CE-2812, Lab #1, Programming Environment Setup

1 Purpose

The purpose of this lab is to setup our development environment, and create a template project to allow our embedded system to implement stdin/stdout via the serial console.

2 Prerequisites

- The Nucleo-F446RE board had been mounted onto the Computer Engineering Development board.

3 Activities

3.1 Install CubeIDE (if needed)

Either download from the Box/Tools folder or directly from STMicroelectronics (requires registration), unzip, and install. All defaults may be selected during installation.

The first time you run CubeIDE, you will be asked to select a workspace directory. The default should be fine, but you may locate anywhere on your system. You may then land on a welcome page. If you close that tab, you will be in the normal Eclipse workspace.

3.2 Install RealTerm

Located in Box/Tools folder. Default options are fine.

3.3 Setup Template Project

Follow the instructions provided in the document **"Creating Template Project in CubeIDE with Serial Console.pdf"** which is in the shared Box folder. The code needed is there too in **uart_driver.zip**.

Create a new project based on the template. Build and run to verify is works with RealTerm (or other serial terminal program such as PuTTY or Tera Term – I recommend RealTerm). Note that there may occasionally be discrepancies between the behavior of the serial console and the behavior of a real console. In RealTerm, enabling "newline mode" will offer best compatibility.

3.4 Copy-Paste Program

In the new project, replicate (copy-paste or type) example program Example 1.4 from TCB (The C Book). It starts on page 26 of the pdf version of the book and spans to the next page.

Tweak the program for our environment – specifically, you will need the #include "uart_driver.h" and the call to init_usart2 at the beginning of main. Also note that the program calls the function "exit" at the end. You may leave that intact. You may need to fix up the ' and " characters which do not copy-paste properly from the pdf and there is at least one other misprint that you will need to fix.

Build and run program. You should be able to decipher the purpose and behavior of the program by skimming the code.

3.5 DEBUGGING

Experiment with the program in the debugger. We are now able to step by lines of 'C' code which may be several assembly instructions each. You can also inspect and watch the 'C' variables. Be sure to observe the following:

- What is the behavior of the function 'getchar()'? Does it behave the way you might expect it to just based on its name? Search for 'getchar' on the Interweb. What do you find for documentation?
- Embedded systems should not let the program end. This program was not written for an embedded system and does end. What does our system do when main is allowed to end? Use the debugger to investigate. As part of your investigation, remove the last line of the program (the call to exit()) and investigate that behavior as well.

3.6 DELIVERABLES

Demonstrate a working program before leaving lab in week 1. There is no explicit deliverable to be uploaded to Canvas.