2.2 INTERFACING SEVEN SEGMENT DISPLAY WITH LPC1768 MICROCONTROLLER

Learning Outcomes:

- Understanding Seven-Segment Display Interfacing
  - Learn how a seven-segment display works and how each segment (A-G, DP) is controlled to display numbers.
  - Differentiate between common anode and common cathode displays.
- Multiplexing Technique
  - Understand the multiplexing concept to control multiple displays using fewer GPIO pins.
  - Learn how time-division multiplexing creates the illusion of multiple displays being ON simultaneously. Microcontroller GPIO Utilization
  - Learn how LPC1768 GPIO pins are used to control multiple outputs efficiently.
  - Understand the importance of current-limiting resistors in protecting GPIO pins and LEDs.
- Transistor as a Switching Device
  - Understand how NPN transistors (BC547) function as electronic switches.
  - Learn how base current control (via a 1kΩ resistor) determines whether the display is active or not.

2.2.1 INTERFACING DIAGRAM

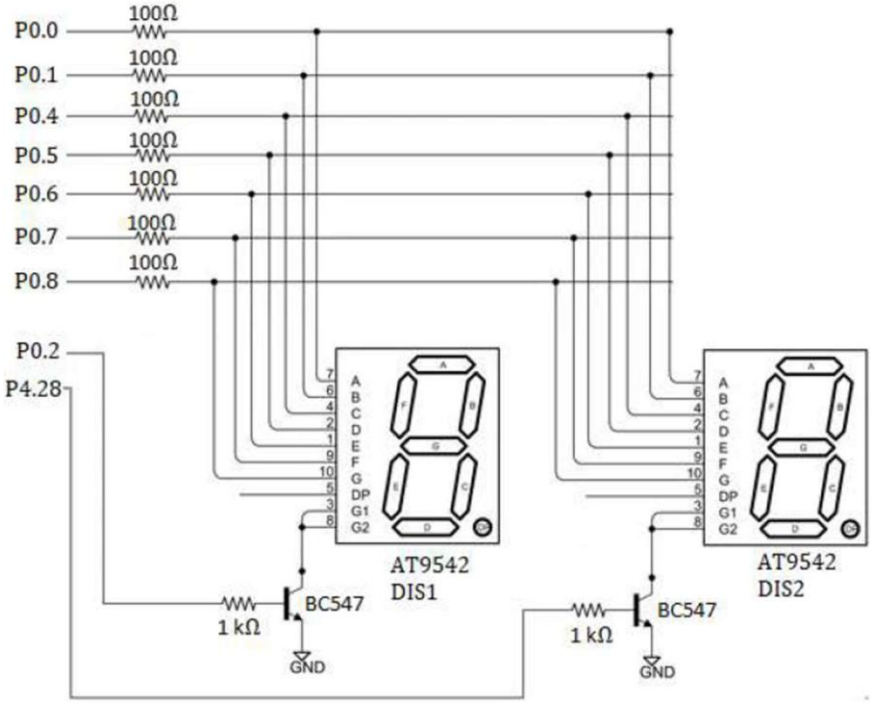


Fig.: Interfacing 7-Segment Display with LPC1768 Microcontroller

The provided circuit diagram illustrates the interfacing of two common anode sevensegment displays (AT9542) with an LPC1768 microcontroller. The segments of both displays are

multiplexed, meaning they share the same data lines (P0.0 to P0.8), while separate control lines (P0.2 and P4.28) are used to enable each display independently.

Seven-Segment Data Connection (P0.0 - P0.8)

- The segment pins (A to G and DP) of both displays are connected in parallel to P0.0 to P0.8 via 100Ω resistors.
- Since the displays are common anode, the anode (VCC) is connected to a positive voltage, and the segments light up when a LOW (0V) signal is applied.
- The microcontroller must output LOW (0) to turn on a segment and HIGH (1) to turn it off.

Multiplexing Control Using Transistors

- Each display has a separate common anode pin connected to an NPN transistor (BC547).
- P0.2 controls Display 1 (DIS1) and P4.28 controls Display 2 (DIS2).
- When a transistor is activated (ON), it connects the display's common anode to GND, allowing the display to function.
- The 1kΩ resistor at the transistor's base limits the base current to ensure proper switching.

Working Mechanism

The LPC1768 microcontroller drives the segment lines by outputting LOW (0V) for active segments. Multiplexing technique is used to display numbers on both displays:

- Activate Display 1 (DIS1)
  - Set P0.2 HIGH to turn ON the transistor, connecting the anode to GND.
  - Drive P0.0–P0.8 LOW (0V) for the required segments.
  - Keep P4.28 LOW to turn off Display 2.
- Activate Display 2 (DIS2)
  - Set P4.28 HIGH while P0.2 LOW to switch displays.
  - Update P0.0–P0.8 accordingly.
  - The microcontroller rapidly alternates between displays (every few milliseconds), making them appear continuously ON due to Persistence of Vision.

2.2.2 DESIGN CALCULATIONS

Current Limiting Resistor Calculations:

Each segment in a seven-segment display is essentially an LED. The key parameters to consider:

- Typical Forward Voltage of LED (V_f): 2V (for red LEDs, varies for other colours).
- Operating Voltage (V_{cc}): 3.3V (LPC1768 GPIO output voltage).
- Maximum Current per Segment (I_{seg}): 10mA to 20mA (safe range for LEDs).

Here,

V = 3.3 V V = 2V I = 10 mA (Typical Value)

$$\therefore R = \frac{V_{CC} - V_F}{I_F} = \frac{3.3 - 2}{10x10^{-3}} = 130 \text{ Ohm}$$

Selecting Standard Value of Resistance **R = 100 Ohm** If we use a 100Ω resistor, the segment current will be:

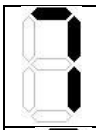
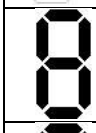
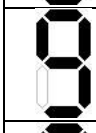
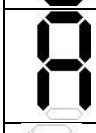
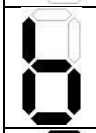
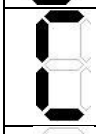
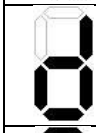
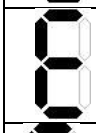
$$I = \frac{1.3}{100} = 13 \text{ mA (This is within the safe range of LED current)}$$

The power dissipation in each resistor is calculated as

P = I R = (13x10) x 100 = 0.00169W < 0.25 W of Standard Resistor

2.2.3 SEVEN SEGMENT CODES

Pins	P0.8	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	Hex Code in 32 bit
Digit	g	f	e	d	c	NC	EN1	b	a	
0	1	0	0	0	0	0	0	0	0	0x0000 0100
1										
2										
3										
4										
5										
6										

FIODIR Register Configuration:

All the port pins should be configured as output pins. Write '1' to configure as output.

P0.31	P0.30	P0.29	P0.28	P0.27	P0.26	P0.25	P0.24	P0.23	P0.22	P0.21	P0.20	P0.19	P0.18	P0.17	P0.16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

P0.31	P0.30	P0.29	P0.28	P0.27	P0.26	P0.25	P0.24	P0.23	P0.22	P0.21	P0.20	P0.19	P0.18	P0.17	P0.16
0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1

LPC_GPIO0->FIODIR = 0x0000 01F7

2.2.4 EXAMPLE CODES

2a) Write a program in C to display the digit '5' on first seven segment display.

Code:

```
#include <lpc17xx.h>
 int main(void) {
        // Make all used pins output LPC_GPIO0->FIODIR |=
        0x000001F7;

        // Send the 7-Seg. Display Code LPC_GPIO0->FIOPIN = 0x00000042;

        // Enable first display unit LPC_GPIO0->FIOSET |= (1 << 2);
         while(1);
}
```

2b) Write a program in C to display the character 'F' on second seven segment display.

Code:

```
#include <lpc17xx.h>


int main(void) {
        // Make all used pins output LPC_GPIO0->FIODIR |=
        0x000001F7;

        // Configure the control pin of 2nd 7-Seg display as o/p
        LPC_GPIO4->FIODIR |= (1 << 28);

        // Send the 7-Seg. Display Code for character 'F' LPC_GPIO0->FIOPIN =
        0x00000032;

        // Enable first display unit LPC_GPIO4->FIOSET |= (1 << 28);
         while(1);
}
```

2c) Write a program in C to implement Hexadecimal Up-Counter that counts from 0 to F on Display-1 of 7-Segment display. Code:

```
#include<lpc17xx.h> void delay(unsigned
int count)
{
        unsigned int i, j;          for (i=0; i<count; i++)
                        for(j=0; j<1275; j++);
}

int main(void)
{       unsigned long int sevensegcodes[] =  {
                _____, _____, _____,
```

Complete these by writing the 32-bit hex codes for all digits.

```
                    _____________, _____________, _____________,
                    _____________, _____________, _____________,
                    _____________, _____________, _____________,
                    _____________, _____________, _____________,
                    _____________
            };
                unsigned char i = 0;
                  // Make all used pins output
                LPC_GPIO0->FIODIR |= 0x000001F7;
              while(1)
            {
                        for(i = 0; i < 16; i++)
                {
                            LPC_GPIO0->FIOPIN = sevensegcodes[i];
                              // Enable first display unit
                            LPC_GPIO0->FIOSET |= (1 << 2);
                          delay(5000);
                }
            }
        }
```

2d) Write a program in C to implement Hexadecimal Down-Counter that counts from FF to 00 on 7-Segment display.

Code:

```
#include<lpc17xx.h>
 void delay(unsigned int count)
{       unsigned int i,j;
        for (i=0; i<count; i++)
for(j=0;j<1275;j++);
}  int main(void)
{ unsigned long int sevensegcodes[] =  {
                    _____________, _____________, _____________,
```

```
                    _____, _____, _____,
                    _____, _____, _____,
                    _____, _____, _____,
                    _____, _____, _____,
                    _____
        };
            unsigned char number, digit, i = 0;

          // Make all used pins output
         LPC_GPIO0->FIODIR|= 0x000001F7;
          LPC_GPIO4->FIODIR|= (1 << 28);


        while(1)
        {
                    for(number = 0xFF; number >= 0x00; number--)
            {
                        for(i = 0; i < 50; i++)
                    {
  // Extract first digit and display on unit 1   digit = (number & 0xF0)>>4;   LPC_GPIO0->FIOPIN = sevensegcodes[digit];
                        LPC_GPIO0->FIOSET |= (1 << 2);
LPC_GPIO4->FIOCLR |= (1 << 28);
                        delay(50);

                                                // Extract first digit and display on unit 2
                      digit = (number & 0x0F);
                        LPC_GPIO0->FIOPIN = sevensegcodes[digit];
                      LPC_GPIO0->FIOCLR |= (1 << 2);
LPC_GPIO4->FIOSET |= (1 << 28);
                        delay(50);
                    }
            }
        }
}
```