LIST OF FIGURES

1. INTRODUCTION

1.1 Overview

- Apply the microprocessor into daily life, this clock can be used in many aspects which require high accuracy.
- Having basic knowledge about timer, interrupt, and I2C communications between IC.

1.2 Research's background

- Base on TI microprocessor open resource tutorial about MSP430G family.
- Nowadays most digital clock project based on Arduino with the given library, students cannot understand deeply about basic embedded system, clock frequency and interrupts.

1.3 Research purposes

- Make a digital clock can display real time, having 3 functions: alarm, timer, stopwatch.
- Essential task:

  Stage 1: Understand about IO pin of microprocessor.

  - Using data sheet, and online open source tutorial.
  - Using MSP430x2xx Family User Guide [1]

Table 8-2. Digital I/O Registers

Port	Register	Short Form	Address	Register Type	Initial State
P1	Input	P1IN	020h	Read only	-
	Output	P1OUT	021h	Read/write	Unchanged
	Direction	P1DIR	022h	Read/write	Reset with PUC
	Interrupt Flag	P1IFG	023h	Read/write	Reset with PUC
	Interrupt Edge Select	P1IES	024h	Read/write	Unchanged
	Interrupt Enable	P1IE	025h	Read/write	Reset with PUC
	Port Select	P1SEL	026h	Read/write	Reset with PUC
	Port Select 2	P1SEL2	041h	Read/write	Reset with PUC
	Resistor Enable	P1REN	027h	Read/write	Reset with PUC
P2	Input	P2IN	028h	Read only	-
	Output	P2OUT	029h	Read/write	Unchanged
	Direction	P2DIR	02Ah	Read/write	Reset with PUC
	Interrupt Flag	P2IFG	02Bh	Read/write	Reset with PUC
	Interrupt Edge Select	P2IES	02Ch	Read/write	Unchanged
	Interrupt Enable	P2IE	02Dh	Read/write	Reset with PUC
	Port Select	P2SEL	02Eh	Read/write	0C0h with PUC
	Port Select 2	P2SEL2	042h	Read/write	Reset with PUC
	Resistor Enable	P2REN	02Fh	Read/write	Reset with PUC

Figure 1: Digital I/O Registers

Stage 2: Study about Interrupt and Timer on Microprocessor

- Using Continuos Mode, Pull Interrupt every 50 cycles for global period every 50ms with clock 8MHz.

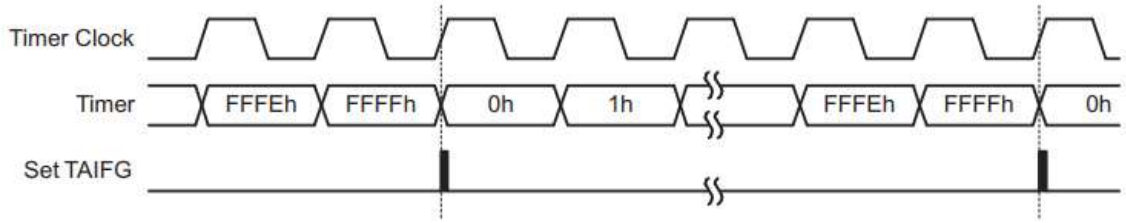

Figure 12-5. Continuous Mode Flag Setting

Figure 3: Continuous Mode Flag Setting

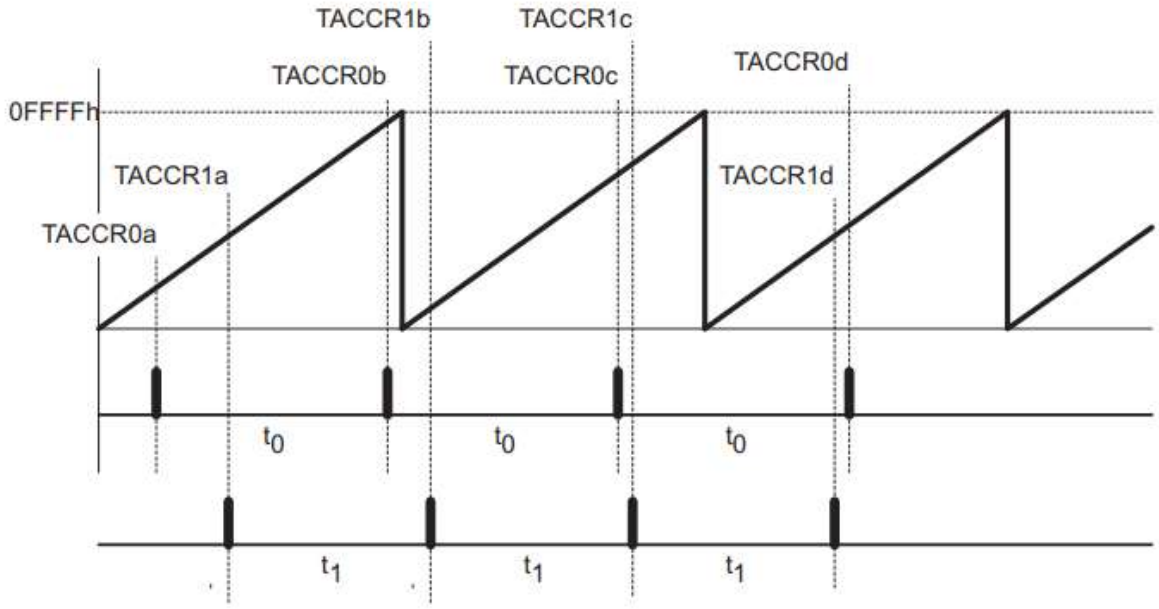

Figure 2: Continuous Mode Time Intervals

Stage 3: Communication and display data between kit and LCD module.

- Using IO port to communicate with LCD base on its Pin functions and instruction code.
- Build a send command function to communicate with LCD and other support function like: PrintString, PrintNumber, …to LCD which base on those command functions.

Stage 4: I2C communication interface between MSP430 and Real Time module DS1307.

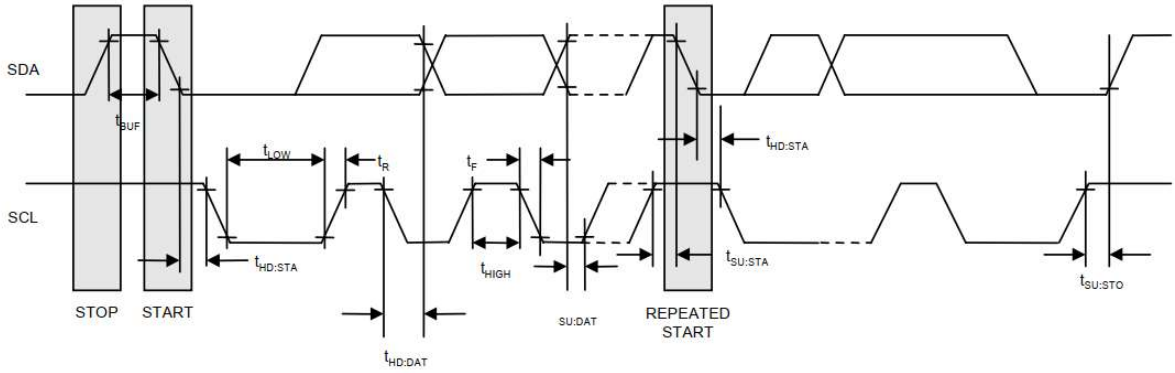

Figure 4: Timing Diagram of DS1307 I2C connection.

- Study about I2C interface between ICs.
- Build function to read and write from MSP to DS1307 using I2C connection.

Stage 5: Make schematic of the clock.

- Searching for Altium tutorial.
- Using basic knowledge about electronic circuit to design digital clock base on design requirement.

Stage 6: Make PCB.

- Learn how to use Altium via YouTube and other open sources.

2. THEORIES

Chapter 1: Clock and Frequency of Microprocessor.

The basic clock Module+ supports low system cost and ultralow power consumption. Using three internal clock signals, the user can select the best balance of performance and low power consumption. The basic clock module+ includes two, three or four clock sources:

- LFXT1CLK: Low-frequency/high-frequency oscillator that can be used with low-frequency watch crystals or external clock sources of 32768 Hz or with standard crystals, resonators, or external clock sources in the 400-kHz to 16-MHz range.
- XT2CLK: Optional high-frequency oscillator that can be used with standard crystals, resonators, or external clock sources in the 400-kHz to 16-MHz range.
- DCOCLK: Internal digitally controlled oscillator (DCO).
- VLOCLK: Internal very low power, low frequency oscillator with 12-kHz typical frequency.

Chapter 2: LCD module

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits.

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD

Chapter 3: Using I2C connection to store read and write day and time to Real Time Clock Module.

ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	FUNCTION	RANGE
00H	CH	10 Seconds			Seconds				Seconds	00–59
01H	0	10 Minutes			Minutes				Minutes	00–59
02H	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM
		24	PM/AM							00–23
03H	0	0	0	0	0	DAY			Day	01–07
04H	0	0	10 Date		Date				Date	01–31
05H	0	0	0	10 Month	Month				Month	01–12
06H	10 Year				Year				Year	00–99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08H-3FH									RAM 56 x 8	00H–FFH

0 = Always reads back as 0.

Figure 5: Timekeeper Registers on DS1307.

Chapter 4: Using pull - down button to change between state of the system.

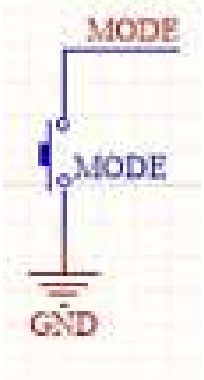

Figure 6: Pulled – down button

Pulled-down button: when the button is inactive, the data wire is connected to ground (low signal). When the button is active the data wire is connected to source (high signal).

3. DESIGN AND IMPLEMENTATION OF HARDWARE

- Design requirement:
    - Small size, acceptable price.
    - Detail:
        - 1 LCD 16x02.
        - 5 button to switching between mode in clock.
        - 1 MSP4302553.
        - 1 Module IC DS1307.
        - 1 Jack DC 5V Power Supply.
        - 2 Power LED.
        - 1 LM1117 3v3.
- Analyses:
    - Using LM1117 to convert 5vDC from jack to 3v3 which is used by the MSP430.
    - Using high frequency clock in MSP430 to control display with LCD
    - Using I2C to communicate with Real Time Module to assure the correctness of the clock.
- Block Diagram:

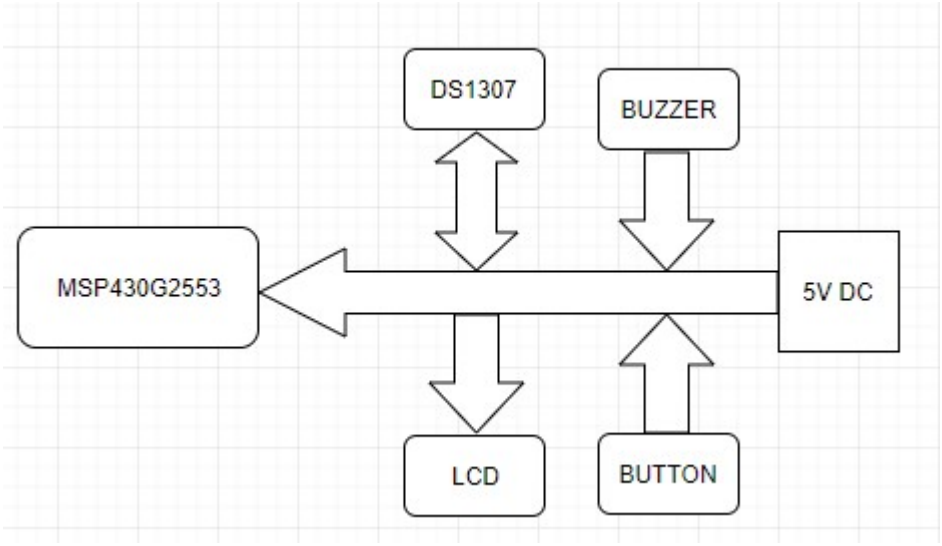

Figure 7: Block diagram of hardware

- Detail hardware:

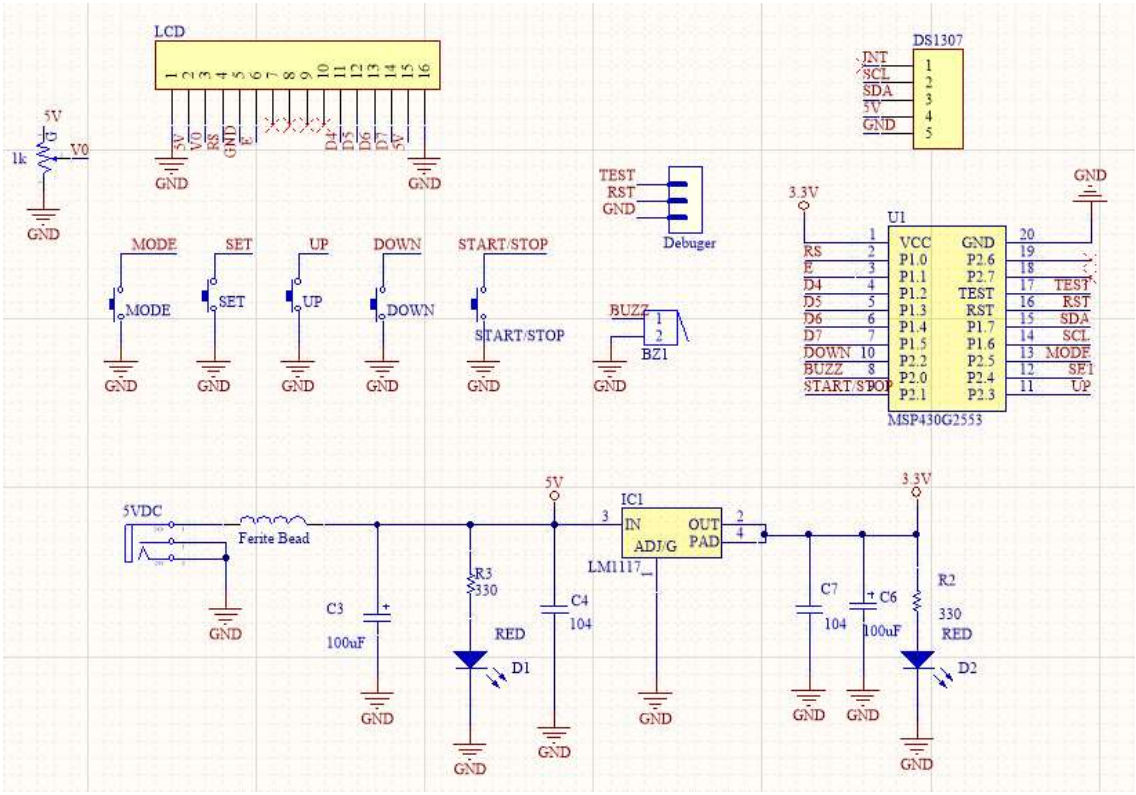

Figure 8: Schematic

4. DESIGN AND IMPLEMENTATION OF SOFTWARE

- Software requirement:
  - Can display day and time on LCD.
  - Read day and time from DS1307 Real Time Module.
  - Change day and time by using MODE, SET, UP, DOWN buttons.
  - Can change between 4 modes immediately when pressing MODE button.
  - When the clock in alarm mode, it can be turn off by pressing UP, DOWN buttons at the same time.
  - Successfully run 4 modes: Alarm, Stop Watch, Timer, display time.
- Analyses:
  - Using interrupt to create global period for CPU.
  - Using I2C connection method to communicate between MSP430 and DS1307.
- Software Diagram:

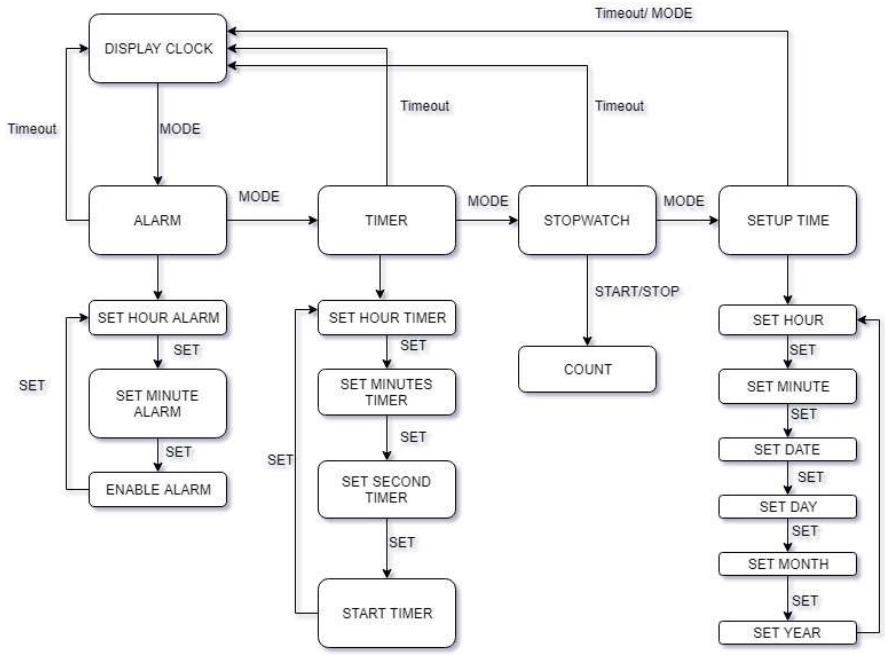

Figure 9: Software State Diagram

- Explanation:
  - Display Clock: in this state, LCD displaying date, time and waiting for user to press button MODE to switch to ALARM mode.
  - ALARM: when first enter this state, the system is in the SET_HOUR_ALARM mode, user can press UP/DOWN to adjust the needed hour. Press SET to accept and move to SET_MINUTE_ALARM, SET to move to ENABLE_ALARM and SET again to return SET_HOUR_ALARM. Press MODE to move to TIMER mode.
  - TIMER: Similar to SET_HOUR_ALARM, we press UP/DOWN to set needed time, SET to change between hour/minute/second and SET again to go in to waiting to start, and START to begin counting down.
  - STOPWATCH: Press START to begin counting up, START again to Pause, and SET to reset back to 0. MODE to move to SETUP_TIME.
  - SETUP_TIME: Press UP/DOWN to adjust, SET to accept.
  - In every state, after 5 second not interacting, the system will automaticly come back to first state which is Displaying Clock.

5. RESULTS AND DISCUSSION

- We have a digital Clock base on MSP430G2553 having 4 function: Display time, alarm, stopwatch, timer.
- The clock can show time accurate up to year 2100.

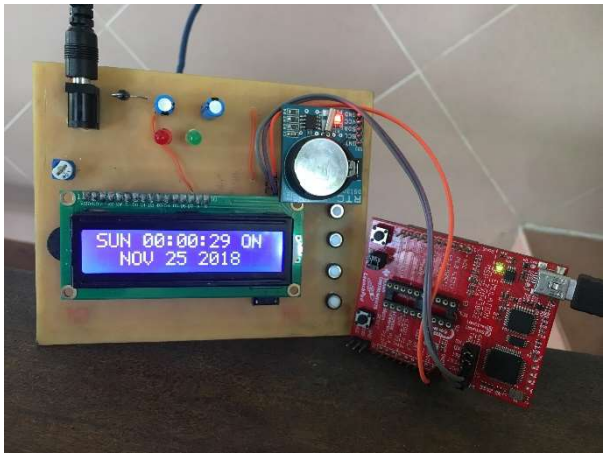

Figure 11: The Completed Board (Ver 2)

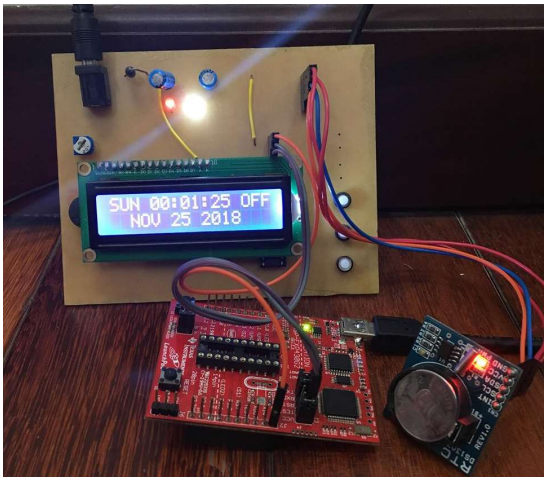

Figure 10: The Completed Board (Ver 1)

- Stimulation result:

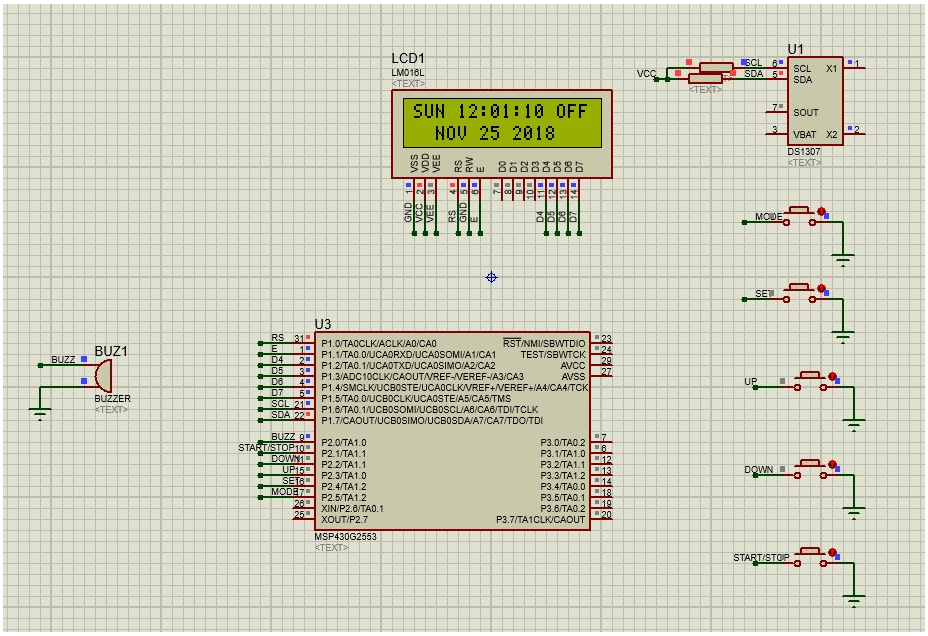

Figure 12: Software Stimulation Result.

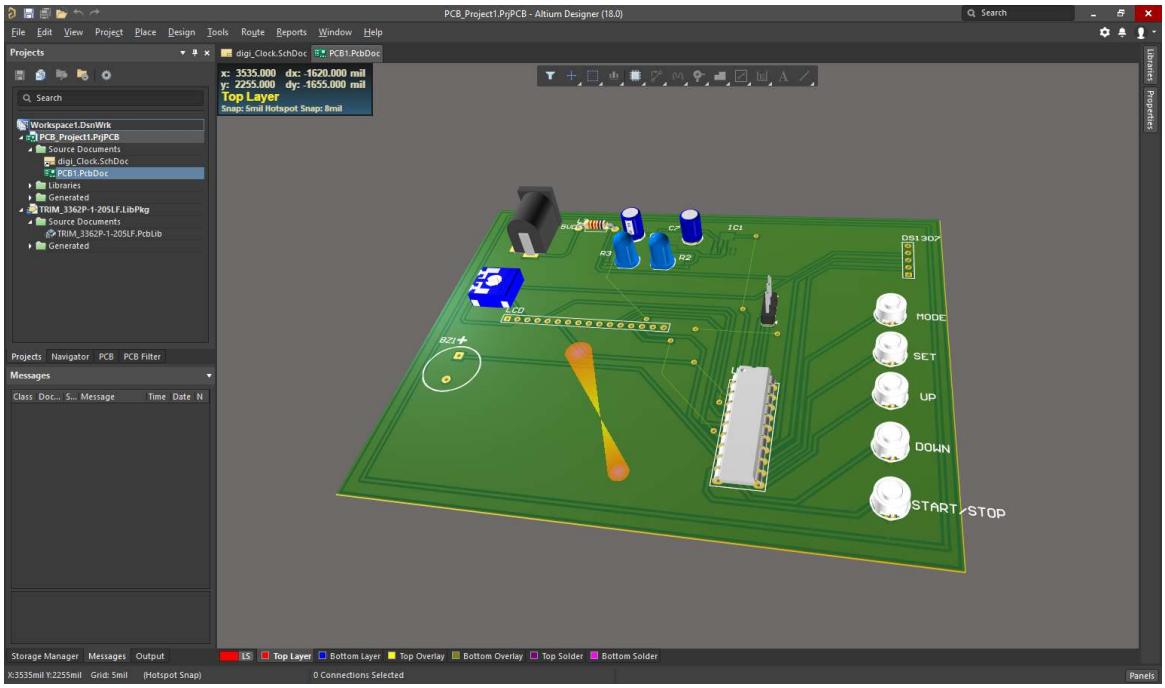

Figure 13: Hardware Stimulation result (Front).

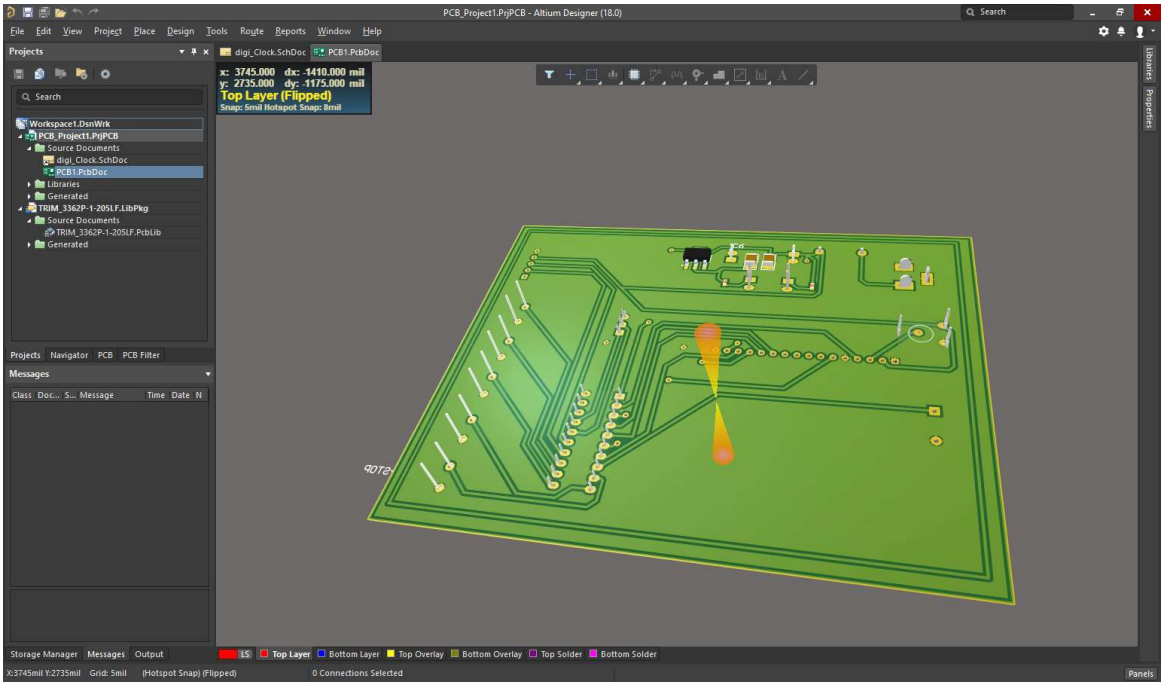

Figure 14: Hardware Stimulation result (Back).

6. CONCLUSION AND FUTURE WORKS

6.1 Conclusion

By this point, our team has completed a digital clock capable of real time display, alarm, timer and stopwatch. Thereby understand about IO pin of microprocessor, Communication and display data between kit and LCD module, I2C communication interface between MSP430 and Real Time module DS1307. However, we have some problems in printing board and product appearance is not attractive. In short, we accomplished the goal set out in part 1.

6.2 Future works

My team has the idea of developing a compact, fully functional product in this project but at a low price, low power consumption and eye-catching looks.

7. REFERENCES

[1] Texas Instruments for DataSheet
http://www.ti.com/product/MSP430G2553?keyMatch=msp430g2553&tisearch=Search-EN-Everything

[2] Pay it forward club in Bach Khoa University

www.payitforward.edu.vn

[3] Introduction to PIC Microprocessor

http://www.cc.hcmut.edu.vn/bai-viet-lap-trinh-ung-dung-vi-dieu-khien-101.aspx

8. APPENDIXS

- The bottom Layer of the board:

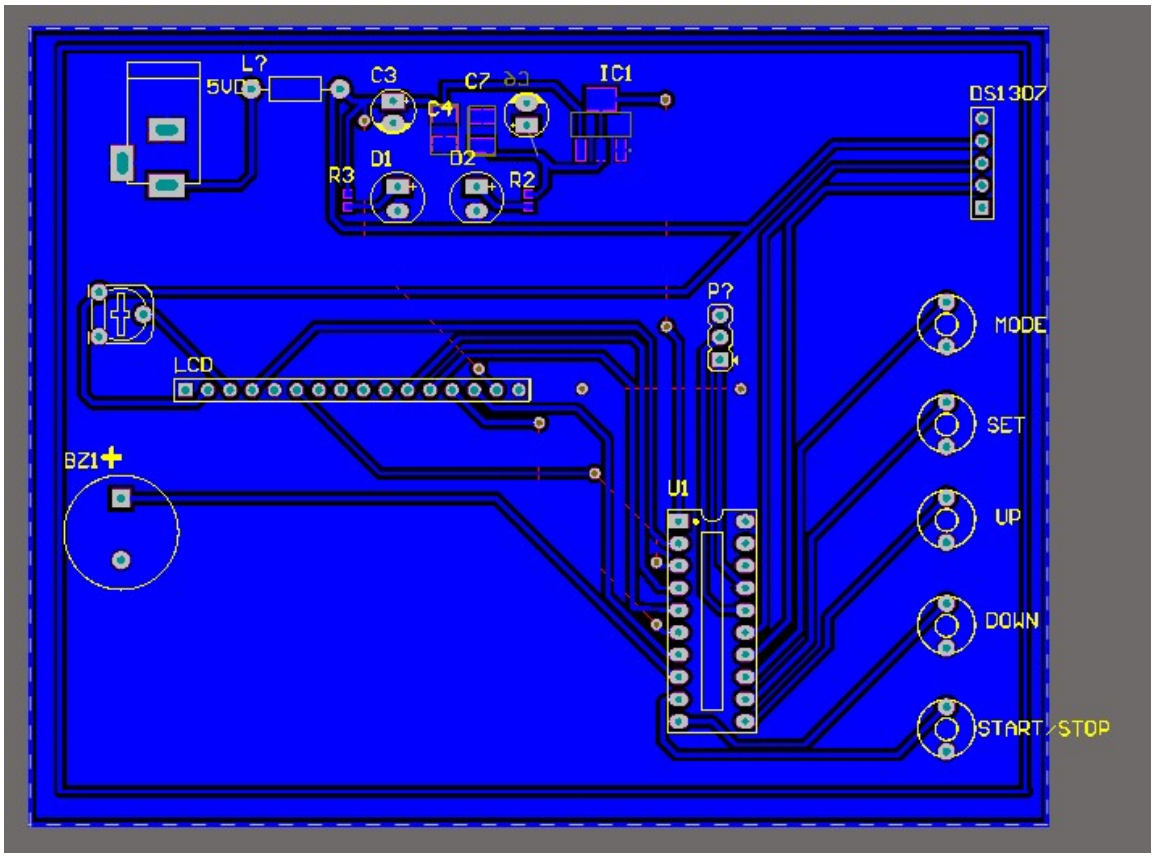

Figure 15: Bottom Layer of the Clock.

- The main function in software:

```
#include "main.h"

#define LED     P2OUT
#define BUTTON  P2IN
#define ON      1
#define OFF     2

unsigned char ledOutput[8] =
{0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
unsigned char ledStatus[8] = {0,0,0,0,0,0,0,0};
unsigned char flag_TimerA = 0;
unsigned char count = 0;
unsigned char datawrite[8],dataread[8];

void init_system(void);
void configLEDs(void);

void OpenOutput(int i);
void CloseOutput(int i);
void ReverseOuput(int i);
```

```
void rtc_settime(int sec, int min, int hour, int day, int date, int
month, int year);
void read_ds1307(void);
unsigned char read_data(unsigned char addr);

unsigned char isButton(unsigned char i);

#define INIT_SYSTEM         0

#define SET_HOUR_ALARM      1
#define SET_MINUTE_ALARM    2
#define SET_ENABLE_ALARM    3

#define SET_HOUR_TIMER      4
#define SET_MINUTE_TIMER    5
#define SET_SECOND_TIMER    6
#define START_TIMER         7

#define START_STOPWATCH     8

#define SET_HOUR            9
#define SET_MINUTE          10
#define SET_DAY             11
#define SET_DATE            12
#define SET_MONTH           13
#define SET_YEAR            14

unsigned char isButtonMode();
unsigned char isButtonSet();
unsigned char isButtonUp();
unsigned char isButtonDown();
unsigned char isButtonTurnOffAlarm();
unsigned char isButtonResetSecond();
unsigned char isButtonStart();

unsigned char timeBlink = 0;
unsigned char statusSetupTime = INIT_SYSTEM;

void DisplayTime(void);
void ResetSecond(void);
unsigned char timeOut = 0;

void DisplayAlarm();
void SetHourAlarm();
void SetMinuteAlarm();
void SetEnableAlarm();
void Alarm();
char enableAlarm = 0, hourAlarm = 2, minAlarm = 0, flagAlarm = 0;

void DisplayTimer();
void SetHourTimer();
void SetMinuteTimer();
void SetSecondTimer();
void Timer();
void TimeOutTimer();
char hourTimer = 0, minTimer = 0, secTimer = 0, enableTimer = 0;

void DisplayStopWatch();
void StopWatch();
void ResetStopWatch();
```

```
        char hourSW = 0, minSW = 0, secSW = 0, msecSW = 0, enableSW = 0;

        void SetupTime();
        void SetHour();
        void SetMinute();
        void SetDay();
        void SetDate();
        void SetMonth();
        void SetYear();
        char sec = 0,min = 0,hour = 0;
        unsigned char day = 1,date = 25,month = 11,year = 18;
//=============================================================================
```