

AutoRoboController

基础程序说明

简述:

该版本【CodeForRobotac_v2019_BaseType】基础程序适配【RobotacMainBoard_2019】核心控制器,功能囊括核心控制器所有外设接口以及其它核心功能,请配合硬件说明书使用。除了满足“大学 Robotac 机器人大赛”机器人控制功能外,也多范围适用大学生电子电控 DIY 学习及实践。

程序编译及下载:

编译环境

该程序基于【 μ Vision Keil_v5】集成编译环境开发完成,软件详情及下载+安装+破解方法自行百度,此外,还需安装 J-Link 程序下载器驱动【J-Link Manual】,以满足正常的程序下载和在线调试需求。

1)、Keil_v5



2)、J-Link 驱动



程序下载

控制器已配置【SWD】程序模式,首先在软件编译环境中设置为 SWD 模式,并确认下载器驱动已安装、硬件连接无误。


1)、硬件接线:

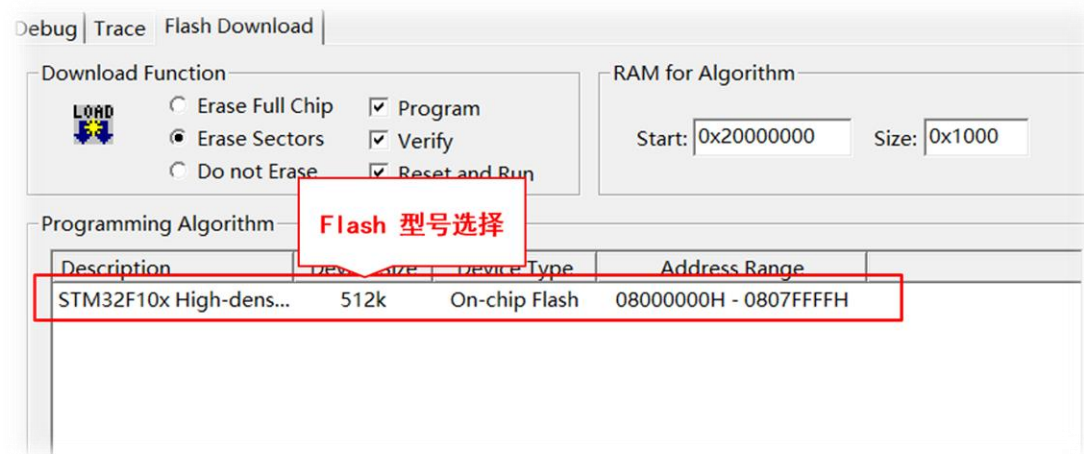
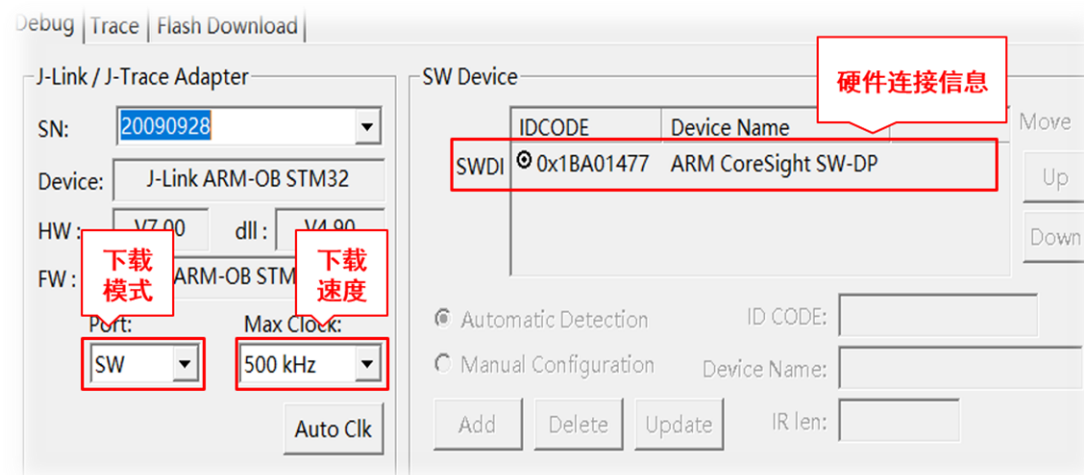
把下载器的 GND、SWDIO、SWCLK 分别与控制器【SWD】接口对应连接,给控制器上电,确保控制器处于正常工作状态。常见的 J-Link 下载器如下:





2)、软件设置:

原则上基础版程序工程已经设置好程序下载配置,自行设置步骤

- ①、点击软件界面【Options】按钮 ;
- ②、选择【Debug】菜单栏,选择【J-LINK/J-TRACE】下载模式,点击【Settings】按钮;



⑥、点击【编译】按钮  完成程序编译，确认无 error 之后，点击【下载】按钮  开始下载程序。

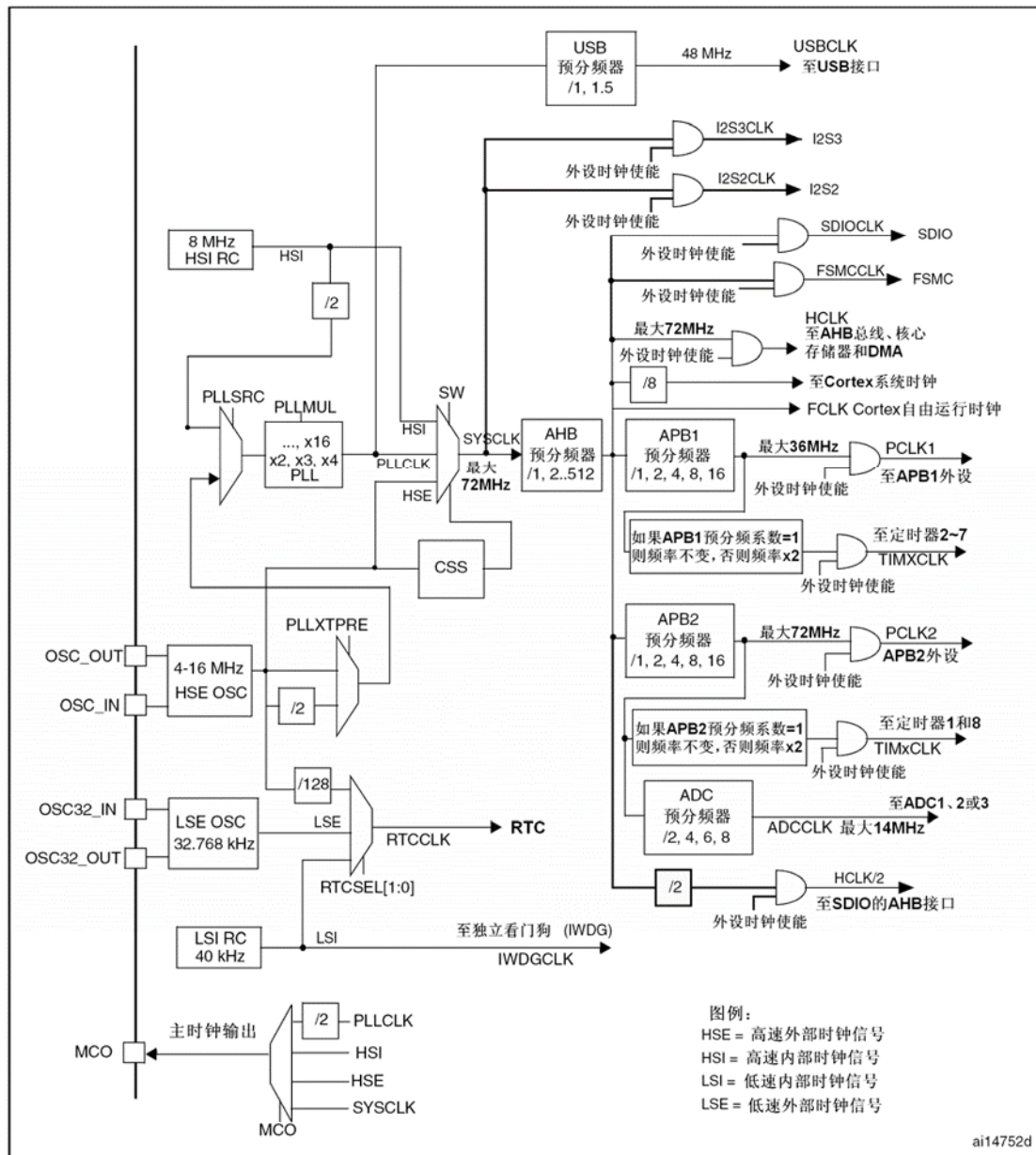
时钟及定时器:

时钟选择

控制器已配置一颗【8M 有源石英晶振】，系统主频最高可达 72MHz，也可以自行选择片内时钟，最高可达 64MHz。

函数：【SystemInit()】初始化系统时钟；

芯片时钟树：



定时器及延时

通常需要配置定时器中断实现周期巡检任务，相应实现了系统心跳中断及 TIM2 时钟中断的配置及中断函数；

函数：【delay_init()】初始化系统心跳；

函数：【delay_ms()】延时函数 ms；

函数：【delay_us()】延时函数 us；
变量：【delay_ostickspersec】为心跳频率；
函数：【SysTick_Handler()】心跳中断函数；
函数：【Timer2_Init()】初始化定时器 2；
函数：【TIM2_IRQHandler ()】定时器中断函数；

涉及文件：【delay】【timer】

蜂鸣器+LED:

蜂鸣器

控制器配置一颗常规 5v 有源蜂鸣器，由三极管驱动工作。

函数：【Hummer_On()】蜂鸣器开始发声；
函数：【Hummer_Off()】蜂鸣器停止发声；
函数：【Hummer_SystemStart()】系统初始化完成发声；
函数：【Hummer_SystemOk()】系统‘Ok’发声；



LED

板载两颗可控 LED 状态指示灯，分布在核心芯片两边。

函数：【Led_On()】LED 灯亮；
函数：【Led_Off()】LED 灯灭；
函数：【PCI_Init()】蜂鸣器+LED 引脚初始化；

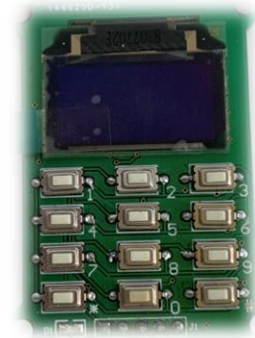
涉及文件：【gpio】

键盘+OLED 屏:

控制器配置一路 I2C 协议下的键盘+OLED 显示屏外设，12 键键位，外部中断触发按键事件；OLED 液晶屏分辨率 128x64，可烧写汉字字库和符号字母，字体大小 6*8、8*16、16*16 可选，字库 Flash 大小 32k，每次烧写完毕后注释字库函数再次下载程序；

函数：【keyboardConfig ()】键盘初始化；
函数：【EEPROM_Init ()】烧写字库；
函数：【Get_KeyBoardData ()】获取键盘按键值；
函数：【OLED_EEPROM_disp_num ()】液晶屏显示整形数字；
函数：【OLED_EEPROM_disp_string ()】液晶屏显示字符串；
函数：【OLED_EEPROM_ShowChar1616()】液晶屏显示字符；
函数：【OLED_EEPROM_disp_word ()】液晶屏显示文字；

涉及文件：【ch455】【iic】【oled】【oled_eeprom】【Keyboard】



电机控制 (PWM/UART):

控制器原则上两个电机控制接口最多可控制 4 路电机 (PWM)，或者两路电机 (UART)。

PWM 模式

分别占用 TIM5 和 TIM2 的各自两通道，根据电机驱动类型自定义输出频率。

变量：【MOTORx_Period】修改 PWM 频率；

变量：【MOTORx_Prescaler_Value】分频因子，配合修改输出频率；

函数：【Motor_Init()】电机控制初始化

函数：【Set_Motorx_A_Pwm()】占空比控制函数

函数：【Set_Motorx_B_Pwm()】占空比控制函数



UART 模式

自行配置相应的 UART2 和 UART3 即可通过串口通信的方式控制电机驱动工作。

涉及文件：【Motor】

陀螺仪 (MPU6050):

控制器配置一片高精度惯性陀螺传感器，I2C 通信方式，可接收三轴角加速度信息。

函数：【Init_MPU6050()】初始化 MPU6050；

函数：【READ_MPU6050()】获取传感器参数

变量：【A_X】【A_Y】【A_Z】三轴角加速度；

变量：【T_X】【T_Y】【T_Z】温度；



涉及文件：【mpu6050】

传感器 (ADC):

控制器配置 12 通道模拟量采集，其中 ADC11 为控制器电源电压测量通道，ADC12 为板载温度传感器通道，其余 10 通道为板载外设接口。ADC 精度达 12 位（最大值 4095）；选择外部基准电压 (3.3v)。配备相应 DMA 通道，初始化完成后自动转换 AD 数据。

函数：【ADC_Sensor_Init()】ADC 初始化；

函数：【Get_Sensor_Value()】转化电压、温度值；

变量：【ad_value[]】12 通道 ADC 实时数据；

变量：【Voltage】电源电压值；

变量：【Temperature】板载温度；



涉及文件：【Sensor】

串口 (UART):

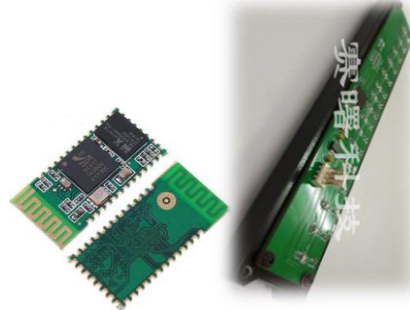
控制器最多可配置 5 路串行通信端口 (USRAT1, USRAT2, USART3, UART4, UART5), 配置和修改方式大同小异, 包括串口初始化、串口发送、波特率选择、串口接收及接收中断函数, 其中涉及到某些数据通信编码和解码协议可自行编写。注意每次接收完成后需要清接收标志位, 每次收/发一个字节数据;

函数: 【UART4_Init()】串口 4 初始化;

函数: 【UART4_Send()】串口 4 发送数据;

函数: 【UART4_IRQHandler()】串口 4 接收中断函数;

涉及文件: 【Bluetooth】【xxb】【uart4】



FLASH 存储:

对于某些重要参数或者控制器运行参数需要长期掉电保存的, 可以将其存入芯片 Flash, 需要时自动读取。使用此项功能时需注两点:

- 1)、存储地址选择, 避免与程序段地址冲突导致控制器无法正常工作。
- 2)、存入时擦除操作, 避免误擦除有效字段;
- 3)、每次操作数据长度为 16 位 (两个字节);

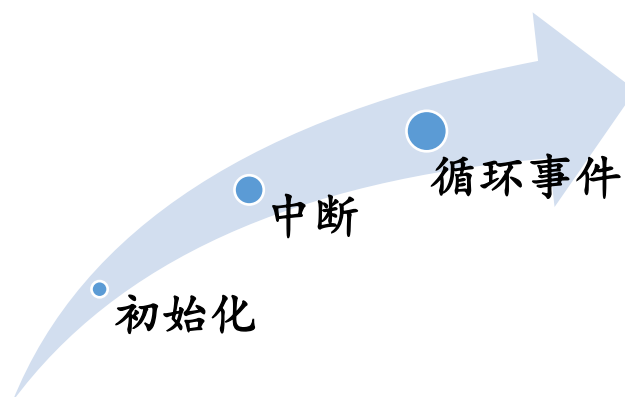
函数: 【FLASH_init()】Flash 初始化;

函数: 【FLASH_WriteByte()】写入数据;

函数: 【ReadFlashNBtye()】读取数据;

涉及文件: 【flash】

程序运行流程:



初始化: 系统所有的功能配置、开机自检等函数;

```
//=====系统初始化=====//

NVIC_Init();           //系统中断
SystemInit();          //系统时钟
delay_init();          //延时+心跳
Motor_Init();          //PWM
PCI_Init();            //I/O
ADC_Sensor_Init();     //ADC
Init_MPU6050();        //陀螺仪
BluetoothUart1_Init(); //串口1
keyboardConfig();      //键盘
UART4_Init();          //串口4
XxbUart5_Init();       //串口5
FLASH_init();          //Flash

//=====系统初始化完成=====//

delay_ms(300); //延时
Hummer_SystemStart(); //蜂鸣器提示
```

中断：定时器中断、串口接收中断、键盘按键中断、传感器接收中断、系统心跳中断等等，可以通过分配不同的中断优先等级来优先执行某些事件。例如传感器数据采集、电机速度控制等等具有一定周期、响应实时性的事件可以放置在定时器中断里面；而相对于紧急预警、紧急停机等等涉及安全因素的事件应当给予更高优先级。

```
//=====系统心跳服务函数=====
void SysTick_Handler(void) //10us触发一次
{
    t10ms = (++t10ms < 1000) ? t10ms : 0;
    if(!t10ms)
    {
        ;
    }
}
```

循环事件：系统无其它紧急事件时循环执行的函数，例如机器人持续运行函数、数据发送函数、显示屏/LED 指示类函数等，一般处理实时性要求不明显、延时性高、内容复杂的事件。

```
//=====循环事件=====//
while(1)
{
    ;//此处放置实时循环事件
}
```