Clock Project Report

EE224BTECH11044 - Muthyala koushik

1 Introduction

This project implements a digital clock using an Arduino Uno with a single SN7447 BCD-to-7-segment decoder. The clock displays hours, minutes, and seconds on six multiplexed 7-segment displays. Time is maintained in Binary Coded Decimal (BCD) format and updated every second using a Timer1 compare match interrupt.

2 Aim

- Design a digital clock that displays time in `HH:MM:SS` format.
- Use a single SN7447 decoder to drive six 7-segment displays through multiplexing.
- Update the time every second via an interrupt-driven timer.
- Implement BCD arithmetic to handle digit carry-overs across seconds, minutes, and hours.

3 Components Required

- Arduino Uno
- SN7447 BCD-to-7-segment decoder
- 6 common anode (or cathode, as required) 7-segment displays
- Current-limiting resistors (typically 220 Ω per segment)
- External 16 MHz clock source (or use the internal oscillator if applicable)
- Breadboard and connecting wires

4 Hardware Configuration

4.1 Pin Connections

The code defines two groups of connections:

a) **BCD Output to SN7447:** Four AVR pins provide the 4-bit Binary Coded Decimal (BCD) value to the SN7447.

SN7447 Input	Function	AVR Port/Pin	Digital Pin (Uno)
A (LSB)	BCD Data Bit 0	PD2	2
B	BCD Data Bit 1	PD3	3
C	BCD Data Bit 2	PD4	4
D (MSB)	BCD Data Bit 3	PD5	5

b) **Display Multiplex Control:** Six separate AVR pins are used to enable the common pin of each 7-segment display in a multiplexed arrangement.

Display Digit	Function	AVR Port/Pin	Digital Pin (Uno)
Hours Tens (H1)	Digit Enable Control	PD6	6
Hours Units (H2)	Digit Enable Control	PD7	7
Minutes Tens (M1)	Digit Enable Control	PB0	8
Minutes Units (M2)	Digit Enable Control	PB1	9
Seconds Tens (S1)	Digit Enable Control	PB2	10
Seconds Units (S2)	Digit Enable Control	PB3	11

4.2 Segment Wiring

- The outputs of the SN7447 decoder (segments `a` through `g`) are connected in parallel to the corresponding segments on all six 7-segment displays.
- Each segment connection should include a current-limiting resistor.
- The common pins of the displays (anode or cathode, depending on the type) are switched by the multiplex control lines.

4.3 Power Supply

Connection	Notes
VCC	+5V supply to SN7447, displays, and AVR
GND	Common ground for all components

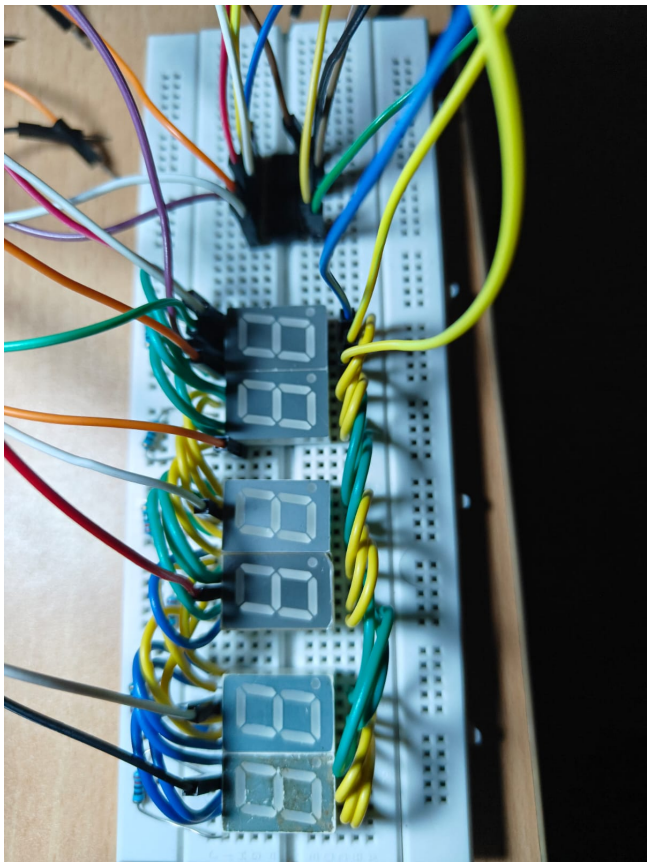

Figure 2: Block diagram of the digital clock system.

5 Software Design

5.1 Overall Architecture

The project firmware is divided into two main modules:

a) **Time Management:**

- Time is stored in three volatile BCD variables: `hours`, `minutes`, and `seconds`.
- A Timer1 Compare Match interrupt (in CTC mode with a 1024 prescaler) updates the time every second.
- BCD arithmetic is used to increment seconds and handle carry-overs for minutes and hours.

b) **Display Multiplexing:**

- The function `displayTime()` extracts individual digits from the BCD time values.
- A single 4-bit BCD digit is sent to the SN7447 via the BCD output pins (PD2–PD5) using the function `displayDigit()`.
- Each digit is displayed in sequence by enabling its corresponding control pin for a brief period (using `_delay_ms()`), creating the illusion of a continuously lit display.

5.2 Interrupt-Driven Time Update

- Timer1 is set to CTC mode with a prescaler of 1024, and the compare register is loaded with a value for a 1-second interval.
- The ISR (`TIMER1_COMPA_vect`) increments the seconds variable in BCD, performs digit carry operations, and updates minutes and hours accordingly.

6 Operation Workflow

1. **Initialization:**
   - Configure BCD output pins (PD2–PD5) and display control pins (PD6, PD7, PB0–PB3) as outputs.
   - Initialize Timer1 and enable global interrupts.
2. **Time Update:**
   - Every second, the ISR updates the time variables using BCD arithmetic and carry management.
3. **Display Refresh:**
   - The main loop continuously calls `displayTime()`, which sequentially enables each display digit.
   - Each digit is shown for a short period, and rapid cycling creates persistence-of-vision.

7 Performance Specifications

- **Voltage:** 5V DC
- **Clock Frequency:** 16 MHz
- **Time Update Accuracy:** 1-second resolution (via Timer1 interrupt)
- **Multiplexing Rate:** Fast enough to achieve persistence-of-vision across 6 digits

8 Code Explanation

The following section explains the main parts of the digital clock code implemented on an Arduino Uno.

8.1 Preprocessor Directives and Includes

- `#define F_CPU 16000000UL`: Sets the CPU clock frequency to 16 MHz. This definition is required by the delay routines from `<util/delay.h>`.
- `#include <avr/io.h>`: Includes the definitions for the input/output registers specific to the Arduino Uno.
- `#include <avr/interrupt.h>`: Provides the interrupt handling functions and macros.
- `#include <util/delay.h>`: Enables the use of `_delay_ms()` for creating precise delays.

8.2 Pin Definitions and Hardware Connections

- **BCD Pins (for SN7447 Decoder):**
  - `A` is defined as `PD2`
  - `B` is defined as `PD3`
  - `C` is defined as `PD4`
  - `D` is defined as `PD5`

  These pins send the 4-bit BCD value (representing a digit) to the SN7447 decoder.
- **Display Control Pins (for Multiplexing):**
  - Hours Tens (H1) is defined as `PD6`
  - Hours Units (H2) is defined as `PD7`
  - Minutes Tens (M1) is defined as `PB0`
  - Minutes Units (M2) is defined as `PB1`
  - Seconds Tens (S1) is defined as `PB2`
  - Seconds Units (S2) is defined as `PB3`

  These control pins are used to selectively enable each of the six 7-segment displays via multiplexing.
- **Clock Variables:** The variables `hours`, `minutes`, and `seconds` are declared as volatile `uint8_t` and stored in BCD format. For example, `hours` is initialized to `0b00010010` (i.e., 12 in BCD).

8.3 Display Functions

- `displayDigit(uint8_t digit)`: This function sends a 4-bit BCD digit to the SN7447 decoder. It modifies `PORTD` by:
  - Clearing the bits corresponding to PD2–PD5 (using a bit mask).
  - Shifting the input digit into the correct bit positions.
- `displayTime()`:
  - Extracts individual BCD digits for hours, minutes, and seconds.
  - For each digit (e.g., tens and units of hours, minutes, seconds), the function:
    1. Enables the appropriate display control pin.
    2. Calls `displayDigit()` to send the digit via the SN7447.

3. Uses `_delay_ms()` for a brief delay to ensure the digit is visible.
4. Disables the display control pin before moving to the next digit.

- This rapid cycling through digits (multiplexing) creates the illusion that all six displays are lit continuously.

8.4 Interrupt Service Routine (ISR)

- `ISR(TIMER1_COMPA_vect)`: This routine is executed once per second, as determined by Timer1.
  - The seconds counter is incremented by one in BCD.
  - A series of conditional checks manage BCD carry:
    - When the lower nibble exceeds 9, it carries over by adding 0x10 (i.e., 16 in decimal) to the upper nibble.
    - If the seconds reach 60 (BCD value `0b01100000`), the seconds reset to 0 and minutes are incremented.
    - Similar checks apply for minutes and hours, with hours resetting after reaching 24 (BCD value `0b00100000`).

8.5 Timer Initialization

- `timer1_init()`:
  - Configures Timer1 in CTC (Clear Timer on Compare) mode.
  - Sets the prescaler to 1024, which divides the 16 MHz clock.
  - Loads the Output Compare Register (`OCR1A`) with the value needed to achieve a 1-second interval.
  - Enables the Timer1 compare interrupt via `TIMSK1`.
  - Calls `sei()` to globally enable interrupts.

8.6 Main Function

- `main()`:
  - Configures the data direction registers:
    - `DDRD` is set for the BCD output pins (PD2–PD5) and the display control pins for hours (PD6 and PD7).
    - `DDRB` is set for the display control pins for minutes and seconds (PB0–PB3).
  - Calls `timer1_init()` to initialize Timer1 and enable interrupts.
  - Enters an infinite loop in which `displayTime()` is continuously called. This ensures the display is refreshed rapidly to maintain the appearance of a continuously lit clock.