University of Bahrain

College of Information Technology

Department of Computer Engineering

ITCE230 - Microprocessors

Digital Clock and Alarm System

Prepared by:

Objectives:

1. write c code to implement a fully functioning clock and alarm system

2. design a functioning circuit in Simulide to emulate the functionality of the a Atmega2560

3. debug the code and make it more readable

Introduction

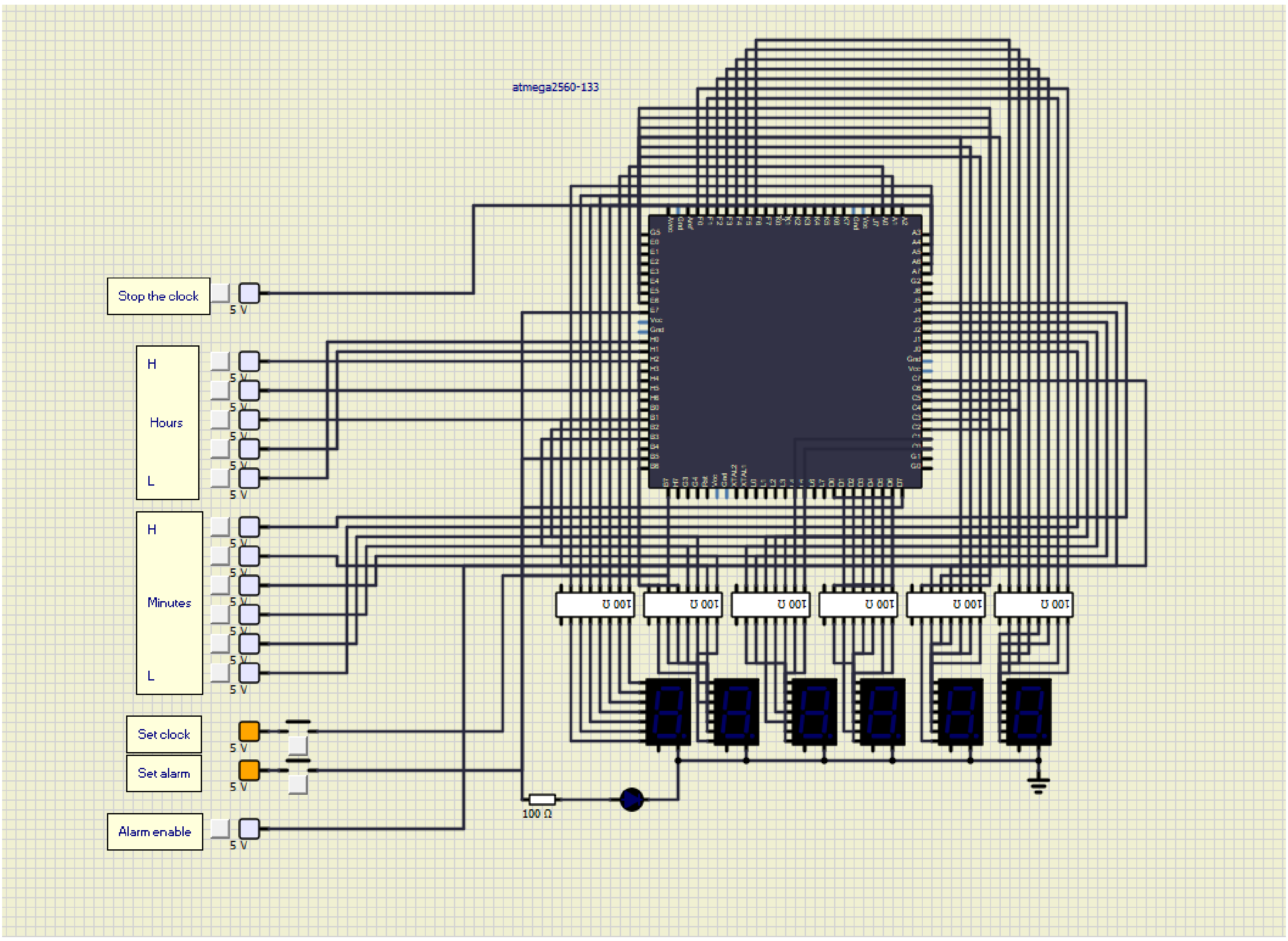
In this project we used the knowledge of what we studied this semester, and we got the idea to develop a clock that stop and sets the time, and this will be displayed by 7-segment display. The clock can also set an alarm and when it reaches a certain time a led will light on for 10 seconds showing that the alarm is now activated. The clock will receive data from fixed volts and switches that will act as inputs.

Design:

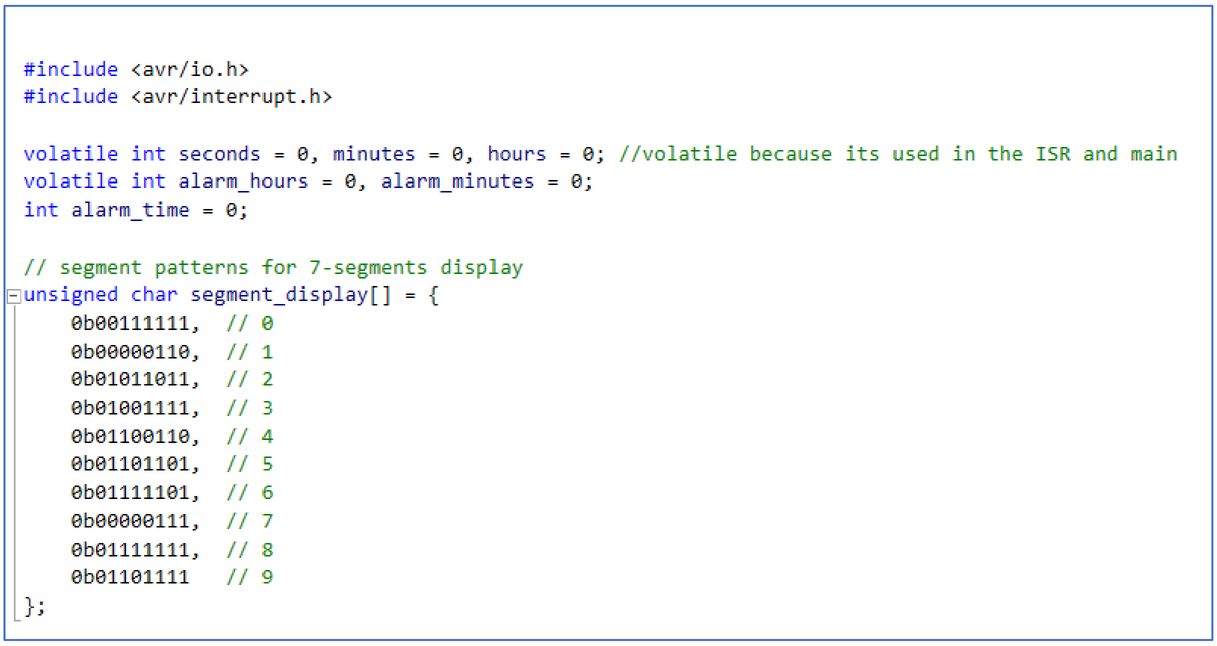
We used these following components for our design:

1. The Atmega2560 microcontroller
2. Switches, push buttons and fixed volts as inputs
3. 7-segment display and led as outputs and some resistors

Full circuit:

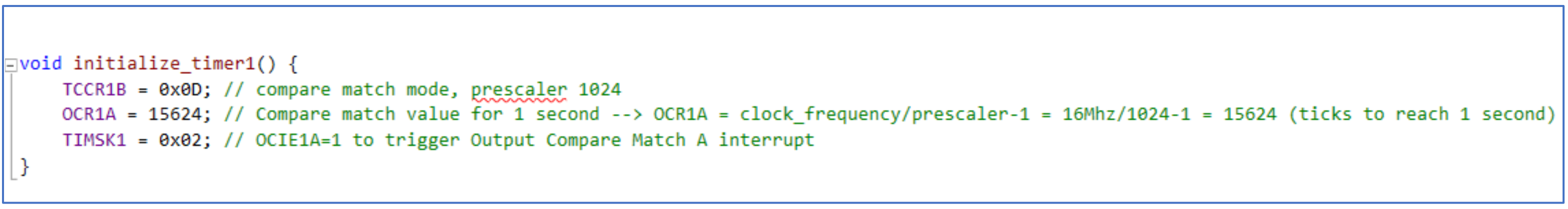


Implementation:



```c
#include <avr/io.h>
#include <avr/interrupt.h>

volatile int seconds = 0, minutes = 0, hours = 0; //volatile because its used in the ISR and main
volatile int alarm_hours = 0, alarm_minutes = 0;
int alarm_time = 0;

// segment patterns for 7-segments display
unsigned char segment_display[] = {
    0b00111111,  // 0
    0b00000110,  // 1
    0b01011011,  // 2
    0b01001111,  // 3
    0b01100110,  // 4
    0b01101101,  // 5
    0b01111101,  // 6
    0b00000111,  // 7
    0b01111111,  // 8
    0b01101111   // 9
};
```

- In this code we used variables to store the current time and alarm settings. Initializing the 7-segements display

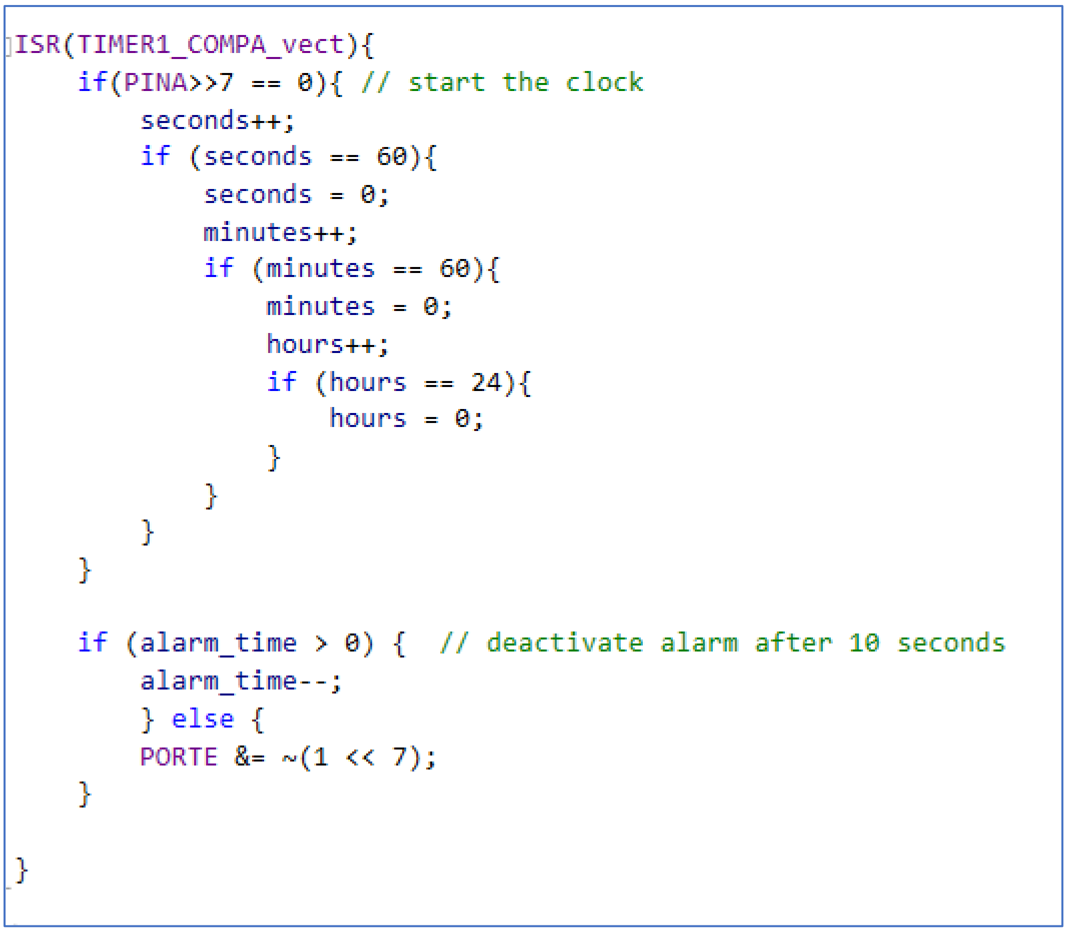


```c
void initialize_timer1() {
    TCCR1B = 0x0D; // compare match mode, prescaler 1024
    OCR1A = 15624; // Compare match value for 1 second --> OCR1A = clock_frequency/prescaler-1 = 16Mhz/1024-1 = 15624 (ticks to reach 1 second)
    TIMSK1 = 0x02; // OCIE1A=1 to trigger Output Compare Match A interrupt
}
```

Initializing the timer

- We made this function set up Timer1, which generates an interrupt every second. This is done using output compare match mode, where the timer compares its count to 15624 (calculated for a 16 MHz clock). The interrupt is enabled to run specific code every time a second passes.

- The interrupt is used for independent counting and more efficient use of the CPU. Also, this makes the clock more reliable and enables multitasking.

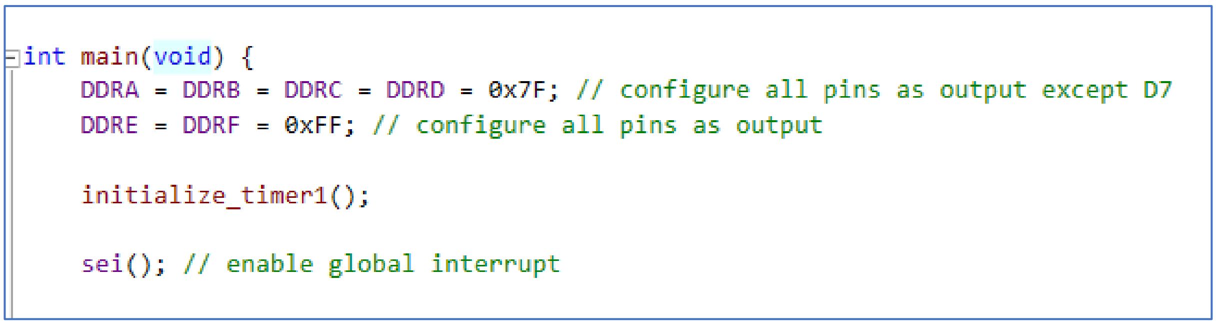


```
ISR(TIMER1_COMPA_vect){
    if(PINA>>7 == 0){ // start the clock
        seconds++;
        if (seconds == 60){
            seconds = 0;
            minutes++;
            if (minutes == 60){
                minutes = 0;
                hours++;
                if (hours == 24){
                    hours = 0;
                }
            }
        }
    }

    if (alarm_time > 0) {  // deactivate alarm after 10 seconds
        alarm_time--;
        } else {
        PORTE &= ~(1 << 7);
    }

}
```

- The interrupt function runs every second. It increases the seconds and resets it to 0 when it reaches 60, updating minutes and hours as needed. However, when PINA.7 is activated, the clock stops.

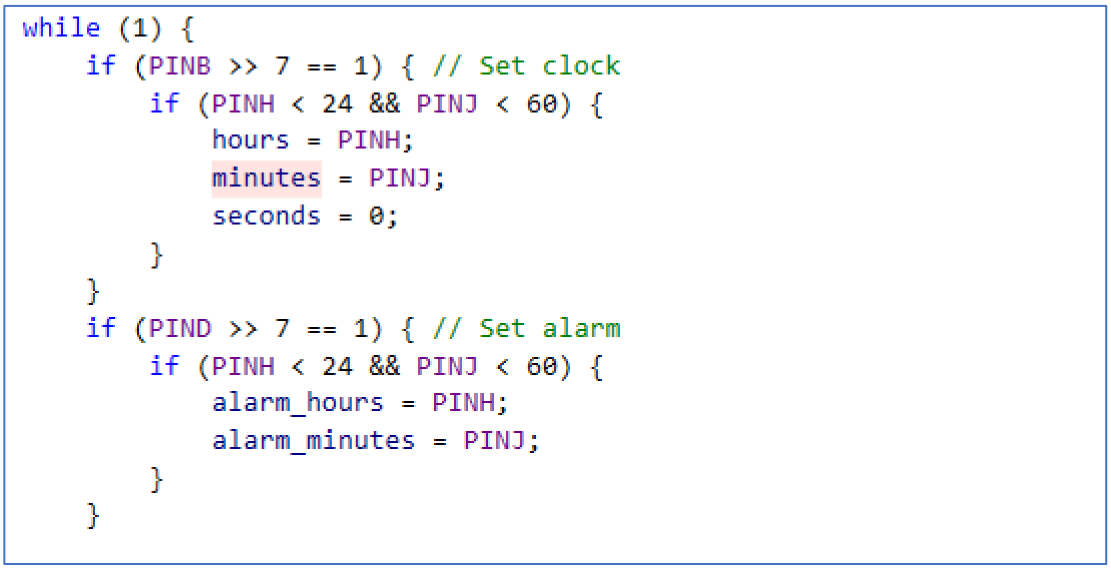
- Also, the part down is related to the alarm time. It's separated from the main because it needs to be inside the ISR since it synchronizes to the timer and since this function is time sensitive.

- PORTE.7 represent the alarm output



```c
int main(void) {
    DDRA = DDRB = DDRC = DDRD = 0x7F; // configure all pins as output except D7
    DDRE = DDRF = 0xFF; // configure all pins as output

    initialize_timer1();

    sei(); // enable global interrupt
```

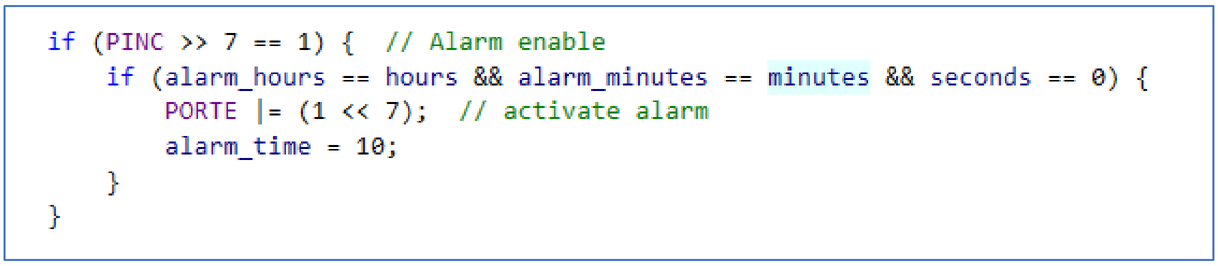
- In the main function, most pins are set as outputs for controlling displays and alarms. The timer setup function is called, and global interrupts are enabled.



```c
while (1) {
    if (PINB >> 7 == 1) { // Set clock
        if (PINH < 24 && PINJ < 60) {
            hours = PINH;
            minutes = PINJ;
            seconds = 0;
        }
    }
    if (PIND >> 7 == 1) { // Set alarm
        if (PINH < 24 && PINJ < 60) {
            alarm_hours = PINH;
            alarm_minutes = PINJ;
        }
    }
```

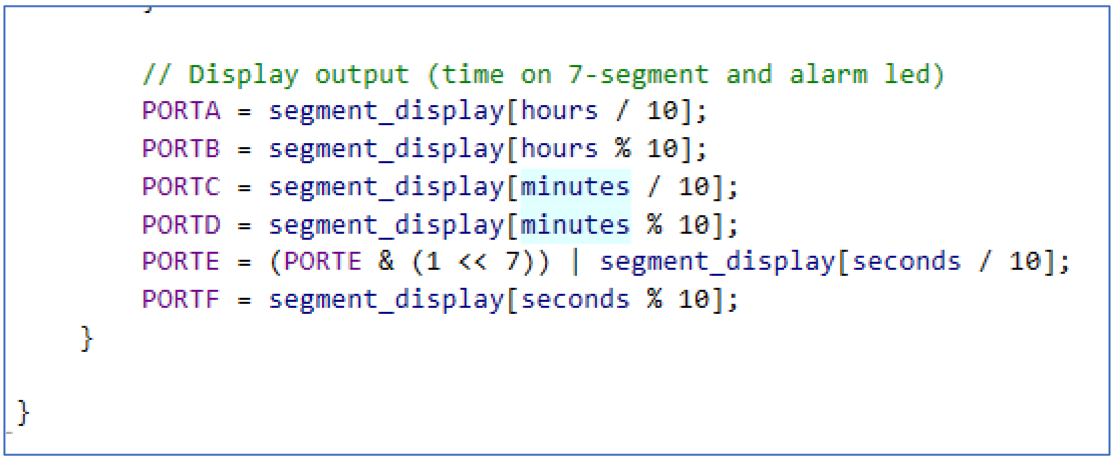
- The program allows you to set time for the clock or the alarm. When PINB.7 triggered, the time values PINH for hours and PINJ for minutes set the clock at that time.

- When PIND.7 triggered, the input for hours and minutes will be saved inside variables. When the clock reaches that time, the alarm will be activated.



```
if (PINC >> 7 == 1) {  // Alarm enable
    if (alarm_hours == hours && alarm_minutes == minutes && seconds == 0) {
        PORTE |= (1 << 7);  // activate alarm
        alarm_time = 10;
    }
}
```

- This will activate the alarm after checking the saved value to the clock then deactivating it after 10 seconds (where it counts down inside the ISR as mentioned before).

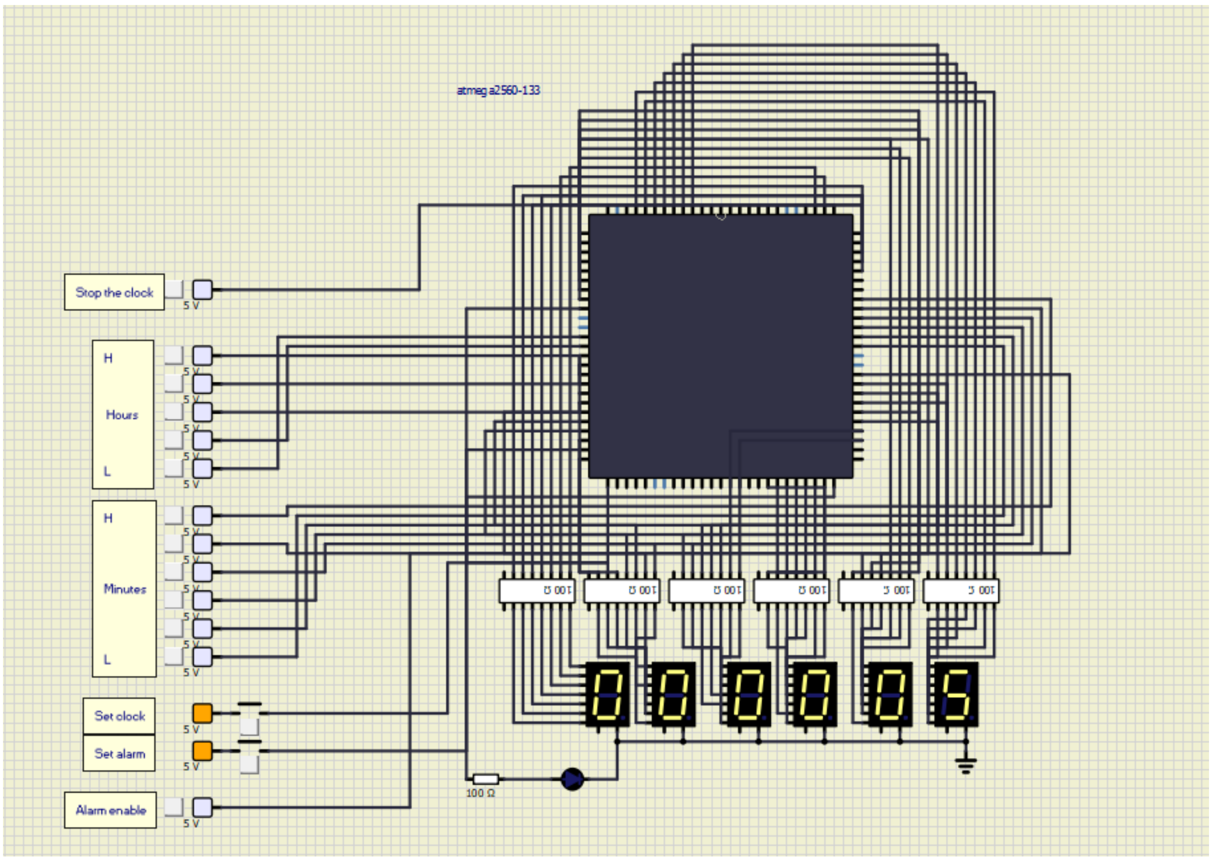


```
        // Display output (time on 7-segment and alarm led)
        PORTA = segment_display[hours / 10];
        PORTB = segment_display[hours % 10];
        PORTC = segment_display[minutes / 10];
        PORTD = segment_display[minutes % 10];
        PORTE = (PORTE & (1 << 7)) | segment_display[seconds / 10];
        PORTF = segment_display[seconds % 10];
    }

}
```

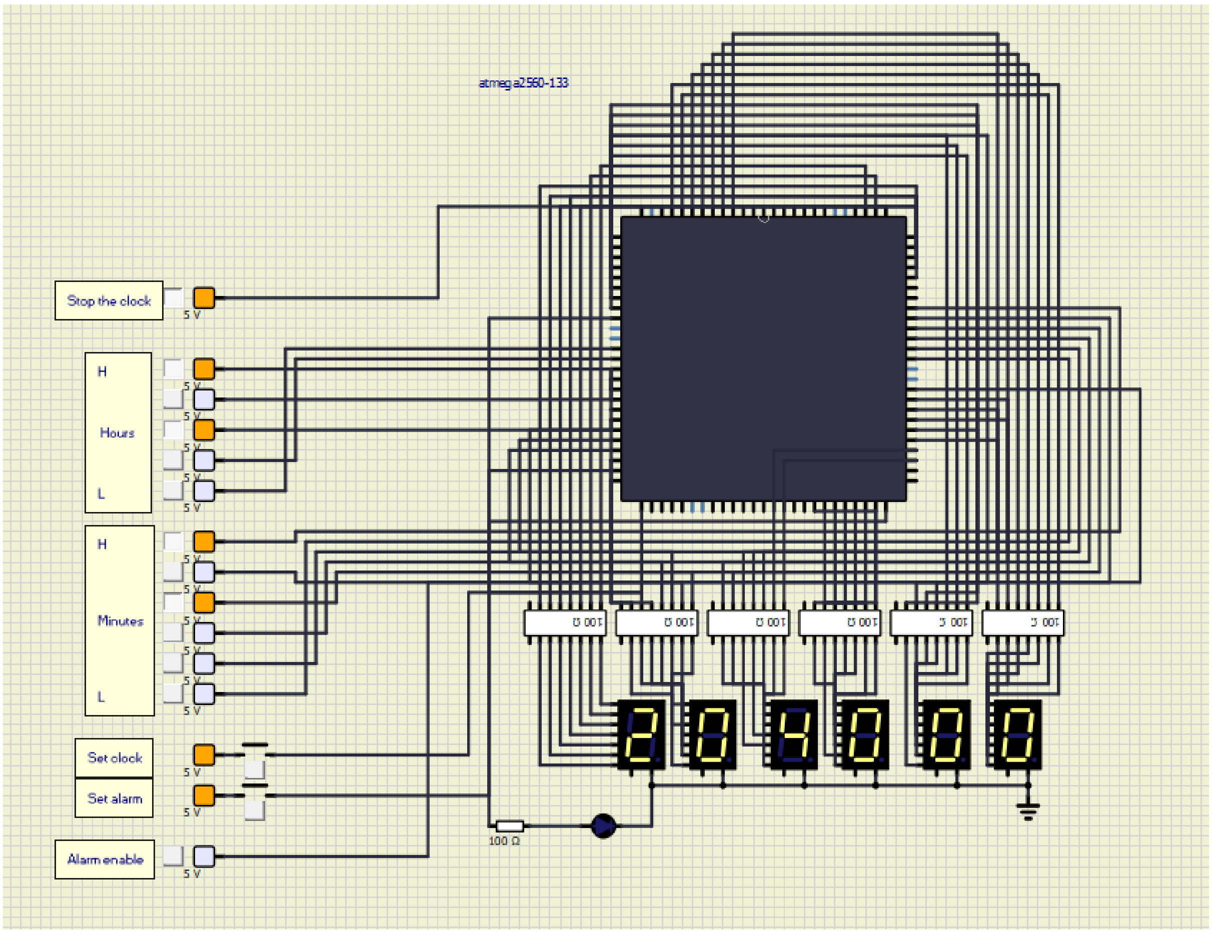
- The current time is shown on a 7-segment display. Each digit is extracted and converted into its corresponding segment pattern using the segment display array. This pattern is then sent to the respective output ports to update the display.

- PORTE is different because it contains the alarm output too.

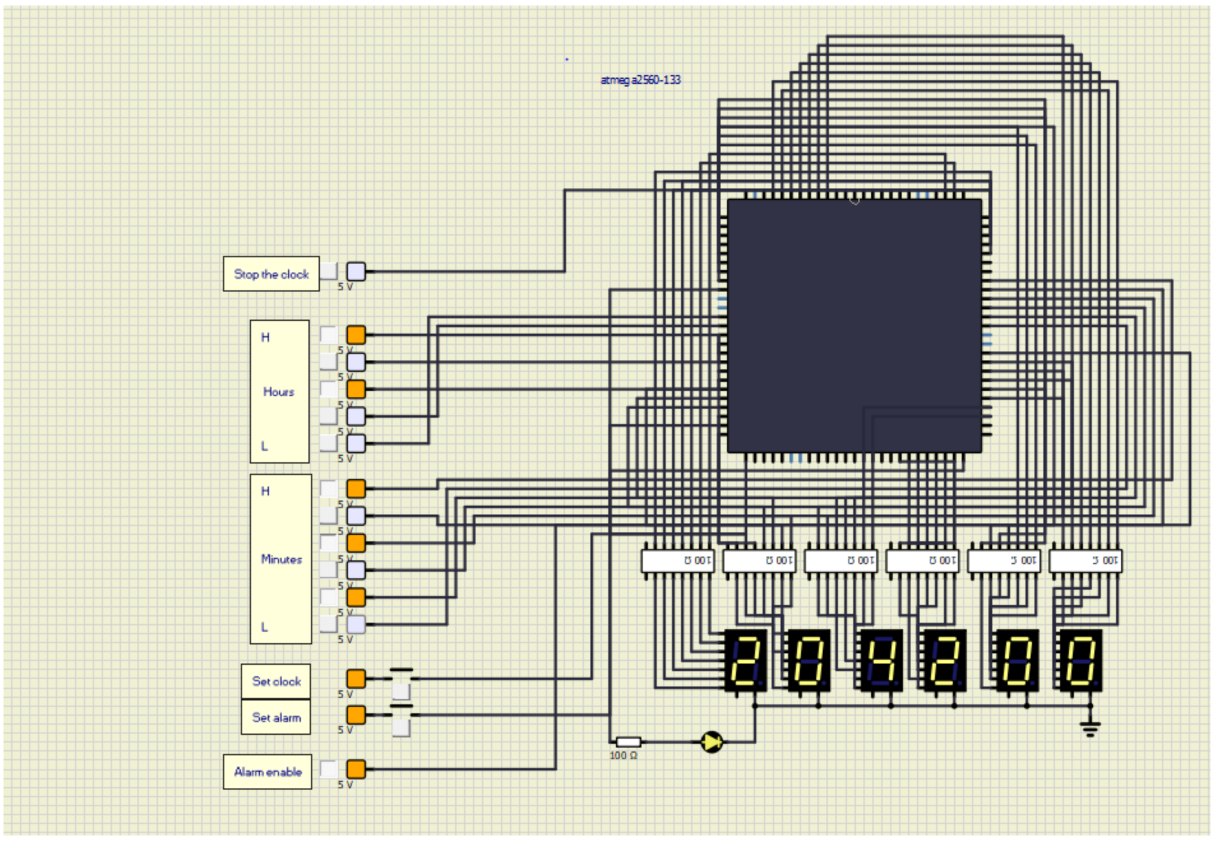
Simulation:



This is the default state, as soon as you turn on the system the timer will start incremating the seconds



Here we set the clock after picking the input as 10100 for (20:00:00) hours and 101000 (00:40:00) for minutes then using the switch for “set clock” to set the time we want.



Setting the alarm is the same as setting the clock we just need to enable the alarm pick an input as well here we picked 10100 for hours and 101010 for minutes. The led activates and will deactivates after ten seconds. The alarm can only be activated if “alarm enable” is turned on.

Conclusion:

We have managed to develop a Clock System that can stop time, has an alarm and can set any given time on the clock and the alarm using the Atmega2560 and the 7-segment display to show the work we done.

References:

- Microcontrollerslab: was used for 7_segment display https://microcontrollerslab.com/7-segment-display-interfacing-with-pic-microcontroller/#google_vig

- Benthomsen.wordpress: was used for timer configuration and programing https://bennthomsen.wordpress.com/arduino/peripherals/timers/