#cec32xNotes, cec32x_ws_10_WS01_running_a_program_in_Keil_simulator

WS 1. Running a program in Keil simulator

We assume the Keil MDK-ARM tools have been installed. If not, please see the handouts of Lab 0 and install the tools.

Verifying the demo code works

- Go to the **Dev tools** section under **Modules** of the CEC 320 Canvas page.
- Read the **Project file structure for simulator-based projects** section of `cec32x_devtool_20_organization_of_proj_files.pdf` to understand the file structure of the simulator-based template project.
- Download `expl_010_template_for_simulator_prjc_with_c.zip`, the zipped file for the simulator-based template project. Unzip the file to have a folder structure as described in the above pdf file.
- Double-click `expl_c_prjt_sim.uvprojx`, the main project management file to open Keil MDK_ARM.
- Follow the instructions in the **Running a program in the Keil simulator** section of `ece32x_devtool_22_running_debugging_a_program_inKeil.pdf` to build and run the program. You should be able to see the following printout in the **Debug (printf) Viewer** window: "The values of my_ids are 1234 and 1522756, respectively."

A demo will be provided in the class.

Programming tasks

(80 points total)

- Change the project folder to `cec32x_ws01_running_a_program_in_keil_sim`. Change the name of the project management files to this name as well.

- Rebuild and run the project to make sure there is nothing wrong.

- (10 points) Replace all `my_id2` by `ID2` and all `my_id` by `ID1` in the `main.c` code. Note that you need to pay attention the order of doing the replacement otherwise, you may introduce bugs.

- (10 points) Assign a 4-digit decimal number to `ID1` according to your team. For a team of two students, you put together the last 2 digits of the ERAU ID of each member to have a 4-digit number; for those working alone, you can use the last 4 digits of your ERAU ID.

- (10 points) Change line 17 to

```
ID2 = ID1 + (ID1 << 16);
```

The intention of this change is to have `ID2` as the repeated version of `ID1` when printed in Hexadecimal. For example, if `ID1 = 0XABCD`, then `ID2 = 0XABCDADCD`. Here the leading `0X` stands for Hexadecimal.

- (20 points) Change line #19 so that the print-out results are in Hexadecimal, instead of decimal. You may want to use `%04X` and `%08X` for `ID1` and `ID2`, respectively. Of course, change the texts for `ID1` and `ID2` in the string as well.

- (20 points) Add one line after line #19 so that you can print out the addresses of `ID1` and `ID2`. You may want to use %p to format the printing of the **addresses** of `ID1` and `ID2`. Note that to get the address of a variable, you need to use the `&` notation. As `ID` and `ID2` are in a data struct, their addresses are related and can be found from the address of that data struct as well. If you have trouble or questions, please ask in class (CEC320).

- (10 points) Add one line just before the line of "while (1)" to print out your names using something like:

```
printf("I certify that I, firstname lastname, finished \n");
printf("this work independently.\n");
```

if you work in a team, or

```
printf("I certify that I, firstname lastname, finished \n");
printf("this work independently based on discussion with \n");
printf("[your team member's name].\n");
```

if you work in a team.

- Build and debug (run) the modified project.

Submission of your work

(20 points)

Each student needs to submit their own results—code snippet for the main function and the screenshots for the above printout results in the form of a pdf file. Put your name at the top of the page before the code snippet. Name your file as cec320_ws1_lastname_firstname(or initial).pdf. You also need to zip all your project files (not including the build files) into a single zip file and upload this file as well.