NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

DIVISION OF ELECTRONICS AND COMMUNICATION ENGINEERING

EC316 MICROPROCESSOR LAB PROJECT REPORT

Mathematical Alarm Clock

AN INTEL 8085 MICROPROCESSOR BASED PROJECT

Submitted by
Prabhat Gupta (124EC15)
Srijan Pabbi (173EC15)

Submitted to
Prof. Dhananjay V. Gadre

May, 2018

Contents

1 Abstract

Everyone must have seen the ordinary alarm clocks that sound only when it is set to be alarmed.Why not to make a normal alarm period an interesting task. It's time to get rid of the conventional alarm clocks. Here what comes the mathematical alarm clock. The mathematical alarm clock is a device that modifies a conventional alarm clock to a more advanced version that not only alarms but it also makes sure that the user gets out of sleep for solving some mathematical problems.This device aims to make turning on an alarm a slightly more involving task. It keeps the trackof time and displays it on the lcd. It does so by using an RTC (parallel data bus). The Alarm can be set using a keypad. At the time of alarm, a buzzer sounds. The buzzing of the alarm continues as long as the user would not give the correct answers of certain number of mathematical problems. At the alarm preset time, the device throws a randomly generated integer based mathematical problem based on simple arithmetic (+). The alarm stops only after the correct answer is given by the user. The project incorporates a display (LCD 20x4),keypad, buzzer as an alarming system, parallel bus RTC

2 Background And Motivation

EC-311/316 has provided us an opportunity to build this project under the able guidance of Prof Dhananjay Gadre. The idea of this project is to learn and portray the skill of designing a microprocessor based system and putting whatever we learn in this course into a useful application. This device aims to make turning on an alarm a slightly more involving task, forcing the user to focus on the device for some time hence successfully breaking the sleep as compared to normal alarm clocks. Making this project will involve almost all the steps required for making a embedded system product namely; planning, designing, getting the pcb fabricated, coding, and a making an enclosure for the final display, and most importantly documenting throughout the process of making it, which will give us experience on how to design a embedded product from zero to finish.

3 Justification

Mathematical Alarm Clock involves the use of a RTC(Real Time Clock) for time keeping, 20x4 character LCD for display of time and the generated math questions, and a user input keypad, these devices cannot be used with simple digital logic devices to perform the job we want out of this project. This project must be able to display time and, perform background mathematical calculations , accept and compare user inputs to the answers of the questions which it generated randomly. The need to do all this for successful working of the project the use of a microprocessor which is fairly fast and can perform slightly complex mathematical calculations is justified.

4 Block Diagram

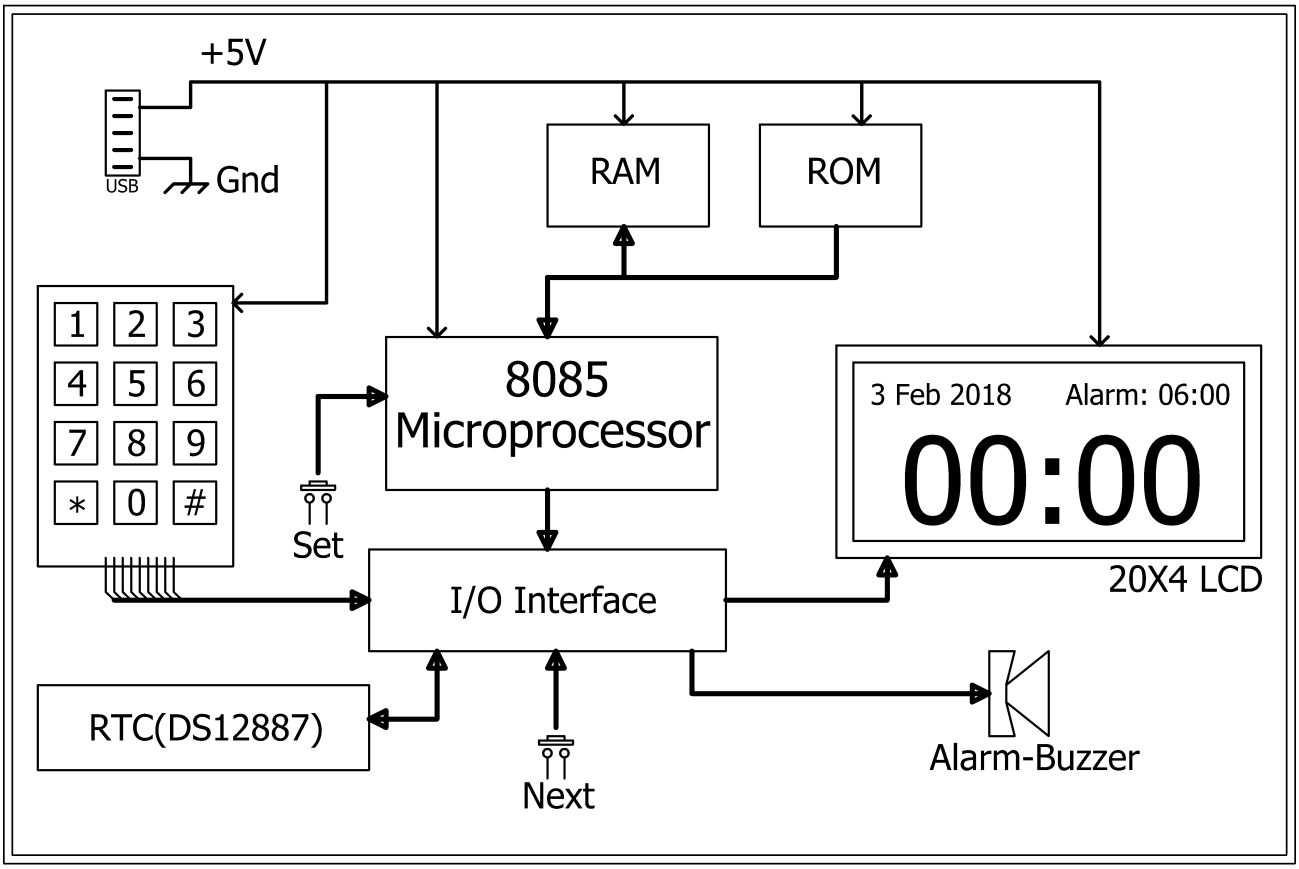


Figure 1: Block Diagram of Mathematical Alarm Clock

5 Technical Details

""

5.1 Hardware

5.1.1 Core Microprocessor (Intel 80C85)

At the heart of this project is an Intel 80C85 microprocessor(μP) from OKI Semiconductors is a CMOS implementation of the original TTL 8085 which has a quiescent current of as low as 10mA. The microprocessor has a 64KB memory address space and 256 bytes of I/O space. In this case the 8085 is running at 3MHz from the clock of a 6MHz crystal oscillator.

Figure 2: OKI 8085

5.1.2 Peripherals And Addressing

This project employs a Atmel ROM 28C256, Intel 8155 PPI+RAM+Timer, Maxim DS12887 RTC, as peripherals for the microprocessor and hence they occupy some part of the 8085's address space.

- *Atmel ROM 28C256*: Is a 32KB rom which is over kill for this project but was chosen due to its ready availability and most of the address space of the 8085 microprocessor is not in use.
  -The address space for the ROM ranges from 0000h to 0$FFFh$.
  -Chip Select($\overline{CS}$) for this IC is connected to A_{15} of the μP
- *Intel 8155*: Is a IC specially developed to be used with the Intel 8085 μP. It has 256bytes of RAM, 3 I/O ports 2 of 8bit and 1 of 6bit, with handshake capability using the former as I/O and the latter for handshaking, and a 8 bit timer with clock out.
  -The address space for the RAM ranges from 8000h to 80FFh.
  -The IO addresses for 8155CWR-00h, PORTA-01h, PORTB-02h, PORTC-03h
  -Chip Select($\overline{CS}$) for this IC is connected to A_{15} of the μP
- *RTC*: This IC is a complete package of an RTC, battery, and crystal, along with a non-volatile RAM in the RTC. This chip implements the Intel Bus Protocol Internally and hence can be memory mapped.
  -The address space for the RTC ranges from 8100h to 8180h.
  -Chip Select($\overline{CS}$) for this IC is connected via a decoding logic as shown in Figure 7

5.1.3 RTC(Real Time Clock)

The real time clock employed for this project DS12887A from Maxim Integrated is a parallel bus RTC with data exchange via the Intel bus protocol implemented internally. This RTC comes in a 24 pin EDIP package with internally implemented crystal oscillator and a battery. It can count time on both binary as well as BCD format with one-time of the day alarm, and an added feature of daylight saving by keeping track of the exact day,date,year,and century of the millennium. Alongside the basic time keeping features, it can interrupt the processor about an alarm situation. It can also produce a square wave out of various frequencies which can be used to drive an alarm sounding system.
The following is the schematic implementation of the RTC.

5.1.4 20x4 Character LCD

The display used in our project is 20X4 LCD display. It is said so as it has 4 rows with 20 characters each. It uses HD44780 driver. The customized characters are generated using CGRAM. 7 special characters are stored at seven locations of CGRAM. Using these seven characters any of the character or digit can be produced. Its contrast and brightness are controlled using two independent presets.

5.1.5 Key-Pad

The keypad used is 4X3 that means it has 4 rows and 3 columns. The switches that are used in the keypad matrix is 40XX tactile switches. It is interfaced using polling technique. In the polling technique, each switch press is detected by polling individual switches. In this, rows are treated as input and columns as output. A switch is identified by outputting through columns and inputting through rows.

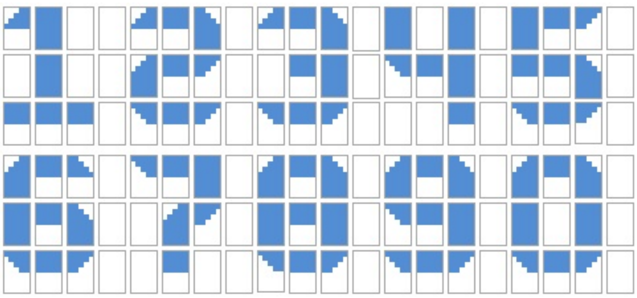
5.1.6 Buzzer

The buzzer used here is the D.C. buzzer. It is derived through a port pin of 8155 using a transistor as a switch.

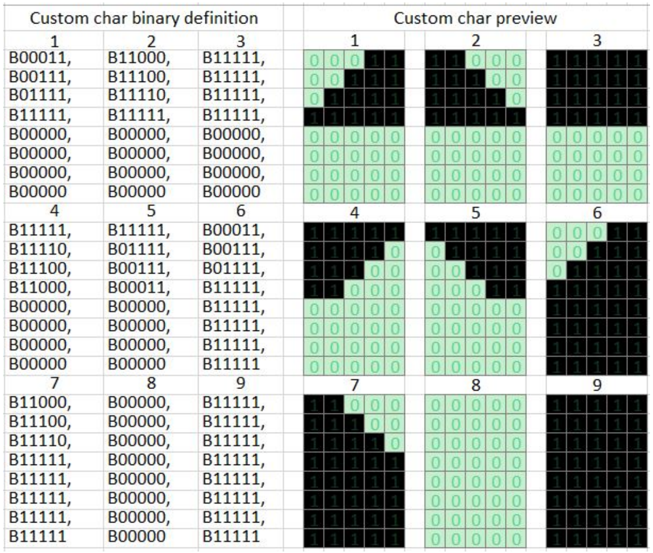
5.2 Software Implementation and Interesting Concepts

5.2.1 Big Font and LCD Posting Techniques

Big Font or customized characters are generated using the CGRAM of the lcd. For generating these characters, some special patterns(7) are stored in the CGRAM. Using these seven patterns, by appropriate positioning, any big character can be generated.



(a) Decimal Digits In Bigfont



Custom char binary definition		
1	2	3
B00011,	B11000,	B11111,
B00111,	B11100,	B11111,
B01111,	B11110,	B11111,
B11111,	B11111,	B11111,
B00000,	B00000,	B00000,
B00000,	B00000,	B00000,
B00000,	B00000,	B00000,
B00000	B00000	B00000
4	5	6
B11111,	B11111,	B00011,
B11110,	B01111,	B00111,
B11100,	B00111,	B01111,
B11000,	B00011,	B11111,
B00000,	B00000,	B11111,
B00000,	B00000,	B11111,
B00000,	B00000,	B11111,
B00000	B00000	B11111
7	8	9
B11000,	B00000,	B11111,
B11100,	B00000,	B11111,
B11110,	B00000,	B11111,
B11111,	B00000,	B11111,
B11111,	B00000,	B11111,
B11111,	B00000,	B11111,
B11111,	B00000,	B11111,
B11111	B00000	B11111

(b) Building Blocks For The Digits

- In order to initialize the LCD the following set of commands need to be sent on the LCD port using the command function explained later. The first command is used to set the LCD into the 8 bit data acceptance mode along with a 2 line spill over, which means that if the first line is filled with characters, the next character sent will be automatically printed in the next line. The second command clears the LCD of any previous garbage value. Next comes the display on command and the cursor mode determination. At the end the entry mode is specifier which determines the movement of data or the cursor on the data input to the LCD.

```
LCD_INIT:   MVI C,38h      ;8bit 2 line mode
            CALL SEND_CMD
            MVI C,01h      ;clear lcd
            CALL SEND_CMD
            MVI C,0Ch      ;display on, cursor off, blink off
            CALL SEND_CMD
            MVI C,06h      ; entry mode
            CALL SEND_CMD
            RET

```

- The following are two subroutines which are used to send either data or commands to the LCD display. Both of them follow a strict protocol in which first the data or command word is made available on the PORTB lines of the 8155. The protocol next requires the LCDRS to be either high or low depending on the type of word, data or command. Next a pulse of LCDEN is required for the LCD controller to take in the data after which a delay is given to perform the operation successfully without any interference.

```
SEND_DATA:  PUSH PSW       ;C for arg
            MOV A,C        ;DATA OUT
            OUT PORTB
            IN PORTA
```

```
            ORI LCDRS   ;for making rs 1
            OUT PORTA   ;sending rs before lcd_en
            ORI LCDEN   ;EN pulse
            OUT PORTA
            ANI 0FBh
            OUT PORTA
            CALL DELAY_2MS
            POP PSW
            RET

SEND_CMD:   PUSH PSW  ;C for arg
            MOV A,C       ;COMMAND OUT
            OUT PORTB
            IN PORTA
            ANI 0FDh  ;for making rs 0
            OUT PORTA ;sending rs before lcd_en
            ORI LCDEN   ;EN pulse
            OUT PORTA
            ANI 0FBh
            OUT PORTA
            CALL DELAY_2MS
            POP PSW
            RET
```

5.2.2 Key-Pad Scanning

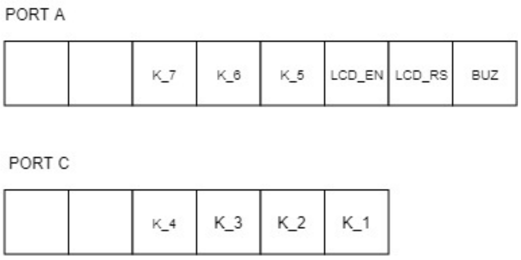


Figure 4: Connections To port A and C

It is interfaced using polling technique. In the polling technique, each switch press is detected by polling individual switches. In this, rows are treated as input and columns as output. A switch is identified by outputting through columns and inputting through rows.

```

KEYPAD: PUSH PSW
        PUSH H
COL1:   IN PORTA      ;input current state of portA
        ANI 0F7h      ;make the K_5 bit as 0 to read inputs for col1 only
        OUT PORTA
        CALL DELAY_2MS;debouncing delay
        IN PORTC      ;read the column
        ANI 0Fh
PRES1:  CPI 0Eh       ;is button in row 1 pressed?
        JNZ PRES4
        MVI A,01h
        JMP ENDKEYPAD
PRES4:  CPI 0Dh       ;is button in row 2 pressed?
        JNZ PRES7
        MVI A,04h
        JMP ENDKEYPAD
PRES7:  CPI 0Bh       ;is button in row 3 pressed?
        JNZ PRESS10
        MVI A,07h
        JMP ENDKEYPAD
PRESS10:CPI 07h       ;is button in row 4 pressed?
        JNZ COL2
        MVI A,0Ah
        JMP ENDKEYPAD

COL2: ...

COL3: ...

ENDKEYPAD:
        CPI 0Fh       ;is no button pressed?
        JZ COL1       ;if that is the case then continue polling
        LXI H,PRESSED
        MOV M,A       ;store the pressed
        CALL DELAY_500MS
        IN PORTA
        ORI 38h       make the K_5,6,7 inactive
        OUT PORTA
        POP H
        POP PSW
        RET

```

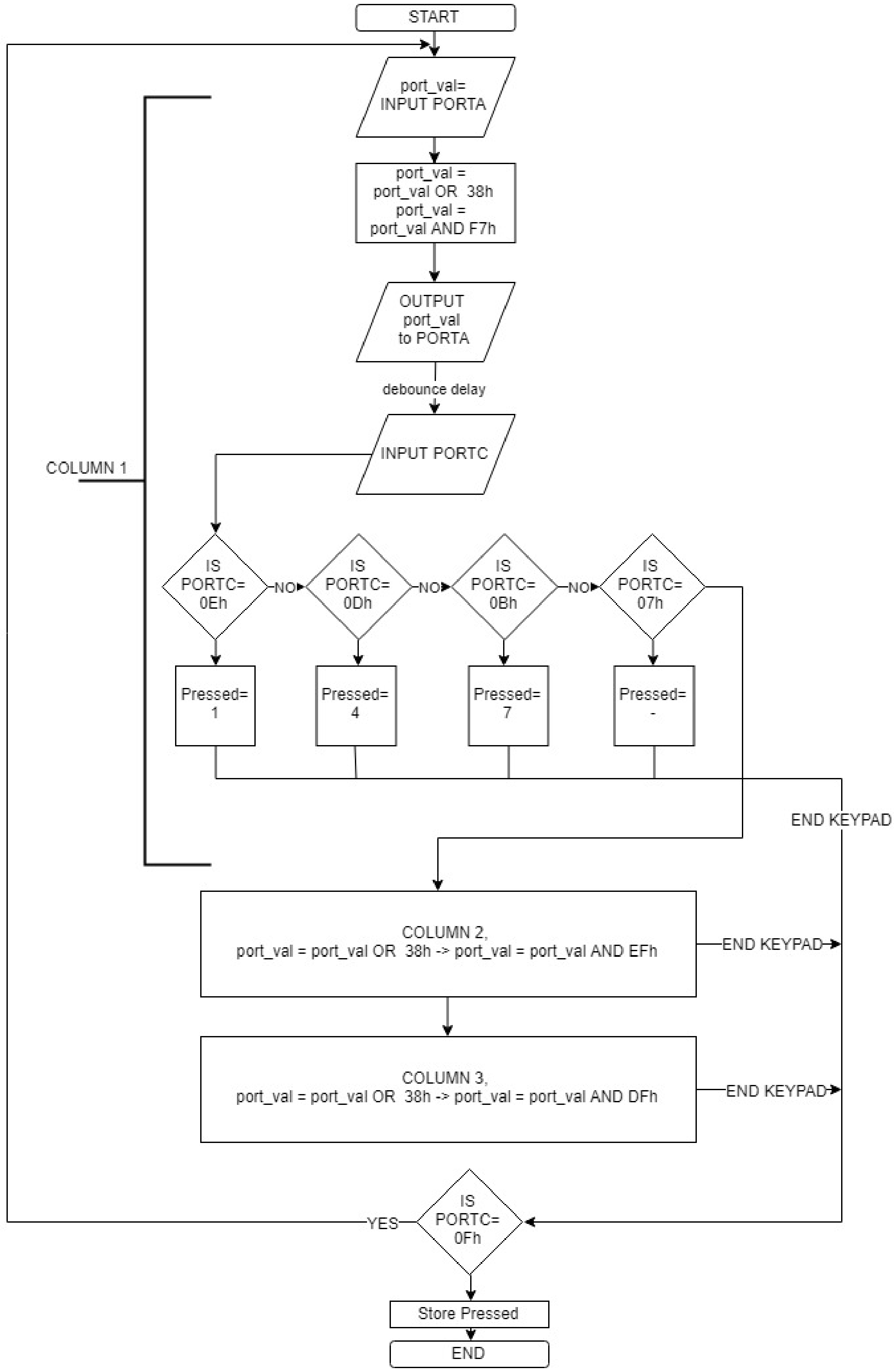


Figure 5: Keypad Scanning Algorithm Flowchart

5.2.3 LFSR(Linear Feedback Shift Register) Implementation

It is an algorithm to generate random numbers. In this method we have two components- seed and mask. Algorithm goes this way: seed value is masked using mask value such that the bits in seed, corresponding to the ones in mask value, are XORed and the resultant is moved to the MSB of seed and the other bits are shifted one to left. In this way we generate a random number each time this cycle is repeated.

```
LFSR:    PUSH PSW  ;RET VAL IN THE SEED LOC
         PUSH B
         PUSH D
         PUSH H
         LXI H,SEED
         MOV A,M ;retrieve seed
         LXI H,BITS
         ANA M    ;retrieve mask bits
         MVI B,00H    ;count
         MVI C,08H    ;8 bits
LOOP1:   RRC          ;LSB in CARRY
         JNC NXT1
         INR B        ;increment B is carry is 1
NXT1:    DCR C
         JNZ LOOP1
         MOV A,B
         RLC
         RLC
         RLC
         RLC
         RLC
         RLC
         RLC
         ANI 80H
         MOV B,A ;store the XOR result
                 ;in the MSB of B
         LXI H,SEED
         MOV A,M
         STC
         CMC
         RAR ;rotate SEED making MSB=0
         ORA B    ;put the new MSB val from B
         STA SEED     ;replace old seed
                      ;val with new rand no.
         POP H
         POP D
         POP B
         POP PSW
         RET
```

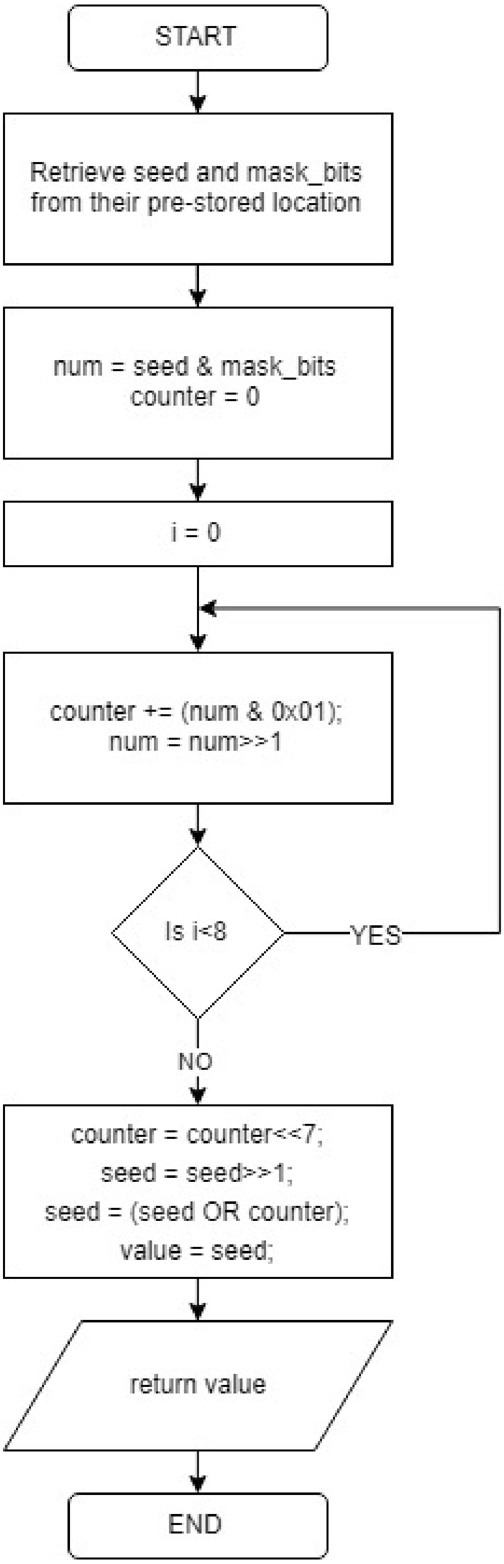


Figure 6: Flowchart of LFSR Algorithm

5.2.4 User Setup Inputs With Interrupts

Interrupt RST7.5 is used as an user interface to update the timings of normal day time and the alarm time. It's a user interacting switch which is connected to RST7.5. Each time this switch is pressed, program transfers to settings window to set either day time or alarm time. According to the option selected, settings transfer to set the required time.

5.2.5 Arithmetic Random Problem Generation With Result Check

Each time alarm time matches the regular time, program transfers to a section in which some random problems(based on addition) are generated using LFSR algorithm. The result to these problems are also calculated and stored n the memory at the same time. The results are checked against the ones that are inputted by the user. Only when the result matches the input the alarm goes off otherwise more random problems are generated.

6 Supporting Materials

””””””””””””””””””””””””””””””””””

6.1 Support Links and References

- Youtube Video: https://www.youtube.com/watch?v=FnkgJ6MPxQ0
- GITHUB Link for Code: https://github.com/srijanpabbi/Mathematical-Alarm-Clock-8085-uP-based
- Programmer and Assembler related Material: https://drive.google.com/drive/folders/0B3dkOo5f1Sb-ZHQ5bHRYb3ZnR2c?usp=sharing
- Other 8085 Projects for Reference: http://www.8085projects.in/
- A forum discussion on LCD bigfont: https://www.avrfreaks.net/forum/alphanumeric-lcd-hd44780-big-font-big-digits-generator-excel-sheet
- Full End to End character LCD tutorial in C as well as Assembly Language
- RTC Datasheet: https://pdfserv.maximintegrated.com/en/ds/DS12885-DS12C887A.pdf
- OKI 80C85 Datasheet: http://elearning.tukenya.ac.ke/course/view.php?id=2424
- LCD Controller Datasheet: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

6.2 Project Schematic And Boards

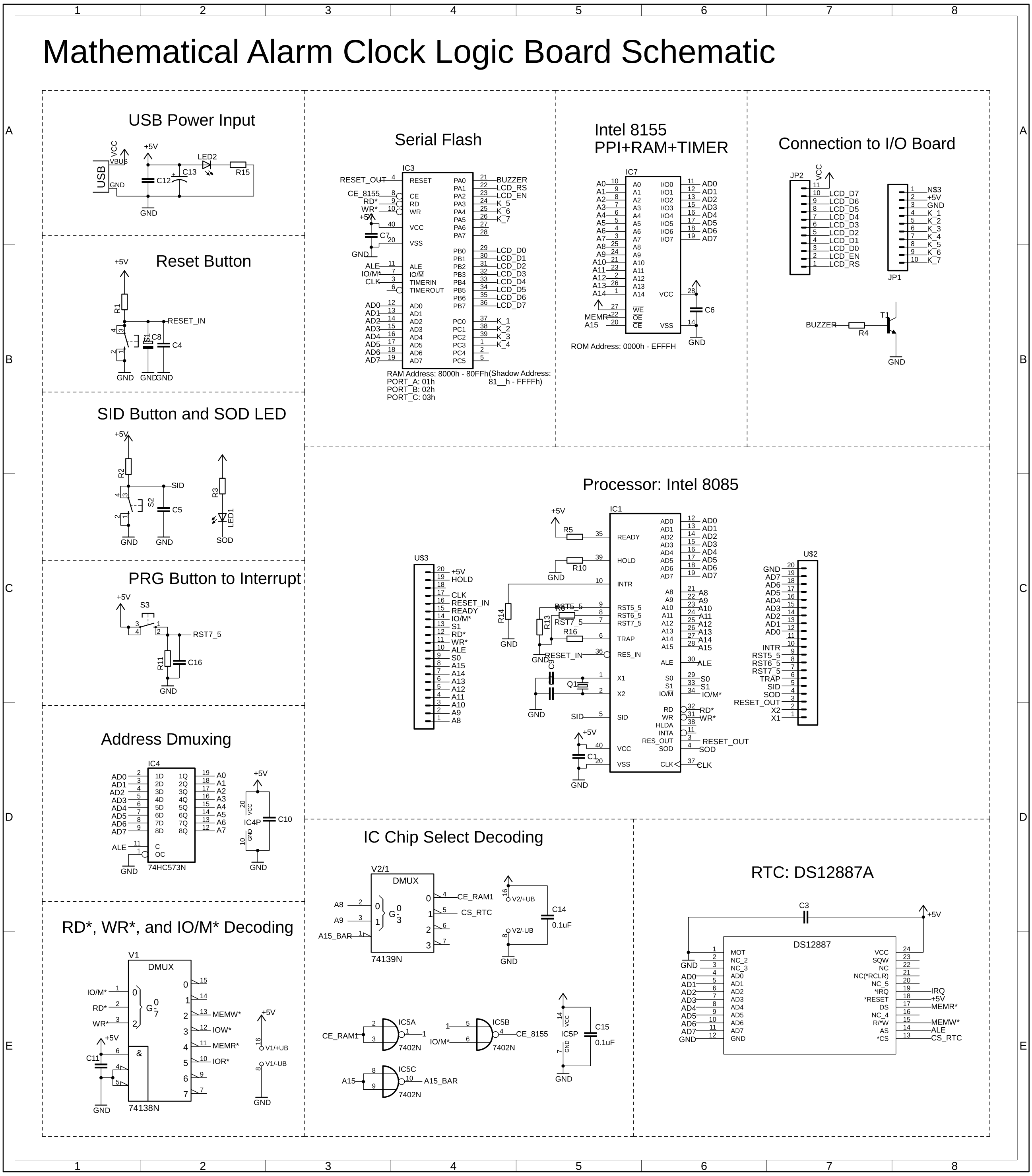


Figure 7: Logic Board Schematic

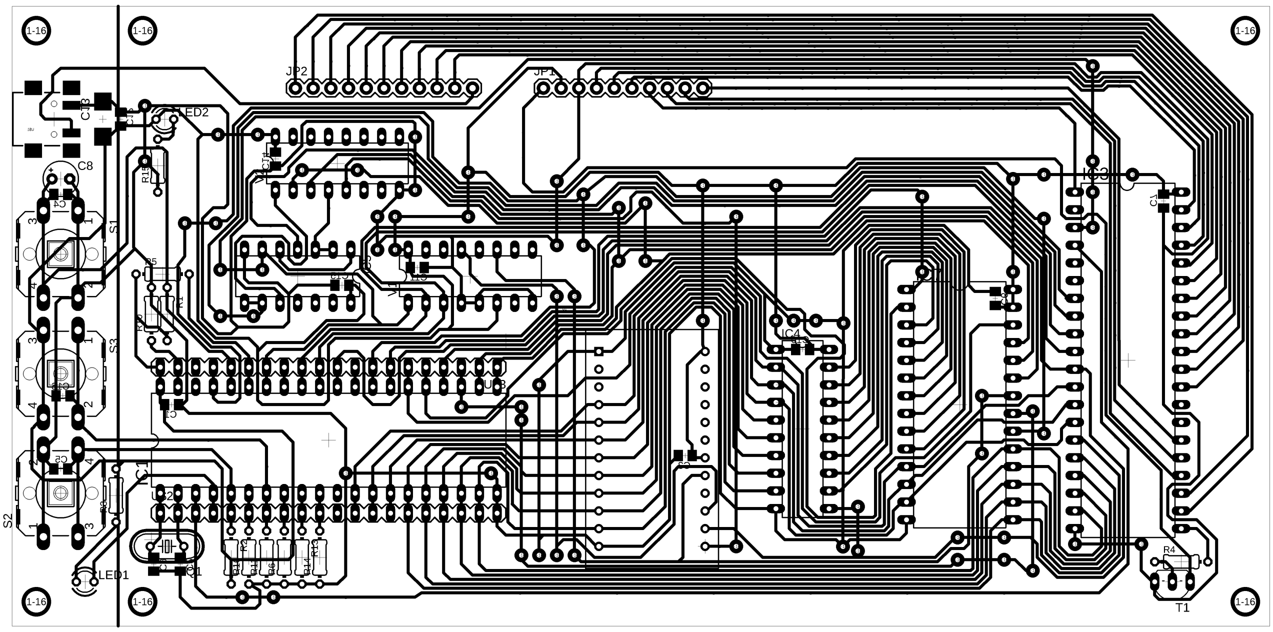


Figure 8: Logic Board Layout

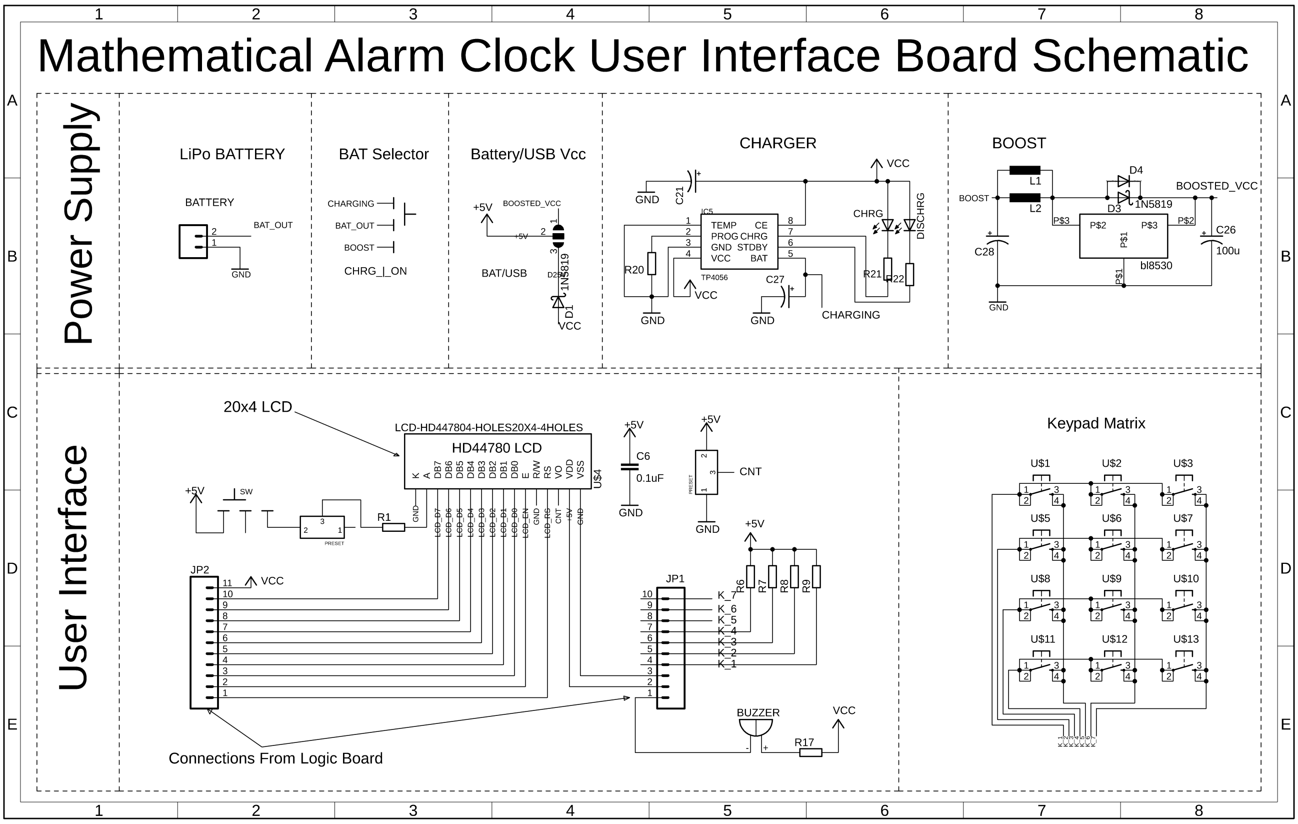


Figure 9: User Interface Board Schematic

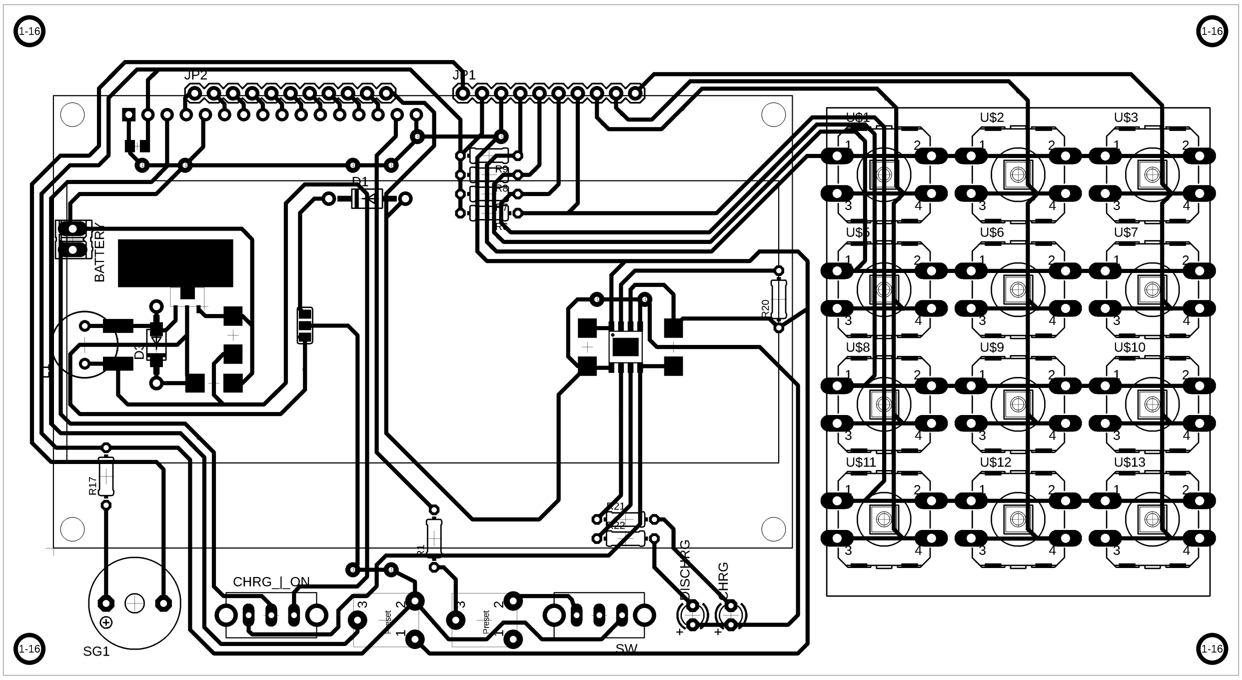


Figure 10: User Interface Board Layout

6.3 Bill of Materials

S.No.	Component	Note
1.	TMP8085A	Microprocessor
2.	ROM	EEPROM
3.	Decoder	3X8
4.	Crystal	
5.	DS12887	Parallel Bus RTC
6.	8255	PPI
7.	8155	PPI
8.	LCD	20X4
9.	Buzzer	
10.	Keypad	4X3
11.	Resistors	
12.	Capacitors	
13.	Omron Switches	
14.	LEDs	
15.	Slide Switch	
16.	USB Power	MINI USB