

DIGITAL CLOCK

EE24BTECH11038 - M.B.S Aravind

March 24, 2025

Contents

1	Introduction	3
2	Hardware Components	3
3	LCD	3
4	Features of 16x2 LCD	4
5	Pin Configuration	4
6	Circuit Diagram	4
7	Working Principle	5
8	Hardware Configuration	5
8.1	LCD Connection	5
8.2	Button Connections	6
9	Software Design	7
9.1	Software Architecture	7
9.2	Key Data Structures	7
9.3	Expression Evaluation	8
9.4	User Interface Design	8
10	Implementation Challenges	8
10.1	Limited Memory	8
10.2	Expression Parsing	9
10.3	User Interface Limitations	9
11	Software Functional Blocks	9
11.1	LCD Control Functions	9
11.2	Button Interface Functions	10
11.3	Expression Handling Functions	10
11.4	Calculation Functions	10

12 Testing Results	10
12.1 Arithmetic Operations	10
12.2 Trigonometric Functions	11
13 Limitations and Future Improvements	11
13.1 Current Limitations	11
13.2 Potential Improvements	11
14 Conclusion	12
A Code Highlights	12
A.1 Expression Evaluation Function	12
B Pin Assignment Table	12

1 Required Components

- Breadboard
- Arduino UNO (or ATmega328P Microcontroller)
- Jumper Cables
- 6x Seven-Segment Displays
- 7447 BCD Decoder
- Resistors
- Push Buttons

2 Hardware Connections

Component	ATmega328P Pin	Connection Description
BCD Input A	Digital Pin 2	Connected to 7447 A input
BCD Input B	Digital Pin 3	Connected to 7447 B input
BCD Input C	Digital Pin 4	Connected to 7447 C input
BCD Input D	Digital Pin 5	Connected to 7447 D input
Common Anode Pins	PORTC Analog Pins	Control individual display digits
Mode Button	PB0	Switch between Clock, Timer, Stopwatch
Start/Stop Button	PB1	Control mode-specific functions

Table 1: Hardware Connections

3 Working Explanation

3.1 Initialization of I/O and Timer

The AVR microcontroller initializes critical system components:

- Configures BCD output pins for 7-segment display control.
- Sets up Timer1 for interrupt-driven time updates.
- Enables pull-up resistors for button inputs.
- Initializes global interrupt mechanism.

3.2 Displaying Time Using Multiplexing

The clock implements efficient display rendering:

- Extracts individual digits for hours, minutes, and seconds.
- Uses Binary-Coded Decimal (BCD) encoding.
- Activates one display digit at a time.
- Rapidly switches between digits to create a persistent vision effect.
- Minimizes I/O pin usage through sequential activation.

3.3 Time Keeping and Increment Logic

Time management follows precise rules:

- Seconds increment every interrupt cycle.
- Automatic rollover for seconds ($60 \rightarrow 00$).
- Minute increment when seconds reach 60.
- Hour increment when minutes reach 60.
- 24-hour cycle completion with hour reset.

3.4 Timer1 Interrupt for Precise Timing

Implemented using Clear Timer on Compare Match (CTC) Mode:

- 1-second interrupt generation.
- Precise time tracking independent of the main loop.
- Automatic time progression.
- Modulo arithmetic for time rollover.

3.5 Main Loop Execution

The main function provides:

- Continuous display refresh.
- Non-blocking operation.
- Smooth time update mechanism.
- Potential for future feature expansion.

4 Code Architecture

4.1 Key Design Characteristics

- Interrupt-driven time management.
- Efficient memory utilization.
- Modular mode switching.
- Debounced button handling.

4.2 Interrupt Service Routine (ISR) Features

- Precise 1-second time incrementation.
- Cascading time update logic.
- Automatic carry propagation.
- Mode-specific time tracking.

5 Conclusion

The implemented digital clock demonstrates:

- Efficient microcontroller-based timekeeping.
- Flexible multi-mode functionality.
- Robust interrupt-driven design.
- Scalable embedded system architecture.

Potential future improvements:

- Real-Time Clock (RTC) module integration.
- Battery backup implementation.
- Enhanced user interface.
- Additional mode functionalities.

6 References

- Code by rongali charan
- AI suggestions