STM32 ???? ? 11 ??RS-485 ??

? ????

1. 理解 **RS-485** 差分信號 - 掌握半雙工通訊
2. 配置 **MAX485** 收發器 - 實現長距離多點通訊
3. 實現發送 /接收自動切換 - 控制方向線 (DE/RE)

? RS-485 ????

RS-485 vs RS-232

特性	**RS-232**	**RS-485**
信號線	單端 (信號 + GND)	差分 (A、B)
距離	≤ 15 公尺	≤ 1200 公尺
速度	最高 115.2 kbps	最高 10 Mbps
設備數	1 對 1	多點 (最多 32)
模式	全雙工	半雙工

RS-485 ???

信號	說明
A 線	不反轉資料線
B 線	反轉資料線 (邏輯電平反轉)
GND	共同接地

邏輯定義:

- 邏輯 **1**: A > B (A 電壓高於 B)
- 邏輯 **0**: A < B (A 電壓低於 B)

MAX485 ?????

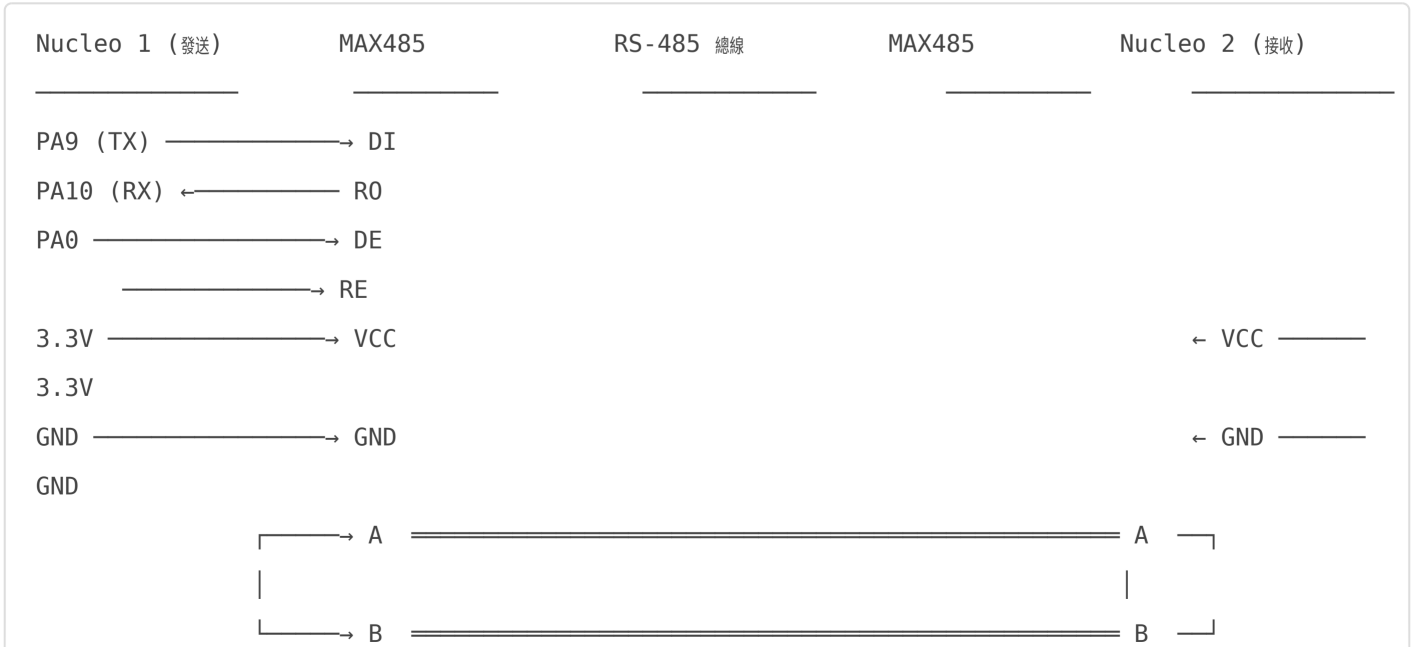
引腳	名稱	方向	功能
DI	Data In	←	來自 UART TX
RO	Receiver Out	→	至 UART RX
DE	Driver Enable	←	發送使能（高 ＝發送）
RE	Receiver Enable	←	接收使能（低 ＝接收）
A、B	差分線	↔	連接到總線

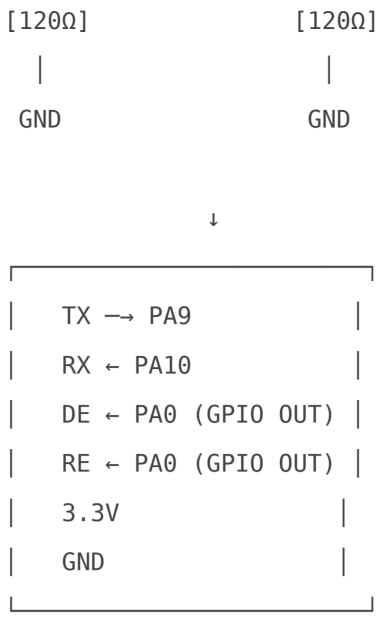
?? ????

????

元件	數量	說明
MAX485 收發器	2	一發一收
STM32 Nucleo	2	主控板
跳線	多條	連接
120Ω 終端電阻	2	總線兩端（可選）

???





?? CubeMX ????

?? 1??? USART1

1. 在 Pinout 中配置:
   - **PA9**: USART1_TX
   - **PA10**: USART1_RX
   - **PA0**: GPIO_Output (用於 DE/RE 控制)

?? 2??? UART ??

1. 左側選 **Connectivity → USART1**
2. **Mode**: Asynchronous
3. 設定:
   - **Baud Rate**: 9600
   - **Word Length**: 8 Bits
   - **Parity**: None
   - **Stop Bits**: 1

?? 3?????

NVIC Settings:

- **USART1 global interrupt** ✓

?? 4??????

? ?????

????main.c

```
/* STM32 Lesson 11 - RS-485 (Sender)
 * 功能: 透過 RS-485 發送資料
 * 難度: 中級
 */

#include "main.h"
#include "usart.h"
#include <stdio.h>
#include <string.h>

#define DE_PORT GPIOA
#define DE_PIN GPIO_PIN_0
#define RE_PORT GPIOA
#define RE_PIN GPIO_PIN_0

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);
void RS485_SetTx(void);   /* 設為發送模式 */
void RS485_SetRx(void);   /* 設為接收模式 */
void RS485_Send(uint8_t *data, uint16_t size);

UART_HandleTypeDef huart1;

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART1_UART_Init();
```

```
  uint8_t counter = 0;

  while (1)
  {
    char buffer[50];

    /* 設為發送模式 */
    RS485_SetTx();
    HAL_Delay(10);  /* 等待方向穩定 */

    /* 準備資料 */
    sprintf(buffer, "Msg%d\r\n", counter++);

    /* 發送 */
    RS485_Send((uint8_t *)buffer, strlen(buffer));

    /* 設為接收模式 */
    RS485_SetRx();

    HAL_Delay(1000);
  }
}

/**
  * @brief 設置為發送模式（DE=高, RE=高）
  */
void RS485_SetTx(void)
{
  HAL_GPIO_WritePin(DE_PORT, DE_PIN, GPIO_PIN_SET);
  HAL_GPIO_WritePin(RE_PORT, RE_PIN, GPIO_PIN_SET);
}

/**
  * @brief 設置為接收模式（DE=低, RE=低）
  */
void RS485_SetRx(void)
{
  HAL_GPIO_WritePin(DE_PORT, DE_PIN, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(RE_PORT, RE_PIN, GPIO_PIN_RESET);
}
```

```
/**
  * @brief □□ RS-485 □□□□
  */
void RS485_Send(uint8_t *data, uint16_t size)
{
  HAL_UART_Transmit(&huart1, data, size, HAL_MAX_DELAY);
  HAL_Delay(5);  /* □□□□□□ */
}

void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = 2;
  RCC_OscInitStruct.PLL.PLLQ = 7;

  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    Error_Handler();

  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
                              | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
    Error_Handler();
}
```

```
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  __HAL_RCC_GPIOA_CLK_ENABLE();

  GPIO_InitStruct.Pin = GPIO_PIN_0;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
}

static void MX_USART1_UART_Init(void)
{
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 9600;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;

  if (HAL_UART_Init(&huart1) != HAL_OK)
    Error_Handler();
}

void Error_Handler(void)
{
  while(1);
}
```

? ?????

????

□ □□□ **UART** 輸出：

```
Msg0
Msg1
Msg2
```

□ □□□ **UART** 輸出：

```
Msg0
Msg1
Msg2
```

????

問題	原因	解決方案
無法接收	方向線控制錯誤	檢查 PA0 GPIO 配置
資料錯亂	波特率不匹配	驗證兩端波特率都是 9600
訊號干擾	缺少終端電阻	在總線兩端各加 120Ω 電阻

? ????

MODBUS ????

RS-485 常用於 MODBUS 協定。基本框架:

```
/* MODBUS RTU 訊息格式 */
typedef struct {
  uint8_t slave_id;          /* 從設備地址 */
  uint8_t function_code;     /* 功能碼 */
  uint8_t data[252];         /* 資料 */
  uint16_t crc;              /* CRC 校驗（結合第 12 節） */
} ModbusFrame;
```

? ????

???? - ? 12 ???? CRC ??

第 12 節將實現:

- **CRC** 多項式與校驗原理
- **STM32** 硬體 **CRC** 引擎
- 資料完整性驗證

□ **RS-485** □□□□□□□□

Revision #2
Created 2026-04-01 02:06:21 UTC by TaipeiTechRacing
Updated 2026-04-06 06:23:50 UTC