Digital Clock

Ellanti Rohith - EE24BTECH11020

Aim

This project is on the design and implementation of a digital clock using an **Arduino**, a **7447 BCD to 7-segment decoder**, six **7-segment displays**, and multiple **push buttons**.

1 Introduction

This project demonstrates the working of a **multiplexed digital clock** with hour, minute, and second displays, updated every second. The implementation utilizes:

- **Arduino Uno** as the main processing unit.
- **7447 BCD to 7-segment decoder** for numerical display.
- **Six 7-segment displays** to show hours, minutes, and seconds.
- **Push buttons** for manual adjustments (Hour, Minute, and Reset).

2 Theory and Background

Since an Arduino has a limited number of output pins, controlling multiple 7-segment displays directly is inefficient. Instead, **multiplexing** is used.

2.1 How Multiplexing Works

- The Arduino **activates one display at a time** while sending the corresponding digit data.
- It **cycles through all six displays rapidly** (about 5ms per digit).
- All digits appear continuously lit.

- This allows **six displays to be controlled using only four BCD lines and six control lines**.

2.2 Multiplexing Process in the Code

1. The time is split into six digits: two for hours, two for minutes, and two for seconds.
2. The Arduino outputs each digit's BCD value to the 7447 decoder.
3. The corresponding 7-segment display is enabled while others are turned off.
4. After a brief delay (5ms), the next display is activated.
5. This cycle repeats continuously, creating a seamless time display.

2.3 7447 BCD to 7-Segment Decoder

The **7447 decoder** takes a **4-bit binary input (BCD)** and converts it into signals for a common-anode 7-segment display. This reduces the number of required Arduino pins.

3 Materials and Components

1. **Arduino Uno**
2. **7447 BCD to 7-segment decoder**
3. **Six 7-segment displays** (Common Anode)
4. **Three push buttons** (Hour, Minute, Reset)
5. **Resistors**
6. **Breadboard and jumper wires**

4 Circuit Design and Wiring

4.1 Circuit Connections

The table below details the wiring of the components:

Component	Arduino Pin	Description
BCD Input A	2	Connects to the 7447 BCD to 7-segment decoder input A
BCD Input B	3	Connects to the 7447 BCD to 7-segment decoder input B
BCD Input C	4	Connects to the 7447 BCD to 7-segment decoder input C
BCD Input D	5	Connects to the 7447 BCD to 7-segment decoder input D
7-Segment Digit 1	6	Controls the first digit of the display (Tens place of Hours)
7-Segment Digit 2	7	Controls the second digit of the display (Units place of Hours)
7-Segment Digit 3	8	Controls the third digit of the display (Tens place of Minutes)
7-Segment Digit 4	9	Controls the fourth digit of the display (Units place of Minutes)
7-Segment Digit 5	10	Controls the fifth digit of the display (Tens place of Seconds)
7-Segment Digit 6	11	Controls the sixth digit of the display (Units place of Seconds)
Hour Button	12	Push button to manually increase the hours value
Minute Button	13	Push button to manually increase the minutes value
Pause Button	A0	Push button to pause or resume the clock
Reset Button	A1	Push button to reset the clock to 12:00:00

5 Software Implementation

The Arduino code below manages the clock's operation:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define BCD_PORT PORTD
#define BCD_DDR DDRD
#define BCD_MASK 0b00111100  // PD2 to PD5

#define COMMON_PORT PORTC
```

```
#define COMMON_DDR DDRC

#define MODE_BUTTON PB0 // Switch between Clock, Timer, and Stopwatch
#define STOPWATCH_BUTTON PB1 // Start/Stop Stopwatch and Timer

volatile int seconds = 0, minutes = 30, hours = 15;
volatile int timer_seconds = 0, timer_minutes = 0, timer_hours = 0;
volatile int stopwatch_seconds = 0, stopwatch_minutes = 0, stopwatch_hours
    = 0;
volatile int mode = 0; // 0 = Clock, 1 = Timer, 2 = Stopwatch
volatile int stopwatch_running = 0; // 1 = Running, 0 = Stopped

void setup() {
    // Set BCD display pins (PD2-PD5) as output
    BCD_DDR |= BCD_MASK;
    BCD_PORT &= ~BCD_MASK;

    // Set digit selector pins (PORTC) as output
    COMMON_DDR = 0xFF;
    COMMON_PORT = 0x00;

    // Enable pull-up resistors for buttons
    PORTD |= (1 << PD6) | (1 << PD7);
    PORTB |= (1 << MODE_BUTTON) | (1 << STOPWATCH_BUTTON);

    // Timer1 Setup: CTC Mode, 1-second interval
    TCCR1B |= (1 << WGM12) | (1 << CS12) | (1 << CS10);
    OCR1A = 15625; // 1-second interrupt
    TIMSK1 |= (1 << OCIE1A);

    // Debug LED on PC7 (Bit 7 of PORTC) to check if ISR is running
    DDRC |= (1 << 7);  // Set PC7 as output
    PORTC &= ~(1 << 7); // Initially turn it off

    sei(); // Enable global interrupts
}

ISR(TIMER1_COMPA_vect) {
    PORTC ^= (1 << 7); // Toggle PC7 to check ISR is running

    // Clock Mode Updates
    if (mode == 0) {
        seconds++;
        if (seconds == 60) {
            seconds = 0;
            minutes++;
            if (minutes == 60) {
                minutes = 0;
                hours = (hours + 1) % 24;
```

```
            }
        }
    }

    // Timer Countdown (only when running)
    if (mode == 1 && stopwatch_running) {
        if (timer_seconds > 0  timer_minutes > 0  timer_hours > 0) {
            if (timer_seconds == 0) {
                if (timer_minutes > 0) {
                    timer_minutes--;
                    timer_seconds = 59;
                } else if (timer_hours > 0) {
                    timer_hours--;
                    timer_minutes = 59;
                    timer_seconds = 59;
                }
            } else {
                timer_seconds--;
            }
        }
    }

    // Stopwatch Increment
    if (mode == 2 && stopwatch_running) {
        stopwatch_seconds++;
        if (stopwatch_seconds == 60) {
            stopwatch_seconds = 0;
            stopwatch_minutes++;
            if (stopwatch_minutes == 60) {
                stopwatch_minutes = 0;
                stopwatch_hours = (stopwatch_hours + 1) % 24;
            }
        }
    }
}

void displayTime();
void setBCD(int value);
void checkButtons();

int main() {
    setup();
    while (1) {
        checkButtons();
        displayTime();
    }
}

// Function to display time on a 6-digit 7-segment display
```

```
void displayTime() {
    int digits[6];

    if (mode == 0) { // Clock Mode
        digits[0] = hours / 10;
        digits[1] = hours % 10;
        digits[2] = minutes / 10;
        digits[3] = minutes % 10;
        digits[4] = seconds / 10;
        digits[5] = seconds % 10;
    } else if (mode == 1) { // Timer Mode
        digits[0] = timer_hours / 10;
        digits[1] = timer_hours % 10;
        digits[2] = timer_minutes / 10;
        digits[3] = timer_minutes % 10;
        digits[4] = timer_seconds / 10;
        digits[5] = timer_seconds % 10;
    } else { // Stopwatch Mode
        digits[0] = stopwatch_hours / 10;
        digits[1] = stopwatch_hours % 10;
        digits[2] = stopwatch_minutes / 10;
        digits[3] = stopwatch_minutes % 10;
        digits[4] = stopwatch_seconds / 10;
        digits[5] = stopwatch_seconds % 10;
    }

    // Multiplex 7-segment display
    for (int i = 0; i < 6; i++) {
        setBCD(digits[i]); // Send the BCD value first
        COMMON_PORT = (1 << i); // Enable the corresponding digit
        _delay_us(500); // Short delay for smooth display
    }
}
// Function to set BCD output for 7-segment display
void setBCD(int value) {
    BCD_PORT = (BCD_PORT & ~BCD_MASK) | ((value << 2) & BCD_MASK);
}

// Function to check button inputs and update mode/settings
void checkButtons() {
    if (!(PIND & (1 << PD6))) {
        _delay_ms(50);
        if (!(PIND & (1 << PD6))) {
            if (mode == 0) {
                hours = (hours + 1) % 24;
                seconds = 0;
            } else if (mode == 1) {
                timer_hours = (timer_hours + 1) % 24;
                seconds = 0;
```

```
            }
            while (!(PIND & (1 << PD6))); // Wait for release
        }
    }

    if (!(PIND & (1 << PD7))) {
        _delay_ms(50);
        if (!(PIND & (1 << PD7))) {
            if (mode == 0) {
                minutes = (minutes + 1) % 60;
                seconds = 0;
            } else if (mode == 1) {
                timer_minutes = (timer_minutes + 1) % 60;
                seconds = 0;
            }
            while (!(PIND & (1 << PD7))); // Wait for release
        }
    }

    if (!(PINB & (1 << MODE_BUTTON))) {
        _delay_ms(50);
        if (!(PINB & (1 << MODE_BUTTON))) {
            mode = (mode + 1) % 3; // Cycle through Clock, Timer, and
    Stopwatch
            while (!(PINB & (1 << MODE_BUTTON))); // Wait for release
        }
    }

    // Modified section: Stopwatch button controls both Timer and
    Stopwatch
    if (!(PINB & (1 << STOPWATCH_BUTTON))) {
        _delay_ms(50);
        if (!(PINB & (1 << STOPWATCH_BUTTON))) {
            if (mode == 2) {  // Toggle Stopwatch running
                stopwatch_running = !stopwatch_running;
            } else if (mode == 1) {  // Toggle Timer running
                stopwatch_running = !stopwatch_running;  // Reuse the same
     flag
            }
            while (!(PINB & (1 << STOPWATCH_BUTTON))) {
                _delay_ms(10);
            }
        }
    }
}
```

Listing 1: 'Code for Digital Clock'

6 Conclusion

The project successfully demonstrated a digital clock using Arduino, 7447 decoder, and 7-segment displays. The clock functions accurately with manual control options, showcasing **real-time updating, display multiplexing**.