Digital Clock Implementation Report

EE24BTECH11033 - Kolluru Suraj

1 Introduction

This report documents the implementation of a **6-digit 7-segment digital clock** using:

- ATmega328P microcontroller (Arduino Uno)
- Hardware BCD encoding via direct GPIO
- Multiplexed common-anode display
- Two-button time adjustment interface

Key features include:

- Precise 1-second interrupt timing
- HH:MM:SS time display format
- Debounced button inputs for time adjustment

2 Hardware Design

2.1 Component Connections

TABLE I: Pin Configuration

Signal	**MCU Pin**	**Description**
BCD Data	PD2-PD5	4-bit BCD output (bits 0-3)
Digit Select	PC0-PC5	Common anode control
Hour Button	PD6	Active-low input with pull-up
Minute Button	PD7	Active-low input with pull-up

2.2 Display Interface

- **BCD Encoding:**
  - Direct GPIO output (PD2-PD5)
  - No external decoder IC used
- **Multiplexing:**
  - 6 digits controlled via PC0-PC5
  - 500 μs display time per digit
  - Full refresh rate ≈ 83Hz

3 Software Implementation

3.1 Display Driver

- **Multiplexing:** Rapidly cycles through digits
- **BCD Conversion:** Uses bitmasking on PORTD
- **Timing:** 500µs per digit for stable display

```
void displayTime() {
    int digits[6] = {
        hours/10, hours%10,
        minutes/10, minutes%10,
        seconds/10, seconds%10
    };

    for (int i=0; i<6; i++) {
        PORTD = (PORTD & ~0b00111100) | ((digits[i] << 2) & 0b00111100);
        PORTC = (1 << i);
        _delay_us(500);
    }
}
```

Listing 1: Display Function

3.2 Button Handling

- **Debouncing:** 50ms software delay
- **Functionality:**
  - Hour button: Increments hours (0-23)
  - Minute button: Increments minutes (0-59)
- **Reset Behavior:** Seconds reset to 0 on adjustment

4 Testing & Validation

4.1 Test Cases

TABLE II: Verification Results

Test	Procedure	Result
Time Accuracy	60-minute continuous run	±1 second drift
Button Response	Rapid button presses	Clean increments
Display Stability	Visual inspection	No flickering
Current Draw	5V supply measurement	85mA (all segments)

4.2 Optimizations

- Reduced ISR overhead by minimizing operations
- Optimized BCD output using bitmasking
- Balanced display brightness vs. refresh rate

5 Conclusion

The implemented digital clock demonstrates:

- Reliable timekeeping through interrupt-driven design
- Efficient GPIO utilization for display control
- Responsive user interface with debounced inputs

```
#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 HOUR_BUTTON PD6
#define MINUTE_BUTTON PD7

volatile int seconds = 0, minutes = 30, hours = 15;

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 << HOUR_BUTTON) | (1 << MINUTE_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
    seconds++;
    if (seconds == 60) {
        seconds = 0;
        minutes++;
        if (minutes == 60) {
            minutes = 0;
            hours = (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];

    digits[0] = hours / 10;
    digits[1] = hours % 10;
    digits[2] = minutes / 10;
    digits[3] = minutes % 10;
    digits[4] = seconds / 10;
    digits[5] = 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 time
void checkButtons() {
    if (!(PIND & (1 << HOUR_BUTTON))) {
        _delay_ms(50);
        if (!(PIND & (1 << HOUR_BUTTON))) {
            hours = (hours + 1) % 24;
            seconds = 0;
            while (!(PIND & (1 << HOUR_BUTTON))); // Wait for release
        }
    }

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

```
}
```

Listing 2: Complete Source Code

Remark: *Code Reference from Rongali Charan -EE24BTECH11052*