November
2025

Digital counter

Using 4 digits 7 Segment

Basant Saad Eldin

Task Definition of the Program

Create a 4-digit counter using a PIC18F6722 microcontroller, displayed on a multiplexed 7-segment display, and controlled by two push-buttons.

Detailed Description of the Task

1 Display a 4-digit number (0000 → 9999)

- The number is shown using **4 multiplexed 7-segment displays.**
- PORTB outputs the **segment patterns (a–g).**
- RC0, RC1, RC2, RC3 select which of the four digits is active.

2 Use multiplexing to display all digits

- Only **one digit lights at a time**, very fast.
- Cycling through digits quickly makes all four appear ON simultaneously.
- The function display(counter):
    - splits the number into thousands, hundreds, tens, ones
    - shows each digit one at a time with small delay

3 Respond to two buttons

- **Button on RC4 → Increment counter by +1**
- **Button on RC5 → Decrement counter by -1**
- Software **debounce** is used to avoid bouncing errors.

4 Ensure counter stays in valid range

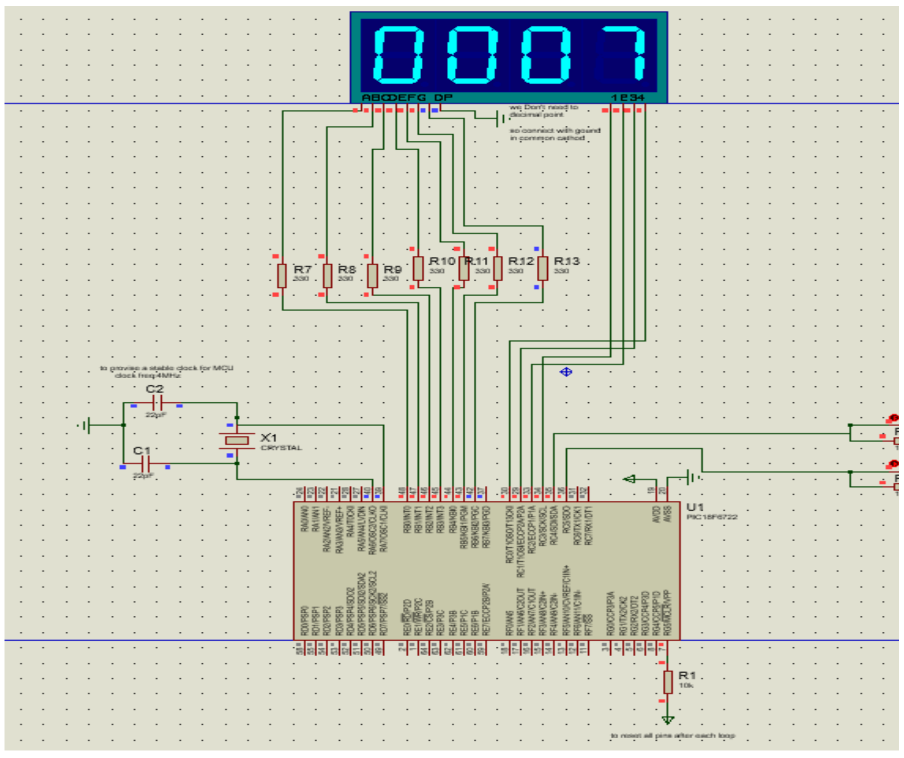
- If it exceeds 9999 → reset to 0000
- If it goes below 0000 → wrap to 9999

5 This system acts as a manual digital counter

You press:

- **INC → number increases**
- **DEC → number decreases**

Display always updates in real-time through multiplexing.



Proteus Simulation , program implements a digital 4-digit counter on a multiplexed 7-segment display using a PIC18F6722

Code Representation (Structured + Explained)

1. Includes & Configuration Bits

```c
#include <p18f6722.h>
#include <delays.h>

#pragma config OSC = HS      // High-speed external oscilla
#pragma config WDT = OFF     // Watchdog Timer OFF
#pragma config LVP = OFF     // Low-Voltage Programming OFF
```

2. 7-segment dislay digit patterns

Each value corresponds to the segments of a common-cathode display:

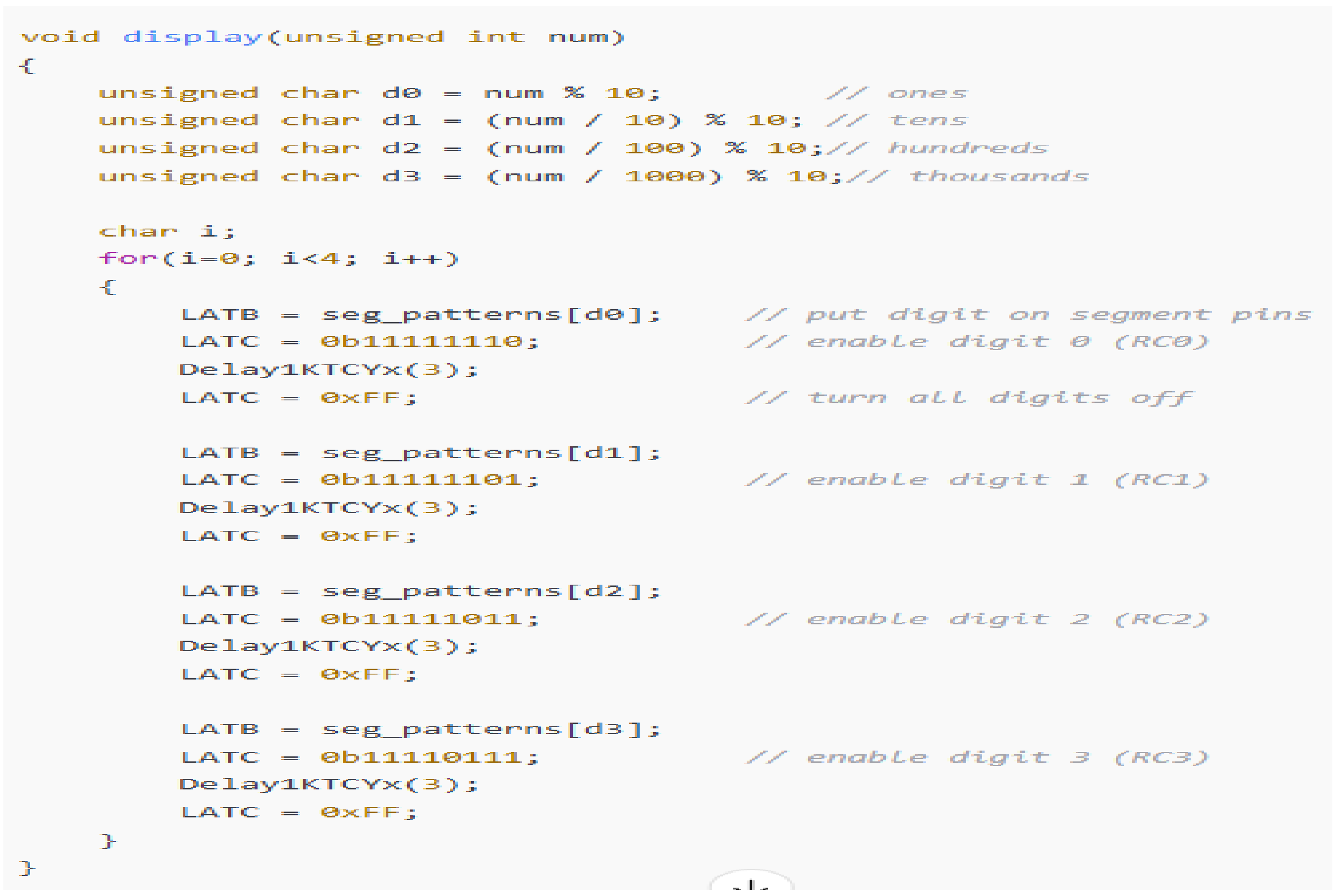
```c
const rom unsigned char seg_patterns[10] = {
    0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F
};
```

3. Button definitions

Using pins RC4 and RC5 as buttons:

```c
#define BUTTON_INC PORTCbits.RC4    // Increment button
#define BUTTON_DEC PORTCbits.RC5    // Decrement button
```

4. Display function (multiplexing four digits)



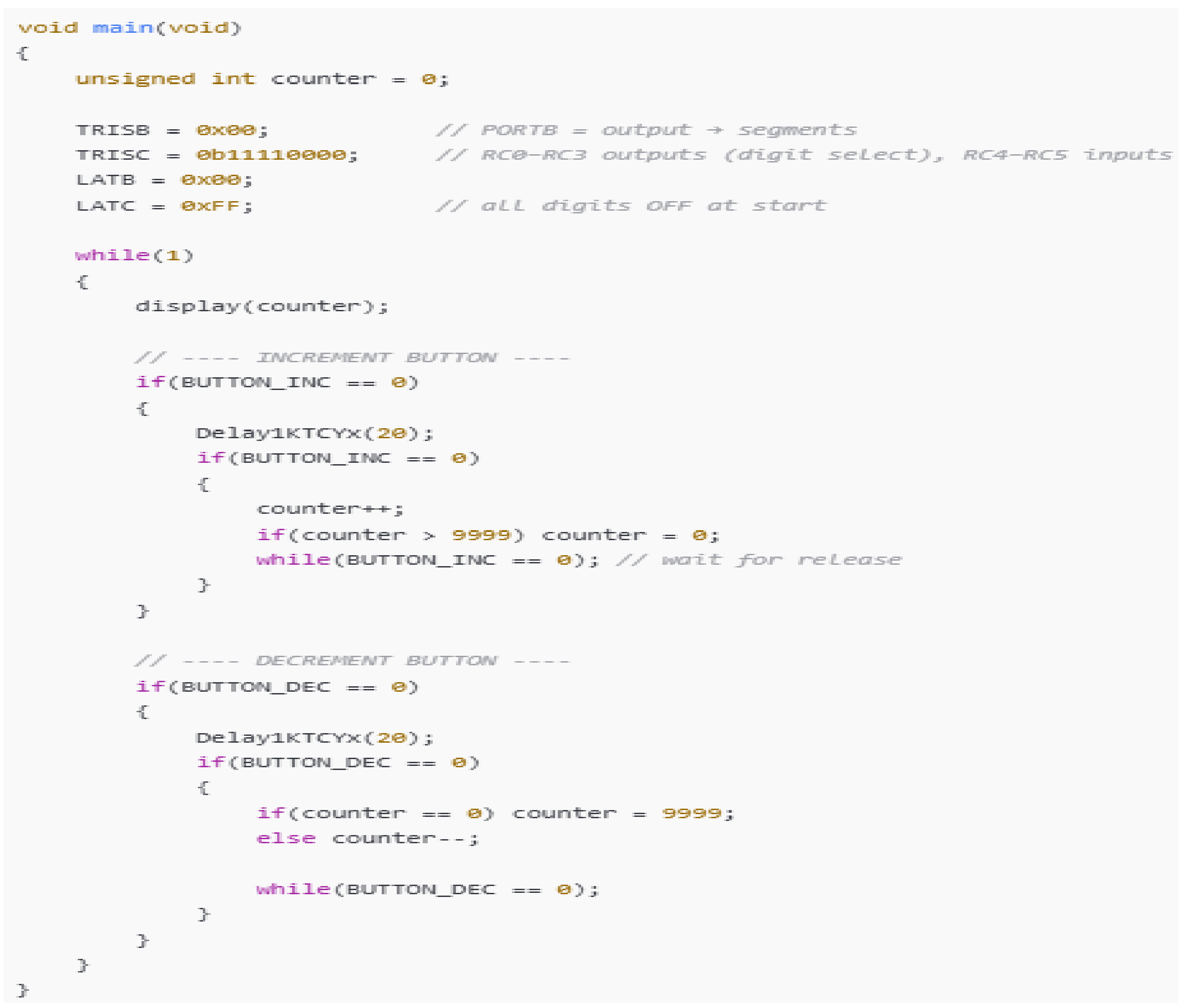
```
void display(unsigned int num)
{
    unsigned char d0 = num % 10;          // ones
    unsigned char d1 = (num / 10) % 10;   // tens
    unsigned char d2 = (num / 100) % 10;// hundreds
    unsigned char d3 = (num / 1000) % 10;// thousands

    char i;
    for(i=0; i<4; i++)
    {
        LATB = seg_patterns[d0];     // put digit on segment pins
        LATC = 0b11111110;           // enable digit 0 (RC0)
        Delay1KTCYx(3);
        LATC = 0xFF;                 // turn all digits off

        LATB = seg_patterns[d1];
        LATC = 0b11111101;           // enable digit 1 (RC1)
        Delay1KTCYx(3);
        LATC = 0xFF;

        LATB = seg_patterns[d2];
        LATC = 0b11111011;           // enable digit 2 (RC2)
        Delay1KTCYx(3);
        LATC = 0xFF;

        LATB = seg_patterns[d3];
        LATC = 0b11110111;           // enable digit 3 (RC3)
        Delay1KTCYx(3);
        LATC = 0xFF;
    }
}
```

What this does

- It splits the number into 4 digits
- Displays each digit quickly one-by-one (multiplexing)
- Repeats the cycle 4 times to reduce flicker

---

5. Main Program



```
void main(void)
{
    unsigned int counter = 0;

    TRISB = 0x00;           // PORTB = output → segments
    TRISC = 0b11110000;     // RC0–RC3 outputs (digit select), RC4–RC5 inputs
    LATB = 0x00;
    LATC = 0xFF;            // all digits OFF at start

    while(1)
    {
        display(counter);

        // ---- INCREMENT BUTTON ----
        if(BUTTON_INC == 0)
        {
            Delay1KTCYx(20);
            if(BUTTON_INC == 0)
            {
                counter++;
                if(counter > 9999) counter = 0;
                while(BUTTON_INC == 0); // wait for release
            }
        }

        // ---- DECREMENT BUTTON ----
        if(BUTTON_DEC == 0)
        {
            Delay1KTCYx(20);
            if(BUTTON_DEC == 0)
            {
                if(counter == 0) counter = 9999;
                else counter--;

                while(BUTTON_DEC == 0);
            }
        }
    }
}
```

What the main loop does

✔ Continuously refreshes the 7-segment display

✔ Reads two buttons:

- RC4 → increment counter
- RC5 → decrement counter