Name: TANG QI
Student I.D.: M-B5-5515-2
Academic year: 2015/2016 1st semester
Instructor: Prof. Vai Mang I

ELCE721 Project Logbook

Scheduler in Embedded IP Core 8051

Based on Altera DE2

for Real-Time Pressure Sensing

Date: 08/09/2015 – 24/11/2015

Date: Sep. 8th - Sep. 15th, first week

Objective:

Get familiar with the **softcore MC8051** and design **software Quartus II**. Learn the procedures of implementing IP core 8051 as well as arrange arrays on **Altera DE2 development board (FPGA)** via the design software Quartus II, via detailed instructions on Internet.

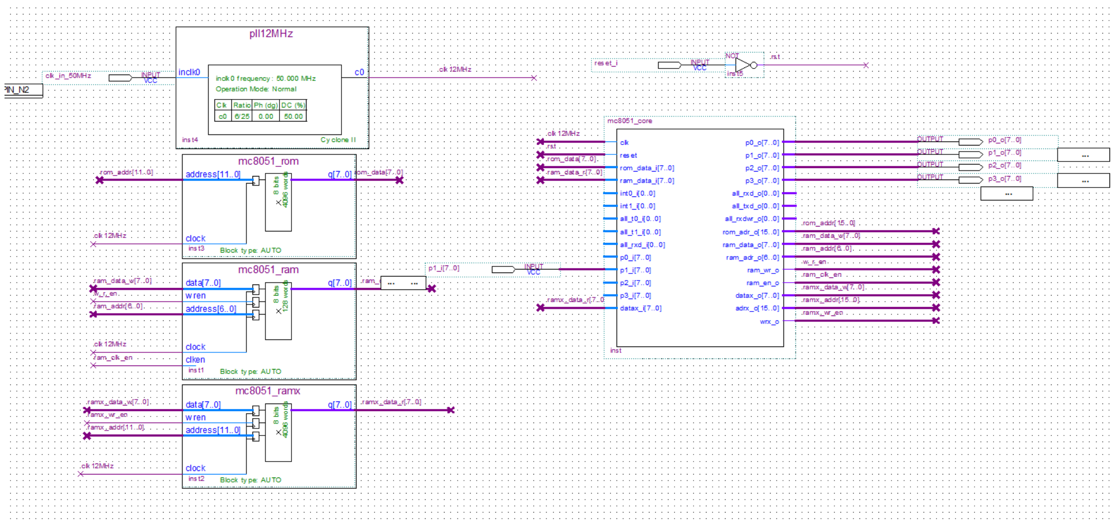
Task Name:

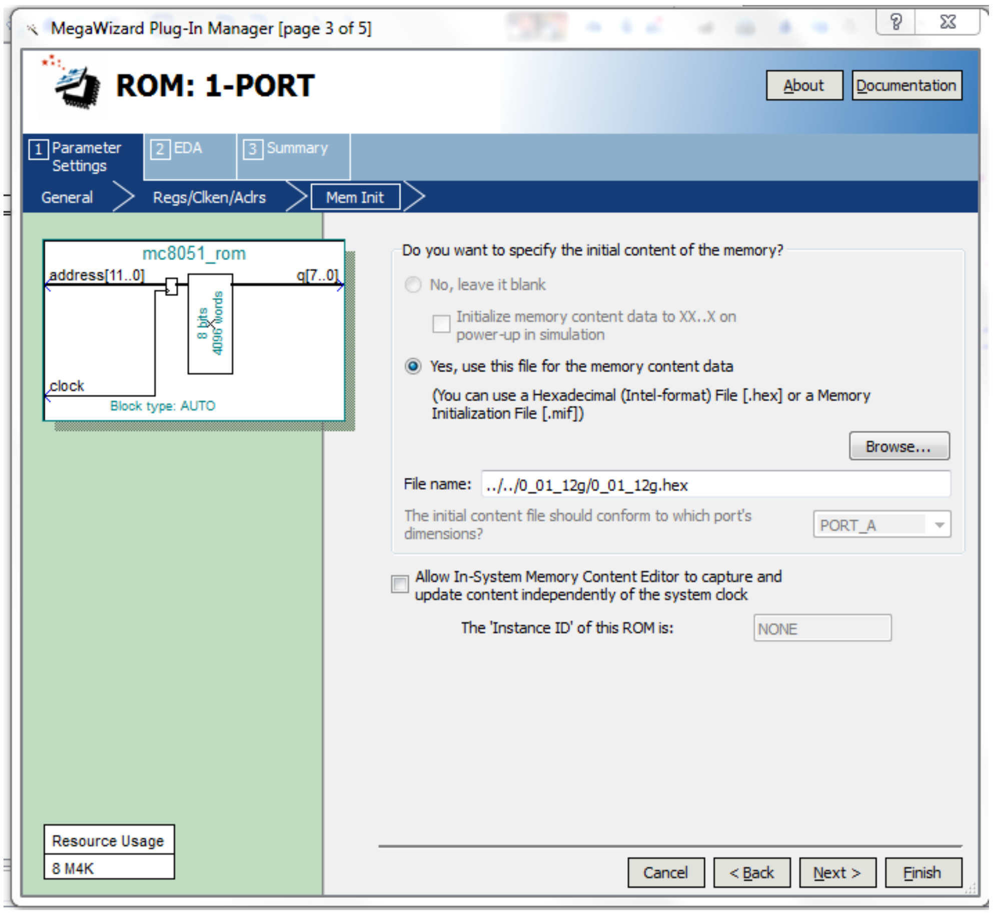
1. Knowledge on 8051 IP core and Altera DE2
2. Building 8051 into development board via the design software Quartus II.

Summary Description:

Finished both tasks. Knowing the characteristics of 8051 and FPGA device (Cyclone II EP2C35F672C6) on development board, the 8051 IP core is added and created via Quartus II. Mega- function wizard is used to create the RAM and ROM for the core, and inputs/ outputs are defined for future steps. Keil.hex file is used for the initial setting of ROM.

Flow Chart:




MegaWizard Plug-In Manager [page 3 of 5]

ROM: 1-PORT

Do you want to specify the initial content of the memory?

No, leave it blank

Initialize memory content data to XX..X on power-up in simulation

[✓ Yes, use this file for the memory content data]

(You can use a Hexadecimal (Intel-format) File [.hex] or a Memory Initialization File [.mif])

File name: ../../0_01_12g/0_01_12g.hex

The initial content file should conform to which port's dimensions? PORT_A

Allow In-System Memory Content Editor to capture and update content independently of the system clock

The 'Instance ID' of this ROM is: NONE

Findings:

1. A convenient way is taught to assign I/O pins to components on development board. Which is, name the I/O pins as the components' names, such as LEDG [7..0] (total 8 pins).
2. I also learnt that the naming of pins should be very careful, for example, P1_O [7…0], it does not work with just one more dot between 7 and 0.

Sep. 15th – Sep. 29th , second and third week

Objective:

Test the operation of 8051 core via simple functions to check the implement/schematic, and learn to do pin assignments based on different function output purposes.

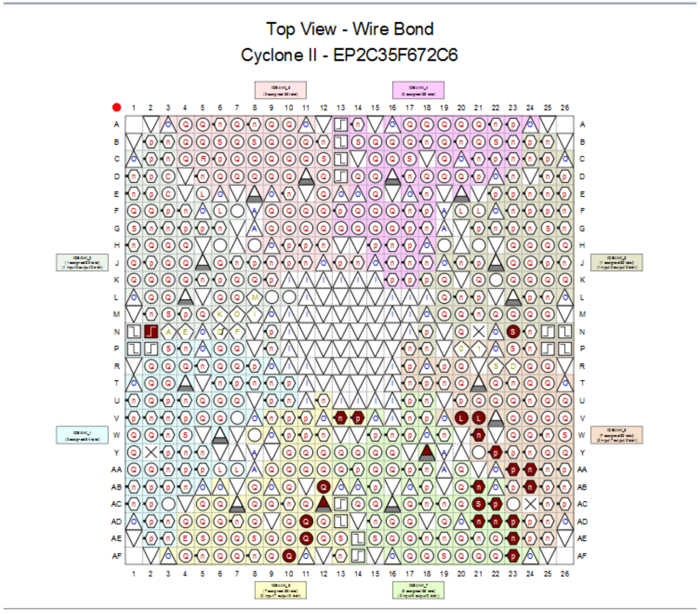
Task Name:

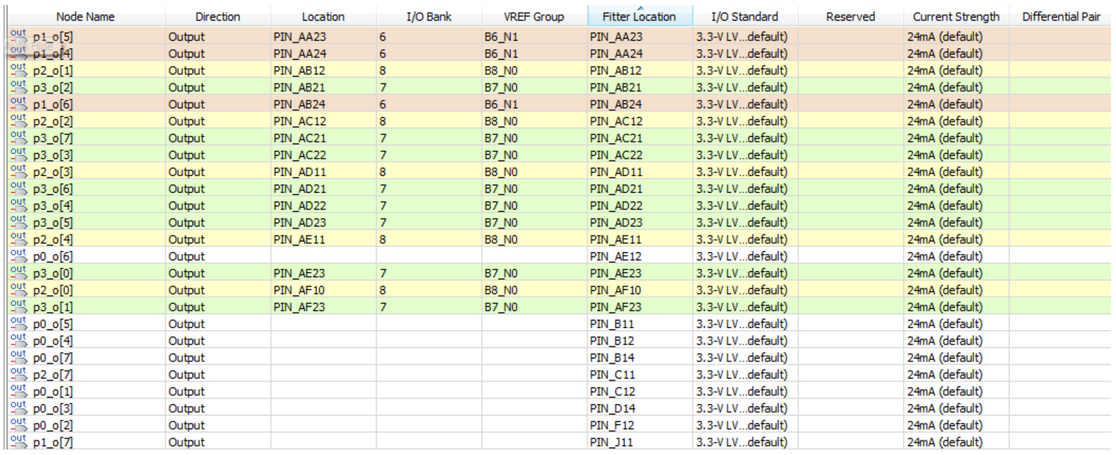
1. Pin assignment and running LED;
2. Get corresponding .hex file;
3. Start compilation and get **.sof** output file.

Summary Description:

Finished all three tasks. First is to know about the function that needed. Then write the corresponding codes. C language is comprehensive and of great convenience, thus Keil C is used. Corresponding codes are built and get the **.hex** file. Then the .hex file is added into the ROM of our IP core 8051. Start compilation and get the **.sof** output file for future programming.

Flow Chart:





Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Differential Pair
p1_o[5]	Output	PIN_AA23	6	B6_N1	PIN_AA23	3.3-V LV...default)		24mA (default)	
p1_o[4]	Output	PIN_AA24	6	B6_N1	PIN_AA24	3.3-V LV...default)		24mA (default)	
p2_o[1]	Output	PIN_AB12	8	B8_N0	PIN_AB12	3.3-V LV...default)		24mA (default)	
p3_o[2]	Output	PIN_AB21	7	B7_N0	PIN_AB21	3.3-V LV...default)		24mA (default)	
p1_o[6]	Output	PIN_AB24	6	B6_N1	PIN_AB24	3.3-V LV...default)		24mA (default)	
p2_o[2]	Output	PIN_AC12	8	B8_N0	PIN_AC12	3.3-V LV...default)		24mA (default)	
p3_o[7]	Output	PIN_AC21	7	B7_N0	PIN_AC21	3.3-V LV...default)		24mA (default)	
p3_o[3]	Output	PIN_AC22	7	B7_N0	PIN_AC22	3.3-V LV...default)		24mA (default)	
p2_o[3]	Output	PIN_AD11	8	B8_N0	PIN_AD11	3.3-V LV...default)		24mA (default)	
p3_o[6]	Output	PIN_AD21	7	B7_N0	PIN_AD21	3.3-V LV...default)		24mA (default)	
p3_o[4]	Output	PIN_AD22	7	B7_N0	PIN_AD22	3.3-V LV...default)		24mA (default)	
p3_o[5]	Output	PIN_AD23	7	B7_N0	PIN_AD23	3.3-V LV...default)		24mA (default)	
p2_o[4]	Output	PIN_AE11	8	B8_N0	PIN_AE11	3.3-V LV...default)		24mA (default)	
p0_o[6]	Output				PIN_AE12	3.3-V LV...default)		24mA (default)	
p3_o[0]	Output	PIN_AE23	7	B7_N0	PIN_AE23	3.3-V LV...default)		24mA (default)	
p2_o[0]	Output	PIN_AF10	8	B8_N0	PIN_AF10	3.3-V LV...default)		24mA (default)	
p3_o[1]	Output	PIN_AF23	7	B7_N0	PIN_AF23	3.3-V LV...default)		24mA (default)	
p0_o[5]	Output				PIN_B11	3.3-V LV...default)		24mA (default)	
p0_o[4]	Output				PIN_B12	3.3-V LV...default)		24mA (default)	
p0_o[7]	Output				PIN_B14	3.3-V LV...default)		24mA (default)	
p2_o[7]	Output				PIN_C11	3.3-V LV...default)		24mA (default)	
p0_o[1]	Output				PIN_C12	3.3-V LV...default)		24mA (default)	
p0_o[3]	Output				PIN_D14	3.3-V LV...default)		24mA (default)	
p0_o[2]	Output				PIN_F12	3.3-V LV...default)		24mA (default)	
p1_o[7]	Output				PIN_J11	3.3-V LV...default)		24mA (default)	

Findings:

1. Modules on development board is corresponds to its wire arrangements. While doing the pin assignment, we should consider the connection between the FPGA and pins. Here, we should obey the Altera DE2 pin table on http://www.terasic.com.tw/attachment/archive/30/DE2_Pin_Table.pdf

2. The codes written for running are as shown:

```
while(1)
     {
 temp=0x01;
      for(i=0;i<8;i++)
       {temp=~temp;
              b=temp;
       P1=b;
       delay(1000);
       temp<<=1;
       }
 temp=0x80;

 for(i=0;i<8;i++)
     {
     temp=~temp;
              b=temp;
```

```
            P1=b;
        delay(1000);
        temp>>=1;
        }
            temp=0xFE;
for(i=0;i<8;i++)
        {
        P1=temp;
        delay(1000);
        temp<<=1;
        }
        temp=0x7F;
        for(i=0;i<8;i++)
          {
        P1=temp;
        delay(1000);
        temp>>=1;
        }
}
```

Sep. 29nd –Oct. 6th , forth and fifth week

Objective:

Test the operation of 8051 core via simple functions to check the implement/ schematic. Learn to do pin assignments based on different function output purposes. Add hardware of usb-blaster and build **.sof** output file into programmer of our projects Quartus II.

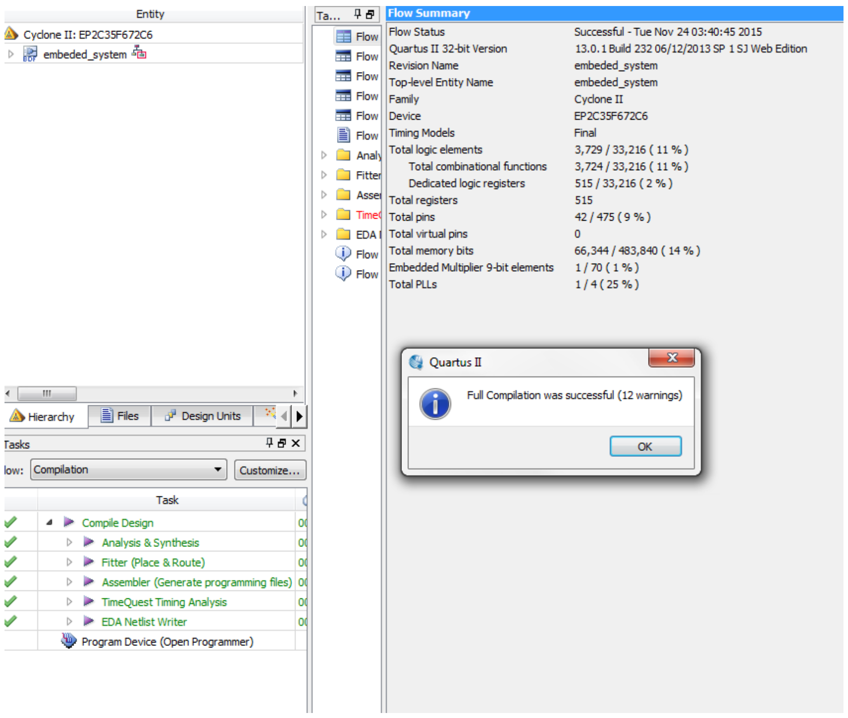
Task Name:

1. Add hardware of **usb-blaster**.
2. Build .**sof** output file into programmer of our projects Quartus II.
3. Finish the testing operation and make sure IP core be correctly built.

Summary Description:

Have trouble in adding hardware but finally solved with the help of professor and TA. All three tasks finished and now it is for sure that the previous works of implement, schematic, compilation and pin assignment were correct.

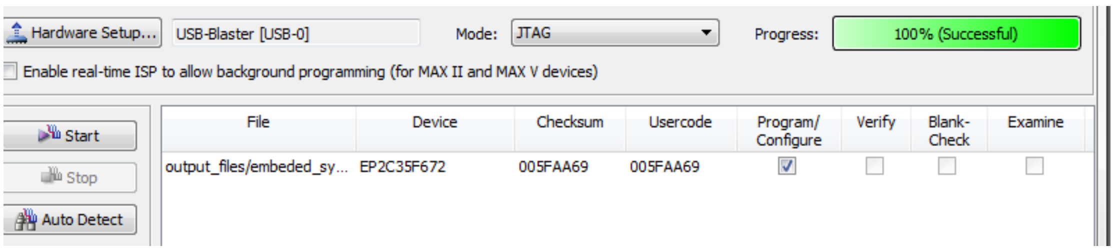
Flow Chart:



	Task
✓	Compile Design
✓	Analysis & Synthesis
✓	Fitter (Place & Route)
✓	Assembler (Generate programming files)
✓	TimeQuest Timing Analysis
✓	EDA Netlist Writer
	Program Device (Open Programmer)

Flow Summary	
Flow Status	Successful - Tue Nov 24 03:40:45 2015
Quartus II 32-bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	embeded_system
Top-level Entity Name	embeded_system
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	3,729 / 33,216 (11 %)
Total combinational functions	3,724 / 33,216 (11 %)
Dedicated logic registers	515 / 33,216 (2 %)
Total registers	515
Total pins	42 / 475 (9 %)
Total virtual pins	0
Total memory bits	66,344 / 483,840 (14 %)
Embedded Multiplier 9-bit elements	1 / 70 (1 %)
Total PLLs	1 / 4 (25 %)

Full Compilation was successful (12 warnings)


Hardware Setup... USB-Blaster [USB-0] Mode: JTAG Progress: 100% (Successful)

☐ Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
output_files/embeded_sy...	EP2C35F672	005FAA69	005FAA69	✓	☐	☐	☐

Problems and Difficulties:

1. The computer cannot notice there is an Altera DE2 hardware connected when I connected the data line to the second port (the right one) on board. After observation, I found the right port is not a blaster but a "device" port.

2. While the auto detection of usb by computer is ended, we add the hardware drive by hand via control panel.

Oct. 6th – Oct.20th , sixth and seventh week

Objective:

Build scheduler and design multi-tasks to test the feasibility.

Task Name:

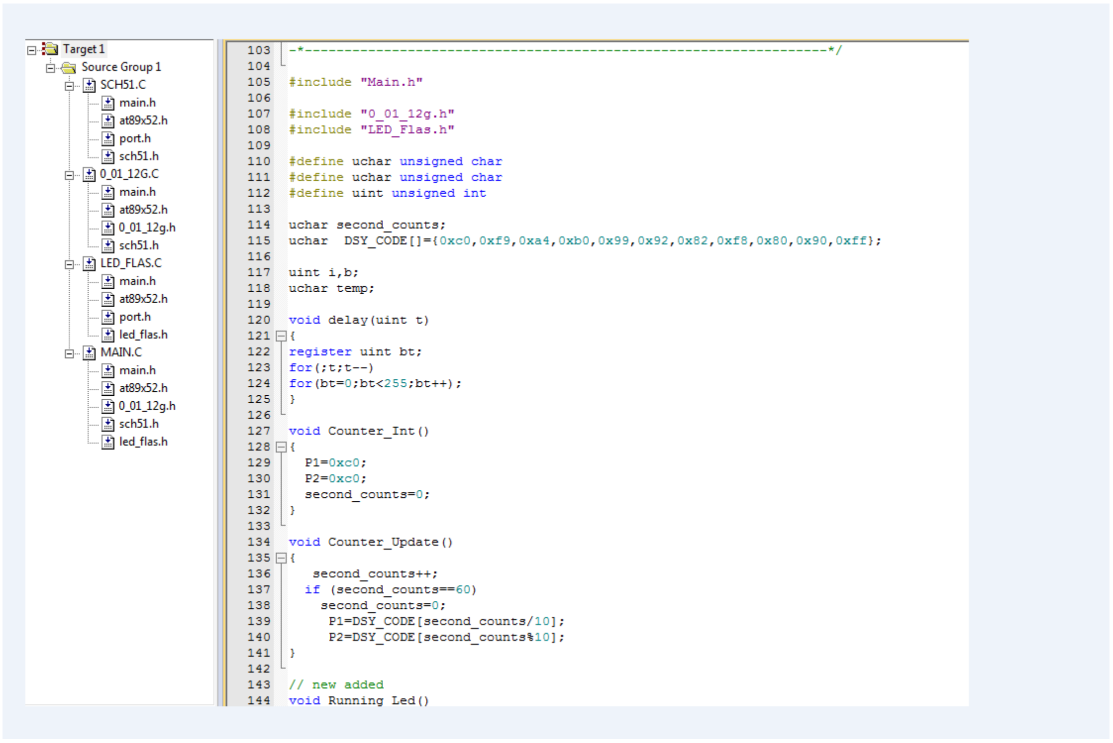
1. Build and adjust the scheduler structure.

2. Implement tasks in scheduler to check and do adjustment.

Summary Description:

Two tasks completed. **.c** and **.h** files of scheduler and three testing tasks are built into Keil C project. The three tasks are running LED ([LEDR7..0]), clock display (three 7-segments LED) and LEDG7 flashed in every one second. In scheduler, every task contains initial state declare and update state declare for scheduling.

Flow chart:



```
Target 1
  Source Group 1
    SCH51.C
      main.h
      at89x52.h
      port.h
      sch51.h
    0_01_12G.C
      main.h
      at89x52.h
      0_01_12g.h
      sch51.h
    LED_FLAS.C
      main.h
      at89x52.h
      port.h
      led_flas.h
    MAIN.C
      main.h
      at89x52.h
      0_01_12g.h
      sch51.h
      led_flas.h
```

```c
-*-------------------------------------------------------------------*/

#include "Main.h"

#include "0_01_12g.h"
#include "LED_Flas.h"

#define uchar unsigned char
#define uchar unsigned char
#define uint unsigned int

uchar second_counts;
uchar  DSY_CODE[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};

uint i,b;
uchar temp;

void delay(uint t)
{
register uint bt;
for(;t;t--)
for(bt=0;bt<255;bt++);
}

void Counter_Int()
{
  P1=0xc0;
  P2=0xc0;
  second_counts=0;
}

void Counter_Update()
{
   second_counts++;
  if (second_counts==60)
    second_counts=0;
     P1=DSY_CODE[second_counts/10];
     P2=DSY_CODE[second_counts%10];
}

// new added
void Running Led()
```

Findings:

1. Every time scheduler is used in a new **.c** file, the initial file should be firstly declared and head file should include corresponding functions.

2. There is no crash in tasks which means that tasks are individually scheduled based on the settled time of initial delay and periods.

Difficulties and Problems:

1. In theory, Keil uvision can auto-detect the head files # included in .c file, however, it is not the same way in practice. After rebuilding, I find Keil cannot recognize its head file under same foler. The solution is “new” a text and put head file contents in it and name the text” .h.”

2. I set the moving bit of running LED the same with the last digit of the counter (P3=2(second_counts%10)). However, there is a problem that the square function “^” in Matlab is not working in Keil because “^” operates as a address symbol here. Finally we apply the “pow” and add corresponding function in Head file.

Oct. 20th – Nov.3rd, eighth and ninth week

Objective:

Because the scheduler works, the drive program of pressure sensor should be implemented. I need to write a drive c. program in keil to achieve the weighing, display and alarming function.

Task Name:

Write functions of weighing, display and alarming for the pressure sensing transistor in Keil C and built into scheduler. In addition, download HX711.c to support the ADC module for the pressure sensing transistor.

Summary Description:

Use LCD to display and because there is no buzzer on DE2 development board, the alarm way is changed to the lighting of LEDG. There are oscillating when pressing Key[3..0], a delay function is needed to cancel the oscillation. The corresponding codes are shown:

```
if( KEY1 == 0 )
{
    Delay_ms(10);
    if( KEY1 == 0 )
    {
        alarm_light();
        while(KEY1 == 0);
        return 1;
    }
}
```

Problems:

Because the floating of weighing data, it is not very convenient to use scanning and update of 7 segment LED. Thus, according to the characteristics of 8051, the pin size limited (only 4*8 outputs) and if the weight is displayed in 7 segments LED, all 32 output pins will be used up.

Nov. 3rd – Nov.17th, tenth and eleventh week

Objective:

Struggling with the pressure sensing program, I decided to change to do a number system conversion decoder (from decimal to binary).

Summary Description:

Finished corresponding code, key part as shown:

```
void trans(unsigned char binaryhtd[10],unsigned int x)
        {
         int m=0,rem;
         do{
           rem=x%2;x=x/2;
           binaryhtd[m]=rem;
           m++;
           }while(x!=0);
         while(m>0)
         binaryhtd[--m];
        }

        void TenChar2TwoChar(unsigned char results[2], unsigned char inputs[10])
        {
         temp=0;
        for(i=0;i<8;i++){
             temp|=(inputs[i]<<i);
                                 }
        results[0]=temp;
        temp=0;

        for(i=0;i<2;i++){
             temp|=(inputs[i+8]<<i);
        }
        results[1]=temp;
        return;
        }

       void division(unsigned int n)
          {
        unsigned char      results[2]={0,0};
        unsigned char TenChar[10]={0,0,0,0,0,0,0,0,0,0};
        trans(TenChar,htd);
            TenChar2TwoChar(results,TenChar);
//                  P1=0;
//                  P2=0;
            P1=results[0];
            P2=results[1];

            while(1);
        }

        trans_intial()
        {
             P1=0;
            P2=0;
            H=0;T=0;D=0;
            adderH=0;adderT=0;adderD=0;
        }

        trans_update()
        {
            if (P1^1==1)
            {
                while(adderH <= 9 && P3^1==1)
                {adderH++;}
                H=adderH++;
            }
        if (P1^2==1)
        {
            while(adderT <=9 && P3_2==1)
            {adderT++;}
            T=adderT++;
        }

         if (P1^3==1)
        {
            while(adderD <=9 && P3_3==1)
            {adderD++;}
            D=adderD++;
        }

        htd=100*H+10*T+1*D;
        division(htd);
      }
```

Problems and difficulties:

Because I tries to convert three digit decimal number to binary, thus, the binary number could be 10 digits (2^10=1024). However, each pin address only contains 8 output pins. I was suggested to separate the number into 2 char, one contains 8 bits, the other contains 2 efficient bits and assign them to 2 output pins. Corresponding codes are as shown:

```
void TenChar2TwoChar(unsigned char results[2], unsigned char inputs[10])
        {
          temp=0;
        for(i=0;i<8;i++){
            temp|=(inputs[i]<<i);
                              }
            results[0]=temp;
            temp=0;

            for(i=0;i<2;i++){
                temp|=(inputs[i+8]<<i);
            }
            results[1]=temp;
            return;
            }

           void division(unsigned int n)
             {
            unsigned char     results[2]={0,0};
            unsigned char TenChar[10]={0,0,0,0,0,0,0,0,0,0};
            trans(TenChar,htd);
                TenChar2TwoChar(results,TenChar);
//                      P1=0;
//                      P2=0;
                P1=results[0];
                P2=results[1];

                while(1);
            }
```

Nov. 18th – Nov.24th, the final week

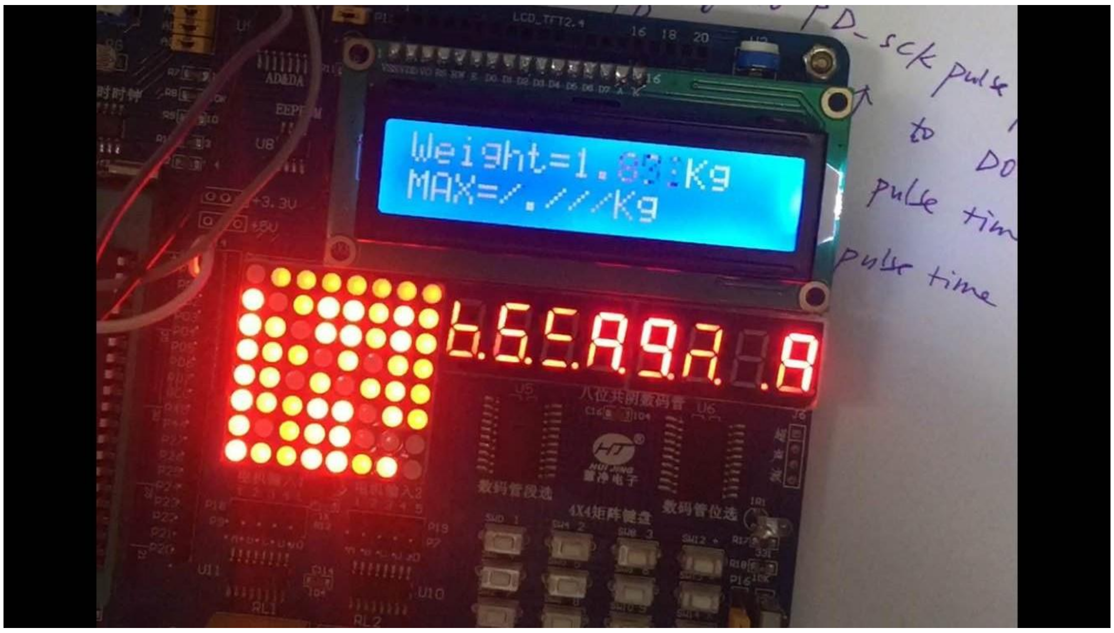
Objective:

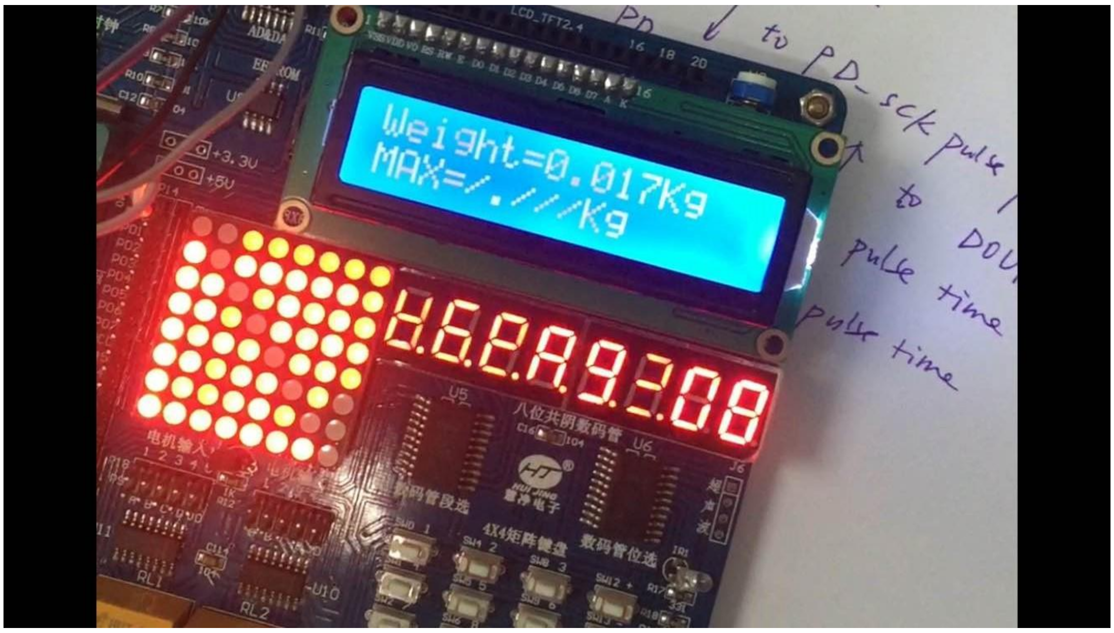
Because previous work of scheduler is of no real – life application. I decided to go back to implement the pressure sensor transistor again. Thus, every time the scheduler updates, the weight on the transistor will also be updated, creating an application of real- time pressure sensing by the timers in schedule program.

Summary Description:

I need to implement the controlling program and functions of several related modules, like the ADC converter HX711, and the LCD display. Also function of get_weight( ) should be written into the scheduler in main.c. After building the programs in Keil C, I make sure there are no errors and no warnings because the existing of warnings could be dangerous in later compilation. After adding the .hex into hardware, I found there is no display on the LCD of DE2 board. To debug, I decided to build the .hex file to a 8051 development board. The whole program works on 8051 normally. Then the next step is to adjust the LCD file of program and implement into DE2.

Flow chart:





Problems and Difficulties

The pin assignments are checked many times and the problem is supposed to be the timing function of the LCD display. Because even there is no input data, the LCD were supposed to display “welcome to use!” However, on the DE2 LCD board, I see no respond. I decided to delay to next week to finish the program after reading related LCD codes and fully understand them.