

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
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:

```

1 #define F_CPU 16000000UL
2 #include <avr/io.h>
3 #include <util/delay.h>
4 #include <avr/interrupt.h>
5
6 #define BCD_PORT PORTD
7 #define BCD_DDR DDRD
8 #define BCD_MASK 0b00111100 // PD2 to PD5
9
10 #define COMMON_PORT PORTC
11 #define COMMON_DDR DDRC

```

```

12
13 #define MODE_BUTTON PB0 // Switch between Clock, Timer, and Stopwatch
14 #define STOPWATCH_BUTTON PB1 // Start/Stop Stopwatch and Timer
15
16 volatile int seconds = 0, minutes = 30, hours = 15;
17 volatile int timer_seconds = 0, timer_minutes = 0, timer_hours = 0;
18 volatile int stopwatch_seconds = 0, stopwatch_minutes = 0, stopwatch_hours
    = 0;
19 volatile int mode = 0; // 0 = Clock, 1 = Timer, 2 = Stopwatch
20 volatile int stopwatch_running = 0; // 1 = Running, 0 = Stopped
21
22 void setup() {
23     // Set BCD display pins (PD2-PD5) as output
24     BCD_DDR |= BCD_MASK;
25     BCD_PORT &= ~BCD_MASK;
26
27     // Set digit selector pins (PORTC) as output
28     COMMON_DDR = 0xFF;
29     COMMON_PORT = 0x00;
30
31     // Enable pull-up resistors for buttons
32     PORTD |= (1 << PD6) | (1 << PD7);
33     PORTB |= (1 << MODE_BUTTON) | (1 << STOPWATCH_BUTTON);
34
35     // Timer1 Setup: CTC Mode, 1-second interval
36     TCCR1B |= (1 << WGM12) | (1 << CS12) | (1 << CS10);
37     OCR1A = 15625; // 1-second interrupt
38     TIMSK1 |= (1 << OCIE1A);
39
40     // Debug LED on PC7 (Bit 7 of PORTC) to check if ISR is running
41     DDRC |= (1 << 7); // Set PC7 as output
42     PORTC &= ~(1 << 7); // Initially turn it off
43
44     sei(); // Enable global interrupts
45 }
46
47 ISR(TIMER1_COMPA_vect) {
48     PORTC ^= (1 << 7); // Toggle PC7 to check ISR is running
49
50     // Clock Mode Updates
51     if (mode == 0) {
52         seconds++;
53         if (seconds == 60) {
54             seconds = 0;
55             minutes++;
56             if (minutes == 60) {
57                 minutes = 0;
58                 hours = (hours + 1) % 24;
59             }

```

```

60     }
61 }
62
63 // Timer Countdown (only when running)
64 if (mode == 1 && stopwatch_running) {
65     if (timer_seconds > 0 timer_minutes > 0 timer_hours > 0) {
66         if (timer_seconds == 0) {
67             if (timer_minutes > 0) {
68                 timer_minutes--;
69                 timer_seconds = 59;
70             } else if (timer_hours > 0) {
71                 timer_hours--;
72                 timer_minutes = 59;
73                 timer_seconds = 59;
74             }
75         } else {
76             timer_seconds--;
77         }
78     }
79 }
80
81 // Stopwatch Increment
82 if (mode == 2 && stopwatch_running) {
83     stopwatch_seconds++;
84     if (stopwatch_seconds == 60) {
85         stopwatch_seconds = 0;
86         stopwatch_minutes++;
87         if (stopwatch_minutes == 60) {
88             stopwatch_minutes = 0;
89             stopwatch_hours = (stopwatch_hours + 1) % 24;
90         }
91     }
92 }
93 }
94
95 void displayTime();
96 void setBCD(int value);
97 void checkButtons();
98
99 int main() {
100     setup();
101     while (1) {
102         checkButtons();
103         displayTime();
104     }
105 }
106
107 // Function to display time on a 6-digit 7-segment display
108 void displayTime() {

```

```

109     int digits[6];
110
111     if (mode == 0) { // Clock Mode
112         digits[0] = hours / 10;
113         digits[1] = hours % 10;
114         digits[2] = minutes / 10;
115         digits[3] = minutes % 10;
116         digits[4] = seconds / 10;
117         digits[5] = seconds % 10;
118     } else if (mode == 1) { // Timer Mode
119         digits[0] = timer_hours / 10;
120         digits[1] = timer_hours % 10;
121         digits[2] = timer_minutes / 10;
122         digits[3] = timer_minutes % 10;
123         digits[4] = timer_seconds / 10;
124         digits[5] = timer_seconds % 10;
125     } else { // Stopwatch Mode
126         digits[0] = stopwatch_hours / 10;
127         digits[1] = stopwatch_hours % 10;
128         digits[2] = stopwatch_minutes / 10;
129         digits[3] = stopwatch_minutes % 10;
130         digits[4] = stopwatch_seconds / 10;
131         digits[5] = stopwatch_seconds % 10;
132     }
133
134     // Multiplex 7-segment display
135     for (int i = 0; i < 6; i++) {
136         setBCD(digits[i]); // Send the BCD value first
137         COMMON_PORT = (1 << i); // Enable the corresponding digit
138         _delay_us(500); // Short delay for smooth display
139     }
140 }
141 // Function to set BCD output for 7-segment display
142 void setBCD(int value) {
143     BCD_PORT = (BCD_PORT & ~BCD_MASK) | ((value << 2) & BCD_MASK);
144 }
145
146 // Function to check button inputs and update mode/settings
147 void checkButtons() {
148     if (!(PIND & (1 << PD6))) {
149         _delay_ms(50);
150         if (!(PIND & (1 << PD6))) {
151             if (mode == 0) {
152                 hours = (hours + 1) % 24;
153                 seconds = 0;
154             } else if (mode == 1) {
155                 timer_hours = (timer_hours + 1) % 24;
156                 seconds = 0;
157             }

```

```

158         while (!(PIND & (1 << PD6))); // Wait for release
159     }
160 }
161
162 if (!(PIND & (1 << PD7))) {
163     _delay_ms(50);
164     if (!(PIND & (1 << PD7))) {
165         if (mode == 0) {
166             minutes = (minutes + 1) % 60;
167             seconds = 0;
168         } else if (mode == 1) {
169             timer_minutes = (timer_minutes + 1) % 60;
170             seconds = 0;
171         }
172         while (!(PIND & (1 << PD7))); // Wait for release
173     }
174 }
175
176 if (!(PINB & (1 << MODE_BUTTON))) {
177     _delay_ms(50);
178     if (!(PINB & (1 << MODE_BUTTON))) {
179         mode = (mode + 1) % 3; // Cycle through Clock, Timer, and
180 Stopwatch
181         while (!(PINB & (1 << MODE_BUTTON))); // Wait for release
182     }
183 }
184
185 // Modified section: Stopwatch button controls both Timer and
186 Stopwatch
187 if (!(PINB & (1 << STOPWATCH_BUTTON))) {
188     _delay_ms(50);
189     if (!(PINB & (1 << STOPWATCH_BUTTON))) {
190         if (mode == 2) { // Toggle Stopwatch running
191             stopwatch_running = !stopwatch_running;
192         } else if (mode == 1) { // Toggle Timer running
193             stopwatch_running = !stopwatch_running; // Reuse the same
194 flag
195         }
196         while (!(PINB & (1 << STOPWATCH_BUTTON))) {
197             _delay_ms(10);
198         }
199     }
200 }

```

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**.