STM32 | 7-SEGMENT DISPLAY

Interfacing 7-Segment Display with STM32F103 using CMSIS

◆ Aim

To interface a common cathode 7-segment display with the STM32F103C8 microcontroller and display numbers 0–9 continuously using GPIO port control via CMSIS registers.

◆ Apparatus Required

Component	Specification / Notes
Microcontroller	STM32F103C8T6 (Blue Pill Board)
Display	7-Segment (Common Cathode)
Resistors	220 Ω × 7 (for current limiting)
Power Supply	+5 V (for display), +3.3 V (for MCU logic)
Breadboard / Proteus	For simulation or testing
Software	STM32CubeIDE / Keil / Proteus 8 Professional

◆ Theory

A 7-segment display consists of 7 LEDs (a–g) arranged to form decimal digits. Each segment lights up when its corresponding pin is set HIGH in common cathode configuration.

Each digit (0–9) can be represented as a binary pattern, where each bit corresponds to one segment:

Segment	Bit	Position
a	0	PA0
b	1	PA1

c	2	PA2
d	3	PA3
e	4	PA4
f	5	PA5
g	6	PA6

Thus, bit pattern 0b0111111 (0x3F) lights all segments except ‘g’, displaying the number 0.

◆ Circuit Diagram (Connection)

Segment	STM32 Pin	Port Pin
a	PA0	GPIOA Pin 0
b	PA1	GPIOA Pin 1
c	PA2	GPIOA Pin 2
d	PA3	GPIOA Pin 3
e	PA4	GPIOA Pin 4
f	PA5	GPIOA Pin 5
g	PA6	GPIOA Pin 6
COM (Common Cathode)	GND	via resistor if needed

Important:
Each segment must have its own 220 Ω resistor between the STM32 pin and segment pin to limit current.

◆ Program Code (main.c)

```c
#include "stm32f1xx.h"

uint8_t seg_code[10] = {
    0x3F, // 0 -> 0b00111111
    0x06, // 1 -> 0b00000110
    0x5B, // 2 -> 0b01011011
    0x4F, // 3 -> 0b01001111
    0x66, // 4 -> 0b01100110
    0x6D, // 5 -> 0b01101101
```

```
    0x7D, // 6 -> 0b01111101
    0x07, // 7 -> 0b00000111
    0x7F, // 8 -> 0b01111111
    0x6F  // 9 -> 0b01101111
};

void delay_ms(int t){
      for (int i = 0; i < t*800; i++){
            __NOP();
      }
}

int main(void){

      RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;

      GPIOA->CRL =0x22222222;

      while(1){
            for (int i = 0; i < 10; i++){
                  GPIOA->ODR = 0x00;
                  GPIOA->ODR = seg_code[i];
                  delay_ms(200);
            }
            for (int i = 9; i > -1; i--){
                  GPIOA->ODR = 0x00;
                  GPIOA->ODR = seg_code[i];
                  delay_ms(200);
            }
      }
}
```

◆ Explanation of Key Registers

Register	Description
RCC->APB2ENR	Enables peripheral clocks. Bit 2 enables GPIOA.
GPIOA->CRL	Configures mode (input/output) and CNF for pins PA0–PA7. Each pin takes 4 bits. 0x2 → Output mode (2 MHz, push-pull).
GPIOA->ODR	Output Data Register — writing a bit HIGH turns that pin ON.
__NOP()	Assembly instruction for "No Operation," used to create software delays.

◆ Working Principle

1. The STM32 sends a 7-bit pattern to the 7-segment display through GPIOA pins.

2. Each bit in the pattern represents one segment (a–g).
3. The lookup table (seg_code[]) stores binary values for digits 0–9.
4. The display continuously counts 0–9 and back down 9–0.

◆ Proteus Simulation Tips

- Use "7SEG-COM-CATHODE" component.
- Connect PA0–PA6 → segments a–g respectively.
- Tie the common pin to GND.
- Add 220 Ω resistors in series with each segment.
- Set MCU clock to 8 MHz internal RC (HSI) or default 72 MHz in Proteus MCU properties.

◆ Expected Output

The 7-segment display will:

- Count 0 → 9 → 0 repeatedly,
- Each digit visible for ~0.5 s,
- All segments light according to the corresponding binary pattern.

◆ Result

Successfully interfaced a 7-segment display with STM32F103 using direct CMSIS register programming and displayed numbers 0–9 continuously.
