages an integer counter that starts from 0 and at each activation of the task is increments by 500.
- Task B, with period 750 msec and priority 1; the first instance of task B shall be executed 1500 msec after starting the operating system. The task manages an integer counter that starts from 1500 and at each activation of the task is increments by 750
- application execution shall last 6000 msec.

The .oil code is the follow :
lab01_ex01.oil

```
OIL_VERSION = "2.5";

IMPLEMENTATION trampoline {

    /* This fix the default STACKSIZE of tasks */
    TASK {
        UINT32 STACKSIZE = 32768 ;
    } ;

    /* This fix the default STACKSIZE of ISRs */
    ISR {
        UINT32 STACKSIZE = 32768 ;
    } ;
};

CPU only_one_periodic_task {
  OS config {
    STATUS = EXTENDED;
    TRACE = TRUE {
        FORMAT = json;
        PROC = TRUE;
        RESOURCE = TRUE;
        ALARM = TRUE;
        EVENT = TRUE;
    };
    BUILD = TRUE {
      APP_SRC = "lab01_ex01.c";
      TRAMPOLINE_BASE_PATH = "../../..";
      CFLAGS="-ggdb";
      APP_NAME = "lab01_exe";
      LINKER = "gcc";
      SYSTEM = PYTHON;
    };
  };

  APPMODE stdAppmode {};

  ALARM a500msec {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = TaskA; };
    AUTOSTART = TRUE { APPMODE = stdAppmode; ALARMTIME = 50; CYCLETIME = 50; };
  };

  ALARM a750msec {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = TaskB; };
    AUTOSTART = TRUE { APPMODE = stdAppmode; ALARMTIME = 150; CYCLETIME = 75; };
  };

  ALARM stopper {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = stop; };
    AUTOSTART = TRUE { APPMODE = stdAppmode; ALARMTIME = 600; CYCLETIME = 0; };
```

```
  };

  TASK TaskA {
    PRIORITY = 2;
    AUTOSTART = TRUE { APPMODE = stdAppmode; };
    ACTIVATION = 1;
    SCHEDULE = FULL;
  };

  TASK TaskB {
    PRIORITY = 1;
    AUTOSTART = FALSE;
    ACTIVATION = 1;
    SCHEDULE = FULL;
  };

  TASK stop {
    PRIORITY = 99;
    AUTOSTART = FALSE;
    ACTIVATION = 1;
    SCHEDULE = FULL;
  };

};
```

The oil code is composed by:

```
CPU only_one_periodic_task {
  OS config {
    STATUS = EXTENDED;
    TRACE = TRUE {
        FORMAT = json;
        PROC = TRUE;
        RESOURCE = TRUE;
        ALARM = TRUE;
        EVENT = TRUE;
    };
    BUILD = TRUE {
      APP_SRC = "lab01_ex01.c";
      TRAMPOLINE_BASE_PATH = "../../..";
      CFLAGS="-ggdb";
      APP_NAME = "lab01_exe";
      LINKER = "gcc";
      SYSTEM = PYTHON;
    };
  };
```

This part of the code is useful because we define all the specification that are needed for the program.

TASK DEFINE

```
TASK TaskA {
    PRIORITY = 2;
    AUTOSTART = TRUE { APPMODE = stdAppmode; };
    ACTIVATION = 1;
    SCHEDULE = FULL;
  };
```

The taskA have PRIORITY = 2 , this is bigger respect to the taskB so the taskA is more important than the taskB (greater it is the number, greater is the priority).AUTOSTART =TRUE so the Task A is placed in the ready list upon started the Operating System, if AUTOSTART = FALSE the task isn't placed in the ready list, so it needs an external system call.
ACTIVATION = 1 means that we can have only one instance of Task A in the ready list; SCHEDULE = FULL means the scheduler is fully preemptible

```
TASK TaskB {
    PRIORITY = 1;
    AUTOSTART = FALSE;
    ACTIVATION = 1;
    SCHEDULE = FULL;
  };
```

AUTOSTART = FALSE means that the task is not instantiate at the beginning of the execution of the operating system, so the task is not in the ready list, so it needs an external system call.

```
TASK stop {
    PRIORITY = 99;
    AUTOSTART = FALSE;
    ACTIVATION = 1;
    SCHEDULE = FULL;
  };

};
```

The task stop has the higher value of priority = 99 so when it starts it have the priority respect the other task

ALARM DEFINE

```
ALARM a500msec {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = TaskA; };
    AUTOSTART = TRUE { APPMODE = stdAppmode; ALARMTIME = 50; CYCLETIME = 50; };
  };
```

In this part we set the first alarm that is useful to activate the taskA. This task is performed each 500ms so I set in the AUSTART the ALARMTIME=50 because the tick time is 10ms, this tick is created in the function `COUNTER = SystemCounter,` and because is a periodic task I set the CYCLETIME=50; the AUTOSTART is true so the alarm is placed in the ready list upon started the Operating System.
The same structure are used to perform the taskB and the stop function

```
ALARM a750msec {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = TaskB; };
    AUTOSTART = TRUE { APPMODE = stdAppmode; ALARMTIME = 150; CYCLETIME = 75; };
  };
```

```
ALARM stopper {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = stop; };
    AUTOSTART = TRUE { APPMODE = stdAppmode; ALARMTIME = 600; CYCLETIME = 0; };
  };
```

The C code

```
#include <stdio.h>
#include <math.h>
#include "tpl_os.h"
int main(void)


{
    StartOS(OSDEFAULTAPPMODE);
    return 0;
}
```

```
DeclareAlarm(a500msec);
DeclareAlarm(a750msec);
DeclareAlarm(stopper);
TASK(TaskA)
{
    static int contA=0;
    printf("task A:%d\r\n",contA);
    contA += 500;
    TerminateTask();
}

TASK(TaskB)
{
    static int contB=1500;
    printf("task B:%d\r\n",contB);
    contB += 750;

  TerminateTask();
}

TASK(stop)
{
  CancelAlarm(a500msec);
  CancelAlarm(a750msec);
  printf("Shutdown\r\n");
  ShutdownOS(E_OK);
  TerminateTask();
}
```

Where I first of all define the three alarms with the function `DeclareAlarm` after that I call the task, so when the taskA is on the code perform that part

```
TASK(TaskA)
{
    static int contA=0;
    printf("task A:%d\r\n",contA);
    contA += 500;
    TerminateTask();
}
```

I initialize the variable `contA=0` and at each iteration I add 500 after each iteration the task end (`TerminateTask();`)

```
TASK(stop)
{
  CancelAlarm(a500msec);
  CancelAlarm(a750msec);
  printf("Shutdown\r\n");
  ShutdownOS(E_OK);
  TerminateTask();
}
```

Task stop is defined, it stops the execution, write on the terminal the end of the process and shutdown the OS.

The program's execution

To run the program first I enter in the folder where there is the file

```
 giuliodimartino@MacBook-Pro-di-Giulio lab01_ex01 %
```

than I export the viper:

```
export VIPER_PATH=/Users/giuliodimartino/Desktop/trampoline/viper
```

After that I run the goil compile:

```
giuliodimartino@MacBook-Pro-di-Giulio lab01_ex01 % ../../../goil/makefile-macosx/goil --
target=posix/Darwin --templates=../../../goil/templates/ lab01_ex01.oil
Created 'lab01_ex01/tpl_os.c'.
Created 'lab01_ex01/tpl_os.h'.
Created 'build.py'.
Created 'make.py'.
Created 'lab01_ex01/tpl_app_custom_types.h'.
Created 'lab01_ex01/tpl_app_config.c'.
Created 'lab01_ex01/tpl_app_config.h'.
Created 'lab01_ex01/tpl_app_define.h'.
```

```
Created 'lab01_ex01/tpl_static_info.json'.
Created 'readTrace.py'.
executing plugin gdb_commands.goilTemplate
Created
'/Users/giuliodimartino/Desktop/trampoline/examples/posix/lab01_ex01/build/lab01_ex01.oil.dep'.
```

And whit

```
giuliodimartino@MacBook-Pro-di-Giulio lab01_ex01 % ./build.py
Making "build/os" directory
[  4%] Compiling ../../../os/tpl_os_kernel.c
[  9%] Compiling ../../../os/tpl_os_timeobj_kernel.c
[ 14%] Compiling ../../../os/tpl_os_action.c
[ 19%] Compiling ../../../os/tpl_os_error.c
[ 23%] Compiling ../../../os/tpl_os_os_kernel.c
[ 28%] Compiling ../../../os/tpl_os_os.c
[ 33%] Compiling ../../../os/tpl_os_interrupt_kernel.c
[ 38%] Compiling ../../../os/tpl_os_task_kernel.c
[ 42%] Compiling ../../../os/tpl_os_resource_kernel.c
[ 47%] Compiling ../../../os/tpl_os_alarm_kernel.c
[ 52%] Compiling lab01_ex01.c
Making "build/lab01_ex01" directory
[ 57%] Compiling lab01_ex01/tpl_app_config.c
[ 61%] Compiling lab01_ex01/tpl_os.c
Making "build/machines/posix" directory
[ 66%] Compiling ../../../machines/posix/tpl_machine_posix.c
[ 71%] Compiling ../../../machines/posix/tpl_viper_interface.c
[ 76%] Compiling ../../../machines/posix/tpl_posix_autosar.c
[ 80%] Compiling ../../../machines/posix/tpl_posix_irq.c
[ 85%] Compiling ../../../machines/posix/tpl_posix_context.c
[ 90%] Compiling ../../../machines/posix/tpl_posixvp_irq_gen.c
[ 95%] Compiling ../../../machines/posix/tpl_trace.c
[100%] Linking lab01_exe
```

In the end I run the exe programmer with the command:

```
giuliodimartino@MacBook-Pro-di-Giulio lab01_ex01 % ./lab01_exe
task A:0
task A:500
task A:1000
task A:1500
task B:1500
task A:2000
task B:2250
task A:2500
task A:3000
task B:3000
task A:3500
task B:3750
task A:4000
task A:4500
task B:4500
task A:5000
task B:5250
task A:5500
Shutdown
Exiting virtual platform.
```

So when the task A is on the start point is 0 and each time 500 is added; instead, when the task B starts the first value at 1500 and at each iteration 750 is added

Exercise #2

Each time that Task A is activated the Built-in LED (pin 13) is turned on, instead when the Task B is active the same LED turns off.

```
OIL_VERSION = "2.5" : "test";


CPU test {
  OS config {
    STATUS = STANDARD;
    BUILD = TRUE {
      TRAMPOLINE_BASE_PATH = "/Users/giuliodimartino/Desktop/trampoline";
      APP_NAME = "lab01";
      APP_SRC = "lab01_ex02.cpp";
      CPPCOMPILER = "avr-g++";
      COMPILER = "avr-gcc";
      LINKER = "avr-gcc";
      ASSEMBLER = "avr-gcc";
      COPIER = "avr-objcopy";
      SYSTEM = PYTHON;
    };
    SYSTEM_CALL = TRUE;
  };

  APPMODE stdAppmode {};

  ALARM a500msec {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = TaskA; };
    AUTOSTART = TRUE { APPMODE = stdAppmode; ALARMTIME = 488; CYCLETIME = 488; };
  };

  ALARM a750msec {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = TaskB; };
    AUTOSTART = TRUE { APPMODE = stdAppmode; ALARMTIME = 1464; CYCLETIME = 732; };
  };

  ALARM stopper {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = stop; };
    AUTOSTART = TRUE { APPMODE = stdAppmode; ALARMTIME = 5859; CYCLETIME = 0; };
  };

  TASK TaskA {
    PRIORITY = 2;
    AUTOSTART = TRUE { APPMODE = stdAppmode; };
    ACTIVATION = 1;
    SCHEDULE = FULL;
  };

  TASK TaskB {
    PRIORITY = 1;
    AUTOSTART = FALSE;
    ACTIVATION = 1;
    SCHEDULE = FULL;
  };

  TASK stop {
    PRIORITY = 99;
    AUTOSTART = FALSE;
    ACTIVATION = 1;
    SCHEDULE = FULL;
  };

};
```

the oil file is more or less the same as before;

```
CPU test {
  OS config {
    STATUS = STANDARD;
    BUILD = TRUE {
```

```
      TRAMPOLINE_BASE_PATH = "/Users/giuliodimartino/Desktop/trampoline";
      APP_NAME = "lab01";
      APP_SRC = "lab01_ex02.cpp";
      CPPCOMPILER = "avr-g++";
      COMPILER = "avr-gcc";
      LINKER = "avr-gcc";
      ASSEMBLER = "avr-gcc";
      COPIER = "avr-objcopy";
      SYSTEM = PYTHON;
    };
    SYSTEM_CALL = TRUE;
  };
```

In this part as before I set all the instruction for the Arduino environment.

```
ALARM a500msec {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = TaskA; };
    AUTOSTART = TRUE { APPMODE = stdAppmode; ALARMTIME = 488; CYCLETIME = 488; };
  };
```

The time in the alarm instruction change because in the previous case the counter performed a tick every 10 ms. A 16Mhz Arduino board performs a tick every 1024 µs, to have the same task execution time we had to modify the ALARMTIME and CYCLETIME with the following proportion:

$$1024{:}\,1000 = time_{desired}{:}\,X$$

so we find an alarmtime of 488 and uncycletime of 488 for task A, an alarmtime of 1464 and uncycletime of 732 for task B and an alarmtime of 5859 and uncycletime of 0 for task stop.

```
#include <stdio.h>
#include <math.h>
#include "tpl_os.h"
#include "Arduino.h"

DeclareAlarm(a500msec);
DeclareAlarm(a750msec);

void setup()
{
    StartOS(OSDEFAULTAPPMODE);
    pinMode(13, OUTPUT);
}

TASK(TaskA)
{
  digitalWrite(13, HIGH);
  TerminateTask();
}

TASK(TaskB)
{
  digitalWrite(13, LOW);
  TerminateTask();
}

TASK(stop)
{
  CancelAlarm(a500msec);
  CancelAlarm(a750msec);
  ShutdownOS(E_OK);
  TerminateTask();
}
```

```
void setup()
{
    StartOS(OSDEFAULTAPPMODE);
    pinMode(13, OUTPUT);
}
```

In this part of code I initialized the pin in ArduinoUno and the command StartOS(OSDEFAULTAPPMODE) initialized the OSEK ;

```
TASK(TaskA)
{
  digitalWrite(13, HIGH);
  TerminateTask();
}
```

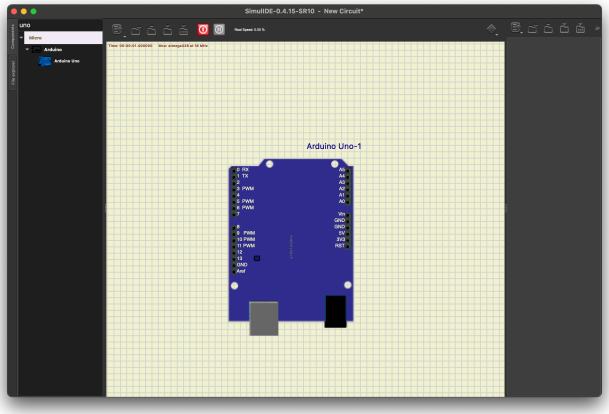
This part is useful to activate the pin each time the task A is performed.

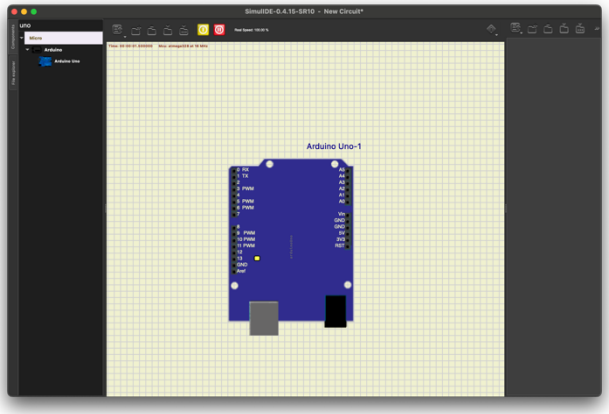
The program's execution

```
giuliodimartino@MacBook-Pro-di-Giulio lab01_ex02 % ../../../../goil/makefile-macosx/goil --
target=avr/arduino/uno --templates=../../../../goil/templates/ lab01_ex02.oil
Created 'lab01_ex02/tpl_dispatch_table.c'.
Created 'lab01_ex02/tpl_invoque.S'.
Created 'lab01_ex02/tpl_invoque_isr.S'.
Created 'lab01_ex02/tpl_os.h'.
Created 'lab01_ex02/tpl_service_ids.h'.
Created 'build.py'.
Created 'make.py'.
Created 'lab01_ex02/tpl_app_custom_types.h'.
Created 'lab01_ex02/tpl_app_config.c'.
Created 'lab01_ex02/tpl_app_config.h'.
Created 'lab01_ex02/tpl_app_define.h'.
Created 'lab01_ex02/tpl_static_info.json'.
Created 'lab01_ex02/tpl_interrupts.c'.
executing plugin gdb_commands.goilTemplate
Created
'/Users/giuliodimartino/Desktop/trampoline/examples/avr/arduinoUno/lab01_ex02/build/lab01_ex02.oil.d
ep'.
No warning, no error.
```

```
giuliodimartino@MacBook-Pro-di-Giulio lab01_ex02 % ./make.py
Nothing to make.
[  5%] Compiling /Users/giuliodimartino/Desktop/trampoline/os/tpl_os_kernel.c
[ 10%] Compiling /Users/giuliodimartino/Desktop/trampoline/os/tpl_os_timeobj_kernel.c
[ 15%] Compiling /Users/giuliodimartino/Desktop/trampoline/os/tpl_os_action.c
[ 21%] Compiling /Users/giuliodimartino/Desktop/trampoline/os/tpl_os_error.c
[ 26%] Compiling /Users/giuliodimartino/Desktop/trampoline/os/tpl_os_os_kernel.c
[ 31%] Compiling /Users/giuliodimartino/Desktop/trampoline/os/tpl_os_os.c
[ 36%] Compiling /Users/giuliodimartino/Desktop/trampoline/os/tpl_os_interrupt_kernel.c
[ 42%] Compiling /Users/giuliodimartino/Desktop/trampoline/os/tpl_os_task_kernel.c
[ 47%] Compiling /Users/giuliodimartino/Desktop/trampoline/os/tpl_os_resource_kernel.c
[ 52%] Compiling /Users/giuliodimartino/Desktop/trampoline/os/tpl_os_alarm_kernel.c
[ 57%] Compiling lab01_ex02.cpp
[ 63%] Compiling lab01_ex02/tpl_app_config.c
[ 68%] Compiling lab01_ex02/tpl_dispatch_table.c
[ 73%] Compiling lab01_ex02/tpl_invoque.S
[ 78%] Compiling lab01_ex02/tpl_interrupts.c
[ 84%] Compiling /Users/giuliodimartino/Desktop/trampoline/machines/avr/tpl_machine.c
[ 89%] Compiling
/Users/giuliodimartino/Desktop/trampoline/machines/avr/arduino/cores/arduino/main.cpp
[ 94%] Compiling /Users/giuliodimartino/Desktop/trampoline/machines/avr/arduino//tpl_trace.cpp
[100%] Linking lab01
       Generating binary lab01.hex from lab01
```

I generate the lab01.hex, putting this file into SimulIDE I obtain the turn on and off of the pin 13.


SimulIDE-0.4.15-SR10 - New Circuit*
Arduino Uno-1


SimulIDE-0.4.15-SR10 - New Circuit*
Arduino Uno-1