Laboratory 5
Dr. Don Evans
CSE 2240 –Spring_2017_Lab_Assignments
Assembly Language Programming and Machine Organization
TA: Naseer Jan

Section N11 - Tu 12:00PM to 1:50PM Junkins 215
Section N12 - Th 3:00PM - 4:50PM Junkins 215
Section N13 - Tu 8:00AM - 9:50AM Junkins 217

Laboratory 5

This lab comprises three parts: part-I (Board Set Up and Blinking LEDs Example using potentiometer), part II (i.e. project setup) & Part-III (ARM assembly programming for LED Blinking using potentiometer). This lab enables you that how to control LEDs and introduce delay among LEDs using potentiometer

STEP I: Board Set Up and Blinking LEDs Example

a) Open the ARM evaluation board user's guide
http://www.keil.com/support/man/docs/mcbstm32c/default.htm
b) Follow steps given in sub-topic 'setup' to setup your board
The MCBSTM32 board requires a power connection and a ULINK-USB adapter between your development PC and the JTAG connector.
To use the MCBSTM32C Evaluation Board with ULINK, you must:

- Connect the ULINK adapter to the JTAG connector of the MCBSTM32C board
- Connect the ULINK adapter to the PC via a standard USB cable
- Connect power from your PC to the board using either a USB A to Micro A or USB A to Micro B cable

c) Load and Debug Blinky project to understand the concept which is as follows:

You can see this in action using uVision Debugger.
Go to: C\ Keil_v5\ARM\Boards\Keil\MCBSTM32C\Blinky\ Blinky.uvproj

Click the **Download to Flash** toolbar button. This executes the Flash download program (**Target Driver** or **External**) selected in **Project — Options for Target — Utilities** and downloads the application program into the STM32F10x device

Start the Debugger.


Use the **Start/Stop Debug Session** toolbar button to start debugging the program.

Or

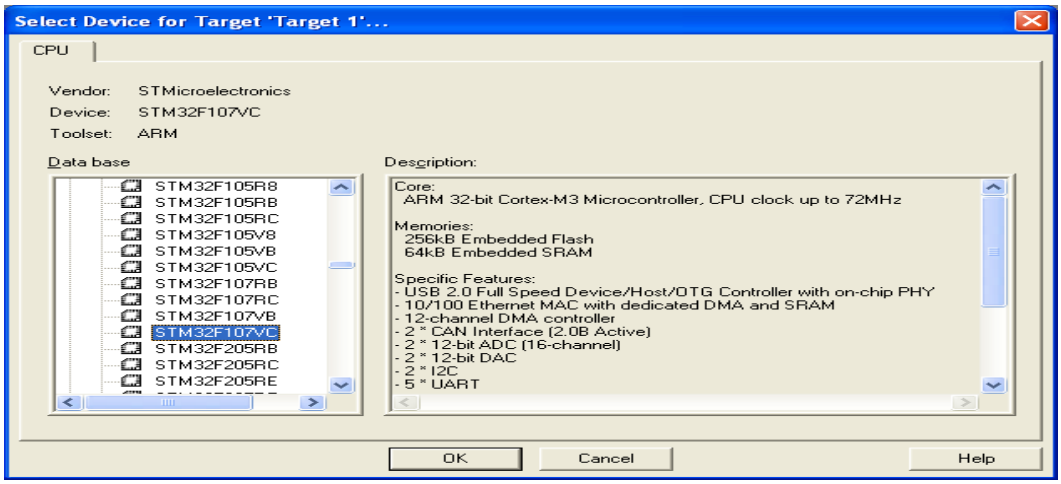
Use the µVision command, **Debug — Start/Stop Debug Session**, to start debugging the program.

STEP II: Project setup

a) Create new project under **Project → New → µVision Project**, name it 'led' and save it under project directory. Select the device STM32F107VC under STMicroelectronics. DO NOT add the default startup file. The default startup file is meant to work with C-programs. We have edited the default startup file so that it works with ASM code.

Device selection

Select the microcontroller from the Device Database
Vendor: STMicroelectronics
Device: STM32F107VC



b) Please check the appendix to verify if your project settings are correct.
c) Create groups 'startup' and 'source' to organize all project files
d) Copy the file ***system_stm32f10x_cl.s*** in the project directory and in Keil add under group 'system'
e) Add a new source file under 'source'. You can now start writing assembly codes for your lab (refers to **step III**).

STEP III: ARM assembly programming for LED Blinking

Write an ARM assembly program using c code for Blinky project. The 'Blinky' project is a simple program for the 'STM32F107VC' microcontroller using Keil 'MCBSTM32C' Evaluation Board.

Functionality: 8 LEDs blinking with speed depending on potentiometer

In this lab, we are going to use both input (Potentiometer) and output (LEDs) devices. In lab#3, we wrote ARM code to turn ON/OFF LEDs on board sequentially (CSE2240lab#3) and controlled LEDs from different values in look up table (CSE2240lab#4). Since these LEDs turn ON/OFF at high speed we used delays between each LED being turned ON/OFF so that we can observe it. Previously this delay value was an arbitrary fixed value. For this lab, we are going to use the potentiometer to change the delay and thus change the frequency at which the LEDs turn ON/OFF. The voltage applied to the input pin of the microcontroller using the potentiometer (blue knob on board) is converted to a digital value using Analog to Digital (AD) converter module of the microcontroller. This value will be much smaller than the fixed delay values we used in (CSE2240lab#3) and (CSE2240lab#4). So, we are going to use the digital value obtained from AD converter to generate a bigger value to use in our delay loop (delays between each LED being turned ON/OFF).

Hints:

- /* Setup and initialize ADC converter in C program i.e. from line # 29 to line#53. Use ADC converter instead of user specify value

- Setup the clock for port E (GPIOE) and configure port E to work as an output port. LEDs are connected to this port.

- Calculate and determine the addresses of registers required to configure and use port E by referring to section 2.3 Memory map, Table 1. Register boundary addresses for RCC and GPIOE.

- RCC_APB2ENR refers section 6.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)
- GPIOE_CRH and GPIOE_BSRR refers to section 8.2 GPIO registers.

- For example: address of GPIOE_BSRR (Port bit set/reset register) is GPIOE boundary address =0x4001 1800 + and BSRR = 0x10
- GPIOE_BSRR = 0x40011810

- You will have to store these addresses in one the general purpose registers to access these port registers.

- To setup the clock for port E store value 0x40 in APB2ENR

- To setup port E as an output port store value 0x33333333 in GPIOE_CRH

- To turn ON any port pins on port E set appropriate bits in lower half of GPIOE_BSRR. Please refer to the 'ARM STM32 Reference Manual' for Cortex M3 for more information about BSRR.

- To turn OFF any port pins on port E set appropriate bits in lower half of GPIOE_BSRR. Please refer the reference manual for Cortex M3 for more information about BSRR

```
#include <stm32f10x_cl.h>

#define LED_NUM     8                   /* Number of user LEDs                      */

const long led_mask[] = { 1<<15, 1<<14, 1<<13, 1<<12, 1<<11, 1<<10, 1<<9, 1<<8 };


int main (void) {
int AD_val, i;
int num = -1;
int dir =  1;

SystemInit();

/* Setup GPIO for LEDs                                                        */
RCC->APB2ENR |=  1 <<  6;           /* Enable GPIOE clock                     */
GPIOE->CRH    = 0x33333333;         /* Configure the GPIO for LEDs            */

/* Setup and initialize ADC converter                                         */
RCC->APB2ENR |=  1 <<  9;           /* Enable ADC1 clock                      */
GPIOC->CRL   &= 0xFFF0FFFF;         /* Configure PC4 as ADC.14 input          */
ADC1->SQR1    = 0x00000000;         /* Regular channel 1 conversion           */
ADC1->SQR2    = 0x00000000;         /* Clear register                         */
ADC1->SQR3    = 14 <<  0;           /* SQ1 = channel 14                       */
ADC1->SMPR1   =  5 << 12;           /* Channel 14 sample time is 55.5 cyc     */
ADC1->SMPR2   = 0x00000000;         /* Clear register                         */
ADC1->CR1     =  1 <<  8;           /* Scan mode on                           */
ADC1->CR2     = (1 << 20) |         /* Enable external trigger                */
(7 << 17) |                         /* EXTSEL = SWSTART                       */
(1 <<  1) |                         /* Continuous conversion                  */
(1 <<  0) ;                         /* ADC enable                             */
ADC1->CR2    |=  1 <<  3;           /* Initialize calibration registers       */
while (ADC1->CR2 & (1 << 3));       /* Wait for initialization to finish      */
ADC1->CR2    |=  1 <<  2;           /* Start calibration                      */
while (ADC1->CR2 & (1 << 2));       /* Wait for calibration to finish         */
ADC1->CR2    |=  1 << 22;           /* Start first conversion                 */

for (;;) {                          /* Loop forever                           */
if (ADC1->SR & (1 << 1)) {          /* If conversion has finished             */
AD_val = ADC1->DR & 0x0FFF;         /* Read AD converted value                */
ADC1->CR2 |= 1 << 22;               /* Start new conversion                   */
}

/* Calculate 'num': 0, 1, ... , LED_NUM-1, LED_NUM-1, ... , 1, 0, 0, ...     */
num += dir;
if (num >= LED_NUM) { dir = -1; num = LED_NUM-1; }
else if   (num < 0) { dir =  1; num = 0;         }

GPIOE->BSRR = led_mask[num];        /* Turn LED on                            */
for (i = 0; i < ((AD_val << 8) + 100000); i++);
```

```
GPIOE->BSRR = led_mask[num] << 16;  /* Turn LED off                    */
}
}
```

Demonstrate result to the lab instructor

Print and submit the file led.s