STM32 ???????? VESC ???? (????????? CAN Buffer)

STM32 ???????? VESC ???? (????????? CAN Buffer)

? ?????????

本系統使用STM32透過CAN Bus控制4顆VESC驅動馬達，並整合了方向盤絕對值編碼器，實現以下進階車輛動態控制：

1. 動態模式切換**(4WD/RWD)**：低速時啟動4WD增加起步循跡性；高速巡航時（> 10000 ERPM）前輪斷電利用單向離合器滑行，後輪切換為100%功率輸出。
2. 軟體環形緩衝區**(Ring Buffer)**：針對STM32傳統CAN不支援DMA的硬體限制，實作了中斷專用的Ring Buffer。接收中斷能在數微秒內完成資料搬移，實現**「非同步解析」**，徹底杜絕CAN Bus高負載時的漏包與主迴圈卡頓問題。
3. 阿克曼電子差速**(Ackermann E-Diff)**：利用方向盤角度動態計算車輛過彎時的內外輪迴轉半徑（依據軸距與輪距）。外側輪獲得較高比例電流，內側輪減少電流，解決車輛推頭並提升過彎靈敏度。

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

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : STM32 VESC 4WD/RWD Dynamic Control System with E-Diff
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
```

```
#include "main.h"
#include <math.h> // 提供電子差速所需的 tanf() 與 fabs()

/* 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

#define MAX_TOTAL_CURRENT_MA  ((POWER_LIMIT_W * 1000) / BATTERY_VOLTAGE_NOM)

// 速度切換閾值 (ERPM: Electrical RPM)
#define ERPM_THRESHOLD_RWD    10000
#define ERPM_HYSTERESIS       500     // 遲滯區間

// 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

// 車體物理尺寸 (電子差速計算用，單位: 公尺)
#define TRACK_WIDTH_M         0.8f    // 輪距 (左右輪中心距離)
#define WHEELBASE_M           1.2f    // 軸距 (前後軸中心距離)

// CAN 軟體環形緩衝區大小 (取代 DMA)
#define CAN_RX_BUFFER_SIZE    16
/* USER CODE END PD */

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

```
CAN_HandleTypeDef hcan1;

/* USER CODE BEGIN PV */
int32_t current_erpm = 0;
uint16_t throttle_adc = 0;
int32_t target_total_ma = 0;
uint16_t speed_adc = 0;

float current_angle = 0.0f;
float steering_angle = 0.0f;    // -180 ~ +180 度 (電子差速輸入)
uint32_t raw_encoder_value = 0;

// 軟體 Ring Buffer 結構宣告
typedef struct {
    CAN_RxHeaderTypeDef header;
    uint8_t data[8];
} CAN_RxPacket_t;

CAN_RxPacket_t can_rx_buffer[CAN_RX_BUFFER_SIZE];
volatile uint8_t can_rx_head = 0; // 寫入指標 (中斷端)
volatile uint8_t can_rx_tail = 0; // 讀取指標 (主程式端)
/* 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 */
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);
void Process_CAN_Rx_Buffer(void);
void Apply_Electronic_Differential(float steering_angle_deg, int32_t axle_total_current,
int32_t *cmd_left, int32_t *cmd_right);
/* USER CODE END PFP */

/**
  * @brief  The application entry point.
  */
int main(void)
{
  HAL_Init();
  SystemClock_Config();

  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_CAN1_Init();
  MX_ADC2_Init();

  /* USER CODE BEGIN 2 */
  HAL_CAN_Start(&hcan1);
  HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING);

  // 啟動 ADC 連續轉換模式
  HAL_ADC_Start(&hadc1);
  HAL_ADC_Start(&hadc2);
  /* USER CODE END 2 */

  /* Infinite loop */
  while (1)
  {
      Process_CAN_Rx_Buffer(); // 隨時消化 Ring Buffer 內的 CAN 訊號
      Control_Loop_1kHz();     // 主控制邏輯
  }
}

/* USER CODE BEGIN 4 */

/**
 * @brief 電子差速分配 (Ackermann E-Diff)
 */
```

```
void Apply_Electronic_Differential(float steering_angle_deg, int32_t axle_total_current,
int32_t *cmd_left, int32_t *cmd_right)
{
    // 直行防呆區間 (轉角小於 3 度視為直行，電流平分)
    if (fabs(steering_angle_deg) < 3.0f) {
        *cmd_left  = axle_total_current / 2;
        *cmd_right = axle_total_current / 2;
        return;
    }

    // 角度轉弧度
    float theta_rad = fabs(steering_angle_deg) * (3.1415926f / 180.0f);

    // 計算假想中心迴轉半徑
    float R_center = WHEELBASE_M / tanf(theta_rad);

    // 計算內外側輪半徑
    float r_inner = R_center - (TRACK_WIDTH_M / 2.0f);
    float r_outer = R_center + (TRACK_WIDTH_M / 2.0f);

    if (r_inner < 0.1f) r_inner = 0.1f; // 防奇異點

    // 依半徑比例分配電流
    float total_r = r_inner + r_outer;
    int32_t current_inner = (int32_t)(axle_total_current * (r_inner / total_r));
    int32_t current_outer = (int32_t)(axle_total_current * (r_outer / total_r));

    // 依據左右轉方向賦予對應輪胎
    if (steering_angle_deg > 0.0f) {
        // 右轉: 右輪是內側
        *cmd_right = current_inner;
        *cmd_left  = current_outer;
    } else {
        // 左轉: 左輪是內側
        *cmd_left  = current_inner;
        *cmd_right = current_outer;
    }
}

/**
 * @brief 主控制迴圈 (約 500Hz)
```

```
 */
void Control_Loop_1kHz(void)
{
    static uint32_t last_control_tick = 0;
    uint32_t current_tick = HAL_GetTick();

    if ((current_tick - last_control_tick) < 2) {
        return;
    }
    last_control_tick = current_tick;

    throttle_adc = HAL_ADC_GetValue(&hadc1);
    speed_adc = HAL_ADC_GetValue(&hadc2);
    current_erpm = ((int32_t)speed_adc * 20000) >> 12;

    target_total_ma = ((int32_t)throttle_adc * MAX_TOTAL_CURRENT_MA) >> 12;

    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;
        }
    }

    int32_t current_front_axle = 0;
    int32_t current_rear_axle = 0;

    if (drive_mode == MODE_4WD) {
        current_front_axle = target_total_ma / 3;
        current_rear_axle  = target_total_ma - current_front_axle;
    } else {
        current_front_axle = 0;
        current_rear_axle  = target_total_ma;
    }

    // □□□□□□□□□ (□□□□□□□)
```

```
    int32_t cmd_front_L = 0, cmd_front_R = 0;
    int32_t cmd_rear_L = 0, cmd_rear_R = 0;

    Apply_Electronic_Differential(steering_angle, current_front_axle, &cmd_front_L,
&cmd_front_R);
    Apply_Electronic_Differential(steering_angle, current_rear_axle,  &cmd_rear_L,
&cmd_rear_R);

    // 輪詢發送 CAN, 避免 Bus Flooding
    static uint8_t motor_step = 0;
    switch(motor_step)
    {
        case 0:
            VESC_Send_Current(VESC_ID_FRONT_L, cmd_front_L);
            motor_step = 1;
            break;
        case 1:
            VESC_Send_Current(VESC_ID_FRONT_R, cmd_front_R);
            motor_step = 2;
            break;
        case 2:
            VESC_Send_Current(VESC_ID_REAR_L,  cmd_rear_L);
            motor_step = 3;
            break;
        case 3:
            VESC_Send_Current(VESC_ID_REAR_R,  cmd_rear_R);
            motor_step = 0;
            break;
    }
}

/**
 * @brief 非同步處理 CAN 緩衝區資料 (置於 main 的 while 迴圈中)
 */
void Process_CAN_Rx_Buffer(void)
{
    while (can_rx_tail != can_rx_head)
    {
        CAN_RxPacket_t *packet = &can_rx_buffer[can_rx_tail];

        // 解析編碼器資料
```

```
        if (packet->header.IDE == CAN_ID_STD && packet->header.StdId == 0x01)
        {
            if (packet->data[0] == 0x07 && packet->data[1] == 0x01 && packet->data[2] == 0x01)
            {
                raw_encoder_value = (uint32_t)((packet->data[6] << 24) |
                                               (packet->data[5] << 16) |
                                               (packet->data[4] << 8)  |
                                                packet->data[3]);

                // 換算為電子差速需要的方向盤絕對角度 (-180 ~ 180)
                float temp_angle = (float)raw_encoder_value - STEERING_CENTER_OFFSET;
                steering_angle = (temp_angle * 360.0f) / ENCODER_RESOLUTION;

                // 處理邊界
                if (steering_angle > 180.0f) {
                    steering_angle -= 360.0f;
                } else if (steering_angle < -180.0f) {
                    steering_angle += 360.0f;
                }
            }
        }
        can_rx_tail = (can_rx_tail + 1) % CAN_RX_BUFFER_SIZE;
    }
}

/**
 * @brief CAN 接收中斷 (極速存入 Buffer 隨即離開)
 */
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    if (hcan->Instance == CAN1)
    {
        uint8_t next_head = (can_rx_head + 1) % CAN_RX_BUFFER_SIZE;

        if (next_head != can_rx_tail)
        {
            if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0,
                                     &can_rx_buffer[can_rx_head].header,
                                     can_rx_buffer[can_rx_head].data) == HAL_OK)
            {
                can_rx_head = next_head;
```

```
            }
        }
        else
        {
            // 从 Buffer 中读取一帧 FIFO 数据丢弃
            CAN_RxHeaderTypeDef dummy_header;
            uint8_t dummy_data[8];
            HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &dummy_header, dummy_data);
        }
    }
}

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;

    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);

    uint32_t timeout = 0;
    while (HAL_CAN_GetTxMailboxesFreeLevel(&hcan1) == 0)
    {
        timeout++;
        if (timeout > 50000) return;
    }
    HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox);
}

void ENCODE_Step_up(void)
{
    CAN_TxHeaderTypeDef TxHeader;
    uint32_t TxMailbox;
```

```
    uint8_t TxData[8];

    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);

    TxHeader.DLC = 5;
    TxData[0] = 0x05;
    TxData[1] = 0x01;
    TxData[2] = 0x05;
    TxData[3] = 0x88;
    TxData[4] = 0x13;
    HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox);
}
/* USER CODE END 4 */
```

Revision #1
Created 2026-04-01 02:06:38 UTC by TaipeiTechRacing
Updated 2026-04-01 02:12:26 UTC by TaipeiTechRacing