????CODE??

STM32 CAN ??? + UART ????

專案名稱 : TTR9.5_eco_VCU

> 本文檔針對提供的 STM32F4 韌體進行詳細分析，說明如何使用 CAN1、CAN2 和 UART 通訊模組。

? ??

- 硬體初始化
- CAN1 使用區段
- CAN2 使用區段
- UART 使用區段
- 訊息接收與發送
- 實現步驟

?????

???????

```
/* main() 函數中的初始化流程 */
MX_GPIO_Init();           // GPIO 初始化
MX_CAN1_Init();           // CAN1 初始化
MX_CAN2_Init();           // CAN2 初始化
MX_USART1_UART_Init();    // UART1 初始化
MX_USART6_UART_Init();    // UART6 初始化
```

image

初始化順序重要性:

- GPIO 必須先初始化，為其他外設提供引腳支援
- CAN 和 UART 初始化順序可互換

CAN1 ????

1?? **CAN1 ?????** (`MX_CAN1_Init`)

```
static void MX_CAN1_Init(void)
{
  hcan1.Instance = CAN1;
  hcan1.Init.Prescaler = 5;               // 波特率預分頻: 決定 CAN 速度
  hcan1.Init.Mode = CAN_MODE_NORMAL;      // 正常模式
  hcan1.Init.SyncJumpWidth = CAN_SJW_1TQ;
  hcan1.Init.TimeSeg1 = CAN_BS1_15TQ;     // 位元時序段 1
  hcan1.Init.TimeSeg2 = CAN_BS2_2TQ;      // 位元時序段 2
  hcan1.Init.TimeTriggeredMode = DISABLE;
  hcan1.Init.AutoBusOff = DISABLE;
  hcan1.Init.AutoWakeUp = DISABLE;
  hcan1.Init.AutoRetransmission = DISABLE;
  hcan1.Init.ReceiveFifoLocked = DISABLE;
  hcan1.Init.TransmitFifoPriority = DISABLE;

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

關鍵參數:

參數	值	說明
`Prescaler`	5	影響 CAN 波特率計算
`TimeSeg1`	15TQ	時序段 1 = 15 時間量子
`TimeSeg2`	2TQ	時序段 2 = 2 時間量子
`SyncJumpWidth`	1TQ	同步跳躍寬度

波特率計算:

```
CAN_BaudRate = APB1_Clock / (Prescaler × (1 + TimeSeg1 + TimeSeg2))
             = 45MHz / (5 × (1 + 15 + 2))
             = 45MHz / (5 × 18)
             = 500 kbps
```

2?? CAN1 ?????

```
CAN_FilterTypeDef sFilterConfig;

// 配置 CAN1 過濾器
sFilterConfig.FilterBank = 0;                // 使用第 0 號過濾器
sFilterConfig.FilterMode = CAN_FILTERMODE_IDMASK;
sFilterConfig.FilterScale = CAN_FILTERSCALE_32BIT;
sFilterConfig.FilterIdHigh = 0x0000;
sFilterConfig.FilterIdLow = 0x0000;
sFilterConfig.FilterMaskIdHigh = 0x0000;
sFilterConfig.FilterMaskIdLow = 0x0000;
sFilterConfig.FilterFIFOAssignment = CAN_RX_FIFO0;
sFilterConfig.FilterActivation = ENABLE;
sFilterConfig.SlaveStartFilterBank = 14;   // CAN2 從第 14 號過濾器開始

HAL_CAN_ConfigFilter(&hcan1, &sFilterConfig);
HAL_CAN_Start(&hcan1);
HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING);
```

過濾器說明:

- **FilterBank 0-13**: CAN1 使用
- **FilterBank 14-27**: CAN2 使用（ Slave 模式）
- **FilterIdLow = 0x0000 + FilterMaskIdLow = 0x0000**: 接收所有訊息
- **CAN_RX_FIFO0**: 接收訊息存儲到 FIFO0
- **ActivateNotification**: 啟用 FIFO0 新訊息中斷

3?? CAN1 ???? (`CAN1_Send_Test`)

```
void CAN1_Send_Test(uint8_t counter)
{
    // 設定發送訊息頭
```

```
    TxHeader1.StdId = 0x101;           // CAN ID: 0x101
    TxHeader1.RTR = CAN_RTR_DATA;      // 資料幀（非遠端幀）
    TxHeader1.IDE = CAN_ID_STD;        // 標準 ID（11-bit）
    TxHeader1.DLC = 8;                 // 資料長度: 8 bytes
    TxHeader1.TransmitGlobalTime = DISABLE;

    // 準備 8 bytes 資料
    TxData1[0] = counter;              // byte 0: 計數器
    TxData1[1] = 0x22;
    TxData1[2] = 0x33;
    TxData1[3] = 0x00;
    TxData1[4] = 0x00;
    TxData1[5] = 0x00;
    TxData1[6] = 0x00;
    TxData1[7] = 0x00;

    // 檢查信箱有無空位並發送
    if (HAL_CAN_GetTxMailboxesFreeLevel(&hcan1) > 0)
    {
        if (HAL_CAN_AddTxMessage(&hcan1, &TxHeader1, TxData1, &TxMailbox1) != HAL_OK)
        {
            // 發送失敗處理
        }
    }
}
```

發送流程:

1. 配置 `TxHeader1` 結構體
2. 填充 `TxData1` 陣列（最多 8 bytes）
3. 檢查發送信箱是否有空位
4. 調用 `HAL_CAN_AddTxMessage()` 發送

在主循環中調用:

```
while(1)
{
    if (HAL_GetTick() - last_uart_tick >= 300)
    {
        Send_uart[86]++;
        HAL_UART_Transmit(&huart1, Send_uart, sendData_lenght + 3, 1000);
```

```
        CAN1_Send_Test(test_count++);  // ← 每 300ms 發送一次 CAN 訊息
        last_uart_tick = HAL_GetTick();
    }
}
```

CAN2 ????

1?? **CAN2 ?????** (`MX_CAN2_Init`)

```
static void MX_CAN2_Init(void)
{
  hcan2.Instance = CAN2;
  hcan2.Init.Prescaler = 5;
  hcan2.Init.Mode = CAN_MODE_NORMAL;
  hcan2.Init.SyncJumpWidth = CAN_SJW_1TQ;
  hcan2.Init.TimeSeg1 = CAN_BS1_15TQ;
  hcan2.Init.TimeSeg2 = CAN_BS2_2TQ;
  hcan2.Init.TimeTriggeredMode = DISABLE;
  hcan2.Init.AutoBusOff = DISABLE;
  hcan2.Init.AutoWakeUp = DISABLE;
  hcan2.Init.AutoRetransmission = DISABLE;
  hcan2.Init.ReceiveFifoLocked = DISABLE;
  hcan2.Init.TransmitFifoPriority = DISABLE;

  if (HAL_CAN_Init(&hcan2) != HAL_OK)
  {
    Error_Handler();
  }
}
```

與 **CAN1** 相同的配置參數

2?? **CAN2 ?????**

```
// 配置 CAN2 過濾器
sFilterConfig.FilterBank = 14;                  // ← 使用第 14 號過濾器 (Slave 區間起點)
sFilterConfig.FilterMode = CAN_FILTERMODE_IDMASK;
```

```
sFilterConfig.FilterScale = CAN_FILTERSCALE_32BIT;
sFilterConfig.FilterIdHigh = 0x0000;
sFilterConfig.FilterIdLow = 0x0000;
sFilterConfig.FilterMaskIdHigh = 0x0000;
sFilterConfig.FilterMaskIdLow = 0x0000;
sFilterConfig.FilterFIFOAssignment = CAN_RX_FIFO0;
sFilterConfig.FilterActivation = ENABLE;

HAL_CAN_ConfigFilter(&hcan2, &sFilterConfig);
HAL_CAN_Start(&hcan2);
HAL_CAN_ActivateNotification(&hcan2, CAN_IT_RX_FIFO0_MSG_PENDING);
```

CAN1 vs CAN2 過濾器分配:

```
CAN1: FilterBank 0-13   (14 個過濾器)
CAN2: FilterBank 14-27  (14 個過濾器, Slave 模式)
```

UART ????

1?? **UART1 ???** (Titania ????)

```
static void MX_USART1_UART_Init(void)
{
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 9600;              // ← 波特率: 9600 bps
  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;       // 收發模式
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;

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

UART1 用途:　　與 Titania 無線模組通訊

UART1 配置參數:

參數	值	說明
`BaudRate`	9600	低速通訊
`WordLength`	8B	8 位元資料
`StopBits`	1	1 個停止位元
`Parity`	NONE	無奇偶校驗

2?? **UART1 ????** (Titania ????)

```
// 全域變數定義
uint8_t sendData_lenght = 84;         //預傳送資料長度
uint8_t Send_uart[87];               // sendData_lenght + 3 (84 + 3)
uint32_t last_uart_tick = 0;         // 時間戳記
uint8_t Send_head[3] = {0x02, 0x00, 0x3A};  // 訊息標頭
uint8_t Send_checksum = 0;

// 主循環中的發送邏輯
while (1)
{
    if (HAL_GetTick() - last_uart_tick >= 300)
    {
        // 設定訊息頭
        Send_uart[0] = Send_head[0];   // 0x02
        Send_uart[1] = Send_head[1];   // 0x00
        Send_uart[2] = Send_head[2];   // 0x3A

        Send_uart[86]++;               // ← 計數器(最後一個 byte)

        // 發送 87 bytes(頭 3 bytes + 84 bytes 資料)
        HAL_UART_Transmit(&huart1, Send_uart, sendData_lenght + 3, 1000);

        last_uart_tick = HAL_GetTick();  // 更新時間戳記
    }
}
```

Titania 無線模組訊息格式:

| 頭 (3B) | 資料 (84B) |

發送間隔: 每 300ms 發送一次

3?? **UART6 ???** (??/????)

```
static void MX_USART6_UART_Init(void)
{
  huart6.Instance = USART6;
  huart6.Init.BaudRate = 115200;              // ← 波特率: 115200 bps (高速)
  huart6.Init.WordLength = UART_WORDLENGTH_8B;
  huart6.Init.StopBits = UART_STOPBITS_1;
  huart6.Init.Parity = UART_PARITY_NONE;
  huart6.Init.Mode = UART_MODE_TX_RX;
  huart6.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart6.Init.OverSampling = UART_OVERSAMPLING_16;

  if (HAL_UART_Init(&huart6) != HAL_OK)
  {
    Error_Handler();
  }
}
```

UART6 用途: IMU

UART6 配置參數:

參數	值	說明
`BaudRate`	115200	高速通訊
`WordLength`	8B	8 位元資料
`StopBits`	1	1 個停止位元
`Parity`	NONE	無奇偶校驗

???????

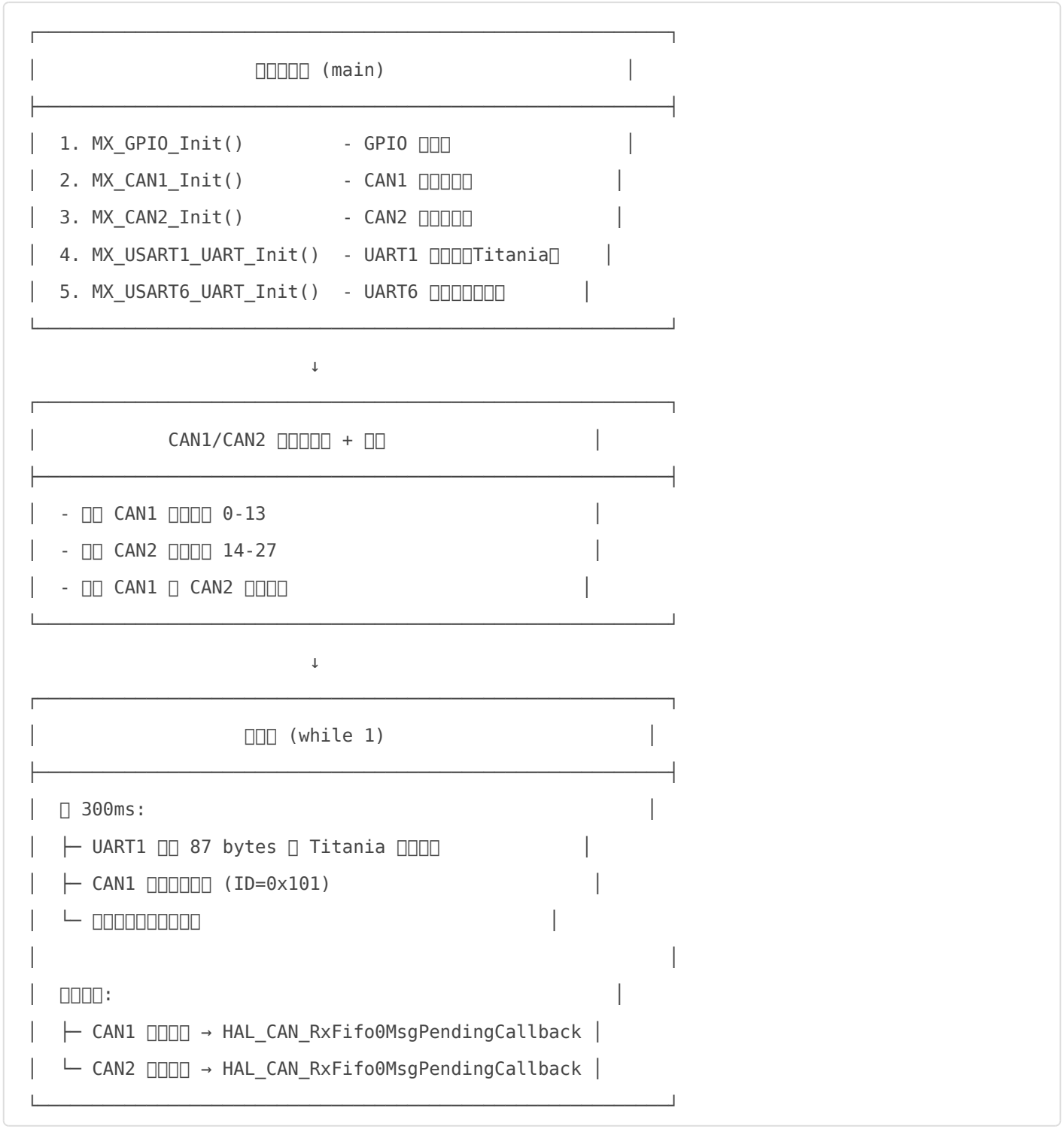
? **CAN ??????**

```
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    CAN_Counter++;  // 計算接收訊息總數

    if (hcan->Instance == CAN1)
    {
        // CAN1 接收訊息處理
        if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader1, RxData1) == HAL_OK)
        {
            // 判斷訊息 ID
            if (RxHeader1.StdId == 0x101)
            {
                // 處理 CAN1 ID=0x101 的訊息
                // RxData1[0] ~ RxData1[7]: 8 bytes 資料
            }
        }
    }
    else if (hcan->Instance == CAN2)
    {
        // CAN2 接收訊息處理
        if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader2, RxData2) == HAL_OK)
        {
            // 從 RxData2 提取編碼器數值 (4 bytes)
            // 資料順序: RxData2[3] (LSB) → RxData2[6] (MSB)
            raw_encoder_value = (uint32_t)(
                (RxData2[6] << 24) |    // 最高位 byte
                (RxData2[5] << 16) |
                (RxData2[4] << 8)  |
                RxData2[3]              // 最低位 byte
            );
        }
    }
}
```

關鍵流程:

1. 中斷觸發 : CAN FIFO0 有新訊息時自動執行
2. 判斷來源 : 檢查 `hcan->Instance` 區分 CAN1 或 CAN2
3. 讀取訊息 : 調用 `HAL_CAN_GetRxMessage()` 讀取訊息和資料
4. 處理資料 : 根據訊息 ID 進行相應處理



????

? ???????

```
┌─────────────────────────────────────────────────────────┐
│                    系統初始化 (main)                    │
├─────────────────────────────────────────────────────────┤
│  1. MX_GPIO_Init()         - GPIO 初始化                │
│  2. MX_CAN1_Init()         - CAN1 硬體初始化            │
│  3. MX_CAN2_Init()         - CAN2 硬體初始化            │
│  4. MX_USART1_UART_Init()  - UART1 初始化 (Titania)     │
│  5. MX_USART6_UART_Init()  - UART6 初始化 (調試)        │
└─────────────────────────────────────────────────────────┘
                          ↓
┌─────────────────────────────────────────────────────────┐
│             CAN1/CAN2 過濾器配置 + 啟動                 │
├─────────────────────────────────────────────────────────┤
│  - 配置 CAN1 過濾器組 0-13                              │
│  - 配置 CAN2 過濾器組 14-27                             │
│  - 啟動 CAN1 和 CAN2 接收中斷                           │
└─────────────────────────────────────────────────────────┘
                          ↓
┌─────────────────────────────────────────────────────────┐
│                    主循環 (while 1)                     │
├─────────────────────────────────────────────────────────┤
│  每 300ms:                                              │
│  ├─ UART1 發送 87 bytes 到 Titania 無線模組             │
│  ├─ CAN1 發送測試訊息 (ID=0x101)                        │
│  └─ 更新計數器和時間戳記                                │
│                                                         │
│  中斷事件:                                              │
│  ├─ CAN1 收到訊息 → HAL_CAN_RxFifo0MsgPendingCallback │
│  └─ CAN2 收到訊息 → HAL_CAN_RxFifo0MsgPendingCallback │
└─────────────────────────────────────────────────────────┘
```

? ??????

CAN1 ??

- [ ] `hcan1.Instance = CAN1`
- [ ] `Prescaler = 5` (波特率 500kbps)
- [ ] `FilterBank = 0` (使用第 0 號過濾器)
- [ ] `FilterFIFOAssignment = CAN_RX_FIFO0`
- [ ] 已調用 `HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING)`

CAN2 ??

- [ ] `hcan2.Instance = CAN2`
- [ ] `Prescaler = 5` (波特率 500kbps)
- [ ] `FilterBank = 14` (Slave 區間起點)
- [ ] `FilterFIFOAssignment = CAN_RX_FIFO0`
- [ ] 已調用 `HAL_CAN_ActivateNotification(&hcan2, CAN_IT_RX_FIFO0_MSG_PENDING)`

UART1 ???Titania?

- [ ] `BaudRate = 9600`
- [ ] 訊息格式: 頭 (3B) + 資料 (84B) = 87 bytes
- [ ] 發送間隔: 300ms
- [ ] 計數器在 `Send_uart[86]`

UART6 ??????

- [ ] `BaudRate = 115200`
- [ ] 用於高速調試輸出

? ?????

功能	CAN1	CAN2	UART1	UART6
實例	CAN1	CAN2	USART1	USART6
波特率	500kbps	500kbps	9600	115200
過濾器	Bank 0-13	Bank 14-27	-	-
用途	數據收發	編碼器 /感測器	無線模組	調試
中斷	FIFO0	FIFO0	-	-

功能	CAN1	CAN2	UART1	UART6
發送函數	`CAN1_Send_Test()`	-	`HAL_UART_Transmit()`	`HAL_UART_Transmit()`

? ????

????

Q1: 如何改變 **CAN** 波特率?

```
// 修改 Prescaler 計算
// CAN_BaudRate = 45MHz / (Prescaler × 18)
// 例如要 250kbps: Prescaler = 10
hcan1.Init.Prescaler = 10;  // 250 kbps
```

Q2: 如何接收特定 **CAN ID** 的訊息?

```
// 修改過濾器設定
sFilterConfig.FilterIdHigh = 0x2020;     // ID 高位
sFilterConfig.FilterIdLow = 0x0000;      // ID 低位
sFilterConfig.FilterMaskIdHigh = 0xFFFF; // 遮罩高位
sFilterConfig.FilterMaskIdLow = 0x0000;  // 遮罩低位
```

Q3: UART 傳送超時怎麼辦?

```
// 增加超時時間或使用非阻斷模式
HAL_UART_Transmit(&huart1, Send_uart, 87, 5000);  // 5秒超時


// 或使用 DMA 模式
HAL_UART_Transmit_DMA(&huart1, Send_uart, 87);
```

文檔版本: 1.0
最後更新: 2026年 3月 5日
適用裝置: STM32F4 系列

Revision #2
Created 2026-04-01 02:06:37 UTC by TaipeiTechRacing
Updated 2026-04-11 14:33:45 UTC by AI Agent