STM32 | UART LED Control

Objective

- To control an LED connected to PA0 of STM32F103C8T6 using UART commands ('1' / '0').
- To simulate UART communication in Proteus (using Virtual Terminal or HC-05 Bluetooth module).
- To understand GPIO and USART registers in STM32F103 using CMSIS.

Components Required (Proteus)

Component	Quantity	Proteus Model
STM32F103C8T6	1	Blue Pill MCU
LED	1	Generic LED
Resistor	220 Ω	Generic resistor
Virtual Terminal	1	Proteus instrument (for UART)
HC-05 Bluetooth Module (optional)	1	Proteus HC-05 module
Wires	As needed	Proteus wires for connections

Circuit Diagram (Proteus)

Option A: Using Virtual Terminal

STM32 Pin	Connection
PA0	LED anode → 220Ω → cathode → GND
PA9	TX → Virtual Terminal RX
PA10	RX ← Virtual Terminal TX
VCC	5V
GND	GND

Option B: Using HC-05 (Optional)

STM32 Pin	Connection
PA0	LED anode → 220Ω → cathode → GND
PA9	TX → HC-05 RX
PA10	RX ← HC-05 TX
VCC	5V
GND	GND

Note: In Proteus, you can choose Virtual Terminal for simplicity instead of HC-05.

Registers Used and Explanation

Peripheral	Register	Function
GPIOA	CRL	Configure PA0 as Output Push-Pull 10MHz
GPIOA	ODR	Output Data Register: 1 → LED ON, 0 → LED OFF
RCC	APB2ENR	Enable clock for GPIOA and USART1
USART1	BRR	Baud rate control (9600 bps @ 8 MHz HSI → 0x341)
USART1	CR1	Enable USART, Transmitter, Receiver
USART1	SR	Status Register: TXE (Transmit empty), RXNE (Receive not empty)
USART1	DR	Data register: read/write characters

Code (CMSIS, Proteus-ready)

```
#include "stm32f1xx.h"
#include <string.h>

void USART1_Init(void);
void USART1_SendChar(char c);
void USART1_SendString(char *str);
char USART1_GetChar(void);
void LED_Init(void);
void delay(int t);

int main(void)
{
    USART1_Init();
    LED_Init();
```

```c
    USART1_SendString("UART Ready\r\n");

    while(1)
    {
        char c = USART1_GetChar();  // Receive char
        USART1_SendChar(c);         // Echo back

        if(c == '1')
        {
             GPIOA->ODR |= (1 << 0);    // LED ON
             USART1_SendString("\r\nLED is ON\r\n");      // Store received char
        }
        else if(c == '0')
        {
            GPIOA->ODR &= ~(1 << 0);  // LED OFF
            USART1_SendString("\r\nLED is OFF\r\n");
        }
    }
}

//---------------- USART ----------------
void USART1_Init(void)
{
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN | RCC_APB2ENR_USART1EN;

    // PA9 TX - AF Push Pull
    GPIOA->CRH &= ~(0xF << 4);
    GPIOA->CRH |=  (0xB << 4);

    // PA10 RX - Floating input
    GPIOA->CRH &= ~(0xF << 8);
    GPIOA->CRH |=  (0x4 << 8);

    USART1->BRR = 0x341;    // 9600 bps @ 8MHz HSI
    USART1->CR1 |= (1 << 13) | (1 << 3) | (1 << 2);
}

void USART1_SendChar(char c)
{
    while(!(USART1->SR & (1 << 7)));
    USART1->DR = c;
}

void USART1_SendString(char *str)
{
    while(*str) USART1_SendChar(*str++);
}

char USART1_GetChar(void)
{
    while(!(USART1->SR & (1 << 5)));
    return USART1->DR;
}

//---------------- LED Init ----------------
void LED_Init(void)
{
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;  // GPIOA clock
    GPIOA->CRL &= ~(0xF << (0 * 4));     // Clear PA0
```

```
    GPIOA->CRL |=  (0x1 << (0 * 4));     // MODE=01 (Output 10MHz), CNF=00 (Push-Pull)
}

//---------------- Delay ----------------
void delay(int t)
{
    for(int i=0; i<800*t; i++){
       __NOP();
    }
}
```

Step-by-Step Proteus Simulation Instructions

1. Open Proteus and place:
    - STM32F103C8T6
    - LED + 220Ω resistor
    - Virtual Terminal (or HC-05 module)
2. Connect PA0 → LED → GND.
3. Connect PA9 → Virtual Terminal RX, PA10 ← Virtual Terminal TX.
4. Connect VCC and GND.
5. Compile the code in STM32CubeIDE and generate .hex file.
6. Load .hex file into STM32 in Proteus.
7. Run the simulation.
8. Virtual Terminal should display "UART Ready".
9. Type '1' → LED turns ON; '0' → LED turns OFF.
10. Observe the echo messages confirming the LED state.

Observations

- On simulation start: "UART Ready" appears on terminal.
- Sending '1' → LED turns ON and message "LED is ON" appears.
- Sending '0' → LED turns OFF and message "LED is OFF" appears.
- Characters are echoed back for verification.

Notes for Simulation

- Use Virtual Terminal for simple testing in Proteus.
- Ensure PA9 TX → Terminal RX, PA10 RX ← Terminal TX.
- Baud rate in USART must match terminal (9600 bps).
