Digital Clock

EE24BTECH11017 - Karthik

1 Introduction

This report presents the design and implementation of a digital clock using an Arduino UNO and AVR programming.This project presents a digital clock using an Arduino Uno, a 7447 BCD to 7-segment decoder IC, and six common cathode 7-segment displays. The clock is designed to display time in HH:MM:SS format. It utilizes multiplexing to control all displays using a single 7447 IC. The Arduino manages time updates and ensures proper digit transitions. The clock runs in real-time and resets after 24 hours. This project demonstrates the practical implementation of BCD to 7-segment conversion and multiplexing in digital electronics.

2 Components and Materials

Components used in the project:

- Six 7-segment displays
- Six resistors (220Ω)
- Arduino UNO
- Breadboard
- Jumper wires
- Power source (cell phone)

3 Circuit Design and Implementation

The circuit was designed on a breadboard, where each 7-segment display was connected to the Arduino through resistors. The wiring was done carefully to ensure proper control of the segments.

4 Software Implementation

The clock functionality was implemented using AVR programming. The Arduino was programmed to control the 7-segment displays to show the correct time. The following is the AVR code used in the project:

4.1 AVR Code

```
// setting cpu frequency to 16 mega hz (for atmega328p)
// atmega328p microcontroller is in the arduino
// frequency is needed for timing calculations
#define F_CPU 16000000UL

// including required libraries
```

```
// they provide access to hardware registers, interrupts, delays
#include <avr/io.h> // standard input-output functions for avr
#include <avr/interrupt.h> // interrupt handling
#include <util/delay.h> // delay functions

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

volatile uint8_t hh = 0, mm = 0, ss = 0;
volatile uint8_t set_mode = 0;

const uint8_t seg_map[10] = {
    0b11000000, 0b11111001, 0b10100100, 0b10110000, 0b10011001,
    0b10010010, 0b10000010, 0b11111000, 0b10000000, 0b10010000
};

void set_segments(uint8_t num) {
    uint8_t pattern = seg_map[num];
    PORTD = (PORTD & 0x03) | ((pattern << 2) & 0xFC);
    if (pattern & (1 << 6)) PORTB |= (1 << PB0);
    else PORTB &= ~(1 << PB0);
}

void enable_digit(uint8_t digit) {
    PORTB &= ~((1 << PB2) | (1 << PB3) | (1 << PB4));
    PORTC &= ~((1 << PC0) | (1 << PC1) | (1 << PC2));

    switch (digit) {
        case 0: PORTB |= (1 << PB2); break;
        case 1: PORTB |= (1 << PB3); break;
        case 2: PORTB |= (1 << PB4); break;
        case 3: PORTC |= (1 << PC0); break;
        case 4: PORTC |= (1 << PC1); break;
        case 5: PORTC |= (1 << PC2); break;
    }
}

ISR(TIMER1_COMPA_vect) {
    if (!set_mode) {
        ss++;
        if (ss == 60) { ss = 0; mm++; }
        if (mm == 60) { mm = 0; hh++; }
        if (hh == 24) { hh = 0; }
    }
```

```
}

void check_buttons(void) {
    if (!(PINB & (1 << PB1))) { // Toggle set mode
        _delay_ms(50);
        if (!(PINB & (1 << PB1))) set_mode = !set_mode;
        while (!(PINB & (1 << PB1))); // Wait for release
    }
    if (set_mode) {
        if (!(PINB & (1 << PB5))) { // Increment hour
            _delay_ms(50);
            if (!(PINB & (1 << PB5))) hh = (hh + 1) % 24;
            while (!(PINB & (1 << PB5)));
        }
        if (!(PINC & (1 << PC3))) { // Minute +1
        _delay_ms(200); // Debounce
        mm = (mm + 1) % 60;
    }

    if (!(PINC & (1 << PC5))) { // Minute +10
        _delay_ms(200); // Debounce
        mm = (mm + 10) % 60;
    }
      if (!(PINC & (1 << PC4))) { // Increment seconds
    _delay_ms(200); // Debounce
    ss = (ss + 1) % 60; // Increment and roll over at 60
}

    }
}

int main(void) {
    DDRD |= 0xFC;
    DDRB |= (1 << PB0) | (1 << PB2) | (1 << PB3) | (1 << PB4);
    DDRC |= (1 << PC0) | (1 << PC1) | (1 << PC2);
  PORTB |= (1 << PB1) | (1 << PB5); // Enable pull-ups for PB1, PB5
PORTC |= (1 << PC3) | (1 << PC4) | (1 << PC5); // Enable pull-ups for
     PC3, PC4, PC5

    TCCR1B |= (1 << WGM12) | (1 << CS12) | (1 << CS10);
    OCR1A = 15625;
    TIMSK1 |= (1 << OCIE1A);
    sei();

    while (1) {
```

```
        check_buttons();
        uint8_t digits[6] = {hh / 10, hh % 10, mm / 10, mm % 10, ss /
            10, ss % 10};
        for (uint8_t digit = 0; digit < 6; digit++) {
            set_segments(digits[digit]);
            enable_digit(digit);
            _delay_ms(3);
        }
    }
}
```

5 Conclusion

This project demonstrated the ability to create a functional digital clock using basic electronic components and AVR programming. Future improvements could include adding a real-time clock (RTC) module for increased accuracy and incorporating additional features like an alarm system.

6 Result

The digital clock successfully displays hours, minutes, and seconds using a common anode 7-segment display. Time can be adjusted using the provided push buttons. The multiplexing technique ensures clear and stable digit display.