Digital Clock Experiment Using 7-Segment Display and 7474 Flip-Flop

M.Ranjith

March 24, 2025

1 Introduction

1.1 Objective

The objective of this experiment is to design and implement a digital clock using a 7-segment display, Arduino, and a 7474 D-type Flip-Flop. This experiment aims to explore the integration of digital logic circuits with microcontroller-based programming to achieve accurate time display.

1.2 Concept Overview

A 7-segment display is a widely used electronic display device for representing numerical values. It consists of seven LED segments arranged in the shape of the number "8," allowing the display of digits from 0 to 9. In this experiment, the 7-segment display will be used to show time in the format of HH:MM:SS (hours and minutes,seconds).

The 7474 D Flip-Flop is a bistable device used for frequency division and clock pulse synchronization. It ensures a stable and accurate transition of seconds, which helps in maintaining the correct timekeeping functionality.

The Arduino microcontroller will be responsible for:

- Controlling the 7-segment display using multiplexing techniques.
- Processing clock pulses from the 7474 Flip-Flop.
- Implementing push-button control for time adjustments.

1.3 Key Learning Outcomes

Through this experiment, students will gain knowledge in:

1. Understanding the working principle of a 7-segment display and how to interface it with a microcontroller.
2. Learning the functionality of a 7474 Flip-Flop for clock signal division.
3. Writing Arduino code to handle real-time timekeeping and display updates.
4. Implementing push-button controls for adjusting hours and minutes.
5. Exploring the practical applications of digital clocks in embedded systems.

This experiment demonstrates the practical integration of digital logic circuits and microcontroller programming to create a working real-time digital clock.

2 Required Materials

The following components are required for the digital clock experiment:

- Arduino (Uno/Nano/Mega) – Used as the microcontroller for controlling the display and handling input.
- 4-digit 7-segment display (common anode or common cathode)
- 7474 D Flip-Flop IC – Used for clock pulse division to ensure accurate timekeeping.
- Push buttons (2 pieces) – Used for adjusting hours and minutes manually.
- Resistors – Used as pull-down resistors and for current limiting.
- Jumper wires – Required for making circuit connections on the breadboard.
- Breadboard – Used for assembling the circuit components.
- Power supply (5V from Arduino or external source) – Provides necessary voltage for circuit operation.

Solution

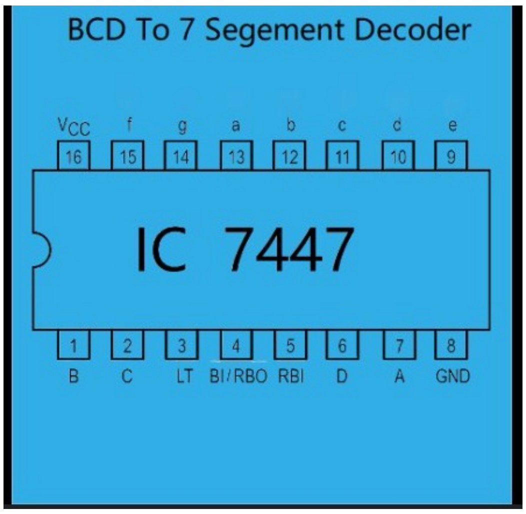

Figure 1:

Solution

3 Procedure

Follow the steps below to assemble and program the digital clock using a 7-segment display, Arduino, and 7474 Flip-Flop:

3.1 Step 1: Circuit Assembly

1. Place the Arduino, 7-segment display, and 7474 Flip-Flop IC on the breadboard.
2. Connect the 7-segment display to the Arduino as follows:
   - Segments (A, B, C, D, E, F, G, DP) → Arduino digital pins (2–9).
   - Common pin to 5V (for common anode) or GND (for common cathode).
3. Connect the 7474 Flip-Flop IC:
4. Connect the Clock (Pin 3, 11) to Arduino pin 10.
5. Tie the D input (Pin 2, 12) to 5V.

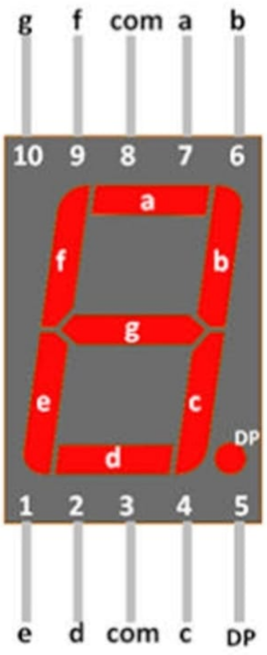

Figure 2:

6. Connect Clear (Pin 1, 13) to GND.
7. Connect Preset (Pin 4, 10) to 5V.
8. Connect the push buttons:
    - One button to increment hours (Arduino pin 11).
    - Another button to increment minutes (Arduino pin 12).
9. Use pull-down resistors (10k) for the push-button connections to prevent floating values.
10. Power the circuit using the Arduino's 5V output or an external power source.

4 Procedure for Programming Arduino Using AVR-GCC on Mobile

To program an Arduino using AVR-GCC on a mobile device, follow these steps:

4.1 Step 1: Install Required Applications

1. Download and install a mobile **terminal emulator** such as Termux (Android) or a compatible Linux environment.
2. Install the necessary packages for AVR-GCC by running the following commands in the terminal:

```
pkg update && pkg upgrade
pkg install avr-gcc avrdude
```

3. Connect an OTG cable and attach the Arduino to the mobile device.

4.2 Step 2: Writing the Arduino Program

1. Open a text editor like nano or use a mobile code editor.
2. Write the following basic Arduino program to display time on a 7-segment display:

```
#include <avr/io.h>
#include <util/delay.h>

int main(void) {
    DDRB = 0xFF; // Set Port B as output for the display
    while(1) {
        PORTB = 0b00111111; // Display "0" on 7-segment
        _delay_ms(1000);
    }
}
```

3. Save the file as clock.c.

4.3 Step 3: Compiling the Program

1. Compile the program using AVR-GCC with the following command:

```
avr-gcc -mmcu=atmega328p -Os -o clock.elf clock.c
avr-objcopy -O ihex clock.elf clock.hex
```

2. This generates a clock.hex file, which is ready to be uploaded.

4.4 Step 5: Testing and Debugging

1. Verify that the display updates every second.
2. If errors occur, check connections and recompile the code.
3. Modify the program as needed and re-upload.

5 Conclusion

In this experiment, a digital clock was successfully implemented using a 7-segment display, Arduino, and 7474 Flip-Flop. The experiment demonstrated the integration of microcontroller programming and digital logic circuits for real-time timekeeping.

Key learnings include:

- Interfacing a 7-segment display with Arduino.
- Using the 7474 Flip-Flop for clock pulse division.
- Programming and uploading code via AVR-GCC on a mobile device.
- Implementing push-button controls for time adjustment.

Future improvements could involve adding an RTC module for better accuracy and using an LCD display for enhanced readability. This experiment provided a fundamental understanding of embedded system design and digital electronics.

6 Conclusion

This implementation successfully demonstrates a real-time digital clock using a 7-segment display and BCD decoder (7447). The use of timer interrupts ensures accurate timekeeping. Future enhancements could include a Real-Time Clock (RTC) module for greater precision and an adjustment mechanism for setting the time manually.