Presented to the **Electronics and Computer Engineering**

De La Salle University - Manila

Term 3, A.Y1. **2022-2023**

In partial fulfillment

of the course

In **LBYCPM3** (**EQ1**)

Project: PIC Digital Clock

Submitted by:

Boncodin, Carl Patrick

Chua, Kendrick Dayle J.

Manalansan, Sean Patrick T.

Marcelo, Mariella Anne M.

Mayuga, Zachary Brent L.

Submitted to:

Dr. Argel Bandala

August 15, 2023

I. Objective

1. To create an adjustable digital clock using PIC16F877A and 4-digit 7 segment display.
2. To be able to incorporate 4 buttons that function as Set/ Pause, Select digit. Increment and decrement button.
3. To further enhance the current skill set in regard to PIC and assembly language.

II. Materials Needed

1. Laptop/PC running Windows 10 or higher OS
2. Proteus 8 Professional
3. MikroC Pro for PIC
4. 100 x Jumper Wires
5. 1 x PIC16F877A
6. 4 x Buttons
7. 1 x 5V DC Power Source
8. 1 x 4 Digit 7 Segment Display
9. 1 x 20Mhz Crystal Oscillator
10. 2 x 10pF Capacitors

III. Procedure

For Simulation

1. Interface PIC16F84A with 4 digit 7 segment display. Connect it into the PORT B and PORT C of the microcontroller.
2. Add 4 buttons connected to the PORT D of the PIC microcontroller.
3. Display numbers 0 - 9 in the 4 - digit 7 segment display.
4. Create a condition where the clock ranges from 00:00 to 23:59.
5. Assign functions for the 4 buttons namely:
   Button 1- Pause the clock and resume the clock when pressed again.
   Button 2 - Select whether hours or minutes to be adjusted.
   Button 3 - Increment hour/ minutes.
   Button 4- Decrement hour/ minutes.
6. Import the HEX file to the PIC16F877A simulation.
7. Simulate the circuit in Proteus.
8. Check all functionalities if achieved.

For Hardware Implementation

1. Gather the components that will be used. Ensure that the components match with the components in the simulation.
2. Follow the connections in the simulation and put it in the breadboard.
3. Program the PIC Microcontroller by using PICKit 3.
4. Read the Device.
5. Load the hex file obtained from Mikro C.

6. Write and Verify the hex file to the PIC Microcontroller.
7. Connect the Microcontroller to the circuit created.
8. Turn on the power supply from the PICKit 3.
9. Remove the MCLR connection to the PICKit 3 programmer and connect it to VDD.
10. Check all the buttons and the display.

IV. Data and Results

A. Circuit Diagram and Design

Simulation Specifications:
Oscillator Frequency– 20MHz
PIC Processor Clock Frequency - 20MHz

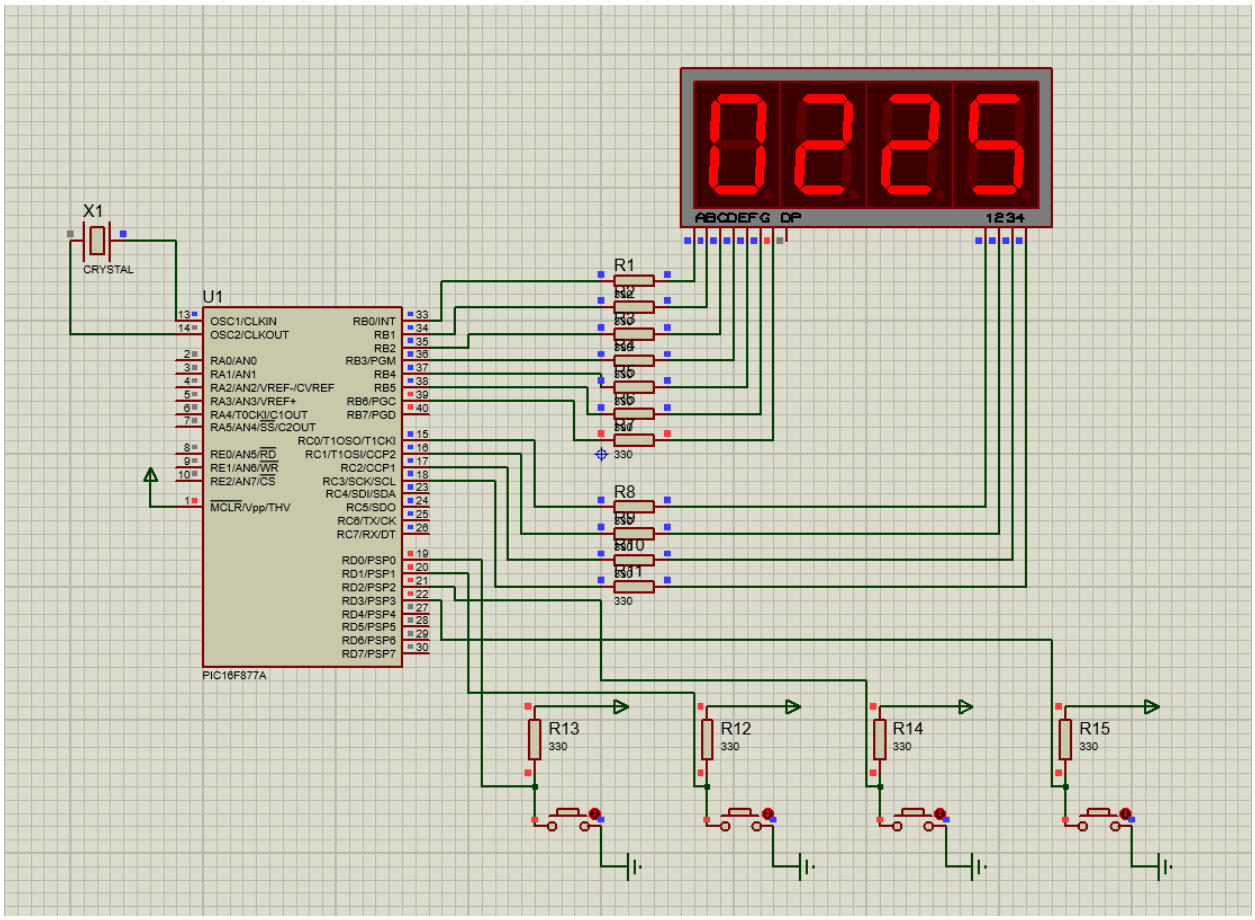

Figure 1. Circuit Diagram

B. Hardware Implementation

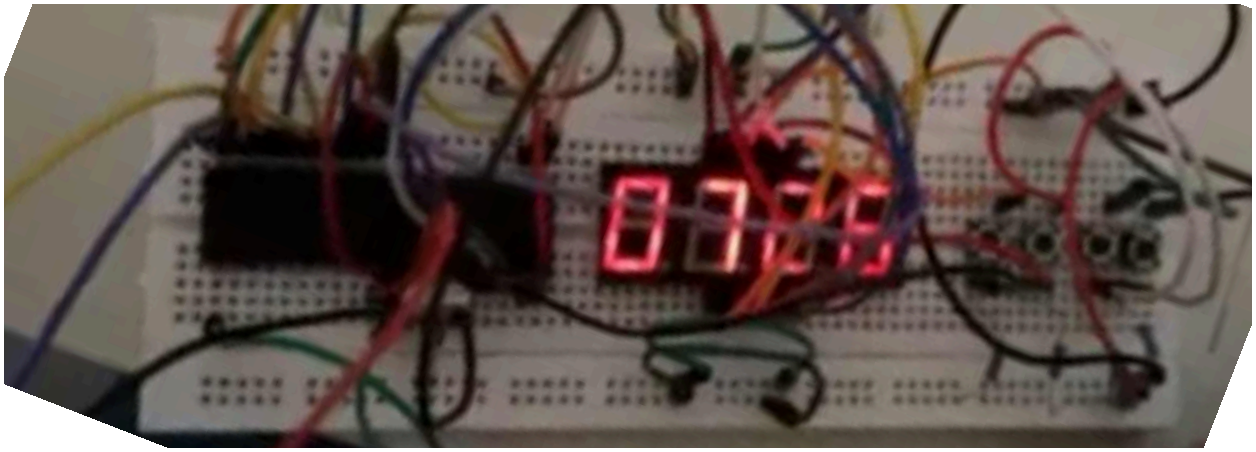

Figure 2. Implementation of the circuit

C. Simulation and Testing

Simulation Video Recording Link:
https://drive.google.com/file/d/10JOPVNyXXw_pbSx24wbqnDEnmmwmhPOP/view?usp=sharing

Hardware Video Recording Link:
https://drive.google.com/file/d/1wj_8OgLX0Si5tIf1AISgijGfFmlP0ibj/view?usp=sharing

D. Code

```
int count[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90};
int d1 = 0;
int d2 = 0;
int d3 = 0;
int d4 = 0;
int start;

int switch1_state = 1;
int switch2_state = 1;
int switch3_state = 1;
int switch4_state = 1;

int clock_enabled = 1;

int is_minutes_selected = 1;

void display() {
   PORTB = count[d1];
   PORTC.B3 = 1;
```

```
    delay_ms(1);
    PORTC.B3 = 0;

    PORTB = count[d2];
    PORTC.B2 = 1;
    delay_ms(1);
    PORTC.B2 = 0;

    PORTB = count[d3];
    PORTC.B1 = 1;
    delay_ms(1);
    PORTC.B1 = 0;

    PORTB = count[d4];
    PORTC.B0 = 1;
    delay_ms(1);
    PORTC.B0 = 0;
}

void clock() {
    if (clock_enabled) {
        d1++;
        if (d1 > 9) {
            d1 = 0;
            d2++;
            if (d2 > 5) {
                d2 = 0;
                d3++;
                if (d3 > 9) {
                    d3 = 0;
                    d4++;
                    if ((d4 == 2 && d3 > 3) || d4 > 2) {
                        d4 = 0;
                    }
                }
            }
        }
    }
}

void main() {
    TRISB = 0x00;
    TRISC = 0x00;
    TRISD = 0xFF;
    PORTB = 0;
    PORTC = 0;

    while (1) {
        clock();
        switch1_state = PORTD.B0;
```

```
switch2_state = PORTD.B1;
switch3_state = PORTD.B2;
switch4_state = PORTD.B3;

if (switch1_state == 0) {
   while (!PORTD.B0) {
   }
   clock_enabled = !clock_enabled;
   start = !start;
}

if (!clock_enabled && switch2_state == 0) {
   while (!PORTD.B1) {
   }
   is_minutes_selected = !is_minutes_selected;
   delay_ms(200);

if (!clock_enabled && switch3_state == 0) {
   while (!PORTD.B2) {
   }

   if (is_minutes_selected) {
      d1++;
      if (d1 == 10) {
         d1 = 0;
         d2++;
         if (d2 == 6) {
            d2 = 0;
         }
      }
   } else {
      d3++;
      if (d3 == 10) {
         d3 = 0;
         d4++;
         if (d4 == 3 && d3 > 2) {
            d4 = 0;
         }
      }
   }
   delay_ms(500);
}

if (!clock_enabled && switch4_state == 0) {
   while (!PORTD.B3) {
   }

   if (is_minutes_selected) {
      if (d1 == 0) {
         if (d2 == 0) {
```

```
                d2 = 5;
                d1 = 9;
            } else {
                d2--;
                d1 = 9;
            }
        } else {
            d1--;
        }
    } else {
        if (d3 == 0) {
            if (d4 == 0) {
                d4 = 2;
                d3 = 3;
            } else {
                d4--;
                d3 = 9;
            }
        } else if (d3 == 3 && d4 == 0 && d1 == 0 && d2 == 0) {
            d3 = 2;
            d4 = 3;
        } else {
            d3--;
        }
    }
    delay_ms(500);
  }

  if (d4 > 2 || (d4 == 2 && d3 > 3)) {
    d4 = 0;
    d3 = 0;
  }
  if (d2 > 5 || (d2 == 5 && d1 > 9)) {
    d2 = 0;
    d1 = 0;
  }

  display(); // Display the current time

  delay_ms(40);
 }
}
```

E. Explanation:

The provided code presents an adjustable clock system that utilizes a microcontroller and 7-segment displays to accurately display and manipulate time. The system allows users to adjust the displayed time, toggle between hours and minutes adjustment modes, and pause or resume the clock's operation. The code is designed to run on a microcontroller, with specific ports configured for input (PORTD) and output (PORTB and PORTC) operations. The microcontroller interacts with four push-button switches (SW1, SW2, SW3, and SW4) connected to PORTD, and it drives a four digit 7-segment display connected to PORTB and PORTC. Each segment of the 7-segment display is represented using hexadecimal values stored in the 'count' array.

The 'clock' function serves as the core of the system's timekeeping mechanism. It runs continuously and simulates the passage of time by sequentially incrementing the variables 'd1', 'd2', 'd3', and 'd4', which represent the display values for minutes (d1 and d2) and hours (d3 and d4). The system's user interface is centered around the four push-button switches and is created within the 'main' function using a series of conditional statements. The 'switch1_state' variable corresponds to SW1, which is used to pause or resume the clock. When SW1 is pressed, the clock's state is toggled, and the clock_enabled variable is updated accordingly.

SW2 controls the selection of adjustment mode (hours or minutes). If the clock is paused (clock_enabled = 0) and SW2 is pressed, the 'is_minutes_selected' flag is toggled. This flag determines whether subsequent adjustments apply to hours or minutes.vSW3 and SW4 handle the increment and decrement operations, respectively, during adjustment modes. When the clock is paused, pressing SW3 or SW4 causes the selected time component (hours or minutes) to increase or decrease. Proper debouncing techniques are applied to ensure accurate input detection.

The code also implements checks to limit the displayed values within valid ranges. For hours, the limit is set to 23, and for minutes, it is set to 59. The 'display' function is responsible for updating the 7-segment displays to reflect the current time. It does so by activating the appropriate segments based on the values stored in the 'count' array for each digit. The display operation is controlled, with each digit being displayed for a short duration before transitioning to the next.

F. Flowchart:

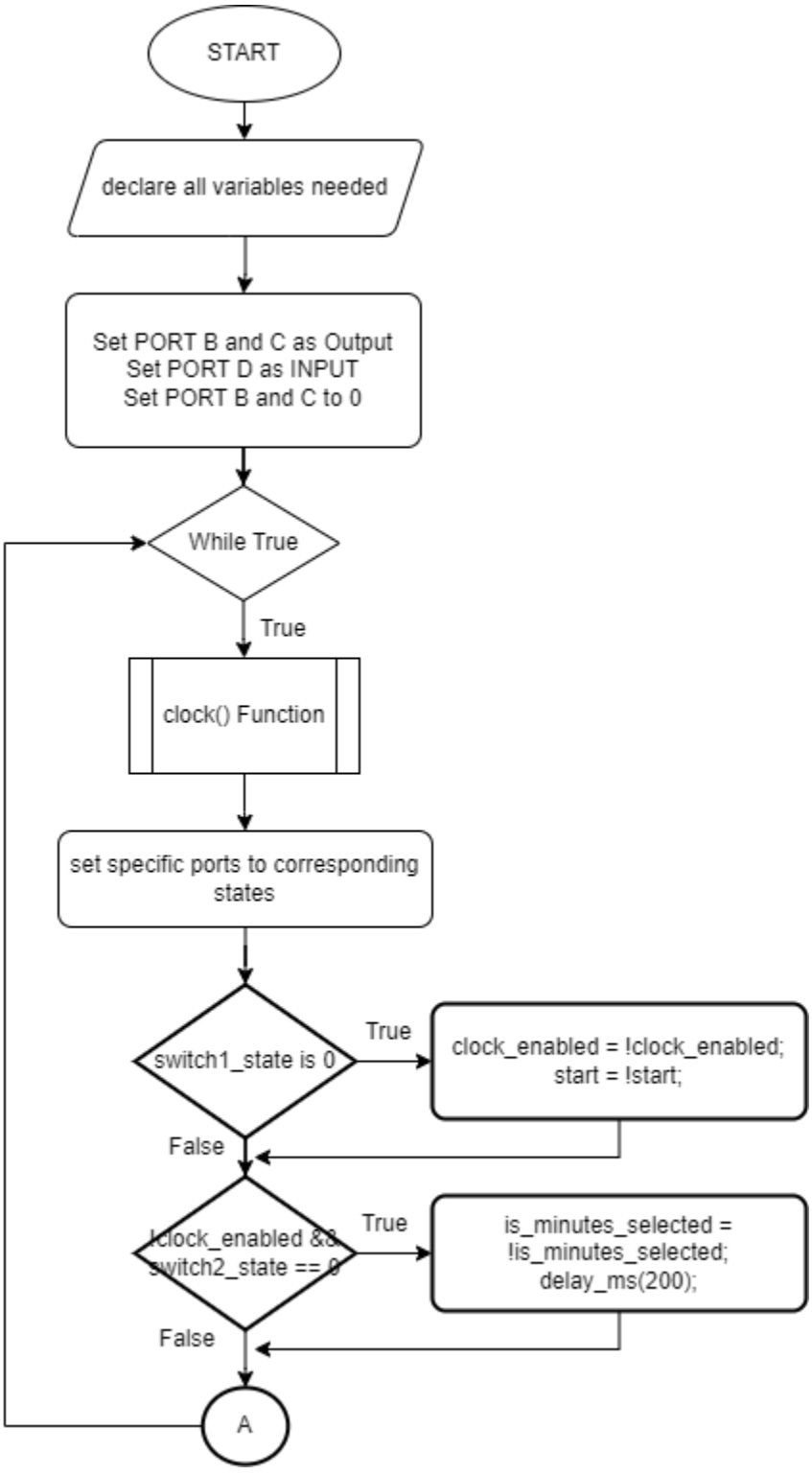

Figure 3. Flowchart of Main 1

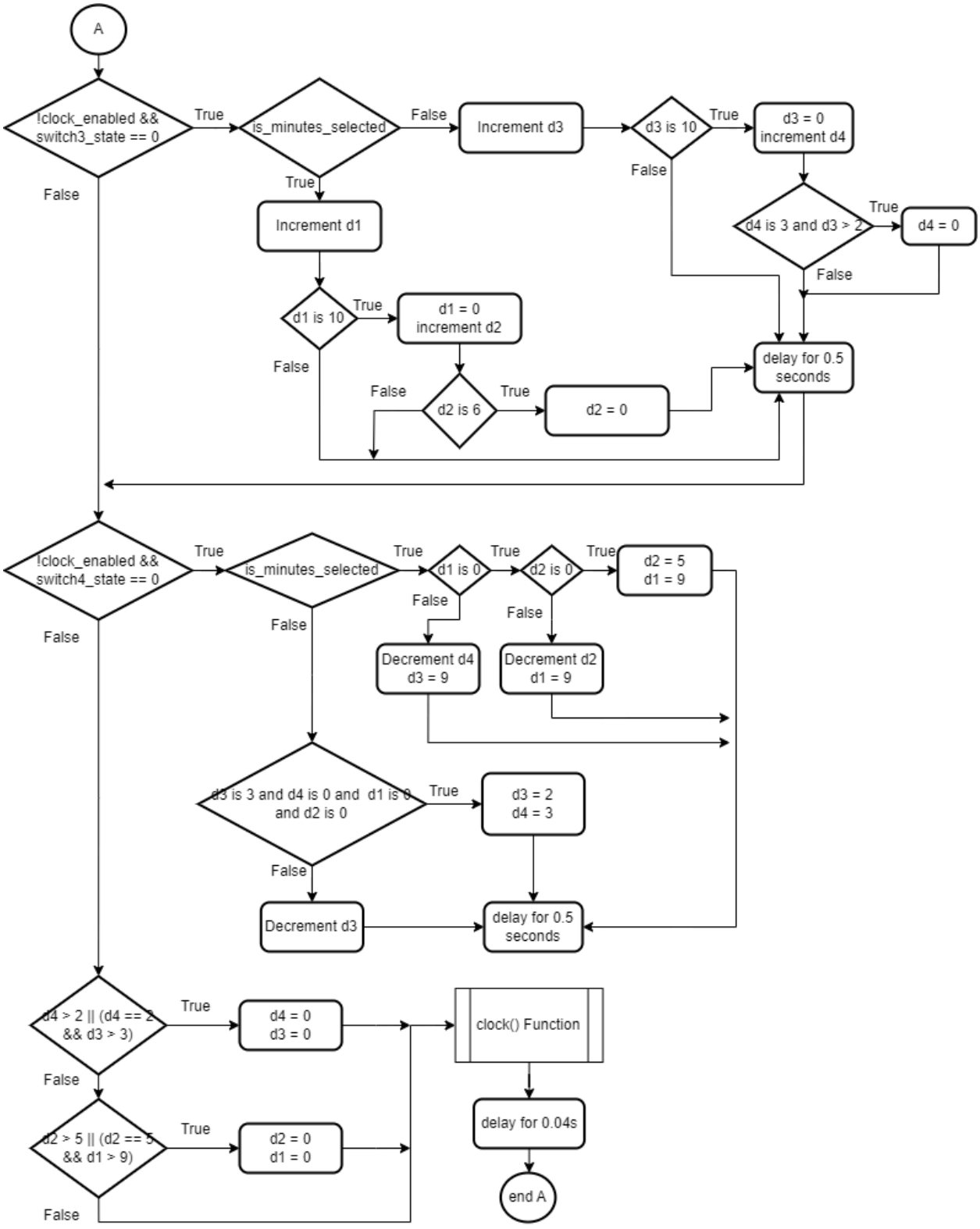

Figure 4. Flowchart of Main 2

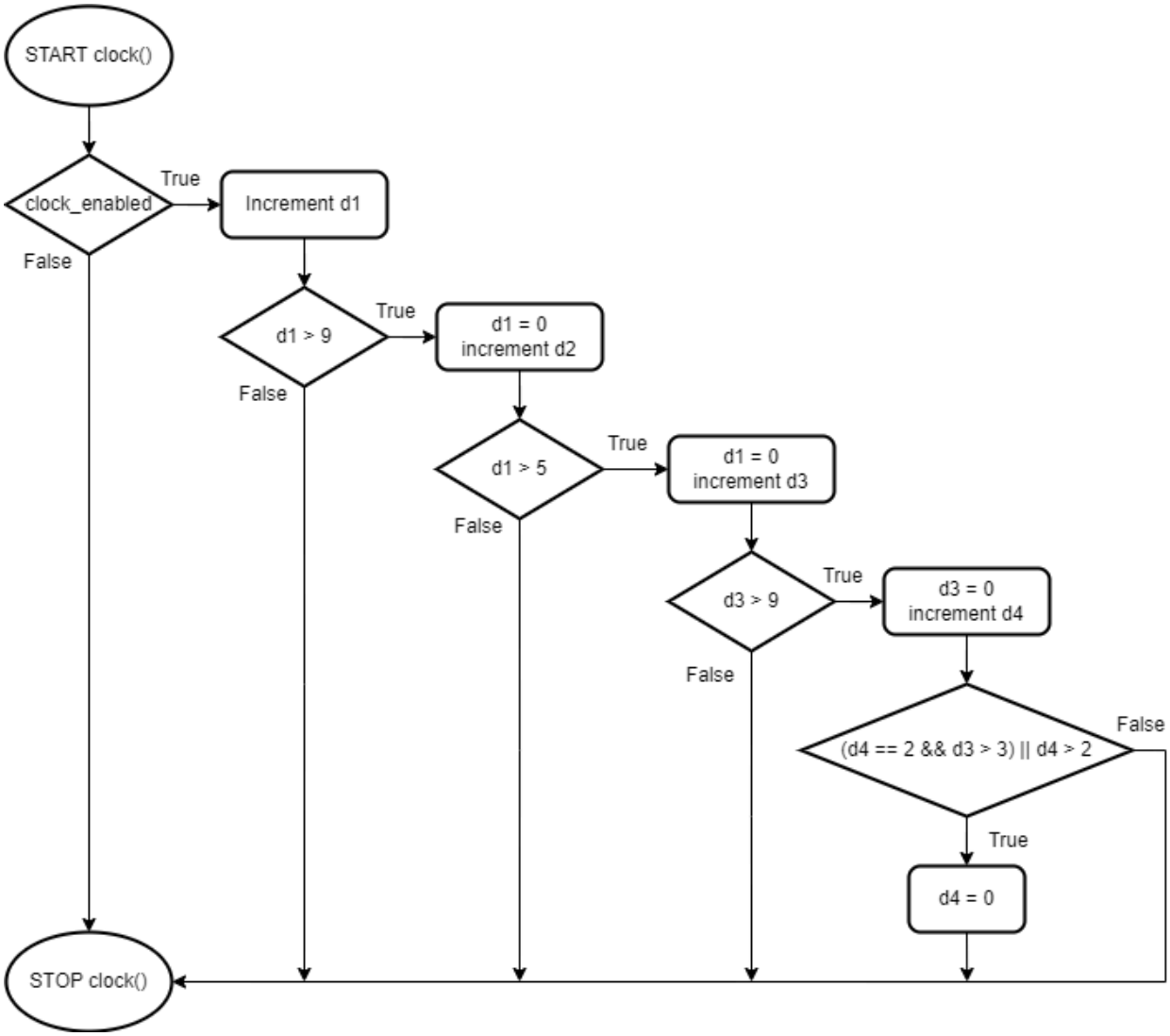

Figure 3. Flowchart of clock() Function

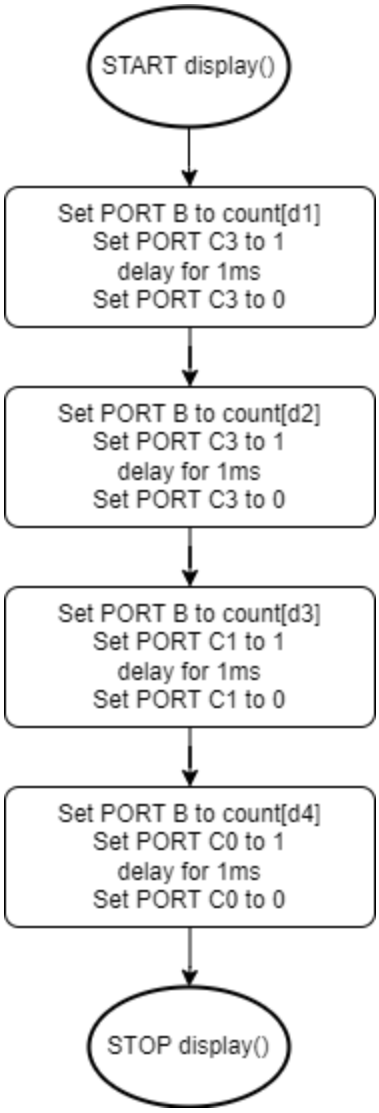

Figure 3. Flowchart of display() Function

V. Analysis and Conclusion:

The objective of the project was to design and implement a digital clock with adjustable functionality utilizing a PIC16F877A microcontroller and a 4-digit 7-segment display. The project consists of the integration of features such as clock setting and pausing, digit selection, and time incrementing and decrementing through button inputs. The project was done in two distinct phases, namely simulation and hardware implementation.

During the simulation phase, the microcontroller was successfully interfaced with the 7-segment display and buttons, demonstrating proper connections and accurate numerical display. The clock function has been successfully implemented, ensuring correct display. Through the utilization of push-buttons, users may adjust the clock. Limit values and conditions has been set to ensure that the clock follows the 24 hours clock display

The hardware implementation accurately follows the connections and components with the simulation. The PIC microcontroller underwent programming operations, resulting in the successful loading of the hex file. The clock, button functionalities, and display successfully loaded into the microcontroller.

The conducted study yielded a substantial enhancement in the participants' understanding of PIC microcontrollers, assembly language programming, and digital circuitry. Through the successful construction of a fully operational digital clock, students were able to refine their skills in microcontroller interfacing, coding, and hardware implementation. Additionally, the study provided a good experience for future projects dealing with PIC microcontrollers and assembly language programming.