Lab 6: ARM MCBSTM32C Interfacing

EE 5385

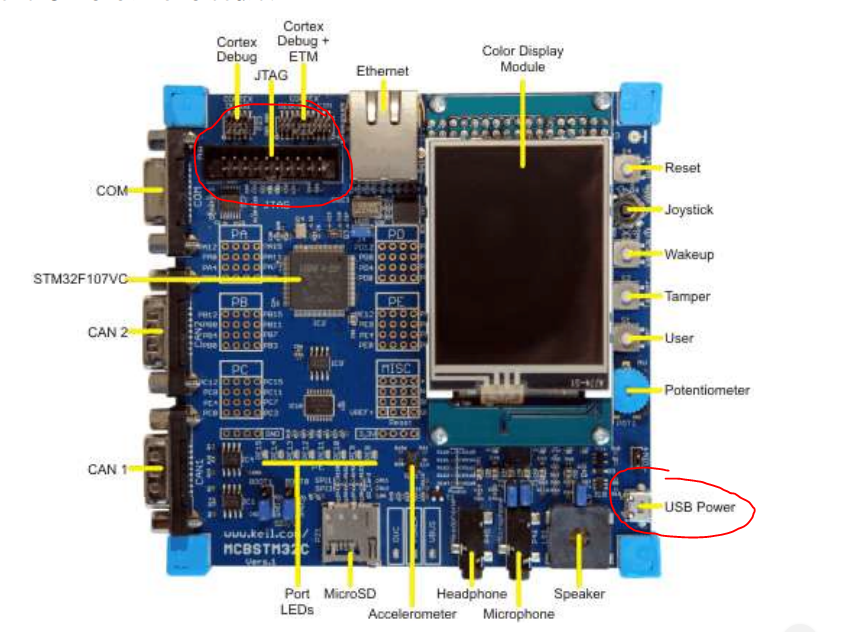
Lab Report

4/16/18

Victor Yuan

Turn on and setup (same as previous two labs)

First, I setup the board like the first lab by connecting an USB cable with an ULINK-ME adapter. Then I connect this adaptor to the JTAG pins in the figure below:



Then I connected the USB to the USB power port on the bottom right of the figure. Connect both ends of the USB cables to the computer one two power the board and one to program the board.

Then I used the program Keil uVision, first I created a new project from scratch. I selected the microcontroller vendor and device in the device database but did not use the system’s default startup code since we were provided with a custom startup file. A created a file and named it logic.s and added this file to the project along with the startup file given to us through this link “http://goo.gl/GXsMOQ”.

Part 1

The goal of this part of the lab was to set the text and back ground color then displays some lines of data containing my name and student ID and then configure the ADC convertor to turn on and off the lights on the microcontroller in sequential repeating order. The following is my counter.c code the section highlighted is the only sections I changed  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Counter.C: LED Flasher

CSE/EE 5385/7385 Microprocessor Architecture and Interfacing

Configure the LCD to SMU colors (Background: blue, Text color: red)

Display your first name on line 0 of the LCD

Display your last name on line 1 of the LCD

Display your student ID on line 2 of the LCD

Flash LEDs PE8, PE9 … PE15 sequentially

Display the counter value on line 3 of the LCD.

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h> /\* standard I/O .h-file \*/

#include <ctype.h> /\* character functions \*/

#include <string.h> /\* string and memory functions \*/

#include <RTL.h> /\* RTX kernel functions & defines \*/

#include <stm32f10x\_cl.h>

#include "GLCD.h"

#define LED\_NUM 8 /\* Number of user LEDs \*/

/\*Converts an integer to a char\*/

char numToChar(int n){

char value;

switch (n){

case 0: value = '0'; break;

case 1: value = '1'; break;

case 2: value = '2'; break;

case 3: value = '3'; break;

case 4: value = '4'; break;

case 5: value = '5'; break;

case 6: value = '6'; break;

case 7: value = '7'; break;

case 8: value = '8'; break;

default: value = '0'; break;

}

return value;

}

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

//moved here to make it a global variable, it was causing a issue for some reason

int main (void) {

int i;

//index=0;

int c;

int z;

SystemInit();

GLCD\_Init(); /\* Initialize the GLCD \*/

GLCD\_Clear(White); /\* Clear the GLCD \*/

/\*\*\*\*\*\*\*\*\* YOUR CODE GOES HERE \*\*\*\*\*\*\*\*\*\*/

GLCD\_SetTextColor (Red);

GLCD\_SetBackColor (Blue); //found using the functions described in the GLCD files

/\*\*\*\*\*\*\*\*\*\* screen set \*\*\*\*\*\*\*\*\*\*/

GLCD\_DisplayString ( 0, 8, 1,"Victor" );

GLCD\_DisplayString ( 1, 8, 1, "Yuan");

GLCD\_DisplayString ( 2, 8, 1, "46642210"); //enter the data required by the lab the first number of each function is the line number the other two numbers sets the formating

/\*\*\*\*\*\*\*\*\*\* ENABLE LEDS \*\*\*\*\*\*\*\*\*\*/

RCC->APB2ENR |= 1<<6; //enables clocks for LEDs thru port e

GPIOE->CRH = 0x33333333; //configures the pins, GPIO's as output

RCC->APB2ENR |= 1<<9; //Enables the clock for ADC-analog digital convertor

GPIOC->CRL &= 0xFFF0FFFF; //setup and initialize the ADC

ADC1->SQR1 = 0x00000000; /

ADC1->SQR2 = 0x00000000;

ADC1->SQR3 = (14<<0);

ADC1->SMPR1 = 6 << 12;

ADC1->SMPR2 = 0x00000000; // these configuration followed the example in the lecture

// I made an endless loop to cycle through all the lights and reset till I turn off the program

for(z = 0; z<2;z++){

for(c = 8; c>-2; c--) {

GPIOE->ODR |= led\_mask[c]; /\*Turn LED num on\*/

for (i = 0; i < (1000000); i++); //delay can reduce it to speed up the cycle

GPIOE->ODR &= ~led\_mask[c]; // turns off the light

GLCD\_DisplayChar ( 3, 8, 1,numToChar(c));

}

z=0; //endless loop

}

/\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*/

}

Part2

In the previous part of the lab all I did was create a a static program that receives no user input. In this section I configured the joystick to manipulate a barograph on the Display.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* bargraph.c

CSE/EE 5385/7385 Microprocessor Architecture and Interfacing

Display an horizontal bar graph of width 200px and height 40px on the LCD

When moving the joystick to the right, the bar graph width should increase to the right

When moving the joystick to the left, the bar graph width should decrease to the left

When pushing the joystick (select), display the current value of the bargraph

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h> /\* standard I/O .h-file \*/

#include <ctype.h> /\* character functions \*/

#include <string.h> /\* string and memory functions \*/

#include <RTL.h> /\* RTX kernel functions & defines \*/

#include <stm32f10x\_cl.h>

#include "GLCD.h"

signed int joystick=100; /\*Joystick offset\*/

int main (void) {

int i;

i = 0;

SystemInit();

GLCD\_Init(); /\* Initialize the GLCD \*/

GLCD\_Clear(White); /\* Clear the GLCD \*/

/\*\*\*\*\*\*\*\*\* YOUR CODE GOES HERE \*\*\*\*\*\*\*\*\*\*/

GLCD\_SetTextColor (Red); //It turns out these configuration works for the bar graph aswell

GLCD\_SetBackColor (Blue);

RCC->APB2ENR |= 1<<5; //5 for port D turns on clock

GPIOD->IDR =0x00000000; //configures for input

RCC->APB2ENR |= 1<<6; //Enables the clock for ADC-analog digital convertor

GPIOC->CRL &= 0xFFF0FFFF; //setup and initialize the ADC

ADC1->SQR1 = 0x00000000;

ADC1->SQR2 = 0x00000000;

ADC1->SQR3 = (14<<0);

ADC1->SMPR1 = 6 << 12;

ADC1->SMPR2 = 0x00000000; //set up of the ADC copied from part one and shown in the lecture

while (i==0){

char val[4];

//unsigned char \* val = &val1;

if(GPIOD->IDR &(1<<15)){ //shifted 15 for Port pin PD15 for left

joystick = joystick +10; //if left is selected a value of 10 was added to the bar graph

if(joystick > 1024) {joystick = 1024;} //if the value exceeded 1024 it would reset back to 1024 therefore limiting the bar to a enforced max

}

if(GPIOD->IDR &(1<<13)){ //shifted 13 for Port pin PD13 for right

joystick = joystick -10; //If a right is selected a value of 10 is subtracted from the bar graph

if(joystick < 0) {joystick = 0;} // same condition as before preventing the bar graph from going negative and messing up the user interface

}

sprintf(val,"%d", joystick);

if(~GPIOD->IDR &(1<<11)){GLCD\_DisplayString(3, 8, 1, val);} //shifted 11 for port pin PD11 for select

GLCD\_Bargraph (100,130, 200,50, joystick); //function from the header file

}

/\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*/

}

Analysis

The biggest issue I had with this lab was simply knowing which bits to shift and then reading and understanding headfiles and how to use functions given. There was also some unfamiliarity with using pointers in C and dereferencing.

Signatures:

