Lab 01

Developed by: Muhammad Imran and Peng Cheng

Date : 2015-02-06

Course : Programming Embedded Systems (ET014G) Mid Sweden University, Sweden.

This lab has three tasks which will enable you to develop understanding of the AVR32 tools i.e., EVK1100 evaluation kit, its software tools including AVR32 studio and framework. Task1 is preparatory task and is required to be done before coming to lab.

Task 1

How to find pin mapping between microprocessor and evaluation board:

In the EVK1100 evaluation board's schematic, you can find different peripherals such as LEDs, push buttons, potentiometer, LDR, etc. These peripherals are connected to microcontroller through GPIOs. In order to use these peripherals, you have to map the pin of the microprocessor to the required peripheral because some pins are multiplexed for three different functions, discussed in later section.

Example: if you want to turn on LED0 (It is LED1 on physical layout of the board), you can find in the schematic that it is PIN27 which connects LED0 (Figure 1), through port PB to the microcontroller PIN 15 of the package (Figure 3). PIN27 is also shared with expansion header J26 (Figure 2). The GPIO number for this pin is 59.

In AVR studio, you can find this mapping in "evk1100.h"

- #define LED0_GPIO AVR32_PIN_PB27

And its low level mapping on actual hardware can be seen in "uc3a0512.h"

- #define AVR32_PIN_PB27 59

Finding port and Pin configuration

Each GPIO line can be assigned to one of 3 peripheral functions; A, B or C as shown in Table 1 and each GPIO line has a unique number ports PA, PB, PC and PX which corresponds to the GPIO pins. For simplicity, the GPIO pins are grouped in different ports. However, the numbering on ports is not directly translated to GPIO numbering. You can use the following formula to find the port and pin number or easy way is to see the AVR datasheet (Table 1) for the pin mapping. NOTE : 32 in the formula corresponds to the fact that "The pins are managed as 32-bit ports" (section 21.5 in datasheet)

Port = floor((GPIO number) / 32), example: floor((59)/32) = 1 (Note: 0 corresponds to A, 1 to B, and so on)
Pin = GPIO number mod 32, example: 59 mod 32 = 27

For example for LED0

Component	**PIN**	**GPIO PIN**	**GPIO port/PIN**
LED0	PB27	GPIO 59	1/27

Table 1. (Table 12.9 in datasheet) GPIO controller function multiplexing

TQFP100	**VQFP144**	**PIN**	**GPIO Pin**	**Function A**	**Function B**	**Function C**
9	14	PB26	GPIO 58	TC – B1	USART1 - RI	
10	15	PB27	GPIO 59	TC –A2	PWM – PWM [4]	

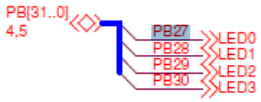

Figure 1. LEDS

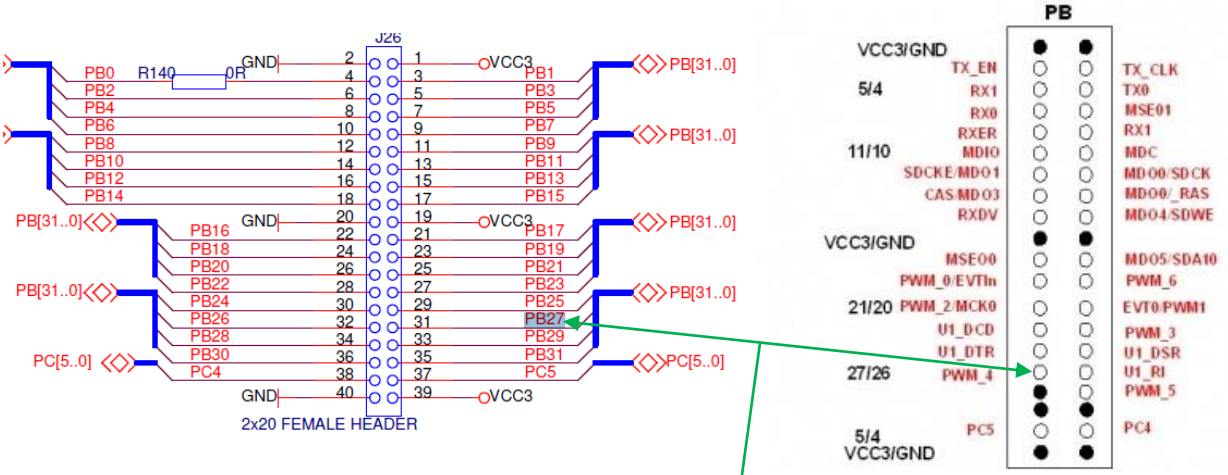

Figure 2. 2x20 header.

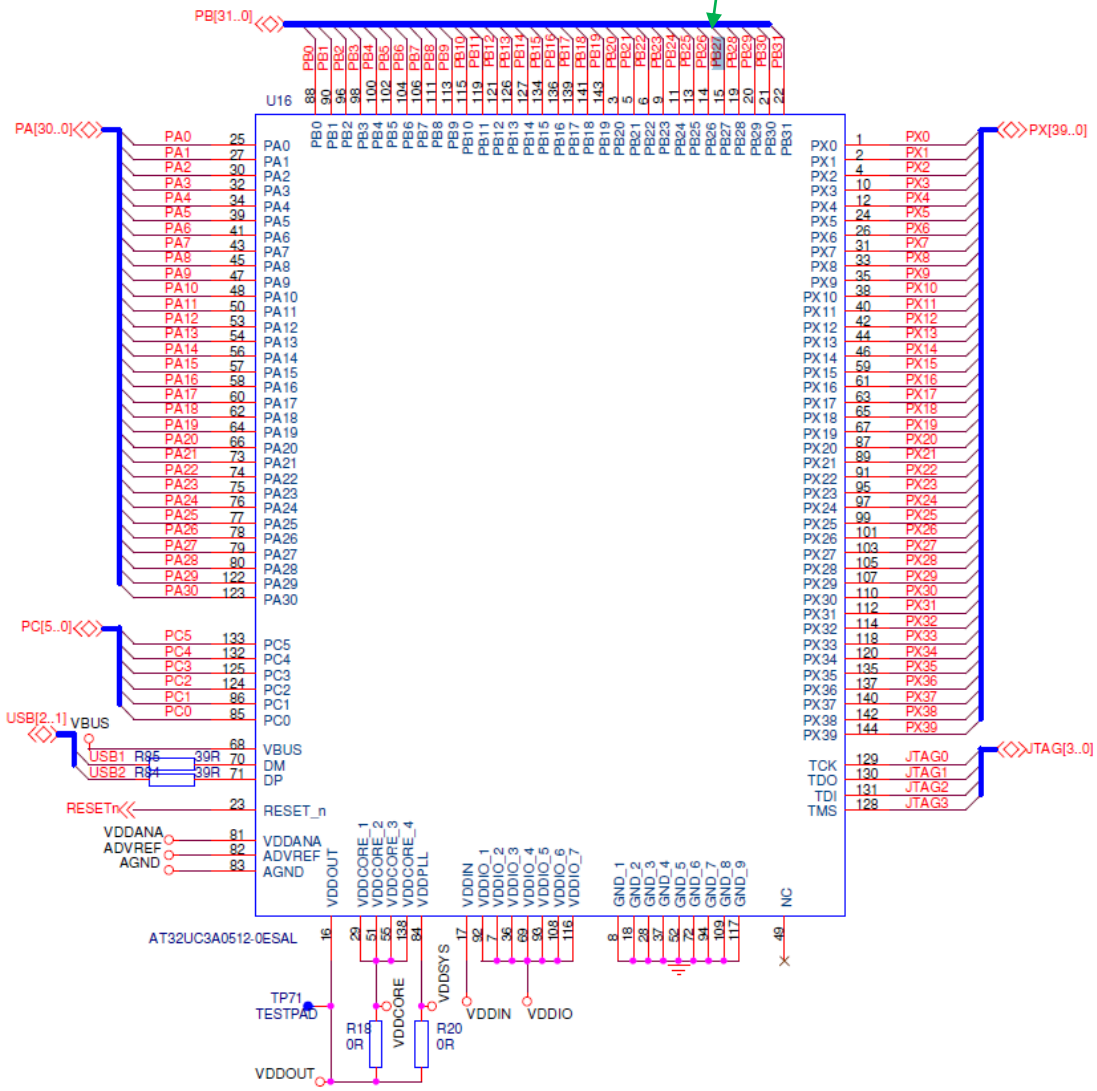

Figure 3. Microcontroller

➡ **Show to the lab demonstrator following sub-tasks,**

How many ports does AT32UC3A0512 has? _____

Names: _____, _____, _____, _____,_____,_____,_____,_____,_____,

Which port and PIN does GPIO 61 corresponds to?. PORT_____, PIN _____.

What are three functions that GPIO 61 can be associated to.

(1) _____

(2) _____

(3) _____

Task 2

You have developed understanding of using the datasheet and schematic of the board and microcontroller. Next, step is to use the software for programming the microcontroller.

Start AVR32 studio

- In the lab computer, AVR studio is located in C:\Program Files\Atmel\AVR Tools\AVR32 Studio.
- Click on the windows start button and locate AVR32 studio under the *Program* menu.
- The first time you start the program you might need to select a location of you workspace.
- If not, click on the File menu and select switch workspace. Select a location for you workspace and create folder at that location called workspace (important)
- Create new project by opening *File/New/Project/AVR32 Project from template* as shown in **Figure 5**Figure 4. Give authors information and click finish.

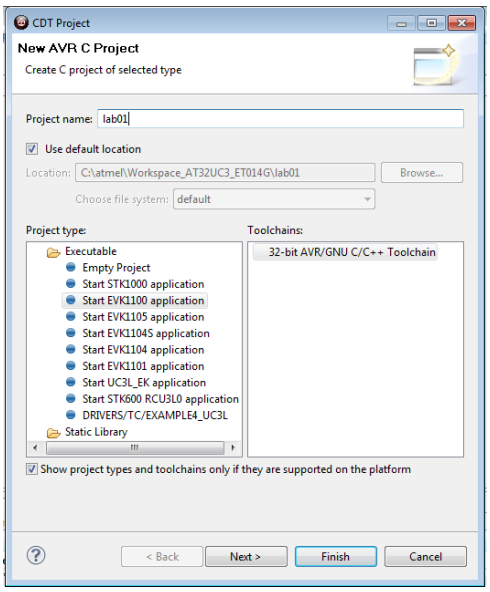

Figure 4. Project template

- Go to the menu bar and click Framework as shown in Figure 5 for selecting required drivers/components/service from the framework. The drivers' window is shown in Figure 6. Similarly select components and services and click finish.

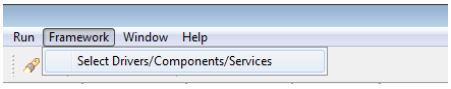

Figure 5. Framework

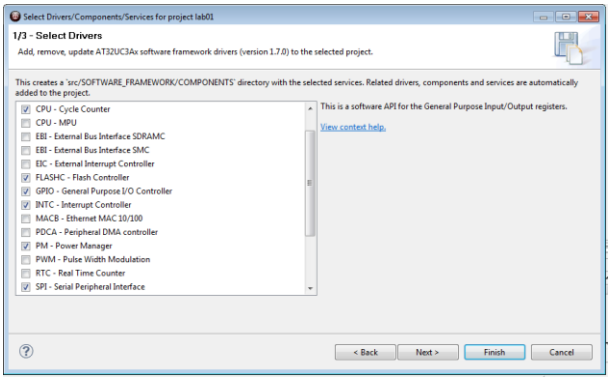

Figure 6. Selecting drivers/components/services

- Now you add/write your required source code/header files in the main as shown in Figure 7.

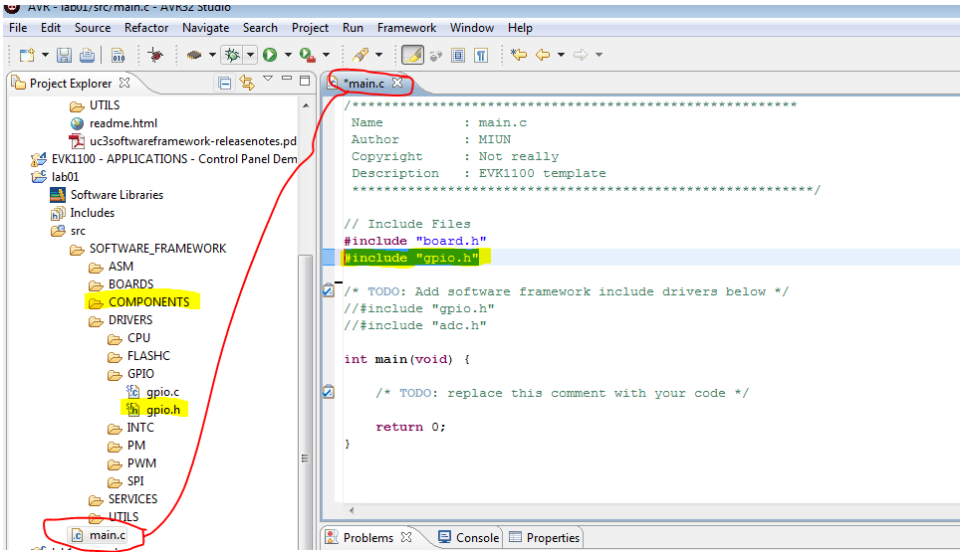

Figure 7. Adding header files in main

➡ Write a program that sets the CPU frequency to 12 MHZ from OSC0. The program gets value from push button0 and turn on LED6 by using polling method. The power manage drivers in the framwork can be used to set the frequency.
You can find functions to LED and push buttons in “gpio.h” and “evk1100.h”.

Programming the device

Create new programming target as shown in Figure 8

- Connect the USB cable between the EVK1100 and the computer
- In the AVR32 target window create a new target as shown in Figure 8
- Right click on the target and select properties
- Select USB DFU programmer
- Select Microcontroller UC3A0512

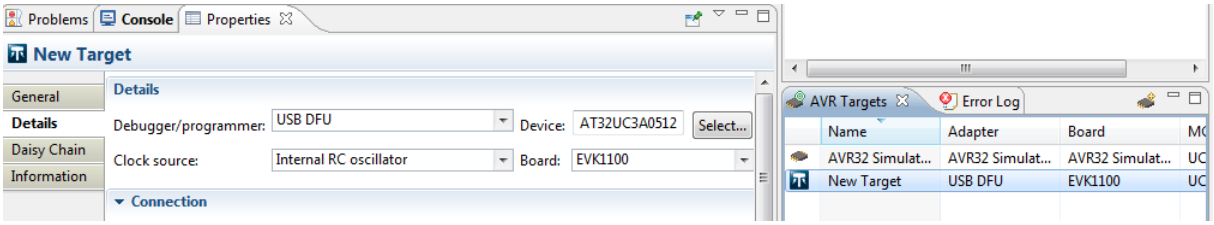

Figure 8. Target selection

Download the program to the processor

- The AVR32 is pre-programmed with a bootloader in order for your program to be downloaded to the correct starting address.
- Set the main power switch to the USB position
- To activate the bootloader, Push and hold the Joystick
- While pushing the joystick, push and release the reset button
- Right click on the USB programming target
    - Select program
    - Locate the .elf file in you Debug folder of you project as shown in Figure 9.
    - Mark all options and press OK
    - Your program will know download to the AVR32

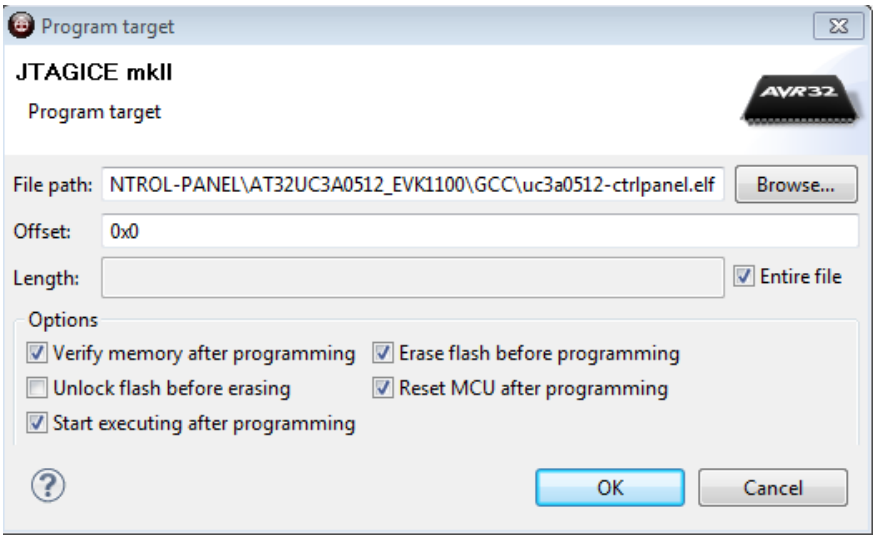

Figure 9. Elf file location

Task 3

➡ Write a C program which calculates the following numbers. Select suitable datatypes for these numbers.

x = 12345678;
y = 87654321;

a = 1234.5678;
b = 8765.4321;

Measurement start here
z = x * y;
Measurement stop here

Measurement start here
c = a * b;
Measurement stop here

- Now, measure their calculation speed by using the CPU cycle counter and use the LCD display of the EVK1100 board to show the measured results.
- Display the time in micro second after converting the CPU cycles with following formula (you can see this formulas in cycle_counter.h).

fcpu_hz = 12000000;
time in us = (CPU cycles* 1000000 + fcpu_hz-1) / fcpu_hz;

The program needs to be analysed for the two cases:
Case1 : (set optimization level to 0)
Case2 : (set optimization level to 1)

Motivate from the ATMEL instruction manuals, what these optimizations are used for.

Optimization 0:
_____

Optimization 1:
_____

TIPS:

You will need these drivers, components and service in this lab

- CPU cycle counter driver: CYCLE_COUNTER".
- LCD display driver : DIP204".
- Power manager driver :PM-power manager

How to set optimization level

You can either set it using graphical way as shown in Figure 10 or changing it in the config.mk file as shown in Figure 11.

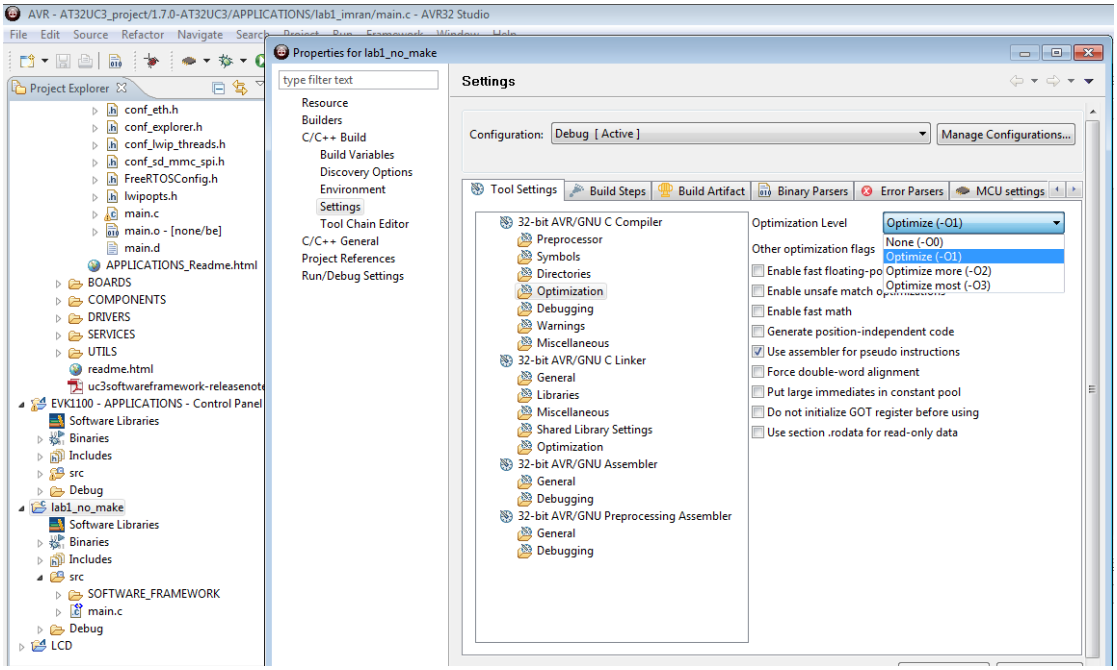

Figure 10. Setting optimization level

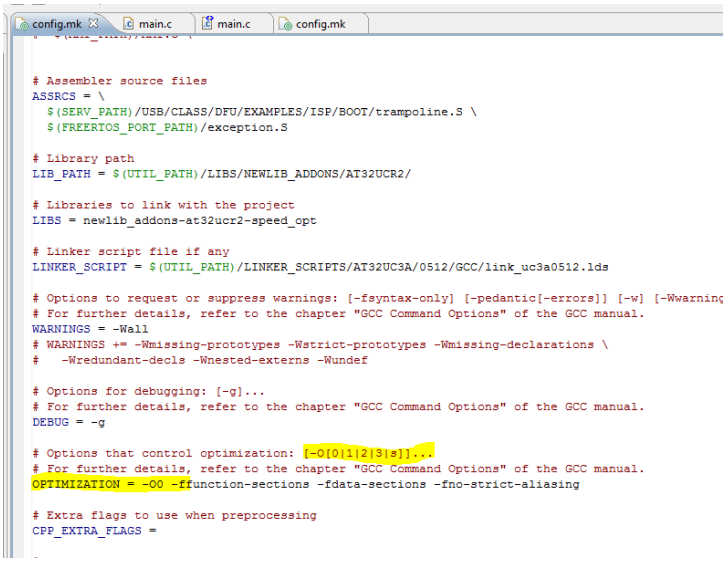

Figure 11. Setting optimization level in config files