**实验报告**

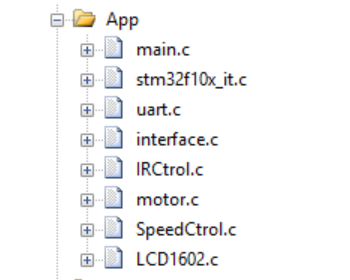
（STM32小车运行控制原理学习，以小车前进功能为例）

学号：2023117184

姓名：唐宇晗

**一、小车运动相关函数介绍**

先对小车运行相关文件进行梳理：



接下来逐一进行介绍：

**Main.c**文件作为嵌入式项目的核心文件，起连接所有功能的作用

**Stm32f10x\_it.c**文件作为stm32中断响应函数，启用中断控制功能后在该文件中写回调函数，也可在启用相关中断的函数下面随时编写，该项目中回调函数在stm32f10x\_it.c中编写

**UART.c**

UART有关函数的使能与逻辑执行

**USART1Conf(u32 baudRate)**

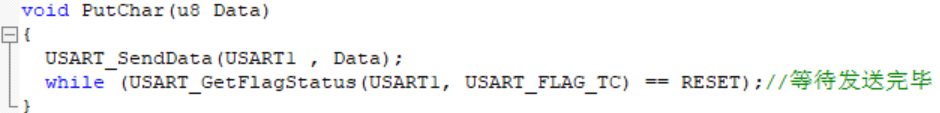


配置USART1串口通信参数，包括波特率、数据位、停止位、奇偶校验等。

初始化USART1的发送和接收引脚（GPIOA9和GPIOA10）。

使能USART1的发送和接收中断。

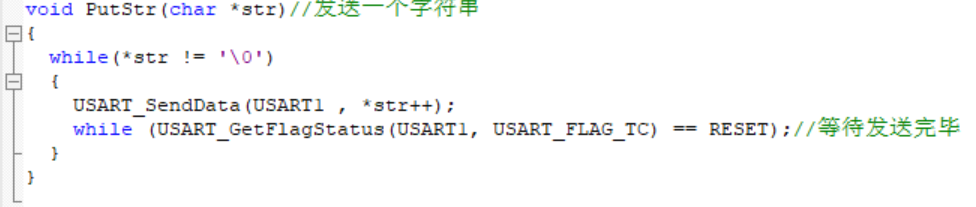
**PutChar(u8 Data)**



发送一个字节的数据。

等待发送完成标志位，确保数据发送完毕。

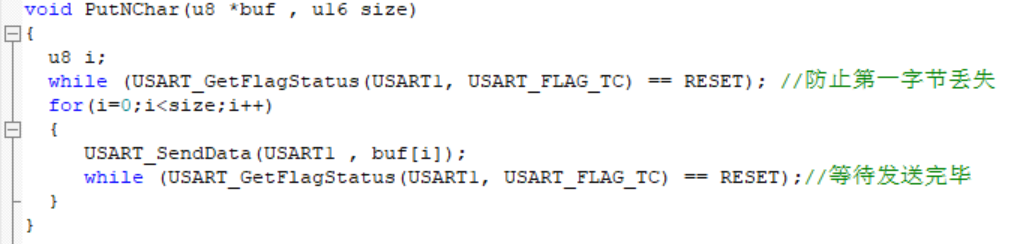
**PutStr(char \*str)**



发送一个字符串。

遍历字符串，逐个发送字符，并等待每个字符发送完毕。

**PutNChar(u8 \*buf, u16 size)**



发送指定长度的数据。

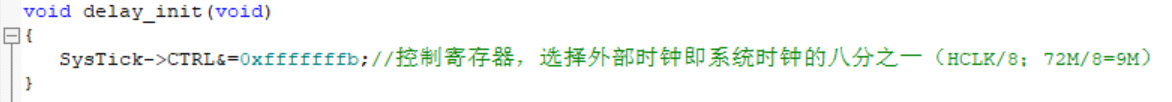
首先等待发送完成标志位，确保不会丢失第一个字节。

循环发送数据缓冲区中的每个字节，并等待每个字节发送完毕。

**Interface.c**文件起使能项目有关GPIO时钟，TIM2定时器和其嵌套中断向量控制器（NVIC），和LED显示屏，舵机等外设，以及LED指示灯的取反

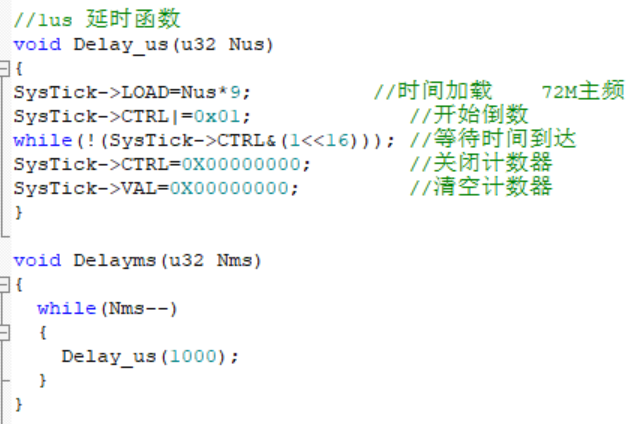
函数介绍：

void delay\_init(void) ：

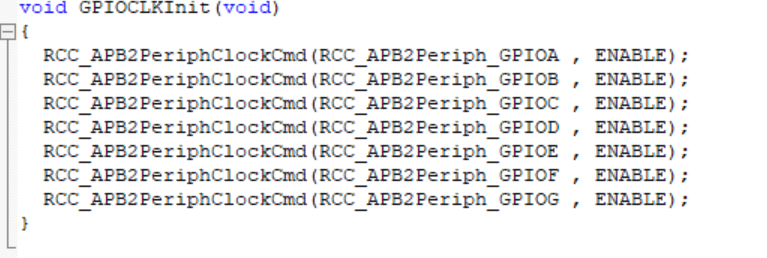
直接编写寄存器，SysTick->CTRL&=0xfffffffb;

对CTRL寄存器与0xfffffffb(1111 1111 1111 1111 1111 1111 1111 1011)，即为将第二位归零，选用外部参考时钟，在STM32F103系列中，主频为72MHZ,外部参考时钟为主频1/8，即为9MHZ

void Delay\_us(u32 Nus)  和void Delayms(u32 Nms)

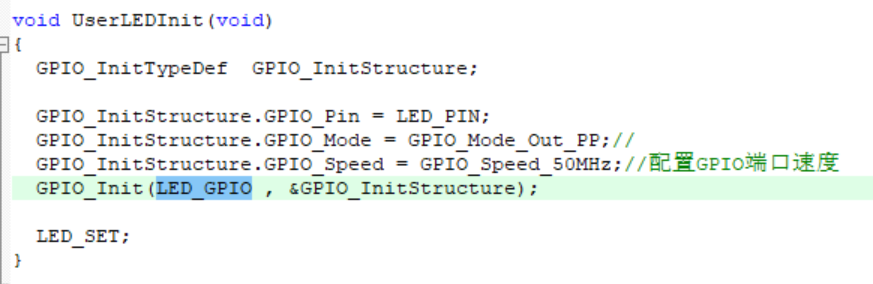
嵌入式项目中常用的延时函数，本项目中，void Delay\_us(u32 Nus)直接对寄存器读写，建议改用nop的方式书写，不会额外占用32的资源，void Delayms(u32 Nms)套用微妙延时函数，实现毫秒级延时

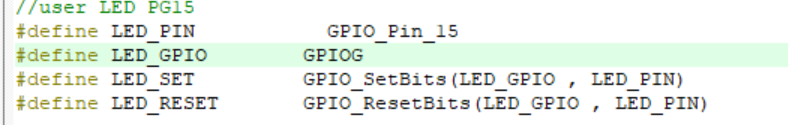
void GPIOCLKInit(void)



使能定时闹钟

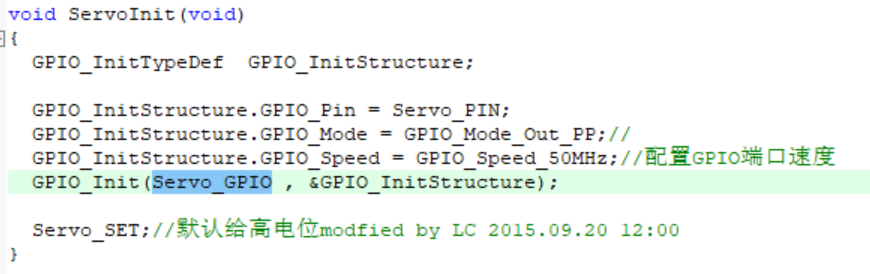
void UserLEDInit(void)

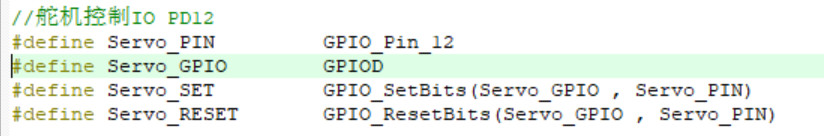




对LED有关GPIO口的简单配置，通过查看声明可知，开启GPIOG时钟下15号端口，推挽输出模式，端口速度50MHz，默认高电位

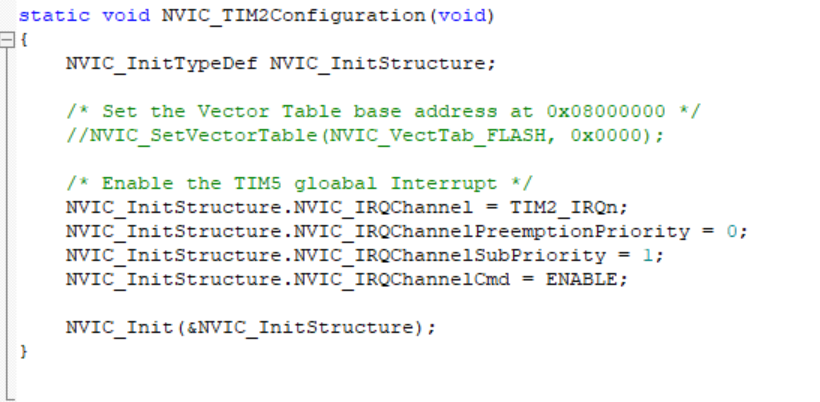
void ServoInit(void)





该函数同理，开启对舵机有关GPIO口的简单配置，通过查看声明可知，开启GPIOD时钟下12号端口，推挽输出模式，端口速度50MHz，默认高电位

static void NVIC\_TIM2Configuration(void)



打开TIM2定时器中断向量控制器，抢占优先级为0，从优先级为1，（数字越低，优先级越高）

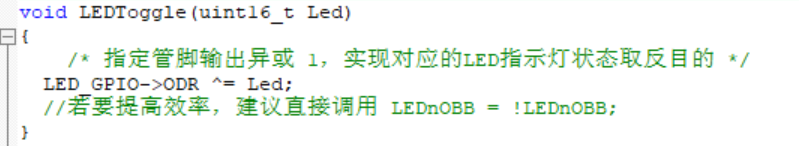
void TIM2\_Init(void)



该函数用于配置TIM2定时器基本功能，在STM32F103中，TIM2为通用定时器，能够完成定时器的基本功能

本次配置TIM2定时器自动重装计数器值为 （100 - 1），预分频器为（72 - 1），0分频，向上计数，同时在清楚中断标志位后打开定时器的更新中断，最后使能TIM2定时器

void LEDToggle(uint16\_t Led)

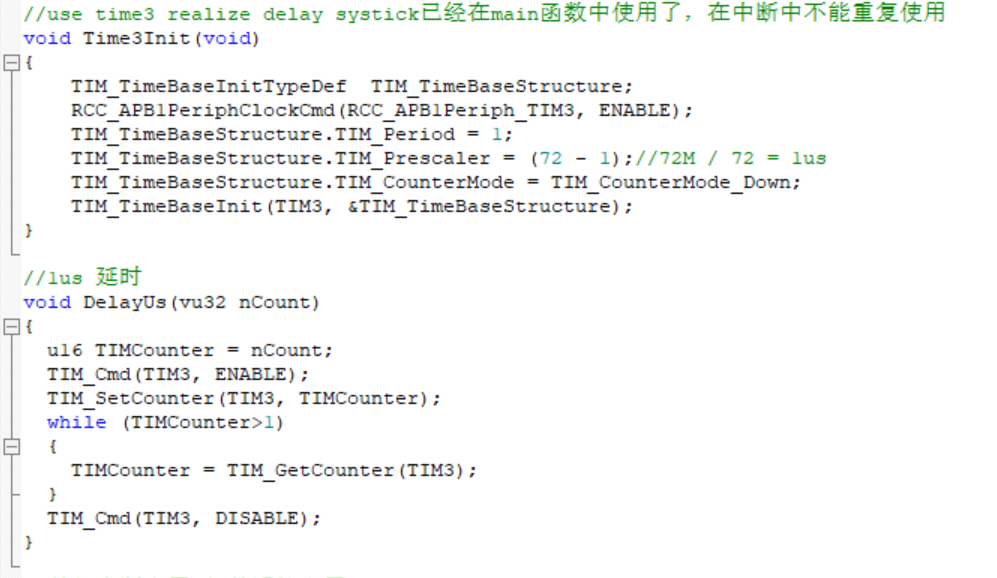


翻转LED电平

**IRCtrol.c**

**void Time3Init(void)**

**void DelayUs(vu32 nCount)**



**延时函数为避免与main函数中冲突，换了一种形式，本质不变**

IRCtrolInit函数：

初始化红外遥控控制

void IRIntIsr(void)：



逻辑判断阈值：高电平时长区分逻辑 0 和 1，需与实际协议匹配

数据存储顺序

高位优先：每次右移 IRCOM[j]，新位补到最高位，符合 NEC 协议的高位先发送顺序。

校验机制

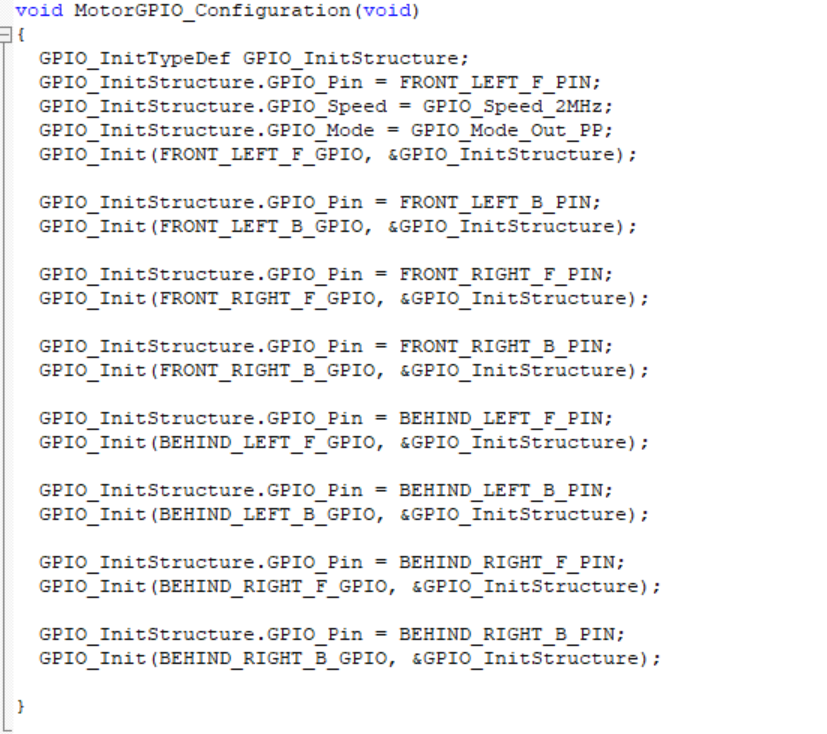
反码校验：第三字节需为第二字节的反码，否则数据无效。

指令映射

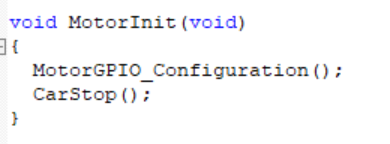
根据 IRCOM[2] 的值映射到控制指令（如上、下、左、右、停止）。

**Motor.c**

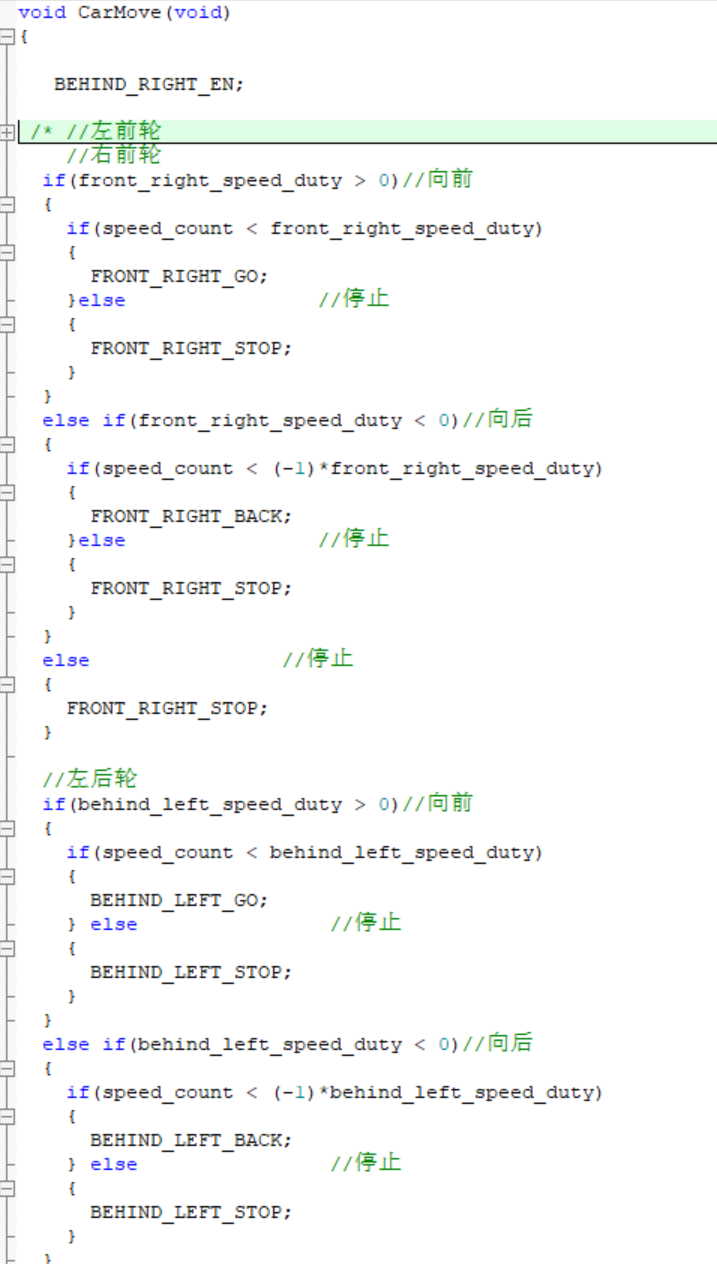
对小车四个电机的设置，控制小车移动

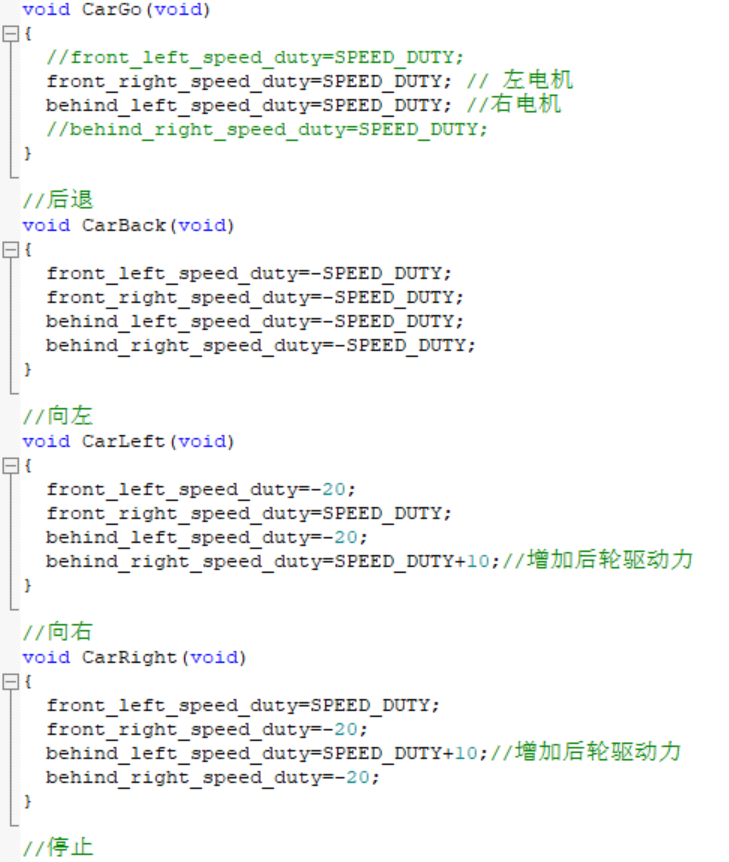


小车四个电机的GPIO口，



小车的基本功能和停止

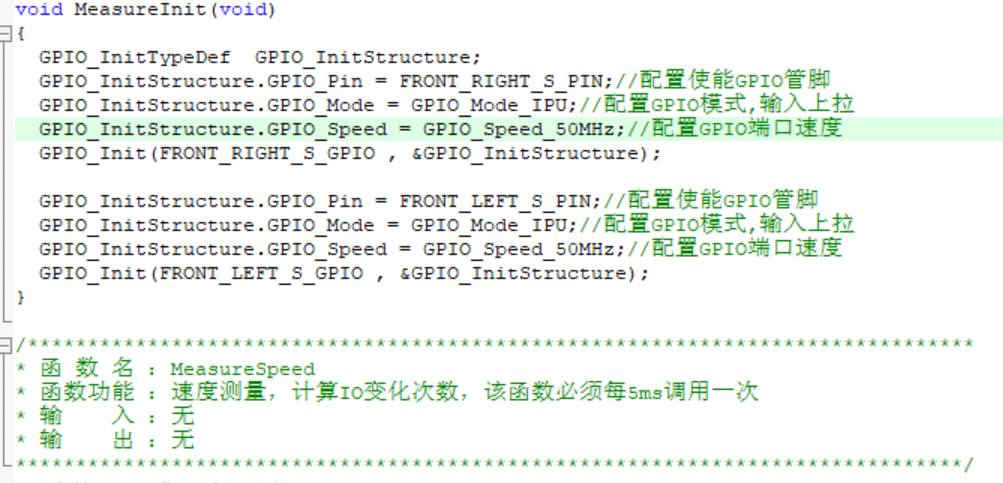




SpeedCtrol.c

小车运动的控制函数

**MeasureInit(void)**



初始化左右轮子的速度测量引脚为输入上拉模式。

**MeasureSpeed(void)**

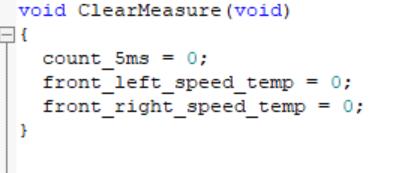


每隔5毫秒调用一次，用于测量左右轮子的速度。

通过检测轮子编码器的电平变化次数来计算速度。

每500毫秒（100次调用）计算一次实际速度，并将其转换为厘米每秒（cm/s）。

**ClearMeasure(void)**



清除测量计数器，用于重新开始速度测量。

**LCD1602.c**

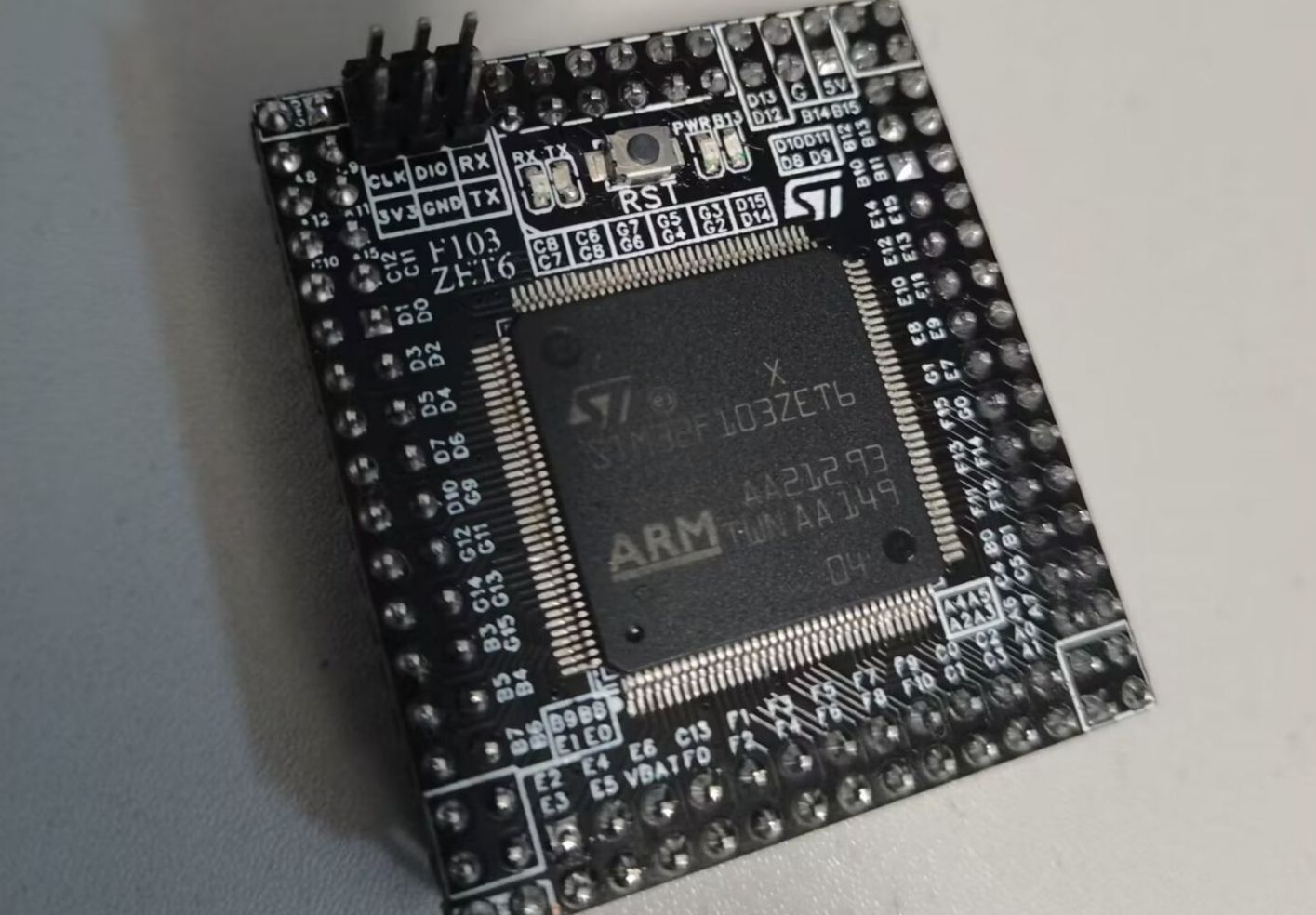
有关LCD1602的函数，基本为开源代码

**二、程序运行过程介绍**

介绍“小车前进功能”中所用到的函数和变量，以及是如何运行使小车前进的。

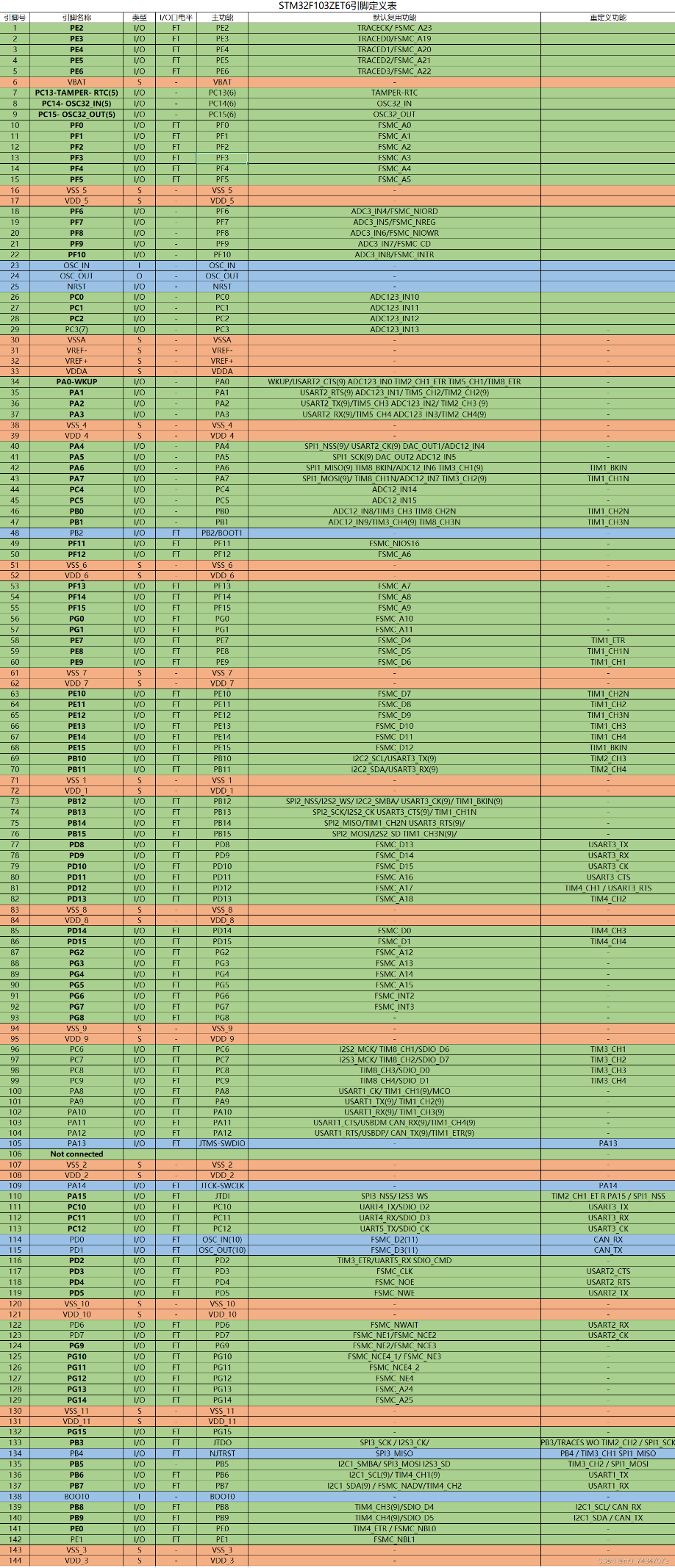
要求300字以上，图文并茂（可以是手工画图并拍照）

如图：STM32F103ZET6



下为引脚功能图

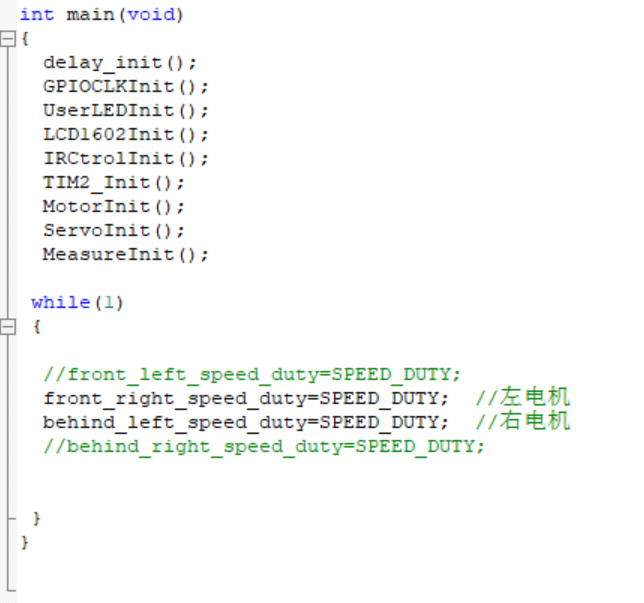
有助于STM32有关功能的开发



代码和具体运行分析

**1. 初始化**

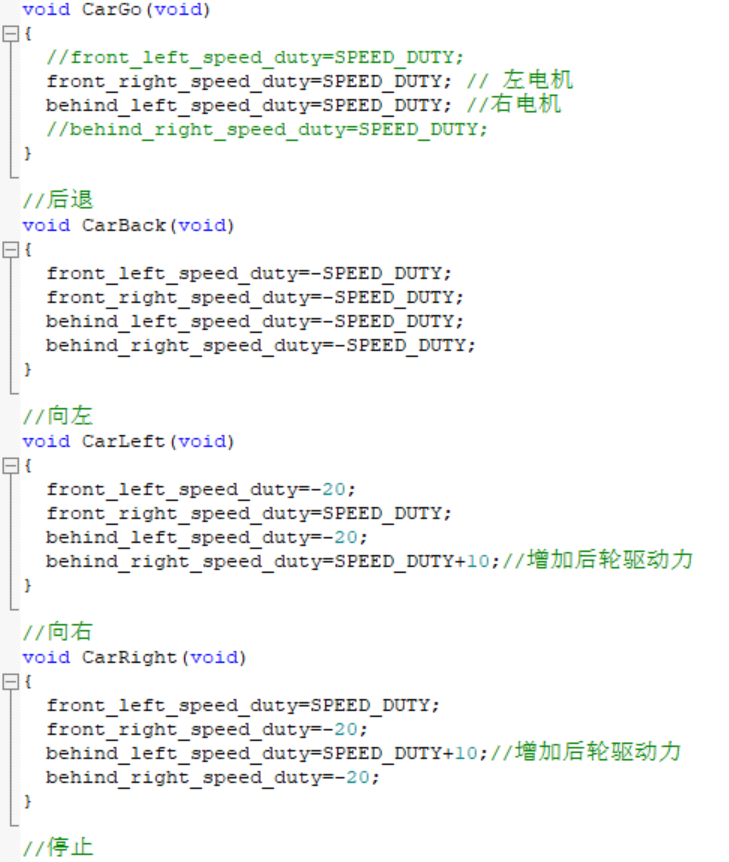
在 main.c 文件中，main 函数首先调用了一系列初始化函数来设置小车的基本功能：

这些初始化函数分别用于：

* delay\_init(): 初始化延时功能。
* GPIOCLKInit(): 初始化GPIO时钟。
* UserLEDInit(): 初始化用户LED。
* LCD1602Init(): 初始化LCD1602显示屏。
* IRCtrolInit(): 初始化红外遥控。
* TIM2\_Init(): 初始化定时器2。
* MotorInit(): 初始化电机。
* ServoInit(): 初始化舵机。
* MeasureInit(): 初始化测量功能。

**2. 电机控制**

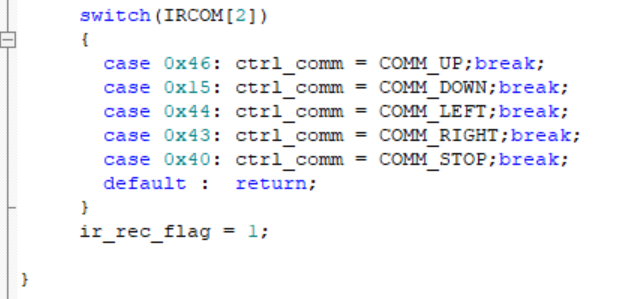
电机控制逻辑在 motor.c 文件中定义，包括小车的前进、后退、左转、右转和停止功能：



* CarGo(): 前进，设置前后左右电机的速度。
* CarBack(): 后退，设置前后左右电机的速度，使小车后退。
* CarLeft(): 左转，设置前后左右电机的速度，使小车向左转。
* CarRight(): 右转，设置前后左右电机的速度，使小车向右转。
* CarStop(): 停止，将所有电机的速度设置为0。

**3. 红外遥控**

红外遥控的逻辑在 IRCtrol.c 文件中定义，通过红外遥控接收指令来控制小车的运动：



case 0x46: ctrl\_comm = COMM\_UP; break; *// 向前*

case 0x15: ctrl\_comm = COMM\_DOWN; break; *// 向后*

case 0x44: ctrl\_comm = COMM\_LEFT; break; *// 向左*

case 0x43: ctrl\_comm = COMM\_RIGHT; break; *// 向右*

case 0x40: ctrl\_comm = COMM\_STOP; break; *// 停止*

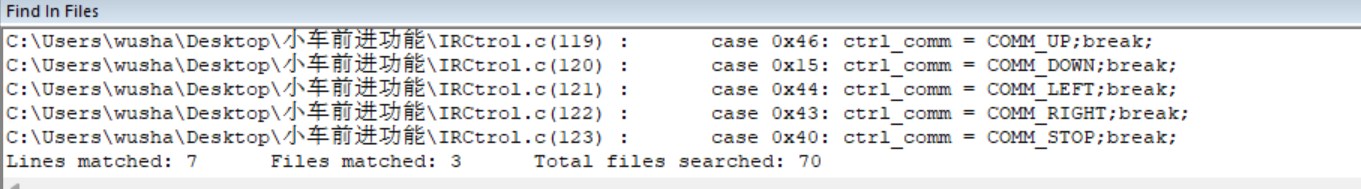
default : return;

红外遥控通过 IRIntIsr 函数解析红外信号，并根据解析结果设置 ctrl\_comm 变量，该变量用于控制小车的运动方向。

**4. 主循环**

在 main.c 的主循环中，根据 ctrl\_comm 变量的值来控制小车的运动：（该逻辑函数存于IRCtrol.c中）





根据 ctrl\_comm 的值，调用相应的函数来控制小车的运动。

**5.总结**

初始化函数设置了小车的基本功能，电机控制函数实现了小车的运动控制，红外遥控函数解析遥控信号并设置控制命令，主循环根据控制命令调用相应的电机控制函数。确保小车能够根据红外遥控的指令进行前进、后退、左转、右转和停止。

最后的实现是通过红外遥控接收指令来控制小车的运动。