Digital Clock Using Arduino and 7-Segment Displays

DHWANITH M DODDAHUNDI - EE24BTECH11016

Objective

The purpose of this project is to design and build a digital clock using:

- **Arduino Uno** as the core processing unit.
- **7447 BCD to 7-segment decoder** to simplify the control of the 7-segment displays.
- **Six 7-segment displays** to represent hours, minutes, and seconds.
- **Push buttons** for manual adjustments.

1 Overview

The digital clock is based on a multiplexing technique to reduce the required output pins from the Arduino. The time is displayed across six 7-segment displays, which are controlled by a BCD to 7-segment decoder.

2 Theory and Design Principles

2.1 Multiplexing Concept

Since the Arduino has a limited number of output pins, directly driving six 7-segment displays would be inefficient. Instead, multiplexing is used:

- The Arduino activates one display at a time, while sending the digit data.
- This creates the illusion that all digits are continuously illuminated.
- This technique reduces the required pins by sharing BCD lines across multiple displays.

2.2 7447 BCD to 7-Segment Decoder

The 7447 decoder translates a 4-bit binary-coded decimal (BCD) input into signals for controlling a 7-segment display. This minimizes the number of Arduino pins required for display control.

2.3 Time Management with Arduino

The clock uses the `millis()` function to manage timekeeping, which tracks milliseconds since the program started. By counting 1000 milliseconds, the clock increments the seconds.

3 Components and Circuit Connections

3.1 Hardware Requirements

- Arduino Uno
- 7447 BCD to 7-segment decoder
- Six 7-segment displays
- Three push buttons (Hour, Minute, and Reset)
- Resistors
- Breadboard and jumper wires

3.2 Wiring Details

The table below shows the connections between the Arduino and the components:

Component	Arduino Pin	Purpose
BCD Input A	2	7447 BCD input A
BCD Input B	3	7447 BCD input B
BCD Input C	4	7447 BCD input C
BCD Input D	5	7447 BCD input D
7-Segment Digit 1	6	Tens place of Hours
7-Segment Digit 2	7	Units place of Hours
7-Segment Digit 3	8	Tens place of Minutes
7-Segment Digit 4	9	Units place of Minutes
7-Segment Digit 5	10	Tens place of Seconds
-Segment Digit 6	11	Units place of Seconds
Hour Button	12	Increases the hour value
Minute Button	13	Increases the minute value
Reset Button	A0	Resets the clock

4 Arduino Code Implementation

The following Arduino code manages the clock's operation, including time updates, display multiplexing, and button handling.

```
// Digital Clock with Arduino and 7-Segment Displays
#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) {
```

```c
                    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: Arduino Code for Digital Clock

5 Conclusion

This project successfully demonstrates a functional digital clock with real-time updates and display multiplexing. The implementation uses an Arduino Uno, a 7447 decoder, and 7-segment displays, making it efficient and easy to operate.