STM32 ???? ? 4 ??UART ???? + ????

? ????

1. 掌握 **UART** 通訊協定 - 理解串列通訊的原理與配置
2. 實現收發功能 - 透過電腦與 STM32 交互通訊
3. 建立命令介面 - 實現簡單的命令解析系統

? UART ????

UART ????

UART (Universal Asynchronous Receiver/Transmitter) 是最常見的串列通訊協定。特點:

- 異步通訊：發送端和接收端不需同步時脈
- 全雙工：可同時收發
- 點對點：通常用於連接兩個設備

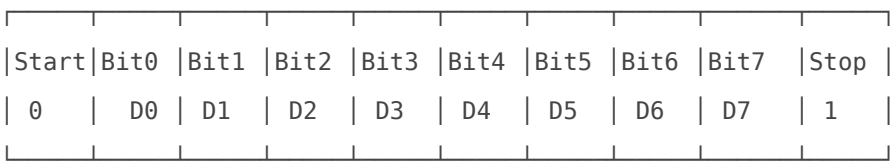
UART ???

STM32F446 UART 最少需要 3 條線:

訊號	功能	連接
TX (USART1_TX)	傳輸	PA9
RX (USART1_RX)	接收	PA10
GND	地線參考	GND

????????

參數	典型值	說明



Baud Rate	9600 / 115200	每秒傳輸位元數
Data Bits	8	每個字符包含 8 位資料
Stop Bits	1	停止位
Parity	None	無奇偶校驗

UART ?????

```
1 個 UART 字符的構成:

┌────┬────┬────┬────┬────┬────┬────┬────┬────┬────┐
│Start│Bit0 │Bit1 │Bit2 │Bit3 │Bit4 │Bit5 │Bit6 │Bit7  │Stop │
│ 0   │  D0 │ D1  │ D2  │ D3  │ D4  │ D5  │ D6  │ D7   │ 1   │
└────┴────┴────┴────┴────┴────┴────┴────┴────┴────┘
```

?? ????

????

元件	數量	說明
USB-UART 轉換器	1	PL2303 或 CP2102 晶片
USB Type-A 傳輸線	1	連接電腦
杜邦線	3	連接 STM32 和轉換器

???

STM32	**USB-UART**	說明
PA9 (TX)	RX	STM32 發送給電腦
PA10 (RX)	TX	STM32 接收來自電腦
GND	GND	共同地線

> ⚠ 重要 - 開發板已內建 ST-Link 調試器, 可透過 USB 進行 UART 通訊, 無需額外轉換器!

?? CubeMX ????

?? 1??? USART1

1. 開啟 CubeMX
2. 在 Pinout 圖上找到 PA9 和 PA10
3. 分別配置為:
   - PA9: **USART1_TX**
   - PA10: **USART1_RX**
4. 或直接點擊 Connectivity → **USART1** 自動配置

?? 2??? USART1 ??

1. 在左側選擇 **Connectivity → USART1**
2. 設定以下參數:
   - **Baud Rate**: 115200
   - **Word Length**: 8 Bits
   - **Stop Bits**: 1
   - **Parity**: None
   - **Mode**: Asynchronous (RX and TX)

?? 3??? USART1 ??

1. 點擊 **NVIC Settings** 標籤
2. 勾選 **USART1 global interrupt**
3. 設定優先級:
   - **Preemption Priority**: 1
   - **Sub Priority**: 0

?? 4??????

點擊 **Generate Code**

? ?????

main.c - UART + ????

```
/* STM32 Lesson 04 - UART Communication with Command Interface
 * 功能: 透過 UART 接收命令並控制 LED
 * 命令:
 *   "ON"  - LED 點亮
 *   "OFF" - LED 熄滅
 *   "TOGGLE" - LED 切換
 * 難度: 中級
 */

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

/* 私有定義 */
#define UART_RX_BUFFER_SIZE 50
#define COMMAND_MAX_LEN 20

/* 函數宣告 */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);
void UART_Print(const char *format, ...);
void Process_Command(char *command);
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);

/* 全域變數 */
UART_HandleTypeDef huart1;
uint8_t uart_rx_buffer[UART_RX_BUFFER_SIZE];
uint8_t uart_rx_index = 0;
char command_buffer[COMMAND_MAX_LEN];

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

```
  /* 初始化 LED 為熄滅狀態 */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);

  /* 列印歡迎訊息 */
  UART_Print("\r\n==============================\r\n");
  UART_Print("STM32 UART Command Interface\r\n");
  UART_Print("Commands: ON, OFF, TOGGLE\r\n");
  UART_Print("==============================\r\n");

  /* 啟用 UART 中斷接收 */
  HAL_UART_Receive_IT(&huart1, uart_rx_buffer, 1);

  while (1)
  {
    /* 主迴圈 */
  }
}

/**
  * @brief UART 傳輸函數（printf 風格）
  */
void UART_Print(const char *format, ...)
{
  char buffer[100];
  va_list args;
  va_start(args, format);
  vsnprintf(buffer, sizeof(buffer), format, args);
  va_end(args);

  HAL_UART_Transmit(&huart1, (uint8_t *)buffer, strlen(buffer), HAL_MAX_DELAY);
}

/**
  * @brief UART 接收完成中斷回調
  */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  if (huart->Instance == USART1)
  {
    uint8_t received_char = uart_rx_buffer[0];
```

```
    /* 處理回車符（\r）或換行符（\n） */
    if (received_char == '\r' || received_char == '\n')
    {
      if (uart_rx_index > 0)
      {
        command_buffer[uart_rx_index] = '\0';
        UART_Print("\r\nReceived: %s\r\n", command_buffer);
        Process_Command(command_buffer);
        uart_rx_index = 0;
      }
    }
    else if (received_char == 0x08 || received_char == 0x7F)  /* Backspace */
    {
      if (uart_rx_index > 0)
      {
        uart_rx_index--;
        UART_Print("\b \b");  /* 刪除顯示 */
      }
    }
    else
    {
      /* 一般字符 */
      if (uart_rx_index < COMMAND_MAX_LEN - 1)
      {
        command_buffer[uart_rx_index++] = received_char;
        HAL_UART_Transmit(&huart1, (uint8_t *)&received_char, 1, HAL_MAX_DELAY);  /* Echo */
      }
    }

    /* 繼續接收下一個字符 */
    HAL_UART_Receive_IT(&huart1, uart_rx_buffer, 1);
  }
}

/**
  * @brief 命令處理函數
  */
void Process_Command(char *command)
{
```

```
  /* 轉換為大寫以便比較 */
  for (int i = 0; command[i]; i++)
  {
    if (command[i] >= 'a' && command[i] <= 'z')
      command[i] -= 32;
  }

  /* 命令解析 */
  if (strcmp(command, "ON") == 0)
  {
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
    UART_Print("LED ON\r\n");
  }
  else if (strcmp(command, "OFF") == 0)
  {
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
    UART_Print("LED OFF\r\n");
  }
  else if (strcmp(command, "TOGGLE") == 0)
  {
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
    UART_Print("LED TOGGLED\r\n");
  }
  else
  {
    UART_Print("Unknown command. Try: ON, OFF, TOGGLE\r\n");
  }

  UART_Print("> ");  /* 命令提示符 */
}

/**
  * @brief 系統時脈配置
  */
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();
  }
}

/**
  * @brief GPIO □□□
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  __HAL_RCC_GPIOA_CLK_ENABLE();

  GPIO_InitStruct.Pin = GPIO_PIN_5;
  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);
}

/**
  * @brief USART1 □□□
  */
static void MX_USART1_UART_Init(void)
{
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  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;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;

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

  __HAL_UART_ENABLE_IT(&huart1, UART_IT_RXNE);
}

/**
  * @brief UART MSP □□□
  */
void HAL_UART_MspInit(UART_HandleTypeDef* huart)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  if(huart->Instance == USART1)
  {
    __HAL_RCC_USART1_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();

    GPIO_InitStruct.Pin = GPIO_PIN_9 | GPIO_PIN_10;
```

```
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    HAL_NVIC_SetPriority(USART1_IRQn, 1, 0);
    HAL_NVIC_EnableIRQ(USART1_IRQn);
  }
}

/**
  * @brief USART1 中斷處理
  */
void USART1_IRQHandler(void)
{
  HAL_UART_IRQHandler(&huart1);
}

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

? ?????

????

□ □□□□□

1. 編譯並燒錄程式
2. 開啟終端機軟體（ PuTTY、Arduino IDE 或 Tera Term）
3. 連接 COM 埠，波特率 115200
4. 應看到歡迎訊息: `STM32 UART Command Interface`
5. 輸入 `ON` → LED 點亮，列印 `LED ON`
6. 輸入 `OFF` → LED 熄滅，列印 `LED OFF`
7. 輸入 `TOGGLE` → LED 切換狀態

????

問題	原因	解決方案
無訊息輸出	UART 未初始化或波特率不對	檢查 CubeMX 中 USART1 設定，確認波特率為 115200
亂碼	波特率不匹配	改變終端機波特率，通常試試 9600 或 115200
無法接收命令	中斷未啟用	確認 `HAL_UART_Receive_IT()` 已調用
命令無反應	命令格式不正確	確保輸入 `ON`、`OFF` 或 `TOGGLE`（大小寫會自動轉換）

? ????

???????

優化版本使用圓形緩衝區防止緩衝區溢出

DMA + UART

使用 DMA 進行高效率的 UART 傳輸（後續課程）

? ????

???? - ? 5 ??ADC???????????? channel?

第 5 節將實現：

- **ADC** 單次轉換與掃描模式
- 軟體觸發與計時器觸發
- 多通道採樣

□ **UART □□□□□□□□□**

Revision #2
Created 2026-04-01 02:06:14 UTC by TaipeiTechRacing
Updated 2026-04-06 06:23:16 UTC