National University of Singapore
Dept of Electrical & Computer Engineering (ECE)

CG2028 Microcontroller Programming and Interfacing

Lab Session 1
Familiarization with Hardware and Development Environment

Objectives of this session

The objectives of this session are:
1. to become familiar with the LPCXpresso Integrated Development Environment (IDE), and
2. to build, download and execute: (a) a simple assembly language program, and (b) a C program, on the NXP LPC1769 ARM Cortex-M3 system on chip on the LPCXpresso board.

Note: In the following Part 1 and Part 2 of this lab session, you will use an assembly language program and a C program, respectively. At this stage, you are not expected to understand fully every single line in these programs (although some explanation is given in the last section of this lab manual). The emphasis of Lab Session 1 is to familiarize yourself with the IDE and its features, the program compilation and download process, and to ensure that your hardware and cables, as well as software configuration, are all working properly.

Reference Materials

- LPCXpresso User Guide (13 January 2014) for LPCXpresso IDE Version 6 onwards. Henceforth, we shall refer to this document as [LUG].
- NXP UM10360 LPC17xx User Manual

Instructions

Log on to the PC in the Digital Electronics Laboratory.

The LPCXpresso Integrated Development Environment (IDE) is already installed on the PC. Henceforth, we shall refer to this IDE as “LXIDE”.

Note:
You can also install LXIDE on your own PC or notebook PC so that you can work on your assignments outside the scheduled lab hours.
Get the installation file from the IVLE “Lab and Assignment Sessions” workbin, or directly from http://www.lpcware.com/lpcxpresso/download (LPCXpresso IDE v6.1.4). You will need to activate the **free edition** of the software by registering with NXP - simply follow the instructions after installation to obtain the activation code and enter it.

You need a USB thumbdrive with at least 50 MB of free space so that it is convenient for you to work with both the lab PC and your own PC (e.g. after lab hours).
Bring your thumbdrive with you for every lab session.

CAUTION: Be careful not to lose your thumbdrive as it contains all your work. You are advised to back up often to another place, e.g. your PC hard disk.

Create a folder called “CG2028workspace” on your thumbdrive.
Copy the file “CG2028 Part 1.zip” (found in the same IVLE workbin folder) into this folder.

Start up LXIDE.

You are prompted to enter the path to your workspace. Click on Browse, Computer, and locate your thumbdrive and the “CG2028workspace” directory there.

Read pages 13-14 of [LUG], i.e. Section 3.4 “Major components of the Develop Perspective”, to familiarise yourself with the 5 main sections of the **Develop** perspective of LXIDE that you will be using mainly.

Read pages 15-24 of [LUG], i.e. Section 4 “Importing and Debugging example projects”, to get an overview of the program compilation, downloading and debugging process. However, **do NOT perform any of the actions yet** as you will be guided step-by-step in the next section of this lab manual.

In Section 4.1 “Importing an Example project”, **instead of** going to the .\lpcxpresso\Examples\NXP directory to load the NXP LPC800 example, go to the “CG2028workspace” directory on your thumbdrive and select “**CG2028 Part 1.zip**”. Click on “Next>”, and “Finish” at the next screen to import all the 7 projects listed. This will take a few minutes.

Part 1. LPC1769_asm_blinky

In this part of the lab session, you will build the **LPC1769_asm_blinky** project in LXIDE. LXIDE produces an executable image for the NXP LXP1769 ARM Cortex-M3 system-on-chip. It then downloads the executable image to the NXP LPC1769 and starts to run it.

Make sure that the correct target platform, i.e. LPC1768 or LPC1769, has been selected. This is indicated in the lower right corner of LXIDE.

Refer to Section 4.2 Building projects and Section 4.3 Debugging a project of [LUG].

Connect the LPCXpresso board (please handle with care) to the PC using the provided Mini-B-to-USB cable.

Instead of clicking on Debug 'Blinky', as in shown in Figure 4.5 of [LUG], left click on the **LPC1769_asm_blinky** project in the Project Explorer (top left) pane.

Left click on “Debug ‘LPC1769_asm_blinky’ [Debug]” in the Quickstart (bottom left) pane.

You will see various messages in the Console tab (bottom pane) as the program is compiled and the executable image is downloaded to the LPC1769.

Select the LPC-Link emulator and click “OK”. Refer to Section 4.3.1 Debug Emulator Selection Dialog of [LUG].

If you have done everything correctly, the red LED on the LPCXpresso board starts to blink slowly (hence the name “blinky”).

Congratulations! You have just successfully compiled and executed your first ARM assembly language program!

Refer to Section 4.3.2 Controlling execution (pages 23-24) of [LUG].

Click on the Pause button (2 yellow vertical bars) to pause the program.

You can inspect and modify the contents of the processor registers (both general purpose and special registers) by clicking on the "Registers" tab (top left pane) in LXIDE.

You can inspect and modify the contents of the memory by clicking on the Memory tab (bottom pane), adding a monitor (click on "+") and entering the address of the memory location you are interested in. The contents of a word in memory starting from the address you entered, as well as the surrounding memory locations, will be shown.

To resume the program, click on the Run/Resume button with the yellow vertical bar and green right arrow.

To stop the program completely, click on the button with the red square.
IMPORTANT: Make sure you do this before you build your program again, e.g. when you modify your program or switch to another project. Otherwise, error messages will appear.

Familiarise yourself with the debugging features of LXIDE as described in Section 4.3.2 Controlling execution (pages 23-24) of [LUG], e.g. toggle Breakpoint, executing the program one instruction at a time (Step Into, Step Over, Step Return), inspect peripheral registers (in the Peripherals tab (top left pane): select the register by ticking the square on the left; a new entry appears for this peripheral register in the Memory tab (bottom pane)) and the condition flags (nzcvq) (in the Registers tab mentioned earlier (scroll down)), Variables tab (bottom left pane) etc.

Part 2. LPC_1769_plainC_blinky

The task of making the red LED (LED2) on the LPCXpresso board blink can also be done using a C program.

Make sure the program in Part 1 has stopped completely. If not, click on the button with the red square.

Left click on the **LPC1769_ plainC_blinky** project in the Project Explorer (top left) pane.

Build the project and download the executable image by left clicking on "Debug 'LPC1769_plainC_blinky' [Debug]" in the Quickstart (bottom left) pane.

You will see various messages in the Console tab (bottom pane) as the program is compiled and the executable image is downloaded to the LPC1769.

If you have done everything correctly, the red LED on the LPCXpresso board starts to blink slowly.

Congratulations! You have just successfully compiled and executed your first C program on the ARM platform!

Similar to the above, you can pause the program and inspect the registers and memory locations, and also trace the running of the program.

You can also see the actual ARM assembly language instructions and the offsets that are computed by the assembler and linker when you select the Disassembly view (click on the

button with ‘i' and yellow right arrow in the Debug pane (top right corner)). A new Disassembly pane will appear.

Food for Thought: Try selecting the Disassembly view at the end of Part 1. Why is there a difference between the assembly language program and the disassembled code?

This concludes this lab session.

Stop any running program completely by clicking on the button with the red square.

IMPORTANT: Shut down the LXIDE application by selecting File – Exit on the menu. It will take about 10 seconds to save and close your workspace. DO NOT remove your USB thumbdrive until the LXIDE application has completely ended. Otherwise, you will corrupt your project files and make them unusable.

For Information Only

What was going on in the ‘blinky’ programs?

After you have completed Part 1 and Part 2 above, i.e. after the lab session, you should read the programs carefully to familiarize yourself with the structure and format of the programs, as well as try to understand how the program instructions caused the red LED on the LPCXpresso board to blink slowly. These programs will be discussed in greater detail during the lectures in Weeks 3 to 5. Assignment 1 will utilize this foundation knowledge to perform an engineering task.

The main parts of the 2 programs with comments are shown in the Appendix to this Lab Manual and an introductory description of what is happening will now be given.

Basically, the programs turn on and off a digital output line in the General Purpose IO (GPIO) port 0 pin 22 that is connected to the red LED. However, the physical pin of the LPC1769 has several functions and the GPIO functionality has to be selected by writing a ‘00’ pattern in bits 12 and 13 of the PINSEL1 register (at address 0x4002C004).

After the GPIO functionality has been selected, we need to set the direction of port 0 pin 22 to be *output* by writing a bit pattern into the FIO0DIR register (at address 0x2009C000).

The state of GPIO port 0 pin 22, i.e. whether it is on/high or off/low, can be determined by reading bit 22 of the FIO0PIN register (at address 0x2009C014).

To turn on the red LED, make the state of GPIO port 0 pin 22 on/high by writing ‘1’ to bit 22 of the FIO0SET register (at address 0x2009C018).

To turn off the red LED, make the state of GPIO port 0 pin 22 off/low by writing ‘1’ to bit 22 of the FIO0CLR register (at address 0x2009C01C). Note that you do not turn off the red LED by writing ‘0’ to a bit in a register.

Finally, there is a delay loop to make the red LED stay on or off for a short duration before switching to the other state. The whole cycle runs repeatedly.

THE END