STM32 ???? ? 10 ??CANbus ?????????

? ????

1. 理解 **CAN** 協定基礎 - 掌握識別符、優先級仲裁與訊息過濾
2. 配置 **TJA1050** 收發器 - 實現 Nucleo 與 Nucleo 的 CAN 通訊
3. 實現雙板訊息收發 - 一片做發送方，另一片做接收方

? CAN ????

CAN ????

CAN (Controller Area Network) 是車用工業級網路協定。特點:

- 強抗干擾：差分信號設計
- 實時性好：優先級仲裁確保關鍵訊息先傳
- 多節點：最多 127 個節點共享一條總線
- 長距離：可達 40km（低速）或 1km（高速）

CAN ????

欄位	長度	說明
ID	11 bit（標準） / 29 bit（擴展）	訊息識別符，用於優先級仲裁
DLC	4 bit	資料長度碼（0-8 bytes）
DATA	0-8 bytes	實際負載資料
CRC	15 bit	循環冗餘碼（硬體自動計算）

CAN ? 2 ????

訊號	說明

CAN_H	CAN 高位線
CAN_L	CAN 低位線
GND	共同接地

STM32F446 CAN ??

特性	規格
CAN 模組	2 個（ CAN1、CAN2）
最高速度	1 Mbps
訊息濾波器	14 個過濾器組
接收 **FIFO**	2 個（每個 3 訊息）
傳輸郵箱	3 個

?? ????

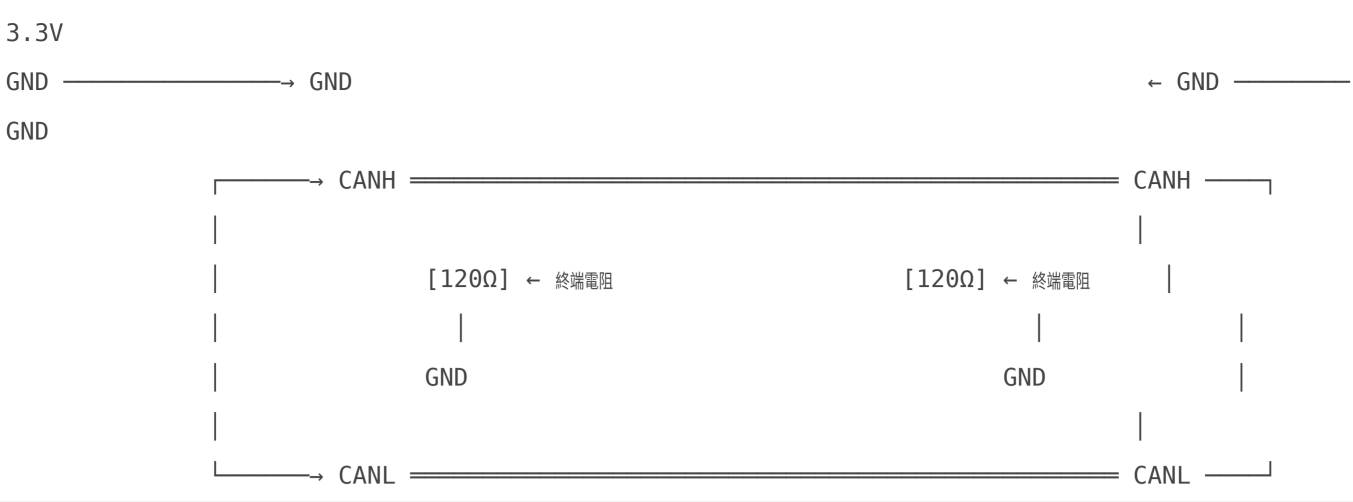
????

元件	數量	說明
Nucleo-F446RC	2	一發一收
TJA1050 CAN 收發器	2	轉換 CAN 信號
120Ω 終端電阻	2	CAN 總線兩端各 1
跳線	多條	連接所有元件

CAN ????

```
Nucleo 1 (發送方)      TJA1050 收發器        CAN 總線       TJA1050 收發器       Nucleo 2 (接收方)
──────────────────     ──────────────────    ──────────     ──────────────────   ──────────────────

PB8 (CAN1_RX) ──────→ RXD                                                ← RXD ─── PB8 (CAN1_RX)
PB9 (CAN1_TX) ←────── TXD                                                ──→ TXD ─── PB9 (CAN1_TX)
3.3V ──────────────→ VCC                                                 ← VCC ──────
```



???

| Nucleo PB8 | → | TJA1050 RXD | | Nucleo PB9 | ← | TJA1050 TXD | | Nucleo 3.3V | → | TJA1050 VCC | | Nucleo GND | → | TJA1050 GND | | TJA1050 CANH | = | CAN 總線 H | | TJA1050 CANL | = | CAN 總線 L |

?? CubeMX ????

?? 1??? CAN1???????????

1. 開啟 CubeMX
2. 在 Pinout 圖中選擇:
   - **PB8**: 配置為 **CAN1_RX**
   - **PB9**: 配置為 **CAN1_TX**

?? 2??? CAN ??

1. 在左側選擇 **Connectivity → CAN1**
2. **Activated**: 勾選啟用
3. **Parameter Settings** 中設定:
   - **Mode**: Normal Mode
   - **Prescaler**: 8 (波特率 = 168MHz / 8 / 13 ≈ 1.615 Mbps, 調整至 1 Mbps)
   - **Time Quanta in Bit Segment 1**: 11
   - **Time Quanta in Bit Segment 2**: 2
   - **Resynchronization Jump Width**: 1
4. **Filter Settings**: 配置過濾器
   - **Number of Master Filters**: 14
   - **Filters Configuration**:

- **Filter ID**：0x000 (接收所有 ID)
- **Filter Mask**：0x000
- **Filter FIFO Assignment**：FIFO0
- **Filter Activation**：Enable

?? 3?????

NVIC Settings 中勾選:

- **CAN1 RX0 interrupt**
- **CAN1 TX interrupt**

?? 4??????

點擊 **Generate Code**

? ?????

??? (Nucleo 1)?main.c

```
/* STM32 Lesson 10 - CAN Bus (Sender)
 * 功能: 發送 CAN 訊息到另一片 Nucleo
 * 難度: 中級
 */

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

#define CAN_ID_TEST 0x123  /* CAN 訊息 ID */

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_CAN1_Init(void);
static void MX_USART1_UART_Init(void);
void UART_Print(const char *format, ...);
```

```
CAN_HandleTypeDef hcan1;
UART_HandleTypeDef huart1;

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_CAN1_Init();
  MX_USART1_UART_Init();

  UART_Print("\r\n=== CAN Bus Sender ===\r\n");

  CAN_TxHeaderTypeDef TxHeader;
  uint8_t TxData[8];
  uint32_t TxMailbox;

  /* 配置發送訊息頭 */
  TxHeader.StdId = CAN_ID_TEST;
  TxHeader.IDE = CAN_ID_STD;     /* 標準 ID */
  TxHeader.RTR = CAN_RTR_DATA;   /* 資料訊息（非遠程） */
  TxHeader.DLC = 8;              /* 資料長度 8 bytes */

  uint32_t counter = 0;

  while (1)
  {
    /* 準備發送資料 */
    TxData[0] = (counter >> 24) & 0xFF;
    TxData[1] = (counter >> 16) & 0xFF;
    TxData[2] = (counter >> 8) & 0xFF;
    TxData[3] = counter & 0xFF;
    TxData[4] = 0xAA;
    TxData[5] = 0xBB;
    TxData[6] = 0xCC;
    TxData[7] = 0xDD;

    /* 發送訊息 */
```

```
    if (HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox) == HAL_OK)
    {
      UART_Print("CAN Sent: ID=0x%03X, Data=[%02X %02X %02X %02X %02X %02X %02X %02X]\r\n",
        CAN_ID_TEST,
        TxData[0], TxData[1], TxData[2], TxData[3],
        TxData[4], TxData[5], TxData[6], TxData[7]);
      counter++;
    }
    else
    {
      UART_Print("CAN Send Failed!\r\n");
    }

    HAL_Delay(1000);
  }
}

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);
}

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)
{
  __HAL_RCC_GPIOA_CLK_ENABLE();
}

static void MX_CAN1_Init(void)
{
  hcan1.Instance = CAN1;
  hcan1.Init.Prescaler = 6;
  hcan1.Init.Mode = CAN_MODE_NORMAL;
  hcan1.Init.SyncJumpWidth = CAN_SJW_1TQ;
  hcan1.Init.TimeSeg1 = CAN_BS1_11TQ;
  hcan1.Init.TimeSeg2 = CAN_BS2_2TQ;
  hcan1.Init.TimeTriggeredMode = DISABLE;
  hcan1.Init.AutoBusOff = DISABLE;
  hcan1.Init.AutoWakeUp = DISABLE;
  hcan1.Init.AutoRetransmission = ENABLE;
  hcan1.Init.ReceiveFifoLocked = DISABLE;
  hcan1.Init.TransmitFifoPriority = DISABLE;
```

```
  if (HAL_CAN_Init(&hcan1) != HAL_OK)
    Error_Handler();

  HAL_CAN_Start(&hcan1);
}

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();
}

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

??? (Nucleo 2)?main.c

```
/* STM32 Lesson 10 - CAN Bus (Receiver)
 * 功能: 接收另一片 Nucleo 的 CAN 訊息
 * 難度: 中級
 */

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

#define CAN_ID_TEST 0x123
```

```
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_CAN1_Init(void);
static void MX_USART1_UART_Init(void);
void UART_Print(const char *format, ...);

CAN_HandleTypeDef hcan1;
UART_HandleTypeDef huart1;

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_CAN1_Init();
  MX_USART1_UART_Init();

  UART_Print("\r\n=== CAN Bus Receiver ===\r\n");

  CAN_FilterTypeDef sFilterConfig;
  CAN_RxHeaderTypeDef RxHeader;
  uint8_t RxData[8];

  /* □□□□□□□□ */
  sFilterConfig.FilterBank = 0;
  sFilterConfig.FilterMode = CAN_FILTERMODE_IDMASK;
  sFilterConfig.FilterScale = CAN_FILTERSCALE_32BIT;
  sFilterConfig.FilterIdHigh = (CAN_ID_TEST << 5) >> 16;
  sFilterConfig.FilterIdLow = (CAN_ID_TEST << 5) & 0xFFFF;
  sFilterConfig.FilterMaskIdHigh = 0xFFFF;
  sFilterConfig.FilterMaskIdLow = 0xFFFF;
  sFilterConfig.FilterFIFOAssignment = CAN_RX_FIFO0;
  sFilterConfig.FilterActivation = ENABLE;

  if (HAL_CAN_ConfigFilter(&hcan1, &sFilterConfig) != HAL_OK)
    Error_Handler();

  if (HAL_CAN_Start(&hcan1) != HAL_OK)
    Error_Handler();
```

```
  /* □□□□□□ */
  if (HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING) != HAL_OK)
    Error_Handler();

  UART_Print("Waiting for CAN messages...\r\n");

  while (1)
  {
    if (HAL_CAN_GetRxFifoFillLevel(&hcan1, CAN_RX_FIFO0) > 0)
    {
      if (HAL_CAN_GetRxMessage(&hcan1, CAN_RX_FIFO0, &RxHeader, RxData) == HAL_OK)
      {
        UART_Print("CAN Received: ID=0x%03X, DLC=%d, Data=[%02X %02X %02X %02X %02X %02X %02X
%02X]\r\n",
          RxHeader.StdId, RxHeader.DLC,
          RxData[0], RxData[1], RxData[2], RxData[3],
          RxData[4], RxData[5], RxData[6], RxData[7]);
      }
    }
    HAL_Delay(100);
  }
}

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);
}

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)
{
  __HAL_RCC_GPIOA_CLK_ENABLE();
}

static void MX_CAN1_Init(void)
{
  hcan1.Instance = CAN1;
  hcan1.Init.Prescaler = 6;
  hcan1.Init.Mode = CAN_MODE_NORMAL;
  hcan1.Init.SyncJumpWidth = CAN_SJW_1TQ;
  hcan1.Init.TimeSeg1 = CAN_BS1_11TQ;
  hcan1.Init.TimeSeg2 = CAN_BS2_2TQ;
  hcan1.Init.TimeTriggeredMode = DISABLE;
```

```
  hcan1.Init.AutoBusOff = DISABLE;
  hcan1.Init.AutoWakeUp = DISABLE;
  hcan1.Init.AutoRetransmission = ENABLE;
  hcan1.Init.ReceiveFifoLocked = DISABLE;
  hcan1.Init.TransmitFifoPriority = DISABLE;

  if (HAL_CAN_Init(&hcan1) != HAL_OK)
    Error_Handler();
}

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();
}

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

? ?????

????

UART

```
=== CAN Bus Sender ===
CAN Sent: ID=0x123, Data=[00 00 00 00 AA BB CC DD]
CAN Sent: ID=0x123, Data=[00 00 00 01 AA BB CC DD]
```

```
CAN Sent: ID=0x123, Data=[00 00 00 02 AA BB CC DD]
```

UART

```
=== CAN Bus Receiver ===
Waiting for CAN messages...
CAN Received: ID=0x123, DLC=8, Data=[00 00 00 00 AA BB CC DD]
CAN Received: ID=0x123, DLC=8, Data=[00 00 00 01 AA BB CC DD]
CAN Received: ID=0x123, DLC=8, Data=[00 00 00 02 AA BB CC DD]
```

????

		Prescaler/TimeSeg
	CAN	120Ω
		HAL_CAN_GetTxMailboxesFreeLevel()

? ????

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

11

- **RS-485**
-
-

CAN

Revision #2
Created 2026-04-01 02:06:25 UTC by TaipeiTechRacing
Updated 2026-04-06 06:24:10 UTC