TI MSPM0G3507 智能小车全能模板使用说明书 (CCS版)

简介： 本工程基于德州仪器 (TI) 高性能 **MSPM0G3507** 微控制器，专为智能车竞赛（如电赛）设计。工程采用 **CCS (Code Composer Studio)** 开发，集成了 TI 最新的 **SysConfig** 配置工具，不仅包含麦克纳姆轮/差速运动控制、PID 闭环、IMU 姿态解算，还支持 OLED 菜单和灰度循迹。

1. 快速上手：如何使用 CCS 打开工程？

不要直接双击文件！请按照以下步骤操作，保证 0 报错：

1. 打开软件：启动 Code Composer Studio (CCS)。
2. 导入工程：
   - 点击菜单栏 `File` -> `Import...`
   - 选择 `C/C++` -> `CCS Projects` -> 点击 `Next`。
   - 点击 `Browse...`，选择本工程的根目录文件夹（即包含 `.ccsproject` 的文件夹）。
   - 在 "Discovered projects" 中勾选 `Template_Car` 或 `Template_Yuntai`，点击 `Finish`。
3. 编译代码：点击工具栏上的 🔨 (锤子图标) 进行编译。
4. 查看配置 **(SysConfig)**：
   - 在左侧项目资源管理器中，双击 `empty.syscfg` 文件。

- 这是一个图形化配置工具，你可以在这里直观地看到所有引脚分配、时钟配置和中断设置。修改引脚无需写代码，直接在这里鼠标点选即可！

2. 代码架构梳理 (文件导航)

工程结构经过精心分层，逻辑清晰：

- **My_App (应用层 - 你的逻辑写这里)**
  - `scheduler.c`：时间片调度器。整个系统的心脏，管理 10ms、50ms、100ms 等周期任务。
  - `motor_app.c`：运动学解算。输入前进速度(m/s)，自动计算 4 个轮子的转速。支持麦轮/差速。
  - `pid_app.c`：PID 闭环控制。包含速度环和位置环逻辑，调整 Kp, Ki, Kd 参数就在这里。
  - `oled_app.c`：菜单显示。负责在屏幕上刷新电压、角度、速度等数据。
  - `gray_app.c`：循迹逻辑。处理灰度传感器的信号，判断黑线位置。
  - `encoder_app.c`：速度测量。计算当前车轮的实时转速。
  - `uart_app.c`：通信任务。处理蓝牙或上位机的数据包。
- **My_Driver (驱动层 - 硬件操作)**
  - `motor_driver.c`：电机底层。直接操作 PWM 寄存器让电机转动。
  - `encoder_driver.c`：编码器底层。读取定时器计数值换算成脉冲。
  - `wit_imu.c`：陀螺仪驱动。解析维特智能 (WitMotion) 传感器的串口协议。
  - `oled_driver.c`：OLED底层。I2C 通信协议实现。

- `No_Mcu_Ganv_Grayscale_Sensor.c`：灰度传感器驱动。
- **MSPM0 (系统层)**
  - `interrupt.c`：中断服务函数。定时器中断、串口中断入口都在这。
  - `clock.c`：系统时钟初始化。

3. 硬件外设与引脚清单

重要提示： MSPM0 的最大优势是引脚映射非常灵活。最准确的引脚定义请直接打开工程中的 `empty.syscfg` 查看。以下为默认模板配置：

外设模块	功能	涉及引脚 (SysConfig中名称)	说明
PWM (Timer)	电机速度控制	`PWM_Motor_1` ~ `PWM_Motor_4`	产生 10-20kHz PWM 波驱动电机桥
GPIO (Output)	电机方向控制	`GPIO_Motor_Dir_1` ~ `4`	高低电平控制正反转
QEI / Timer	编码器读取	`Encoder_Timer_1` ~ `4`	需连接电机编码器 A/B 相
UART (Serial)	陀螺仪 (IMU)	`UART_IMU_TX/RX`	推荐开启 DMA 接收
UART (Serial)	蓝牙/上位机	`UART_DEBUG_TX/RX`	用于无线调参或看波形
I2C	OLED 屏幕	`I2C_OLED_SDA/SCL`	驱动 SSD1306 屏幕
GPIO (Input)	灰度传感器	`GPIO_Gray_1` ~ `x`	数字量读取黑白线状态
GPIO (Input)	按键	`KEY1`, `KEY2`	用于切换模式
SPI	Flash 存储	`SPI_FLASH_...`	连接 W25Q64 等芯片 (可选)

4. 全能函数调用指南 (复制即用)

这里整理了所有你需要调用的功能函数，拿来即用！

4.1 小车运动控制 (`motor_app.h`)

这是让车动起来最简单的办法，内部会自动处理麦克纳姆轮或差速模型。

- 设置目标速度
  - `void Car_Set_Velocity(float v_x, float v_y, float v_w)`
  - 参数：
    - `v_x`：前进/后退速度 (mm/s)。正数前进。
    - `v_y`：左右平移速度 (mm/s)。正数左移 (仅麦轮有效，普通车填0)。
    - `v_w`：自转速度 (rad/s)。正数逆时针旋转。
  - 示例：

```
Car_Set_Velocity(500, 0, 0);    // 以 0.5m/s 前进
Car_Set_Velocity(0, 0, 1.5);    // 原地旋转
Car_Set_Velocity(0, 0, 0);      // 停车
```

4.2 PID 算法与调参 (`pid_app.h`)

系统默认已在 `Scheduler` 中每 10ms 自动调用一次计算。你需要做的是修改参数。

- 修改 **PID** 参数 (宏定义方式)
  - 打开 `pid_app.h` 或 `pid_app.c`，找到宏定义：

```
#define KP_SPEED  5.0f    // 比例系数
#define KI_SPEED  0.5f    // 积分系数
#define KD_SPEED  0.0f    // 微分系数
```

- 动态修改参数 (函数方式)

- `void PID_Set_Param(PID_TypeDef *pid, float kp, float ki, float kd)`
- 示例：`PID_Set_Param(&Motor_1_PID, 8.0, 0.5, 0.1);`

4.3 传感器数据读取

- 读取陀螺仪 **(IMU)** (`wit_imu.h`)
  - `float Wit_Get_Yaw(void)`：获取当前航向角 (-180.0 ~ 180.0 度)。
  - 示例：`float current_angle = Wit_Get_Yaw();`
- 读取灰度传感器 (`gray_app.h`)
  - `uint8_t Get_Gray_Status(void)`
  - 返回值：返回一个字节，每一位代表一个探头状态 (0为白/1为黑，或反之，视硬件而定)。
  - 示例：

```
if (Get_Gray_Status() == 0x18) { // 假设中间两个探头检测到线
    Car_Set_Velocity(500, 0, 0); // 直行
}
```

- 读取编码器速度 (`encoder_app.h`)
  - `float Get_Motor_Speed(uint8_t id)`
  - 参数：电机ID (1~4)。
  - 返回值：当前实测速度 (mm/s)。

4.4 OLED 屏幕绘图 (`oled_app.h` / `oled_driver.h`)

- 显示字符串
  - `void OLED_ShowString(uint8_t x, uint8_t y, char *str)`

- 示例：`OLED_ShowString(0, 0, "Running...");`

- 显示数字
  - `void OLED_ShowNumber(uint8_t x, uint8_t y, int num, uint8_t len, uint8_t size)`
  - 示例：`OLED_ShowNumber(0, 2, (int)current_angle, 4, 16);`

4.5 任务调度 (`scheduler.h`)

如果你想加自己的代码（比如每秒闪烁一次 LED），不要用 `delay()`，请用调度器。

- 添加任务步骤：
  1. 打开 `Scheduler_Task/scheduler.c`。
  2. 编写你的函数 `void My_Task(void) { ... }`。
  3. 在 `Scheduler_Task_List` 数组中添加一行：

```
{My_Task, 1000, 0}, // 1000ms 执行一次
```

5. 常见问题 (FAQ)

- **Q:** 导入工程后有些文件是灰色的/打不开？
  - A: 这是 CCS 的链接文件机制。请确保 `My_App`，`Drivers` 等文件夹都在工程根目录下，不要随意移动文件夹位置。
- **Q:** 修改了 **SysConfig** 里的引脚，编译报错？
  - A: 修改 SysConfig 后，保存文件，CCS 会自动重新生成 `ti_msp_dl_config.c`。如果报错，请检查你应用层代码里使用的宏定义名称（如 `GPIO_MOTOR_PIN_1`）是否和 SysConfig 里设置的名称一致。
- **Q:** 电机转动方向反了？

- A: 无需重新焊接线。打开 `motor_driver.c`，找到 `Set_Motor_Voltage` 函数，交换里面控制 GPIO 高低电平的逻辑即可。

- **Q:** 程序跑飞/死机？
  - A: 检查 `scheduler.c` 中的任务是否耗时太长（超过了任务周期）。不要在任务里使用长延时 `delay_ms`。

祝您比赛取得好成绩！