SCIENTIFIC CALCULATOR

EE24BTECH11011 - Pranay

1 Introduction

A digital clock is an essential timekeeping device widely used in various applications. This project aims to develop an Arduino-based digital clock using a 7-segment display, controlled through the 7447 BCD to 7-segment decoder. The design integrates multiplexing techniques to efficiently display time with minimal hardware requirements.

2 Hardware Components

The following components were utilized in this project:

- Arduino Uno (Microcontroller Unit)
- 7447 BCD to 7-Segment Decoder
- Six 7-Segment Displays
- Push Buttons for Manual Adjustment
- Resistors and Wires for Connections
- External Power Supply (Optional)

3 Working Mechanism

The clock employs multiplexing to control multiple 7-segment displays with fewer microcontroller pins. This is achieved by:

- Assigning individual enable signals to each display.
- Sending BCD values to the 7447 decoder to illuminate the appropriate segments.
- Rapidly switching between displays to create a seamless visual effect.

4 System Implementation

The digital clock is structured into modular functions to ensure efficient handling of display updates and user inputs.

4.1 BCD Conversion Module

This function encodes a decimal number into BCD format and transmits it to the 7447 decoder.

```
void updateBCD(uint8_t number) {
    PORTD = (PORTD & 0xF0) | (number & 0x0F);
}
```

4.2 Multiplexing Display Module

This function cycles through the displays, activating one at a time.

```
void selectDisplay(uint8_t display) {
    PORTB &= ~(1 << display); // Turn off previous display
    PORTB |= (1 << display); // Activate the current display
}
```

4.3 Time Display Update

The function `refreshDisplay()` updates the time by extracting and displaying individual digits.

```
void refreshDisplay() {
    uint8_t digits[6] = {hour/10, hour%10, minute/10, minute%10, second/10, second%10};
    static uint8_t activeDisplay = 0;
    updateBCD(digits[activeDisplay]);
    selectDisplay(activeDisplay);
    activeDisplay = (activeDisplay + 1) % 6;
}
```

4.4 Button Control Module

Push buttons allow manual adjustments for hours, minutes, and seconds.

```
void handleButtons() {
    if (!(PINB & (1 << HOUR_BTN))) {
        _delay_ms(50);
        if (!(PINB & (1 << HOUR_BTN))) { hour = (hour + 1) % 24; }
    }
    if (!(PINC & (1 << MIN_BTN))) {
        _delay_ms(50);
        if (!(PINC & (1 << MIN_BTN))) { minute = (minute + 1) % 60; }
    }
    if (!(PINC & (1 << SEC_BTN))) {
        _delay_ms(50);
        if (!(PINC & (1 << SEC_BTN))) { second = (second + 1) % 60; }
    }
}
```

5 System Deployment

The code was compiled using **AVR-GCC** and uploaded to the Arduino Uno. The following steps were followed:

1) Writing and testing the code in an Arduino-compatible IDE.
2) Compiling and generating a binary file.
3) Uploading the binary to the microcontroller using ArduinoDroid.

6 Conclusion

This project successfully implemented an Arduino-driven digital clock using a multiplexed 7-segment display. The use of modular code enhances readability and ease of maintenance. Future improvements may include integrating an RTC module for precise timekeeping and incorporating an LCD for better display flexibility.