ECE3073 Computer Systems

Laboratory Session 7

- To implement concurrent tasks using the uC/OS-II real time kernel running on the NIOS II processor.

- Display the processor utilization, current time in hours minutes and seconds, and a graduated LED brightness using pulse width modulation.

1. Equipment

§ DE2 (or DE10) FPGA Development Board
§ Quartus/Nios II development software containing the uC/OS-II real time kernel.
§ A USB memory device provided by you to store your design files
§ If you are doing this on your own device, you will need:
- Quartus License file (see "Installation" guide in weeks 1 & 2 for setting up your license)
- Monash VPN (if you are doing this at home)

2. Preliminary work

Read this lab manual and the documentation on the statistic task *OSStatInit()* (and attend lectures☺).

Before you start! Have you checked the "equipment" section? You will need the Quartus license (if you are doing this on your own device) AND you will need to run Monash VPN when trying to compile your Quartus projects at home. If you are compiling your project at Monash, you do NOT need to run VPN but you will still need to have your license

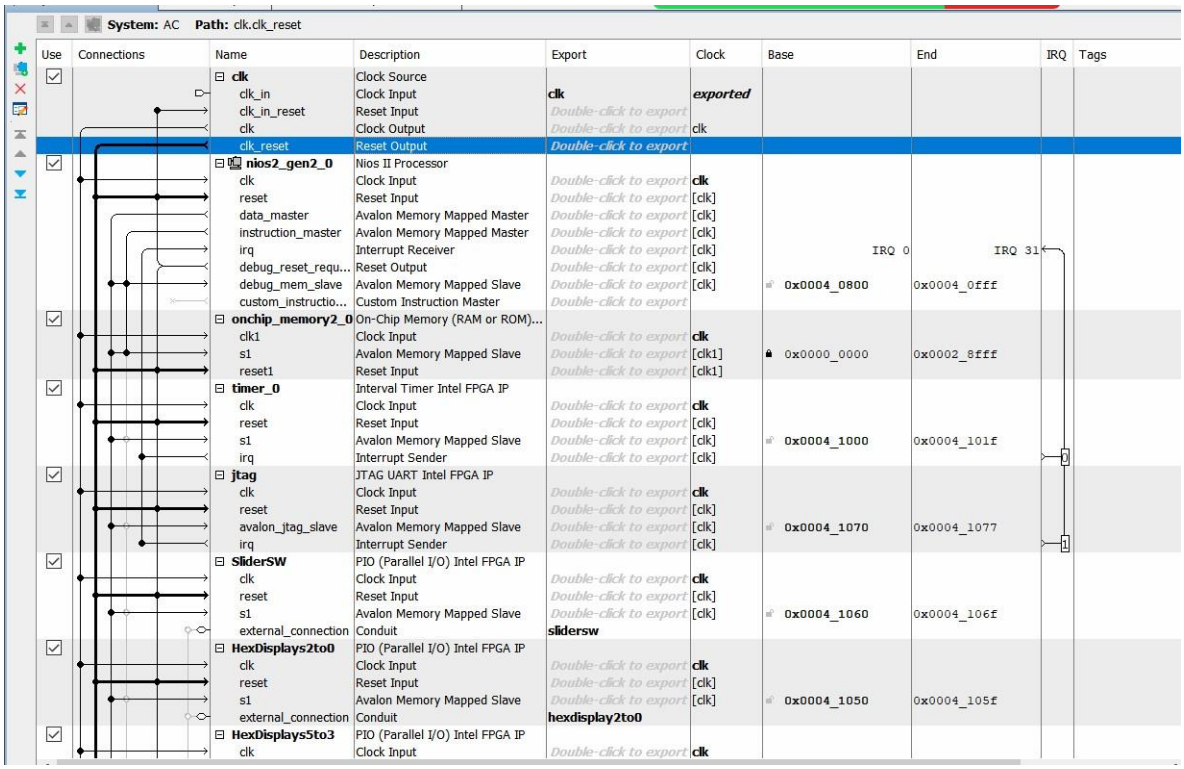
Nios system for this lab exercise

Download the .qar file instead of making the Quartusproject yourself.

Nios system has the following components:

- a Nios II/f processor,
- 164k bytes of RAM (you can type in "164k" into the memory settings, and this will convert to 167936 for you)
- a timer module. Select "Processors and Peripherals", then select "Peripherals" and "Interval Timer Intel FPGA IP". **Put in an IRQ of 0 for Timer.** Note, you can connect the timer module similar to how you have connected other PIOs
- a JTAG UART. **Select the JTAG UART under "Interface Protocols" and "Serial"**. The name of the component is "JTAG UART Intel FPGA IP". **Put in an IRQ of 1 for JTAG**
- 24-bit output PIO for HEX displays HEX2, HEX1, and HEX0.
- 24-bit output PIO for HEX displays HEX5, HEX4, HEX3
- 10-bit output PIO for LEDR[9:0]
- 2-bit input PIO for KEY[1:0]
- 10-bit input PIO for SW[9:0]

Download and install the Quartus project available on the unit website as a Quartus ArchiveFile by double clicking the file with extension ***.qar*** - Quartus should run and ask where to install the project. If you now run Qsys from Quartus and open the ***AC.qsys*** file, the following modules should be available:



- Clock source – ensure the clk_in_reset has been connected (instead of exporting the "reset_n" conduit)
- NIOS II/f 32 bit RISC processor
- JTAG UART
- On Chip Memory
- 8 bit PIO interfaces to all six 7-segment displays
- 10 bit PIO interface to the 10 red LEDs LEDR[9:0].
- 2 bit PIO interface to the 2 push buttons Key[1:0].

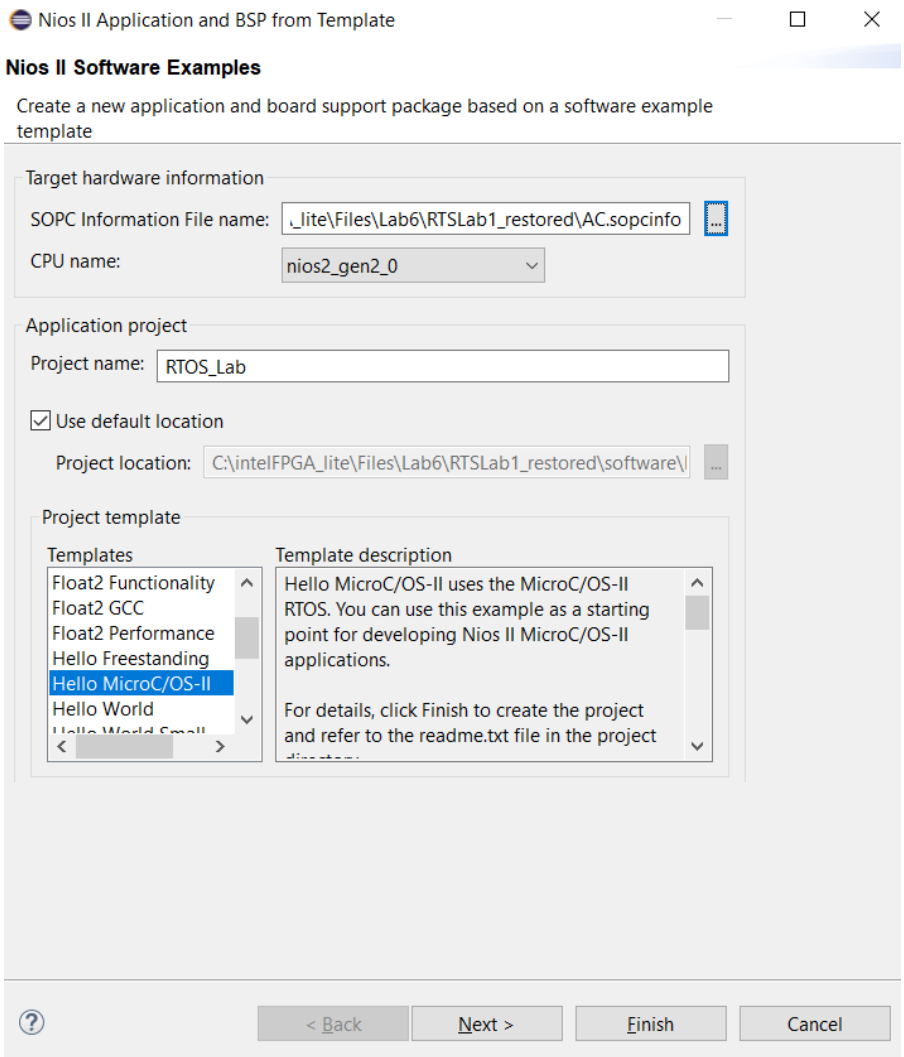
- Interval timer with a 10 msec “simple period interrupt”

This hardware configuration will be the basis for the real time laboratories.

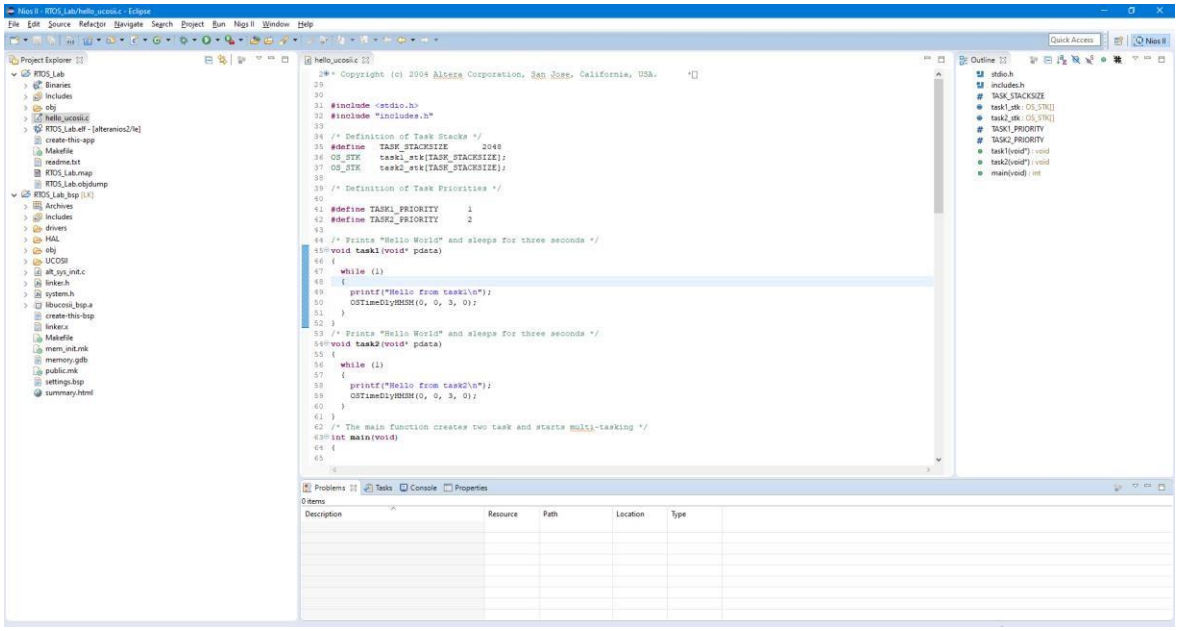
Check for any warnings or errors in the bottom message window – there should be none. The System menu,assign base address, and assign interrupt numbers commands will need to be run if you modify or add new modules. ***Generate*** the Qsys (right tab within Qsys), then compile the Quartus project, and download the hardware configuration to the DE2 -115. Run **Nios II Software Build Tools for Eclipse** for Eclipse from theSTART menu of Windows, under Intel FPGA.

From the Integrated Development Environment (IDE) use the file menu-> New ->Nios II Application and BSP from Template

Select the Hello MicroC/OS-II project template and the AC.sopcinfo file from your project menu (see the dialogue box below for an example), choose a Project name, and click Finish.

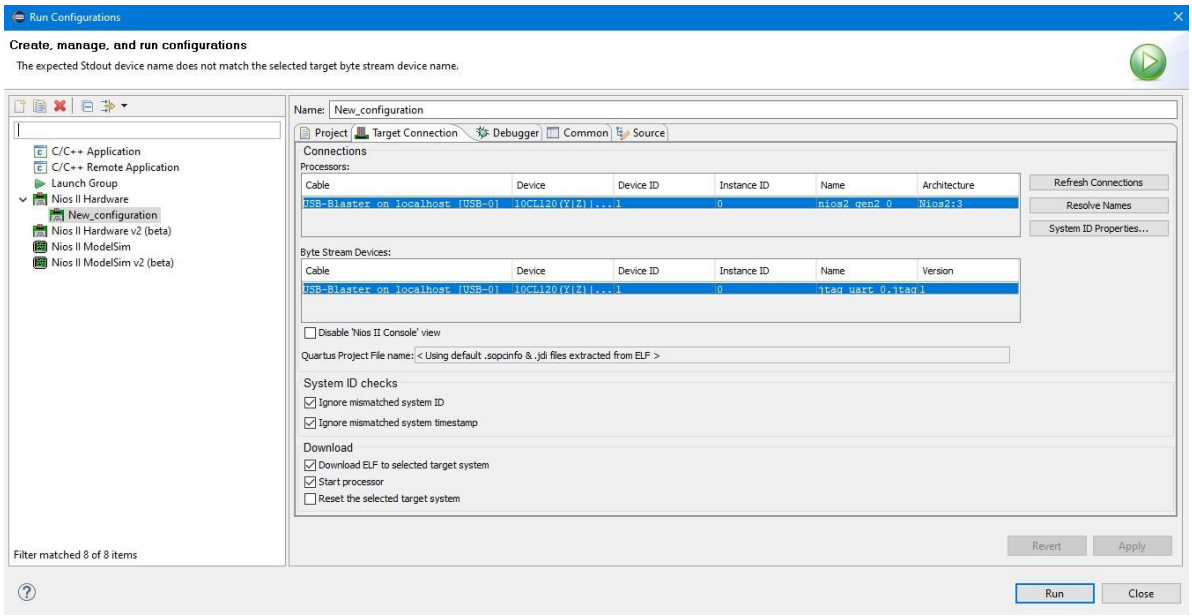


The IDE should look similar the following screen shot. You can open up the hello_ucosii.c file by expanding the folder underneath the Project Explorer, and double clicking the hello_ucosii.c file. To compile the hello_ucosii.c program, Build All (click on the 010 button or use Project Menu-> build all).



Change the defined value for “TASK_STACKSIZE” from 2048 to 1024. Once the compiler has finished with no errors, you can run the program. Ensure that youhave first downloaded the processor configuration from Quartus.

To run the program, press the “Run” menu, select “Run Configurations”. Double click Nios II Hardware, select your “Project Name”. Your “Project ELF file name” should pop up automatically. Ensure that your DE2 (or DE10) board has been detected. This can be done by clicking the “Target Connection Tab” and clicking “Refresh Connections” on the right hand side. USB-Blaster should appear under both “Processors” and “Byte Stream Devices”. Tick the “Ignore mismatched system ID” and “Ignore mismatched system timestamp” options . Press Apply and click Run. The program should download and run.



If you receive a communication error here, check that you have previously downloaded the processor configuration from Quartus. This error can also occur if no clock is connected tothe processor in Quartus or there are no pin assignments in the Quartus project.

Did you remember to change the TASK_STACKSIZE to 1024?

You should observe that there are two tasks running from the “Hello” console messages.

Section 1
Basics of Tasks

Which task runs at the higher priority?

Change the priority of task1 to match task2 in the `OSTaskCreateExt()` kernel call. Run the modified code. Document your findings and then restore the original priorities.

Can two tasks share the same priority in uC/OS-II?

Change the delay in task2 to 10 seconds and test the change.

Remove the time delay call in the lower priority task loop – comment it out using // at the beginning of the line. Does the higher priority task still run? (*Hint:* Reduce the delay in the higher priority task to 10 msec and simplify the output from the lower priority task to one "."to help decipher all the output).

Section– 2
Time of Day Clock

Write suitable code in task1 to perform a simple clock output to the 7 segment display in seconds minutes and hours – use just one task here! Ensure your timer is still set to *1 msec* for this section as it should be after completing the previous section. You should use the `OSTimeDlyHMSM()` kernel function call. Is this always going to be accurate?

Hint:

Including the "io.h" library and using the following macro:

IOWR(base_address, offset, data);

You can use the following macro to read from the base address:

data = IORD(base_address, offset);

The macros are used because in this lab, the processor is NIOS II/f which includes cache memory for higher performance. You need to access PIOs in this section and if this is done via memory pointers, there can be issues related to the cache intercepting the read/write operations and delaying the transactions to actual IO devices. By using the IOWR and IORD macros, the instructions compile into *stwio* and *ldwio* assembly language instructions that bypass the cache. You should use IORD and IOWR to avoid cache issues. Base addresses for each of the PIO devices are already defined under the system.h file.

Ensure you have included the io.h library, otherwise your C code will fail to compile!

Note 2: For the base addresses, you can use pre-defined variables that are generated upon building your Eclipse project. To find these variables, you can expand the .bsp folder in the project navigator and select "system.h"

Appendix A: Updating Qsys/Quartus

The following steps must be followed in order, and each step must have been finished completing first before moving onto the next one.

- In Quartus:
  - Make changes to Qsys
  - Regenerate Qsys
  - Compile top level design in Quartus
  - Program from Quartus onto the board (ensure that Eclipse has been disconnected from the board)
- In Eclipse:
  - Under the "Project Explorer" panel on the right, right click on the folder with the suffix "_bsp". Select the "Nios II" version, and click "Generate BSP"
  - After the BSP is generated, select "Project" and click "Clean…". Make sure the "Build the entire workspace" is selected. Press OK
  - After the Build has finished, open up your "Run Configurations" and press "Run". If you encounter an error, try pressing "Run" again.