

**National University of Science and Technology (NUST)**

**School of Electrical Engineering and Computer Science(SEECS)**

**Microprocessor System**

**Project Report of Digital Clock with Alarm**

|  |  |
| --- | --- |
| **Faulty Member:** Shaiza Malik | **Dated:** 17th May 2024 |
| **Semester:** 4th (Spring 2024) | **Section:** BEE-14D |

|  |  |
| --- | --- |
| **Name** | **CMS** |
| Muhammad Umair | 405215 |
| Irfa Farooq | 412564 |

1. **Introduction:**

A clock is the most used item in houses, offices, and every place. The project is about designing and implementing clocks in an innovative way. The project is based upon AVR microcontroller. The microcontroller displays the time, day, and year on the Liquid crystal Display (LCD).

1. **Problem Statement:**

The project involves developing a system that integrates an AVR microcontroller with 16x2 Liquid Crystal Display (LCD) and coding it to accurately display the current time and date, along with implementing an alarm feature. The key tasks include establishing code of atmega. Additionally, it requires setting up and interfacing the LCD with the AVR, ensuring proper configuration and display functionality along with an alarm feature allowing users to set specific alarm times, with the system comparing the current time to the alarm time and triggering an alert.

1. **Objectives**

The objectives for our project are:

1. Solving real world problems using the knowledge gained in class and lab.
2. Coding AVR in C language.
3. Connecting external devices with AT mega.
4. Serial and parallel data transformation from AT mega to other devices.
5. **Methodology**

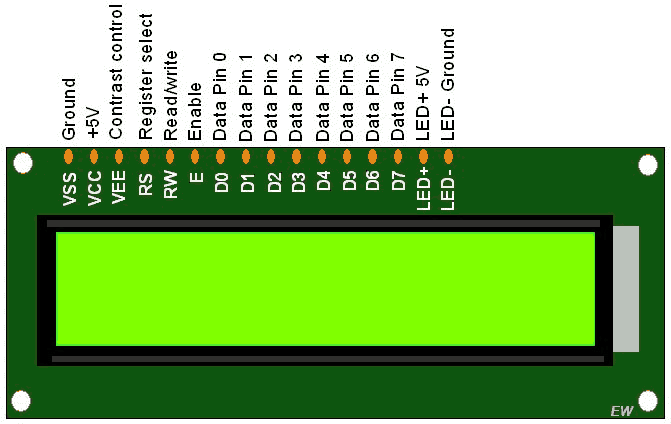
For the project implementation the project was divided into 3 parts

* LDC integration with Atmega.
* Coding Atmega
* Alarm Setup
* Hardware Implementation

1. **LCD integration with ATmega16:**

LCD is the most used component to display data on screen. In total LCD had 14 pins. It has 8 pins to receive data. 1 pin is to enable the LCD, 1 pin is for read or write enable. Other pins are used to select display settings.

In our project, we are giving 4-bit data to display time, day, date, and year. We are using 16x2 LCD as it is sufficient enough for our required data to be displayed.



1. **Coding Atmega**

For the atmega coding the concepts used are:

* Time delay
* Timer Interrupts
* Hardware Interrupts
* LCD to Atmega connection

1. **Alarm**

For the alarm, we have used serial data transmission concept (USART). The user will enter the time on PC through TeraTerm which is bought to ATmega16 using Serial transmission. Then the AVR will read time from RTC and compare it with the user given time. When comparison becomes true, the alarm will turn ON and remain ON for 30 seconds.

For the alarm we have used following concepts:

* Interrupts (To turn off buzzer after 30 sec)
* In built delay functions to create a 30 second delay
* Serial Data transmission (To allow user to set time for alarm)

1. **Resources**

The components used in the project are:

* **Atmega16A:**

Atmega16A is a 40 pins low power 8-bit microcontroller with CMOS technology.

* **LCD (16x2):**

Liquid Crystal display (LCD) is used to display time.

* **Other components**

Other components include breadboard, potentiometer, wires, etc.

1. **Results**

The code is attached below:

**Code:**

**Main.c:**

#include <avr/io.h>

#define *F\_CPU* 1000000UL

#include <util/delay.h>

#include "schedular.h"

int main(void){

start();

while (1){

engine();

}

}

**clock.h:**

#ifndef CLOCK\_H\_

#define CLOCK\_H\_

typedef unsigned char BYTE;

#define PM 0

#define AM 1

void clock\_write(BYTE hr, BYTE min , BYTE sec);

void date\_write(BYTE day , BYTE month , unsigned int year);

#endif /\* CLOCK\_H\_ \*/

**clock.c:**

#include "clock.h"

#include "LCD.h"

#define *F\_CPU* 1000000UL

#include <avr/delay.h>

void clock\_write(BYTE hr, BYTE min, BYTE sec){

BYTE hr0,hr1,min0,min1,sec0,sec1;

hr1 = hr%10;

hr0 = hr/10.0;

min1 = min%10;

min0 = min/10.0;

sec1 = sec%10;

sec0 = sec/10.0;

hr0+=48;

hr1+=48;

min0+=48;

min1+=48;

sec0+=48;

sec1+=48;

LCD\_startfrom(1,1);

*\_delay\_ms*(3);

LCD\_write\_char(hr0);

LCD\_write\_char(hr1);

LCD\_write\_char(':');

LCD\_write\_char(min0);

LCD\_write\_char(min1);

LCD\_write\_char(':');

LCD\_write\_char(sec0);

LCD\_write\_char(sec1);

*\_delay\_ms*(2);

}

void date\_write(BYTE day , BYTE month , unsigned int year){

BYTE day0,day1,month0,month1,year0,year1,year2,year3;

day1 = day%10;

day0 = day/10.0;

month1 = month%10;

month0 = month/10.0;

year0 = year / 1000.0;

year1 = (year%1000) / 100.0;

year2 = (year%100) / 10;

year3 = year%10;

day0+=48;

day1+=48;

month0+=48;

month1+=48;

year0+=48;

year1+=48;

year2+=48;

year3+=48;

LCD\_startfrom(2,1);

*\_delay\_ms*(3);

LCD\_write\_char(day0);

LCD\_write\_char(day1);

LCD\_write\_char('/');

LCD\_write\_char(month0);

LCD\_write\_char(month1);

LCD\_write\_char('/');

LCD\_write\_char(year0);

LCD\_write\_char(year1);

LCD\_write\_char(year2);

LCD\_write\_char(year3);

}

**DIO.h:**

#ifndef DIO\_H\_

#define DIO\_H\_

typedef unsigned char BYTE;

#define INPUT 0

#define OUTPUT 1

#define HIGH 1

#define LOW 0

void pin\_mode (char port , BYTE pin , BYTE mode);

void pin\_write (char port , BYTE pin ,BYTE value);

void port\_write (char port , BYTE value);

BYTE pin\_read (char port , BYTE pin);

BYTE port\_read(char port);

#endif /\* DIO\_H\_ \*/

**DIO.c:**

#include <avr/io.h>

#include "DIO.h"

typedef unsigned char BYTE;

#define INPUT 0

#define OUTPUT 1

#define HIGH 1

#define LOW 0

void pin\_mode (char port , BYTE pin , BYTE mode){

switch (port){

case 'A':{

if (mode == OUTPUT) DDRA |= (1<<pin);

else if (mode == INPUT) DDRA &= ~(1<<pin);

break;

}

case 'B':{

if (mode == OUTPUT) DDRB |= (1<<pin);

else if (mode == INPUT) DDRB &= ~(1<<pin);

break;

}

case 'C':{

if (mode == OUTPUT) DDRC |= (1<<pin);

else if (mode == INPUT) DDRC &= ~(1<<pin);

break;

}

case 'D':{

if (mode == OUTPUT) DDRD |= (1<<pin);

else if (mode == INPUT) DDRD &= ~(1<<pin);

break;

}

default:{

break;

}

}

}

void pin\_write (char port , BYTE pin ,BYTE value){

switch(port){

case 'A':{

if(value == HIGH) PORTA |= (1<<pin);

else if (value == LOW) PORTA &= ~(1<<pin);

break;

}

case 'B':{

if(value == HIGH) PORTB |= (1<<pin);

else if (value == LOW) PORTB &= ~(1<<pin);

break;

}

case 'C':{

if(value == HIGH) PORTC |= (1<<pin);

else if (value == LOW) PORTC &= ~(1<<pin);

break;

}

case 'D':{

if(value == HIGH) PORTD |= (1<<pin);

else if (value == LOW) PORTD &= ~(1<<pin);

break;

}

default:{

break;

}

}

}

void port\_write (char port , BYTE value){

switch(port){

case 'A': PORTA = value;

case 'B': PORTB = value;

case 'C': PORTC = value;

case 'D': PORTD = value;

}

}

BYTE pin\_read (char port , BYTE pin){

BYTE val;

switch(port){

case 'A' : {

val = PINA;

val &= (1<<pin);

val = (val>>pin);

return val;

}

case 'B' : {

val = PINB;

val &= (1<<pin);

val = (val>>pin);

return val;

}

case 'C' : {

val = PINC;

val &= (1<<pin);

val = (val>>pin);

return val;

}

case 'D' : {

val = PIND;

val &= (1<<pin);

val = (val>>pin);

return val;

}

}

}

BYTE port\_read(char port){

switch(port){

case 'A': return PINA;

case 'B': return PINB;

case 'C': return PINC;

case 'D': return PIND;

}

}

**Interrupt.h:**

#ifndef INTERRUPT\_H\_

#define INTERRUPT\_H\_

void interrupt\_init();

#endif /\* INTERRUPT\_H\_ \*/

**Interrupt.c:**

#include <avr/interrupt.h>

#include "interrupt.h"

#include "LCD.h"

#include "clock.h"

#include "schedular.h"

extern BYTE mode;

extern BYTE HR;

extern BYTE MIN;

extern BYTE SEC;

extern BYTE TIME;

extern BYTE DAY;

extern BYTE MONTH;

extern unsigned int YEAR;

// Mode Bottom

ISR(INT0\_vect){

/\* \* MODE 0 IS RUNNING

\* MODE 1 IS EDITING HOURS

\* MODE 2 IS EDITING MINUITES

\* MODE 3 IS EDITING SECONDS

\* MODE 4 IS EDITING DAYS

\* MODE 5 IS EDITING MONTHS

\* MODE 6 IS EDITING YEARS

\*/

if(mode == 6) {mode = 0;}

else {mode++;}

clock\_write(HR , MIN , SEC);

date\_write(DAY , MONTH, YEAR);

}

// Increment Bottom

ISR(INT1\_vect){

// increasing hours

if(mode == 1){

if(HR == 12) {HR = 1;}

else {HR++;}

clock\_write(HR , MIN , SEC);

}

// increasing minuets

else if(mode == 2){

if(MIN == 59) {MIN = 0;}

else {MIN++;}

clock\_write(HR , MIN , SEC);

}

// increasing seconds

else if(mode == 3){

if(SEC == 59) {SEC = 0;}

else {SEC++;}

clock\_write(HR , MIN , SEC);

}

// increasing days

else if(mode == 4){

BYTE max\_days;

if(MONTH==1 ||MONTH==3 ||MONTH==5 ||MONTH==7 ||MONTH==8 ||MONTH==10 ||MONTH==12) max\_days = 31;

else if (MONTH == 2) max\_days = 28;

else max\_days = 30;

if(DAY == max\_days) {DAY = 1;}

else {DAY++;}

date\_write(DAY,MONTH,YEAR);

}

// increasing months

else if(mode==5){

if(MONTH==12) {MONTH = 1;}

else {MONTH++;}

date\_write(DAY,MONTH,YEAR);

}

// increasing years

else if(mode==6){

YEAR++;

date\_write(DAY,MONTH,YEAR);

}

}

// Decrement Bottom

ISR(INT2\_vect){

// decreasing hours

if(mode == 1){

if(HR == 1) {HR = 12;}

else {HR--;}

clock\_write(HR , MIN , SEC);

}

// decreasing minuets

else if(mode == 2){

if(MIN == 0) {MIN = 59;}

else {MIN--;}

clock\_write(HR , MIN , SEC);

}

// decreasing hours

else if(mode == 3){

if(SEC == 0) {SEC = 59;}

else {SEC --;}

clock\_write(HR , MIN , SEC);

}

// decreasing days

else if(mode == 4){

BYTE max\_days;

if(MONTH==1 ||MONTH==3 ||MONTH==5 ||MONTH==7 ||MONTH==8 ||MONTH==10 ||MONTH==12) max\_days = 31;

else if (MONTH == 2) max\_days = 28;

else max\_days = 30;

if(DAY == 1) {DAY = max\_days;}

else {DAY--;}

date\_write(DAY,MONTH,YEAR);

}

// decreasing months

else if(mode==5){

if(MONTH==1) {MONTH = 12;}

else {MONTH--;}

date\_write(DAY,MONTH,YEAR);

}

// decreasing years

else if(mode==6){

YEAR--;

date\_write(DAY,MONTH,YEAR);

}

}

void interrupt\_init(){

SREG |= (1<<7); // Enable global interrupt

// Enable INT0 for rising edge

MCUCR |= (1<<0);

MCUCR |= (1<<1);

MCUCR |= (1<<3);

MCUCR |= (1<<2);

MCUCSR|= (1<<6);

GICR |= (1<<5);

GICR |= (1<<6);

GICR |= (1<<7);

}

**LCD.h:**

#ifndef LCD\_H\_

#define LCD\_H\_

typedef unsigned char BYTE;

void LCD\_cmd(BYTE cmd);

void LCD\_init ();

void LCD\_write\_char(BYTE data);

void LCD\_write(BYTE\* data);

void LCD\_startfrom(BYTE row , BYTE pos);

void LCD\_set\_time(BYTE time);

#endif /\* LCD\_H\_ \*/

**LCD.c:**

#define *F\_CPU* 1000000UL

#include <util/delay.h>

#include "DIO.h"

#include "LCD.h"

#define PM 0

#define AM 1

void LCD\_cmd(BYTE cmd){

port\_write('A' , cmd); // write command on data bus

pin\_write('B',5,0); // RS = 0

pin\_write('B',6,0); // RW = 0

pin\_write('B',7,1); // E pulse high

*\_delay\_ms*(1);

pin\_write('B',7,0); // E pulse low

*\_delay\_ms*(3);

}

void LCD\_init (){

/\* Always connect D0:D7 to PORTA0:PORTA7

Always connect RS to PORTB5

Always connect RW to PORTB6

Always connect E to PORTB7

\*/

pin\_mode('A' , 0 , OUTPUT); //D0

pin\_mode('A' , 1 , OUTPUT); //D1

pin\_mode('A' , 2 , OUTPUT); //D2

pin\_mode('A' , 3 , OUTPUT); //D3

pin\_mode('A' , 4 , OUTPUT); //D4

pin\_mode('A' , 5 , OUTPUT); //D5

pin\_mode('A' , 6 , OUTPUT); //D6

pin\_mode('A' , 7 , OUTPUT); //D7

pin\_mode('B' , 5 , OUTPUT); //RS instruction\data (0\1)

pin\_mode('B' , 6 , OUTPUT); //R\W (1\0)

pin\_mode('B' , 7 , OUTPUT); //E

*\_delay\_ms*(20);

LCD\_cmd(0x38); // Enable 8-bit mode

LCD\_cmd(0x0C); // Display ON & Cursor OFF

LCD\_cmd(0x06); // Auto increment cursor

LCD\_cmd(0x01); // Clear display

LCD\_cmd(0x80); // cursor at home position

}

void LCD\_write\_char(BYTE data){

port\_write('A' , data); // write command on data bus

pin\_write('B',0,5); // RS = 1

pin\_write('B',1,6); // RW = 0

pin\_write('B',7,1); // E pulse high

*\_delay\_ms*(1);

pin\_write('B',7,0); // E pulse low

*\_delay\_ms*(3);

}

void LCD\_write(BYTE\* data){

BYTE i = 0;

while(data[i] != '\0'){

LCD\_write\_char(data[i]);

i++;

}

}

void LCD\_startfrom(BYTE row , BYTE pos){

if (row != 1 && row!=2) return;

if (pos>16) return;

if (row == 1) LCD\_cmd((pos & 0x0F) | 0x80);

else if (row == 2) LCD\_cmd((pos & 0x0F) | 0xC0);

}

**schedular.h:**

#ifndef SCHEDULAR\_H\_

#define SCHEDULAR\_H\_

void start();

void engine();

#endif /\* SCHEDULAR\_H\_ \*/

**Schedular.c:**

#include "schedular.h"

#include "timer.h"

#include "interrupt.h"

#include "clock.h"

#include "LCD.h"

#include "DIO.h"

BYTE mode; // extern to interrupt

BYTE HR; // extern to interrupt

BYTE MIN; // extern to interrupt

BYTE SEC;

BYTE DAY;

BYTE MONTH;

unsigned int YEAR;

BYTE TIME;

BYTE previous\_time;

BYTE flag

void start(){

mode = 0;

HR = 11;

MIN = 24;

SEC = 00;

DAY = 30;

MONTH = 4;

YEAR = 2000;

TIME = PM;

previous\_time = TIME;

flag = 0;

LCD\_init();

interrupt\_init();

timer\_init();

clock\_write(HR,MIN,SEC);

date\_write(DAY,MONTH,YEAR);

}

void engine(){

if(mode == 0){

if(flag == 1) {// one second

SEC++;

flag=0;

if(SEC == 60){

MIN++;

SEC=0;

}

if (MIN > 59){

HR++;

MIN=0;

}

if (HR > 12){

HR=1;

}

if(HR == 12){

if(TIME==AM) {TIME = PM;}

else if(TIME == PM) {TIME = AM;}

}

if(previous\_time == PM && TIME == AM) {//new day

previous\_time = TIME;

BYTE max\_days;

if(MONTH==1 ||MONTH==3 ||MONTH==5 ||MONTH==7 ||MONTH==8 ||MONTH==10 ||MONTH==12) {//31 days in these months

max\_days=31;

}

else if(MONTH==2) max\_days = 28;

else max\_days = 30;

if (DAY==max\_days){

MONTH++;

DAY = 1;

}

else {DAY++;}

if(MONTH == 12){

YEAR++;

MONTH = 1;

}

}

clock\_write(HR,MIN,SEC);

date\_write(DAY,MONTH,YEAR);

}

}

}

**timer.h:**

#ifndef TIMER\_H\_

#define TIMER\_H\_

void timer\_init();

#endif /\* TIMER\_H\_ \*/

**timer.c:**

#include "timer.h"

#define *F\_CPU* 1000000ul

#include "DIO.h"

#include <avr/interrupt.h>

extern BYTE flag;

BYTE tick;

ISR(TIMER0\_COMP\_vect){

if(tick==100){

flag = 1;

tick = 0;

}

else{tick++;}

}

void timer\_init(){

tick = 0;

SREG |= (1<<7); // Enable global interrupt

TIMSK |= (1<<1); // Enable Timer interrupt

// SET CTC MODE

TCCR0 &= ~(1<<6);

TCCR0 |= (1<<3);

// SET COUNTER CLOCK

TCCR0 |= (1<<0);

TCCR0 &= ~(1<<1);

TCCR0 |= (1<<2);

// SET COMPARING VALUE

OCR0 = 8;

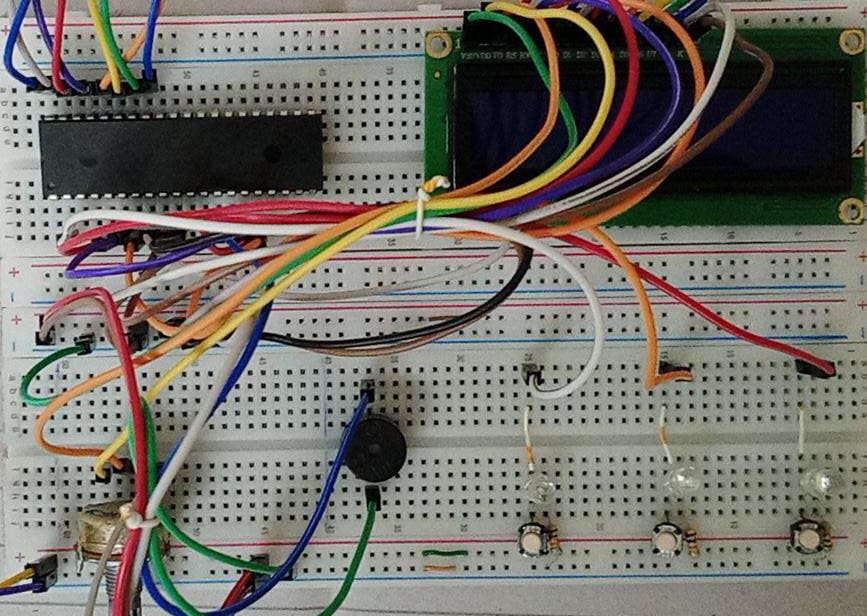
}

**Stimulation:**

**A computer screen shot of a clock

Description automatically generated**

**Hardware:**



1. **Conclusion**

The project is basic, but it has many future applications such as Electrical Appliance automation and Sleep Monitoring. Also, in future we can modify the project to make We can have a bigger microcontroller like AtMega16 or another microcontroller. We can increase the display size. We can design the same project for commercial use. We can add new and exciting features to it.

1. **References**

The idea of the projects was taken from these websites. We have used the basic idea for these websites. We have combined basic ideas with our knowledge to make this project

* <https://www.electronicwings.com/avr-atmega/real-time-clock-rtc-ds1307-interfacing-with-atmega16-32>
* <https://hackaday.io/project/3511-pavapro-portable-avr-programmer/log/12205-keyboard-works-target-avr-programmed>