STM32 ????? VESC ???? (4WD_RWD ????)

? ?????????

本專案使用 STM32 作為核心控制器，透過 CAN Bus 與 4 顆 VESC 進行通訊，實現四輪獨立驅動與動態扭矩分配。

- 電池系統：48V 標稱電壓，總輸出功率限制 1000W。
- 前輪配置：250W 減速電機（內建單向離合器，具備滑行優勢）。
- 後輪配置：500W 減速電機。
- 輪胎規格：20 吋車輪，減速比 1:4.4，馬達額定轉速 390 RPM。
- 控制策略：低速 4WD 起步（1:2 扭矩分配）$\rightarrow$ 高速切換純 RWD 巡航（前輪斷電滑行，後輪滿功率輸出）。

?? ????????? (Bug Fixes & Improvements)

1. 修復 **CAN** 發送死迴圈：在 `VESC_Send_Current` 增加 Timeout 機制，防止 VESC 沒開機或斷線時導致 MCU 系統卡死。
2. 修復記憶體溢位 **(Buffer Overflow)**：在 `ENCODE_Step_up` 中將 `TxData` 陣列加大至 8 bytes，並修正 DLC (Data Length Code) 設定。
3. 優化 **ADC** 讀取：將 `HAL_ADC_Start()` 移出 `while(1)` 迴圈，利用硬體連續轉換模式 (Continuous Mode) 提高穩定性。
4. 增加 **CAN ID** 濾波保護：在接收中斷 `HAL_CAN_RxFifo0MsgPendingCallback` 內加入 `StdId` 檢查，防止雜訊誤觸發編碼器數值更新。

? ????? (`main.c`)

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
```

```
  * @file           : main.c
  * @brief          : STM32 VESC 4WD/RWD Dynamic Control System
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define POWER_LIMIT_W         1000    // 電池總功率限制 1000W
#define BATTERY_VOLTAGE_NOM   48      // 標稱電壓 48V

// 最大總電流 (mA)
#define MAX_TOTAL_CURRENT_MA  ((POWER_LIMIT_W * 1000) / BATTERY_VOLTAGE_NOM)

// 速度切換閾值 (ERPM: Electrical RPM)
// 依據 20吋輪、4.4減速比、390rpm 重新估算切換點
#define ERPM_THRESHOLD_RWD    10000
#define ERPM_HYSTERESIS       500     // 遲滯區間，避免臨界點頻繁切換

// VESC CAN ID 定義 (需與 VESC Tool 設定一致)
#define VESC_ID_FRONT_L       0xB1
#define VESC_ID_FRONT_R       0xB2
#define VESC_ID_REAR_L        0xA3
#define VESC_ID_REAR_R        0xA4

// VESC CAN Packet ID
#define CAN_PACKET_SET_CURRENT 1
#define ENCODER_RESOLUTION     4096.0f
#define STEERING_CENTER_OFFSET 2048.0f
/* USER CODE END PD */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
ADC_HandleTypeDef hadc2;
```

```
CAN_HandleTypeDef hcan1;

/* USER CODE BEGIN PV */
int32_t current_erpm = 0;       // 平均轉速 (由 CAN RX 更新)
uint16_t throttle_adc = 0;      // 0 - 4095
int32_t target_total_ma = 0;    // 總目標電流 (mA)
uint16_t speed_adc = 0;         // 模擬速度的 ADC

float current_angle = 0.0f;     // 0 ~ 360 度的絕對角度
float steering_angle = 0.0f;    // -180 ~ +180 度的方向盤實際轉角
uint32_t raw_encoder_value = 0; // 0 ~ 4095
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_CAN1_Init(void);
static void MX_ADC2_Init(void);

/* USER CODE BEGIN PFP */
// 狀態機定義: 四驅與後驅模式
typedef enum {
    MODE_4WD,
    MODE_RWD
} DriveMode_t;

DriveMode_t drive_mode = MODE_4WD;

void VESC_Send_Current(uint8_t controller_id, int32_t current_ma);
void Control_Loop_1kHz(void);
void ENCODE_Step_up(void);
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan);
/* USER CODE END PFP */

/**
  * @brief  The application entry point.
  */
int main(void)
{
  /* MCU Configuration--------------------------------------------------------*/
```

```
  HAL_Init();
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_CAN1_Init();
  MX_ADC2_Init();

  /* USER CODE BEGIN 2 */
  // 啟動 CAN 總線與中斷
  HAL_CAN_Start(&hcan1);
  HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING);

  // 【優化】將 ADC 啟動移出 while(1) 迴圈，使用連續轉換模式讓硬體自動更新
  HAL_ADC_Start(&hadc1);
  HAL_ADC_Start(&hadc2);
  /* USER CODE END 2 */

  /* Infinite loop */
  while (1)
  {
      Control_Loop_1kHz();
  }
}

/* USER CODE BEGIN 4 */

/**
 * @brief 主控制迴圈 (使用輪詢方式發送 CAN 避免 Bus 壅塞)
 */
void Control_Loop_1kHz(void)
{
    static uint32_t last_control_tick = 0;
    uint32_t current_tick = HAL_GetTick();

    // 控制執行頻率 (大約 500Hz)
    if ((current_tick - last_control_tick) < 2) {
        return;
    }
    last_control_tick = current_tick;
```

```
// 1. 讀取油門與速度模擬 ADC (0 ~ 4095)
throttle_adc = HAL_ADC_GetValue(&hadc1);
speed_adc = HAL_ADC_GetValue(&hadc2);
current_erpm = ((int32_t)speed_adc * 20000) >> 12;

// 2. 計算總需求電流 (mA)
target_total_ma = ((int32_t)throttle_adc * MAX_TOTAL_CURRENT_MA) >> 12;

// 3. 判斷模式 (根據 ERPM 絕對值)
int32_t abs_erpm = (current_erpm < 0) ? -current_erpm : current_erpm;

if (drive_mode == MODE_4WD) {
    if (abs_erpm > (ERPM_THRESHOLD_RWD + ERPM_HYSTERESIS)) {
        drive_mode = MODE_RWD;
    }
} else {
    if (abs_erpm < (ERPM_THRESHOLD_RWD - ERPM_HYSTERESIS)) {
        drive_mode = MODE_4WD;
    }
}

// 4. 動態電流分配 (Distribute Current)
int32_t current_front_axle = 0;
int32_t current_rear_axle = 0;

if (drive_mode == MODE_4WD) {
    // --- 4WD 模式 (起步高扭力) ---
    // 前 250W + 後 500W, 比例 1:2
    current_front_axle = target_total_ma / 3;
    current_rear_axle  = target_total_ma - current_front_axle;
} else {
    // --- RWD 模式 (高速巡航) ---
    // 前輪斷電 (利用離合器滑行), 後輪全功率輸出
    current_front_axle = 0;
    current_rear_axle  = target_total_ma;
}

// 5. 計算單顆電機電流並輪詢發送
int32_t cmd_front = current_front_axle / 2;
int32_t cmd_rear  = current_rear_axle / 2;
```

```
    static uint8_t motor_step = 0;
    switch(motor_step)
    {
        case 0:
            VESC_Send_Current(VESC_ID_FRONT_L, cmd_front);
            motor_step = 1;
            break;
        case 1:
            VESC_Send_Current(VESC_ID_FRONT_R, cmd_front);
            motor_step = 2;
            break;
        case 2:
            VESC_Send_Current(VESC_ID_REAR_L,  cmd_rear);
            motor_step = 3;
            break;
        case 3:
            VESC_Send_Current(VESC_ID_REAR_R,  cmd_rear);
            motor_step = 0;
            break;
    }
}

/**
 * @brief □□□□□□□□ VESC
 */
void VESC_Send_Current(uint8_t controller_id, int32_t current_ma)
{
    CAN_TxHeaderTypeDef TxHeader;
    uint32_t TxMailbox;
    uint8_t TxData[4];

    TxHeader.ExtId = (CAN_PACKET_SET_CURRENT << 8) | controller_id;
    TxHeader.IDE = CAN_ID_EXT;
    TxHeader.RTR = CAN_RTR_DATA;
    TxHeader.DLC = 4;

    // Big Endian packing
    TxData[0] = (uint8_t)(current_ma >> 24);
    TxData[1] = (uint8_t)(current_ma >> 16);
    TxData[2] = (uint8_t)(current_ma >> 8);
```

```
    TxData[3] = (uint8_t)(current_ma);

    // 【安全優化】加入 Timeout 防護，避免 CAN Bus 異常時系統卡死
    uint32_t timeout = 0;
    while (HAL_CAN_GetTxMailboxesFreeLevel(&hcan1) == 0)
    {
        timeout++;
        if (timeout > 50000) {
            return; // 丟棄此包，保護主程式
        }
    }
    HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox);
}

/**
 * @brief 發送編碼器設定命令
 */
void ENCODE_Step_up(void)
{
    CAN_TxHeaderTypeDef TxHeader;
    uint32_t TxMailbox;
    uint8_t TxData[8]; // 【修正】必須開到 8 bytes 以防溢位

    TxHeader.StdId = 0x01;
    TxHeader.IDE = CAN_ID_STD;
    TxHeader.RTR = CAN_RTR_DATA;
    TxHeader.DLC = 4;

    TxData[0] = 0x04;
    TxData[1] = 0x01;
    TxData[2] = 0x04;
    TxData[3] = 0xAA;
    HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox);

    // 第二包命令
    TxHeader.DLC = 5; // 【修正】長度改為 5
    TxData[0] = 0x05;
    TxData[1] = 0x01;
    TxData[2] = 0x05;
    TxData[3] = 0x88;
    TxData[4] = 0x13; // 寫入第 5 個 byte 不再越界
```

```
    HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox);
}

/**
 * @brief CAN 接收中斷回調函式 (處理編碼器回傳角度)
 */
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    CAN_RxHeaderTypeDef RxHeader;
    uint8_t RxData[8];

    if (hcan->Instance == CAN1)
    {
        if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader, RxData) == HAL_OK)
        {
            // 【安全優化】確認封包來源 ID，防止 VESC 雜訊誤更動數據
            if (RxHeader.IDE == CAN_ID_STD && RxHeader.StdId == 0x01)
            {
                if (RxData[0] == 0x07 && RxData[1] == 0x01 && RxData[2] == 0x01)
                {
                    // 小端序解碼
                    raw_encoder_value = (uint32_t)((RxData[6] << 24) |
                                                   (RxData[5] << 16) |
                                                   (RxData[4] << 8)  |
                                                    RxData[3]);
                }
            }
        }
    }
}
/* USER CODE END 4 */
```

Revision #2
Created 2026-04-01 02:06:08 UTC by TaipeiTechRacing
Updated 2026-04-06 06:22:51 UTC