OSES LAB #1 – task and alarms

Please read carefully the assignment description

Purpose of the lab

This lab is intended to assess your knowledge of tasks and alarms.

Deliverables and deadlines

You shall provide a report (pdf file only) where you describe the oil file and the code you implemented for each exercise we proposed. The output produced by your solutions shall also be included (insert in the report the screenshot).
The report shall be uploaded to the portale della didattica using the Elaborati section.
The report shall be provided by October 17th, 2025, 20:00.

Exercise #1

You shall create two basic tasks, as follows:

- 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.

Each time any task is activated it prints on screen the counter value, then it updates the counter by the respective value.
You shall use the Trampoline POSIX target to implement the above application; application execution shall last 6000 msec.

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 = "ex1.c";
      TRAMPOLINE_BASE_PATH = "../..";
      CFLAGS="-ggdb";
      APP_NAME = "ex1_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;
  };

};
```

lab01_ex01.c

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

int main(void)
{
    StartOS(OSDEFAULTAPPMODE);
    return 0;
}

DeclareAlarm(a500msec);
DeclareAlarm(a750msec);

TASK(TaskA)
{
  …

  TerminateTask();
}

TASK(TaskB)
{
  …

  TerminateTask();
}

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

Exercise #2

You must deploy the same application of the Exercise 1, but on ArduinoUno (SimulIDE), instead of POSIX.
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.

You shall refer to the Trampoline example located in:

trampoline/examples/avr/arduinoUno/blink

for the templates of application configuration (.oil) and application behavior (.cpp)