Digital Clock

EE24BTECH11025 - GEEDI HARSHA

1 Introduction

In this experiment, we designed and implemented a digital clock using six seven-segment displays, an ATmega Arduino, and the IC 7447 decoder. The system supports three modes: Clock, Timer, and Stopwatch. The seven-segment displays are controlled using the IC 7447, which converts binary-coded decimal (BCD) inputs into signals that drive the displays. The Arduino manages the timing, mode switching, and display updates. The project demonstrates the use of interrupts, button inputs, and multiplexing for efficient display control.

2 Components Used

The following components were used in this experiment:

- 6 Seven-Segment Displays (Common Cathode)
- 6 Resistors (220 Ω)
- Wires and Jumper Wires
- 1 ATmega Arduino
- 1 IC 7447 (BCD to Seven-Segment Decoder)
- 1 Breadboard
- Power Source

3 Circuit Design

The circuit was designed as follows:

4 Procedure

1. Hardware Setup

- Connect common-anode 7-segment displays to ATmega328P via breadboard.
- Assign segment lines: `PD2-PD7, PB0` (a-g).
- Assign digit select lines: `A0-A2 (PC0-PC2), 10-12 (PB2-PB4).`
- Connect push buttons to `PB1, PB5, PC3-PC5` for input control.
- Any ports can be used if corresponding code changes are made.

2. Software Implementation

- `seg_map`: Lookup table for digit display.
- `set_segments`: Lights up correct segments.
- `enable_digit`: Disables all digits before enabling the correct one.
- `ISR(TIMER1_COMPA_vect)`: Increments time and handles rollovers.
- `check_buttons`: Enables time adjustments when `set_mode = 1`.
- `main`: Configures pins, enables pull-ups, sets 1s interrupts, and runs multiplexing loop.

7447	$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$	$\bar{e}$	$\bar{f}$	$\bar{g}$
Display	a	b	c	d	e	f	g

7447	D	C	B	A
Arduino	5	4	3	2

1. *Power Supply*
   - Arduino `5V` pin connects to V_{CC} of 7447 IC.
   - Common ground between Arduino and 7447.

2. *Display Interface*
   - COM pins of 7-segment displays connect to `220`Ω resistors, then to Arduino analog pins.

Push Button Functions

Four tactile switches are connected to digital pins `6--9`:

Button	Pin	Function
1	6	Hour adjust (`0` to `23`, then resets).
2	7	Minute adjust (`0` to `59`, then resets).
3	8	Mode toggle (switches between clock and stopwatch).
4	9	Pause/resume (halts clock or stopwatch).

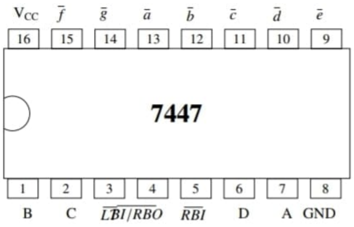

Fig. 3.1: 7447 IC

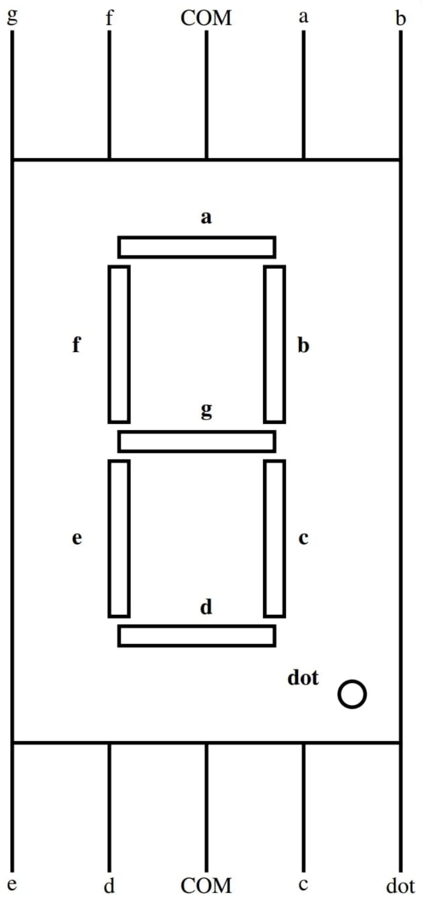

Fig. 2.2: Seven Segment pins

5 Code

The Arduino code for controlling the digital clock, timer, and stopwatch is provided below:

Listing 1: Arduino Code for Digital Clock, Timer, and Stopwatch

```
#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);
            }
        }
    }
}
```

6 Code Explanation

The code is divided into several key sections, each handling a specific functionality of the digital clock, timer, and stopwatch.

6.1 BCD and Digit Selection

- The BCD output is generated using the `setBCD()` function, which takes a digit (0-9) and sets the corresponding BCD bits on PORTD (PD2-PD5).
- The `displayTime()` function splits the time into individual digits and uses multiplexing to display them on the six seven-segment displays. Each digit is displayed for a short duration (`_delay_us(500)`) to create the illusion of simultaneous display.

6.2 Button Configuration

- Two buttons are used: one for mode selection (`MODE_BUTTON`) and one for controlling the stopwatch/timer (`STOPWATCH_BUTTON`).
- The `checkButtons()` function debounces the buttons and handles their actions:
  - `MODE_BUTTON` cycles through the three modes: Clock, Timer, and Stopwatch.
  - `STOPWATCH_BUTTON` starts/stops the stopwatch or timer depending on the current mode.

6.3 Time Tracking Variables

- Three sets of variables track time for the clock, timer, and stopwatch:
  - `hours`, `minutes`, `seconds` for the clock.
  - `timer_hours`, `timer_minutes`, `timer_seconds` for the timer.
  - `stopwatch_hours`, `stopwatch_minutes`, `stopwatch_seconds` for the stopwatch.
- The `mode` variable determines which set of variables is currently active.

6.4 Interrupt-Driven Time Updates

The Timer1 interrupt is configured to trigger every second. The ISR (`TIMER1_COMPA_vect`) handles time updates based on the current mode:

- **Clock Mode:** Increments seconds, minutes, and hours, resetting them as necessary.
- **Timer Mode:** Decrements the timer variables, stopping when all reach zero.
- **Stopwatch Mode:** Increments the stopwatch variables, similar to the clock.

7 Results

The digital clock successfully displayed the time in the format HH:MM:SS. The system seamlessly switched between Clock, Timer, and Stopwatch modes using the buttons. The seven-segment displays were updated every second, and the circuit operated as expected. The resistors ensured that the displays were not damaged by excessive current.

8 Conclusion

This experiment demonstrated the use of seven-segment displays, the IC 7447 decoder, and an ATmega Arduino to create a functional digital clock with additional timer and stopwatch features. The project provided valuable insights into BCD encoding, multiplexing, interrupt handling, and button input processing. Future improvements could include adding a real-time clock (RTC) module for more accurate timekeeping and implementing additional features like alarms or date display.