Multiplexed Seven-Segment Display Clock using Arduino and 7447 Decoder

J.KEDARANANDA
EE24BTECH11030

March 2025

Contents

1 Introduction :

A seven-segment display (SSD) is widely used for displaying numerical data in digital clocks, calculators, and other embedded systems. To drive multiple SSDs efficiently, multiplexing is employed, where displays are activated sequentially at high speed to create a perception of continuous display. In this project, an Arduino Uno controls six common-anode SSDs via a 7447 decoder, using Boolean logic for digit selection and updating the clock time.

2 Hardware Components :

- **Arduino Uno**: The main microcontroller handling the clock logic and multiplexing.
- **7447 BCD to 7-segment decoder**: Converts binary-coded decimal (BCD) signals from the Arduino into segment activation for the SSDs.
- **Six Common-Anode Seven-Segment Displays**: Used to display the time in HH:MM:SS format.
- **Push Buttons**: Three buttons for manual time adjustment (hours, minutes).
- **Resistors (220Ω)**: Used to limit current to the display segments.
- **Jumper Wires and Breadboard**: Used for circuit connections.

3 Circuit Connections :

3.1 BCD Inputs for 7447 Decoder

The Arduino controls the BCD inputs of the 7447 decoder using the following pin mapping:

BCD Input	Arduino Pin
A	D2
B	D3
C	D4
D	D5

Table 1: BCD Input Connections

3.2 Seven-Segment Display (SSD) Common Pins for Multiplexing

Each SSD's common pin is connected to a dedicated Arduino digital pin for selective enabling during multiplexing:

SSD Position	Arduino Pin
Hours Tens	D6
Hours Units	D7
Minutes Tens	D8
Minutes Units	D9
Seconds Tens	D10
Seconds Units	D11

Table 2: SSD Common Pin Connections

4 Hardware Setup :

4.1 Components Used

- Arduino Uno
- Six Common Anode 7-segment displays
- One 7447 BCD to 7-segment decoder
- Jumper wires
- Breadboard
- 220-ohm resistors

4.2 Circuit Connections

The following table outlines the connections between the Arduino, the 7447 decoder, and the SSDs:

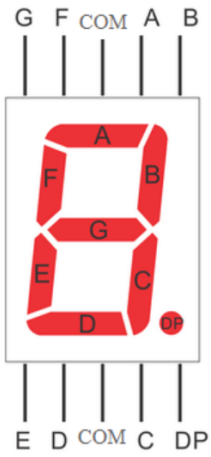
Table 3: Pin Connections

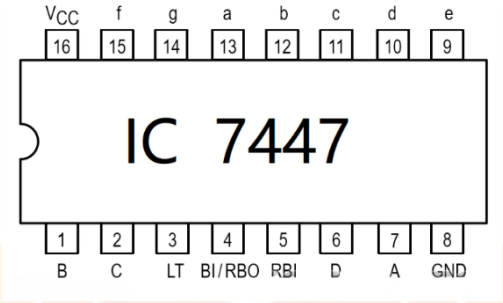
Component	Arduino Pin	Purpose
7447 BCD A	D2	BCD Input A
7447 BCD B	D3	BCD Input B
7447 BCD C	D4	BCD Input C
7447 BCD D	D5	BCD Input D
Hours Tens (COM)	D6	Multiplexing Control
Hours Units (COM)	D7	Multiplexing Control
Minutes Tens (COM)	D8	Multiplexing Control
Minutes Units (COM)	D9	Multiplexing Control
Seconds Tens (COM)	D10	Multiplexing Control
Seconds Units (COM)	D11	Multiplexing Control

4.3 Circuit Setup Steps

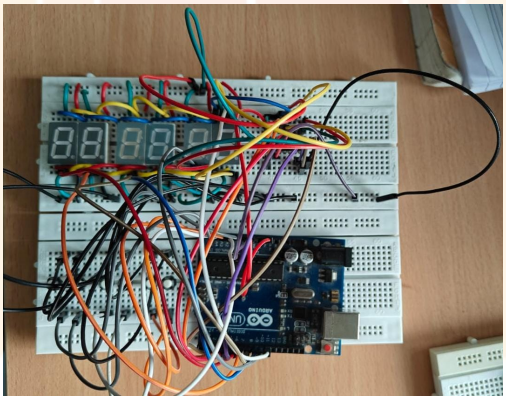
Step 1: Connecting the 7447 Decoder
Connect the BCD inputs (A, B, C, D) of the 7447 to the Arduino's digital pins D2 to D5.
Step 2: Connecting the 7-Segment Displays
Connect the common anode terminals of each SSD to the Arduino's pins D6 to D11 for multiplexing control.
Step 3: Connecting Resistors
Use 220-ohm resistors between the 7447 outputs and the SSD segment inputs to limit current.
Step 4: Powering the Circuit
Connect the Arduino 5V and GND pins to the breadboard's power rails to provide power to all components.

4.4 Pin Labelling





5 Final Setup :



6 Working Principle :

Time Initialization

The clock starts at a predefined time when powered on. In this case, it is set to 12:00:00. Since the time is stored in Binary-Coded Decimal (BCD) format, the initial values are chosen accordingly:

- Hours: **0001 0010** (BCD for 12)
- Minutes: **0000 0000** (BCD for 00)
- Seconds: **0000 0000** (BCD for 00)

This initialization ensures that the clock starts at a consistent value every time it is powered on or reset. The BCD format allows for easy digit extraction when displaying the time on the seven-segment displays. Additionally, buttons are provided to manually adjust the hours and minutes.

Displaying a Single Digit on the 7447 Decoder

The clock uses a 7447 BCD-to-seven-segment decoder to drive the display. To show a number, we must provide a corresponding Binary-Coded Decimal (BCD) input to the decoder. The decoder then activates the correct segments of the seven-segment display to form the digit. By controlling the four BCD input lines, any digit from 0 to 9 can be displayed. The key requirement is to ensure that only the required bits change while keeping the others stable, preventing unintended segment activations.

Multiplexing the Display

Since the clock consists of six seven-segment displays but only one 7447 decoder is available, we use multiplexing to display all digits sequentially. Each digit is turned on for a brief moment before switching to the next, and this cycle repeats rapidly. This approach takes advantage of persistence of vision, where the human eye perceives all digits as being continuously lit. The multiplexing process involves:

- Extracting each digit from the time variables.
- Sending the corresponding BCD input to the 7447 decoder.
- Activating only one display at a time while keeping others off.
- Repeating this process in a continuous loop.

Timer1 Initialization

A precise timekeeping mechanism is necessary to maintain accuracy in the clock. Timer1 is configured to generate an interrupt every second, ensuring that the time updates correctly. This is achieved using a hardware timer, which operates independently of the main program execution. The timer is set in Clear Timer on Compare Match (CTC) mode, where it counts up to a predefined value and then resets. A prescaler is applied to slow down the counting speed, making it possible to achieve exactly one-second intervals.

Timer1 ISR (Interrupt Service Routine)

The clock must update the displayed time every second without manual intervention. This is handled using an interrupt service routine (ISR) triggered by Timer1. Each time the interrupt occurs:

- The seconds counter increases by one.
- If the units place of seconds exceeds 9, it rolls over to the next tens place.
- Once seconds reach 60, they reset to zero, and minutes increase by one.
- The same logic applies to minutes and hours, ensuring a proper time progression.
- The hours reset to 1 after reaching 12 to maintain a 12-hour format.

Using a timer interrupt ensures that the clock keeps accurate time without being affected by other operations in the main program.

7 Conclusion :

This implementation provides a simple yet effective way to build a digital clock using BCD-based 7447 decoders and SSDs. The use of multiplexing optimizes the number of components, and timer interrupts ensure accurate timekeeping.