Nom, prénom :

Lab2 Real-Time Operating Systems for sensors and actuators

Polytech Nice Sophia

You will have to answer on the statement to questions R1, R2, ...
You will have to write, test and provide the codes C1, C2, ...

Open the Nios-II SBT for Eclipse software, create a new "Nios-II application and BSP from template".
Download the project provided on Moodle and choose :

- SOPC information file name: Design_Files\nios1.sopcinfo
- name : a project name without space
- template : Hello_uCOS

Click Finish, cross-compile the project and test the code (see Appendix 1).

The objective of this TP is to resume the behavior of the previous TP but using an additional software layer, that of the uC/OS-II real-time OS.
Documentation about uc/OS-II is available on https://doc.micrium.com/display/osiidoc/home

1) Create Tasks

R1. Recall the parameters of the two uCOS tasks creation functions.

OSTaskCreate Paramètre	Type

OsTaskCreateExt Paramètre	Type

2) Communication and synchronisation services

R2. Cite the different services under uCOS.

Service	Communication or Synchronisation

3) A first code with the RTOS

C1. Adapt the code of the template to create a communication between the 2 tasks, the first one sends the value of a counter then waits for a certain delay D1, the second one displaying on the 7-segments the received value then waits for another delay D2. Test the difference between the mailbox service and the MessageQueue service by adapting the examples provided in chapter 1 of the documentation.
Test the behavior of the MessageQueue for different multiplicative factors between D1 and D2.

Nom, prénom :

D1 = k.D2, for k in [10, 5, 2, 1, 0.5, 0.01].
R3. What do you observe ?

C2. Take the code from the TP1 (reflex test) and pass it in uCOS multitasking code.
To do so, you must first :
- Identify the necessary tasks (3 minimum).
- Set priorities
- Define the necessary IPC services
- Use the **OSTimeGet** service to read time instead of alt_timestamp (first verify that the box os_tmr_en is checked in the menu advanced-> ucosii of the **BSP Editor**).

R4. Draw the Diagram of communications between tasks and ISR, naming the tasks according to their priority (T1, T2, T12...).

Test the code

4) Monitoring the execution

C3. Add the code needed to instrument the code:
- stack size + processor utilization rate, see appendix 2
- execution time, see Appendix 3

Add to your code a specific, low-priority task that will display monitoring information for all tasks.

R5. Note the information obtained with the measurement task.

Task	Number of executions	Mean execution time	Free stack	Used stack

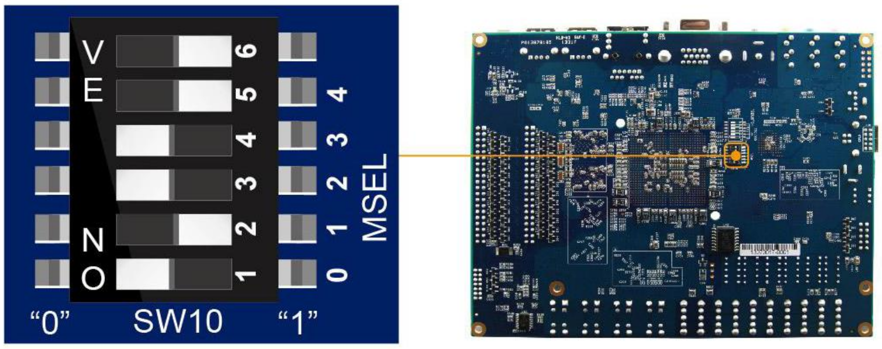
Nom, prénom :

Annex 1 – Programmation of DE1 SoC board

In order to configure the FPGA circuit on the target (one time only) :

- connect the board to the USB Blaster port
- launch Nios II > Quartus II Programming,
- Autodetect,
- add the following file :
- SDRAM_DE1_SoC\output_files\Test_Quartus_161_time_limited.sof
- Click start to configure the FPGA

The micro-switches on the back side of the card must be in the following position:



To download the executable (.elf file) into the memory of the microcontroller :

- compile the project
- Start Run > Run Configuration ...
- New NiosII Hardware
- Refresh connection
- Run

Annex 2 – Measurement of the stack size

The statistical task of uC/OS-II can be used to monitor code execution:
- CPU utilization rate
- Stack size measurement
To allow the execution of this OS_TaskStat() task, the constant OS_TASK_STAT_EN must be set to 1 in OS_CFG.H (to be defined via the eclipse BSP Editor in the menu advanced->ucosii).
For this task to be able to measure the stack actually used by the other tasks, it must be created with the OSTaskCreateExt function (for all the tasks), passing as last parameter the stack measurement flags: OS_TASK_OPT_STK_CHK | OS_TASK_OPT_STK_CLR.

The measure of the utilization rate is provided in the variable OSCPU Usage.

The stack size measurement is performed by the function OSTaskStkChk(int prio, OS_STK_DATA *p), where prio is the priority of the task to be measured, and p the pointer to an OS_STK_DATA structure defined in uCOS_II.h which contains two fields: OSFree and OSused, representing the number of free bytes used by the task whose id is given by the integer prio.

Annexe 3 – Instrumentation of the code (execution time)

You can use uC/OS-II's "Hook" functions to extend your code at particular times of execution. The prototypes are defined in BSP/HAL/src/os_cpu_c.c

- **OSInitHookBegin**, Start the OS,

Nom, prénom :

- OSInitHookEnd, End of initiation of the OS,
- OSTaskCreateHoo, creation of tasks,
- OSTaskDelHook, Delete a task,
- OSTaskIdleHook, Execution of Idle task,
- OSTCBInitHook, Initialisation of TCB,
- **OSTimeTickHook**, Awakening of the OS by the timer,
- **OSTaskSwHook**, Context switch

The context change function will be used to record information about the execution of each task. This information will be stored in a structure:

```
Typedef struct {
        Char TaskName[30] ;
        INT16U      TaskCtr ;
        INT16U      TaskExecTime ;
        INT16U      TaskTotExecTime;
} TASK_USER_DATA;
```

There will be as many structures as there are tasks:

```
TASK_USER_DATA          TaskUserData[NB_TASKS] ;
```

And you will create your own stat function that will print the content of these structures for all the other tasks.

The Hook function code is to be added to your code:

```
void OSTaskSwHook(void)
{
        INT16U taskStopTimestamp, time;
        TASK_USER_DATA *puser;

        taskStopTimestamp = OSTimeGet();

        time =(taskStopTimestamp - taskStartTimestamp) / (OS_TICKS_PER_SEC / 1000); // in ms
        puser = OSTCBCur->OSTCBExtPtr;
        if (puser != (TASK_USER_DATA *)0) {
                puser->TaskCtr++;
                puser->TaskExecTime = time;
                puser->TaskTotExecTime += time;
        }

        taskStartTimestamp = OSTimeGet();
}

void OSInitHookBegin(void)
{
   OSTmrCtr = 0;
   taskStartTimestamp = OSTimeGet();
}

void OSTimeTickHook (void)
{
   OSTmrCtr++;
   if (OSTmrCtr >= (OS_TICKS_PER_SEC / OS_TMR_CFG_TICKS_PER_SEC)) {
       OSTmrCtr = 0;
       OSTmrSignal();
   }
}
```