MT6835 ใช้งานร่วมกับ STM32H723 Nucleo

มันมีไลบรารี่ 2 ชุด

.h	.c
mt6835_stm32_spi_port.h	mt6835_stm32_spi_port.c
mt6835.h	mt6835.c

1.Header File (`mt6835_stm32_spi_port.h`)

- การกำหนดค่าภายในไฟล์นี้:
  - `MT6835_STM32_SPI_PORT_ENABLE`: ตัวแปรนี้ถูกกำหนดเป็น `1` เพื่อเปิดใช้งาน SPI สำหรับการติดต่อกับอุปกรณ์ `mt6835` หากกำหนดค่าเป็น `0` จะไม่ใช้งาน SPI.
- การประกาศฟังก์ชัน:
  - `mt6835_stm32_spi_port_init`: ฟังก์ชันนี้จะทำหน้าที่ในการตั้งค่าและเริ่มต้น SPI และคืนค่า pointer ที่ชี้ไปยังอ็อบเจ็กต์ `mt6835_t` ซึ่งจะใช้ในการควบคุม SPI

2.C File (`mt6835_stm32_spi_port.c`)

- การใช้งาน SPI:
  - ฟังก์ชันในไฟล์นี้ทำงานเกี่ยวข้องกับการควบคุม SPI และการสื่อสารผ่าน SPI ดังนี้:
    - `mt6835_cs_control`: ฟังก์ชันนี้ควบคุมการตั้งค่า Chip Select (CS) โดยการตั้งค่าเป็น High หรือ Low เพื่อเลือกเปิดหรือปิดการสื่อสารกับอุปกรณ์ `mt6835` ตามสถานะที่ต้องการ (เมื่อ CS ถูกตั้งเป็น Low การสื่อสารจะเริ่มทำงาน).
    - `mt6835_spi_send`: ฟังก์ชันนี้ใช้สำหรับการส่งข้อมูลผ่าน SPI โดยใช้คำสั่ง `HAL_SPI_Transmit` ซึ่งเป็นฟังก์ชันของ STM32 HAL ในการส่งข้อมูลจาก `tx_buf` (buffer ที่เก็บข้อมูลที่จะส่ง) จำนวน `len` ไบต์.
    - `mt6835_spi_recv`: ฟังก์ชันนี้ใช้สำหรับการรับข้อมูลผ่าน SPI โดยใช้คำสั่ง `HAL_SPI_Receive` เพื่อรับข้อมูลจาก SPI และเก็บไว้ใน `rx_buf` (buffer ที่ใช้รับข้อมูล) จำนวน `len` ไบต์.
    - `mt6835_spi_send_recv`: ฟังก์ชันนี้จะส่งและรับข้อมูลพร้อมกันในเวลาเดียวกันโดยใช้คำสั่ง `HAL_SPI_TransmitReceive_IT` ซึ่งเป็นการส่งข้อมูลและรับข้อมูลแบบขนาน (duplex) ผ่าน SPI โดยไม่ต้องรอให้การส่งข้อมูลเสร็จสิ้นก่อน.
- การเริ่มต้น SPI (`mt6835_stm32_spi_port_init`):
  - ฟังก์ชันนี้จะสร้างอ็อบเจ็กต์ `mt6835_t` และเชื่อมโยงฟังก์ชันที่เกี่ยวข้องกับการควบคุม SPI (เช่น `mt6835_cs_control`, `mt6835_spi_send`, `mt6835_spi_recv`, `mt6835_spi_send_recv`) กับอ็อบเจ็กต์นี้ ซึ่งจะช่วยให้ `mt6835` สามารถใช้งาน SPI ในการสื่อสารกับอุปกรณ์ได้.

โค้ดนี้เป็นการสร้างไลบรารีสำหรับการใช้งานเซ็นเซอร์ MT6835 ที่ใช้โปรโตคอล SPI (Serial Peripheral Interface) ซึ่งช่วยให้สามารถเชื่อมต่อและควบคุมการทำงานของเซ็นเซอร์ได้ตามคำสั่งที่กำหนดไว้ในไลบรารี มีการใช้ฟังก์ชันต่างๆ เพื่อเชื่อมโยงการส่งและรับข้อมูลกับ SPI, การตรวจสอบ CRC (Cyclic Redundancy Check) เพื่อความถูกต้องของข้อมูล, และการจัดการการตั้งค่าเซ็นเซอร์ เช่น การตั้งค่ามุม (angle) หรือค่าพารามิเตอร์อื่นๆ ในเซ็นเซอร์

โครงสร้างหลักของโค้ด:

mt6835_cmd_enum_t - ใช้สำหรับการกำหนดคำสั่งที่สามารถใช้กับ MT6835 เช่น การอ่าน/เขียนข้อมูล การตั้งค่า zero หรือโหมดการทำงานต่างๆ mt6835_reg_enum_t - กำหนดชื่อของรีจิสเตอร์ในเซ็นเซอร์ MT6835 ซึ่งใช้ในการอ่านหรือเขียนค่า mt6835_t - โครงสร้างหลักที่เก็บข้อมูลและฟังก์ชันต่างๆ ที่เกี่ยวข้องกับการควบคุมเซ็นเซอร์ เช่น การส่งคำสั่งไปยังเซ็นเซอร์ การตั้งค่าฟังก์ชัน SPI การตรวจสอบ CRC mt6835_create() และ mt6835_destroy() - ฟังก์ชันสำหรับสร้างและทำลายอ็อบเจ็กต์ mt6835 mt6835_link_spi_xxx() - ฟังก์ชันเชื่อมโยงการควบคุม SPI กับอ็อบเจ็กต์ mt6835 crc8_table - ตารางสำหรับการคำนวณ CRC เพื่อความถูกต้องของข้อมูลที่รับส่ง

การทำงานหลัก:

การสร้างอ็อบเจ็กต์ - ฟังก์ชัน `mt6835_create()` จะสร้างอ็อบเจ็กต์ mt6835 โดยใช้ฟังก์ชัน malloc สำหรับการจัดการหน่วยความจำ การเชื่อมต่อ SPI - ฟังก์ชัน `mt6835_link_spi_xxx()` ใช้สำหรับเชื่อมโยงการควบคุม SPI โดยการกำหนดฟังก์ชันสำหรับควบคุมการส่งและรับข้อมูลผ่าน SPI การตรวจสอบ CRC - เมื่อเปิดการตรวจสอบ CRC (ผ่านฟังก์ชัน `mt6835_enable_crc_check()` หรือ `mt6835_disable_crc_check()`), ข้อมูลที่ส่งจะถูกตรวจสอบว่าไม่มีข้อผิดพลาดในการส่ง การอ่านข้อมูล - ฟังก์ชันเช่น `mt6835_get_raw_angle()` ใช้ในการดึงค่ามุมจากเซ็นเซอร์ การตั้งค่าศูนย์มุม - ฟังก์ชัน `mt6835_set_zero_angle()` ใช้ในการตั้งค่าศูนย์มุมสำหรับเซ็นเซอร์ MT6835

```c
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "dma.h"
#include "memorymap.h"
#include "spi.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "string.h"
#include "stdio.h"
#include "stdbool.h"
#include <stdarg.h>
#include "mt6835.h"
#include <math.h>
/* USER CODE END Includes */


/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
extern mt6835_t* mt6835_stm32_spi_port_init(void);

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
int _write(int file, char *ptr, int len)
{
        if (HAL_UART_Transmit(&huart3, (uint8_t*) ptr, len, HAL_MAX_DELAY)
            != HAL_OK) {
```

```
            return -1;
        }
        return len;
    }
    uint8_t id;
    uint32_t raw_angle = 0;
    float radian_angle = 0.0f;
    /* USER CODE END 0 */

    /**
      * @brief  The application entry point.
      * @retval int
      */
    int main(void)
    {

      /* USER CODE BEGIN 1 */

      /* USER CODE END 1 */

      /* MCU Configuration--------------------------------------------------------*/

      /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
      HAL_Init();

      /* USER CODE BEGIN Init */

      /* USER CODE END Init */

      /* Configure the system clock */
      SystemClock_Config();

      /* USER CODE BEGIN SysInit */

      /* USER CODE END SysInit */

      /* Initialize all configured peripherals */
      MX_GPIO_Init();
      MX_DMA_Init();
      MX_SPI3_Init();
      MX_USART3_UART_Init();
      MX_SPI1_Init();
      /* USER CODE BEGIN 2 */

        mt6835_t *mt6835 = mt6835_stm32_spi_port_init();
        //mt6835_enable_crc_check(mt6835);
        mt6835_disable_crc_check(mt6835);
      /* USER CODE END 2 */

      /* Infinite loop */
      /* USER CODE BEGIN WHILE */
        while (1) {
            raw_angle = mt6835_get_raw_angle(mt6835, MT6835_READ_ANGLE_METHOD_NORMAL);
            radian_angle = (raw_angle / 2097152.0f) * (M_PI * 2.0f);
```

```
        printf("raw_angle: %lu, radian_angle: %f\n\r", raw_angle, radian_angle);
        HAL_Delay(250);

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    }
  /* USER CODE END 3 */
}
```

1. การนำเข้าไลบรารีและการกำหนดค่า

```
#include "main.h"
#include "dma.h"
#include "memorymap.h"
#include "spi.h"
#include "usart.h"
#include "gpio.h"
#include "string.h"
#include "stdio.h"
#include "stdbool.h"
#include <stdarg.h>
#include "mt6835.h"
#include <math.h>
```

- นำเข้าไลบรารีมาตรฐานของ STM32 และไลบรารีที่เกี่ยวข้องกับ SPI, UART, GPIO
- `mt6835.h` เป็นไลบรารีที่ใช้ในการติดต่อกับเซ็นเซอร์ MT6835
- ใช้ `math.h` เนื่องจากมีการคำนวณค่ามุมในหน่วยเรเดียน

2. ฟังก์ชัน `_write` (ใช้กับ `printf`)

```
int _write(int file, char *ptr, int len)
{
    if (HAL_UART_Transmit(&huart3, (uint8_t*) ptr, len, HAL_MAX_DELAY) != HAL_OK)
{
        return -1;
    }
    return len;
}
```

- ฟังก์ชันนี้ทำให้ `printf()` สามารถใช้กับ UART3 ได้ โดยส่งข้อมูลผ่าน `HAL_UART_Transmit()`
- ใช้ `HAL_MAX_DELAY` เพื่อรอให้การส่งข้อมูลเสร็จสมบูรณ์

3. ตัวแปรที่ใช้

```
uint8_t id;
uint32_t raw_angle = 0;
float radian_angle = 0.0f;
```

- `raw_angle` เก็บค่ามุมที่อ่านจากเซ็นเซอร์ในรูปของ ค่าดิบ (Raw Value)
- `radian_angle` ใช้เก็บค่ามุมที่แปลงเป็นเรเดียน

4. การเริ่มต้นระบบและอุปกรณ์ต่อพ่วง

```
HAL_Init();
SystemClock_Config();
MX_GPIO_Init();
MX_DMA_Init();
MX_SPI3_Init();
MX_USART3_UART_Init();
MX_SPI1_Init();
```

- `HAL_Init()` เป็นฟังก์ชันเริ่มต้นระบบ HAL (Hardware Abstraction Layer)
- `SystemClock_Config()` กำหนดค่า Clock ให้กับ MCU
- `MX_GPIO_Init()`, `MX_DMA_Init()`, `MX_SPI3_Init()`, `MX_USART3_UART_Init()`, `MX_SPI1_Init()` เป็นฟังก์ชันที่ใช้กำหนดค่าอุปกรณ์ต่อพ่วงต่างๆ เช่น GPIO, DMA, SPI, UART

5. การกำหนดค่าและปิด CRC Check

```
mt6835_t *mt6835 = mt6835_stm32_spi_port_init();
// mt6835_enable_crc_check(mt6835);
mt6835_disable_crc_check(mt6835);
```

- ใช้ `mt6835_stm32_spi_port_init()` เพื่อกำหนดค่าเซ็นเซอร์ MT6835
- ปิด CRC Check (`mt6835_disable_crc_check()`) เพื่อให้อ่านค่ามุมโดยไม่ต้องตรวจสอบ CRC

6. การทำงานหลักใน Loop

```
while (1) {
    raw_angle = mt6835_get_raw_angle(mt6835, MT6835_READ_ANGLE_METHOD_NORMAL);
    radian_angle = (raw_angle / 2097152.0f) * (M_PI * 2.0f);
    printf("raw_angle: %lu, radian_angle: %f\n\r", raw_angle, radian_angle);
    HAL_Delay(250);
}
```

- อ่านค่ามุมแบบ Raw จากเซ็นเซอร์ `mt6835_get_raw_angle()`
- แปลงค่าที่อ่านได้ให้เป็นเรเดียน: $\text{radian\_angle} = (\text{raw\_angle} / 2097152.0) \times 2\pi$

    - 2097152 คือค่ามุมเต็มสเกลของเซ็นเซอร์ (21-bit resolution)
    - `M_PI * 2.0f` ใช้แปลงเป็นเรเดียน (1 รอบ = 2π เรเดียน)

- ส่งค่ามุมที่อ่านได้ออกทาง UART โดยใช้ printf()
- หน่วงเวลา 250ms ก่อนอ่านค่าใหม่

ผลลัพธ์ที่คาดหวัง (ตัวอย่าง Output ทาง UART):

```
raw_angle: 1048576, radian_angle: 3.141593
raw_angle: 2097152, radian_angle: 6.283185
raw_angle: 524288, radian_angle: 1.570796
```