STM32 ???? ? 9 ??SPI ?? + DMA

? ????

1. 理解 **SPI** 高速序列通訊 - 掌握主從模式與時序配置
2. 實現 **SPI** 感測器通訊 - 與加速度計、存儲卡等設備通訊
3. 配置 **SPI + DMA** - 實現高速資料轉移

? SPI ????

SPI ????

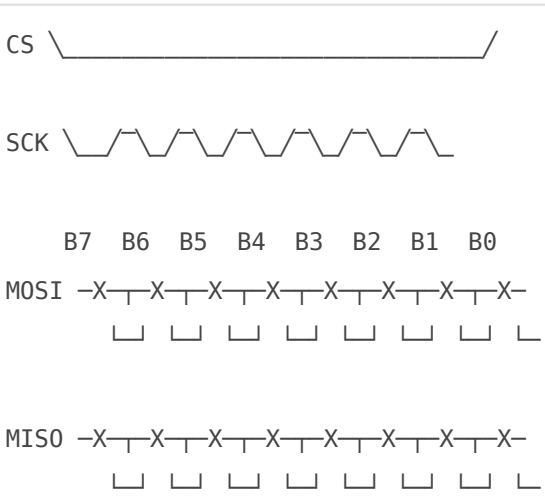
SPI (Serial Peripheral Interface) 是高速同步序列通訊協定。特點:

- 速度快：最高 54 MHz（STM32F446）
- 同步通訊：使用時鐘信號同步
- 全雙工：可同時收發
- 主從結構：一主多從

SPI ? 4 ????

訊號	名稱	方向	功能
MOSI	Master Out Slave In	主 → 從	主發送、從接收
MISO	Master In Slave Out	從 → 主	從發送、主接收
SCK	Serial Clock	主 → 從	時脈（由主產生）
CS/NSS	Chip Select	主 → 從	晶片選擇（低有效）

SPI ???



```
CS \_____________________________/

SCK \__/‾\_/‾\_/‾\_/‾\_/‾\_/‾\_

    B7  B6  B5  B4  B3  B2  B1  B0
MOSI ─X─┬─X─┬─X─┬─X─┬─X─┬─X─┬─X─┬─X─
       └─┘ └─┘ └─┘ └─┘ └─┘ └─┘ └─┘ └─

MISO ─X─┬─X─┬─X─┬─X─┬─X─┬─X─┬─X─┬─X─
       └─┘ └─┘ └─┘ └─┘ └─┘ └─┘ └─┘ └─
```

STM32F446 SPI ??

特性	規格
SPI 模組	3 個（SPI1、SPI2、SPI3）
最高速度	54 MHz（SPI1）、27 MHz（SPI2/3）
DMA 支援	是（分別用於收發）
模式	SPI、I²S、Simplex（單向）

?? ????

????

元件	數量	說明
SPI 從設備（MPU6050、SD卡等）	1	任選
跳線	4	MOSI、MISO、SCK、CS

???

Nucleo 接腳	**SPI1**	功能	從設備連接
PA5	SCK	時鐘	SCL 或 SCK
PA6	MISO	接收	MISO 或 DO
PA7	MOSI	發送	MOSI 或 DIN

Nucleo 接腳	SPI1	功能	從設備連接
PA4	CS	選擇	CS 或 CE

???????

```
Nucleo F446RC              從設備 (如 MPU6050)
──────────────             ──────────────────

PA5 (SCK) ──────────────── SCL
PA6 (MISO) ─────────────── SDA/MISO
PA7 (MOSI) ─────────────── MOSI
PA4 (CS)  ──────────────── CS
3.3V ──────────────────── VCC
GND ──────────────────── GND
```

?? CubeMX ????

?? 1??? SPI1

1. 開啟 CubeMX
2. 在 Pinout 圖中選擇以下接腳:
   - **PA5**: 配置為 **SPI1_SCK**
   - **PA6**: 配置為 **SPI1_MISO**
   - **PA7**: 配置為 **SPI1_MOSI**
   - **PA4**: 配置為 **GPIO_Output** (用於 CS 控制)

?? 2??? SPI ??

1. 在左側選擇 **Connectivity → SPI1**
2. **Mode**: Full-Duplex Master
3. **Configuration** 中設定:
   - **Frame Format**: Motorola (通常使用)
   - **Data Size**: 8 Bits
   - **Prescaler**: 8 (設定 SPI 速度 = 168MHz / 8 = 21 MHz)
   - **CPOL (Clock Polarity)**: Low
   - **CPHA (Clock Phase)**: 1 Edge (根據從設備選擇)
   - **NSS (Chip Select Mode)**: Software (手動控制 CS)

?? 3??? DMA????????

1. 在 **DMA Settings** 中:
   - **Add** → 配置 Tx DMA: DMA2 Stream3 Channel3
   - **Add** → 配置 Rx DMA: DMA2 Stream2 Channel3
2. 分別設定:
   - **Mode**: Normal
   - **Increment Address**: Enable (Memory)
   - **Data Width**: Byte

?? 4?????

NVIC Settings 中勾選:

- **SPI1 global interrupt**
- **DMA2 Stream2 global interrupt**（Rx）
- **DMA2 Stream3 global interrupt**（Tx）

?? 5??????

點擊 **Generate Code**

? ?????

main.c - SPI DMA ??

```
/* STM32 Lesson 09 - SPI with DMA
 * 功能: 使用 SPI + DMA 進行高速通訊
 * 難度: 高級
 */

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

/* 定義 */
```

```
#define SPI_TX_BUFFER_SIZE 256
#define SPI_RX_BUFFER_SIZE 256
#define CS_PORT GPIOA
#define CS_PIN GPIO_PIN_4

/* □□□□ */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
static void MX_DMA_Init(void);
static void MX_USART1_UART_Init(void);
void UART_Print(const char *format, ...);
void SPI_CS_Enable(void);
void SPI_CS_Disable(void);
void SPI_Transmit_DMA(uint8_t *tx_data, uint16_t size);
void SPI_Receive_DMA(uint8_t *rx_data, uint16_t size);

/* □□□□ */
SPI_HandleTypeDef hspi1;
DMA_HandleTypeDef hdma_spi1_rx;
DMA_HandleTypeDef hdma_spi1_tx;
UART_HandleTypeDef huart1;

uint8_t spi_tx_buffer[SPI_TX_BUFFER_SIZE];
uint8_t spi_rx_buffer[SPI_RX_BUFFER_SIZE];

volatile uint8_t spi_tx_complete = 0;
volatile uint8_t spi_rx_complete = 0;

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_SPI1_Init();
  MX_USART1_UART_Init();

  UART_Print("\r\n=== SPI DMA Test ===\r\n");
```

```
  /* 準備發送資料 */
  for (uint16_t i = 0; i < 8; i++)
  {
    spi_tx_buffer[i] = 0xA0 + i;  /* 填充測試資料 */
  }

  while (1)
  {
    UART_Print("Sending via SPI DMA...\r\n");

    /* 使用 DMA 發送 */
    SPI_Transmit_DMA(spi_tx_buffer, 8);

    /* 等待完成 */
    while (!spi_tx_complete);
    spi_tx_complete = 0;

    UART_Print("Sent: ");
    for (uint16_t i = 0; i < 8; i++)
    {
      UART_Print("0x%02X ", spi_tx_buffer[i]);
    }
    UART_Print("\r\n");

    HAL_Delay(1000);
  }
}

/**
  * @brief SPI 片選使能（拉低）
  */
void SPI_CS_Enable(void)
{
  HAL_GPIO_WritePin(CS_PORT, CS_PIN, GPIO_PIN_RESET);
  HAL_Delay(1);  /* 等待片選穩定 */
}

/**
  * @brief SPI 片選禁用（拉高）
  */
```

```
void SPI_CS_Disable(void)
{
  HAL_Delay(1);  /* 等待最後一位傳完 */
  HAL_GPIO_WritePin(CS_PORT, CS_PIN, GPIO_PIN_SET);
}

/**
  * @brief 使用 DMA 發送資料
  */
void SPI_Transmit_DMA(uint8_t *tx_data, uint16_t size)
{
  SPI_CS_Enable();
  HAL_SPI_Transmit_DMA(&hspi1, tx_data, size);
}

/**
  * @brief 使用 DMA 接收資料
  */
void SPI_Receive_DMA(uint8_t *rx_data, uint16_t size)
{
  SPI_CS_Enable();
  HAL_SPI_Receive_DMA(&hspi1, rx_data, size);
}

/**
  * @brief SPI 發送完成中斷回調
  */
void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef *hspi)
{
  if (hspi->Instance == SPI1)
  {
    spi_tx_complete = 1;
    SPI_CS_Disable();
  }
}

/**
  * @brief SPI 接收完成中斷回調
  */
void HAL_SPI_RxCpltCallback(SPI_HandleTypeDef *hspi)
```

```
{
  if (hspi->Instance == SPI1)
  {
    spi_rx_complete = 1;
    SPI_CS_Disable();
  }
}

/**
  * @brief UART □□□□
  */
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 □□□□□□
  */
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();

  /* □□ PA4 □ CS □□ */
  GPIO_InitStruct.Pin = GPIO_PIN_4;
  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);

  /* □□□ CS □□□□□□□□□□ */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
}
```

```
/**
  * @brief SPI1 □□□
  */
static void MX_SPI1_Init(void)
{
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;

  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief DMA □□□
  */
static void MX_DMA_Init(void)
{
  __HAL_RCC_DMA2_CLK_ENABLE();

  HAL_NVIC_SetPriority(DMA2_Stream2_IRQn, 1, 0);
  HAL_NVIC_EnableIRQ(DMA2_Stream2_IRQn);

  HAL_NVIC_SetPriority(DMA2_Stream3_IRQn, 1, 0);
  HAL_NVIC_EnableIRQ(DMA2_Stream3_IRQn);
}

/**
  * @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;

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

/**
  * @brief SPI1 MSP □□□
  */
void HAL_SPI_MspInit(SPI_HandleTypeDef* hspi)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  if(hspi->Instance == SPI1)
  {
    __HAL_RCC_SPI1_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();

    GPIO_InitStruct.Pin = GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    hdma_spi1_rx.Instance = DMA2_Stream2;
    hdma_spi1_rx.Init.Channel = DMA_CHANNEL_3;
    hdma_spi1_rx.Init.Direction = DMA_PERIPH_TO_MEMORY;
    hdma_spi1_rx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_spi1_rx.Init.MemInc = DMA_MINC_ENABLE;
```

```
    hdma_spi1_rx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
    hdma_spi1_rx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
    hdma_spi1_rx.Init.Mode = DMA_NORMAL;
    hdma_spi1_rx.Init.Priority = DMA_PRIORITY_HIGH;
    HAL_DMA_Init(&hdma_spi1_rx);

    __HAL_LINKDMA(hspi, hdmarx, hdma_spi1_rx);

    hdma_spi1_tx.Instance = DMA2_Stream3;
    hdma_spi1_tx.Init.Channel = DMA_CHANNEL_3;
    hdma_spi1_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
    hdma_spi1_tx.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_spi1_tx.Init.MemInc = DMA_MINC_ENABLE;
    hdma_spi1_tx.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
    hdma_spi1_tx.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
    hdma_spi1_tx.Init.Mode = DMA_NORMAL;
    hdma_spi1_tx.Init.Priority = DMA_PRIORITY_HIGH;
    HAL_DMA_Init(&hdma_spi1_tx);

    __HAL_LINKDMA(hspi, hdmatx, hdma_spi1_tx);

    HAL_NVIC_SetPriority(SPI1_IRQn, 1, 0);
    HAL_NVIC_EnableIRQ(SPI1_IRQn);
  }
}

/**
  * @brief SPI1 □□□□
  */
void SPI1_IRQHandler(void)
{
  HAL_SPI_IRQHandler(&hspi1);
}

/**
  * @brief DMA2 Stream2 □□□□
  */
void DMA2_Stream2_IRQHandler(void)
{
  HAL_DMA_IRQHandler(&hdma_spi1_rx);
```

```
}

/**
  * @brief DMA2 Stream3 中斷處理
  */
void DMA2_Stream3_IRQHandler(void)
{
  HAL_DMA_IRQHandler(&hdma_spi1_tx);
}

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

? ?????

????

□ □□□□ :

- UART 輸出 "Sending via SPI DMA..."
- 每秒列印一次發送的資料（0xA0 ~ 0xA7）
- LED 應正常工作（如有連接）

????

問題	原因	解決方案
無 **UART** 輸出	UART 未初始化	檢查 MX_USART1_UART_Init()
SPI 無反應	DMA 配置錯誤	驗證 DMA Stream 和 Channel 設定
傳輸速度慢	Prescaler 過大	減小 Prescaler 值提高速度
資料錯誤	CPOL/CPHA 不匹配	根據從設備調整時序

? ????

SPI ? SD ???

```
/* SD 卡初始化 */
#define SD_CS_PORT GPIOA
#define SD_CS_PIN GPIO_PIN_4

void SD_Init(void)
{
  SPI_CS_Enable();
  /* 發送初始化命令 CMD0 */
  uint8_t cmd[6] = {0x40, 0x00, 0x00, 0x00, 0x00, 0x95};
  HAL_SPI_Transmit(&hspi1, cmd, 6, 100);
  SPI_CS_Disable();
}
```

? ????

???? - ? 10 ??CANbus ?????????

第 10 節將實現:

- **CAN** 協定與識別符
- **TJA1050** 收發器接線
- **Nucleo ↔ Nucleo** 雙板通訊

 SPI □□□□□□□□

Revision #2
Created 2026-04-01 02:06:22 UTC by TaipeiTechRacing
Updated 2026-04-06 06:23:53 UTC