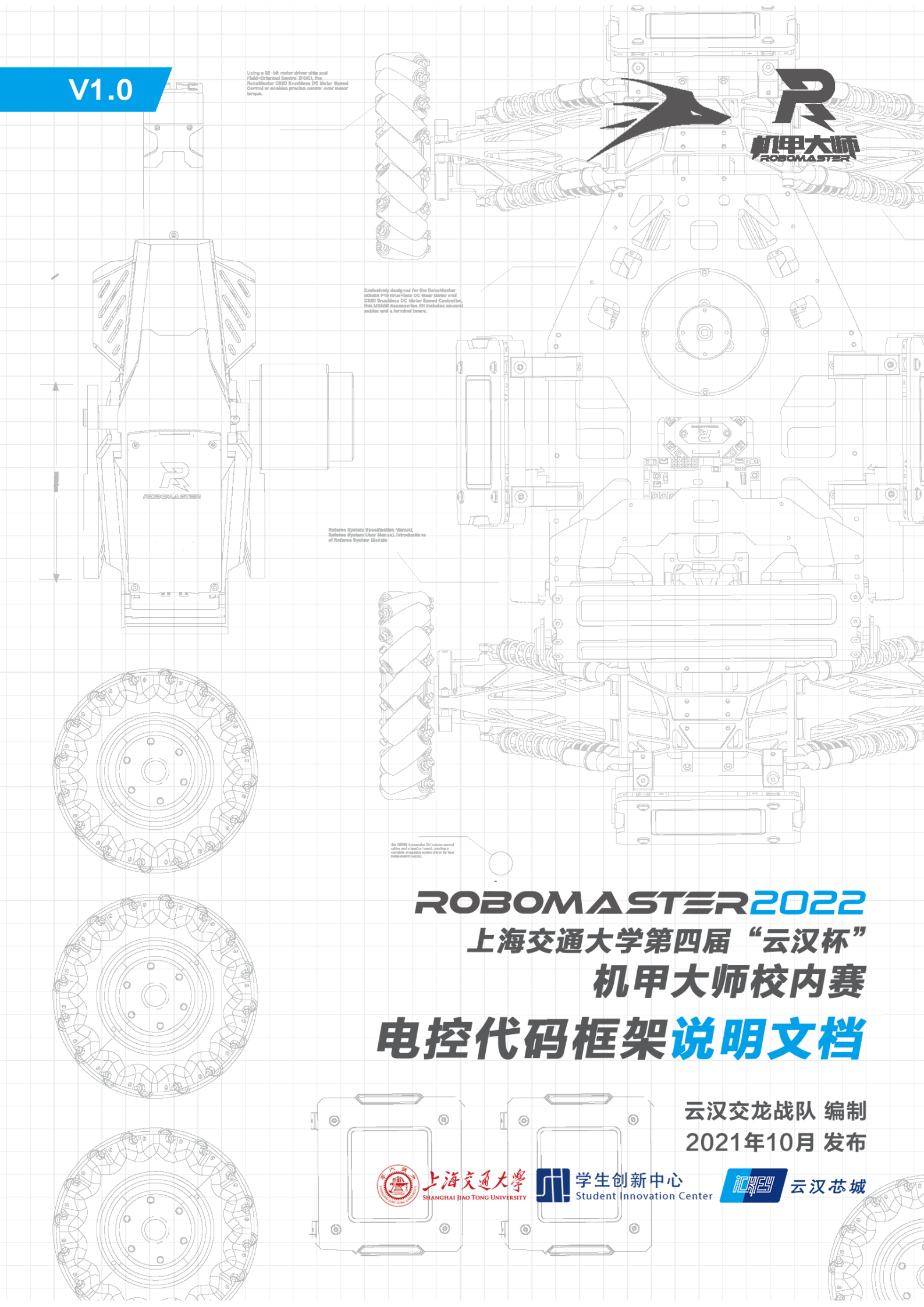
V1.0



ROBOMASTER 2022

上海交通大学第四届“云汉杯”

机甲大师校内赛

电控代码框架说明文档

云汉交龙战队 编制

2021年10月 发布

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

学生创新中心 Student Innovation Center

云汉芯城

环境配置

建议读者先配置电控环境，环境安装包已发在“参赛群”群文件->电控资料->环境配置文件包.zip 中，内含教程文件以及 keil 和 cubeMX 的安装包，其中 cubeMX 安装可跳过，**后续教程和调试不需要 cubeMX，**本说明文档适配 keil 环境。

修改日志

日期	版本	编辑者	修改内容
2021.10.12	V1.0	吕奇正，马嘉悦，郝常升	首次发布

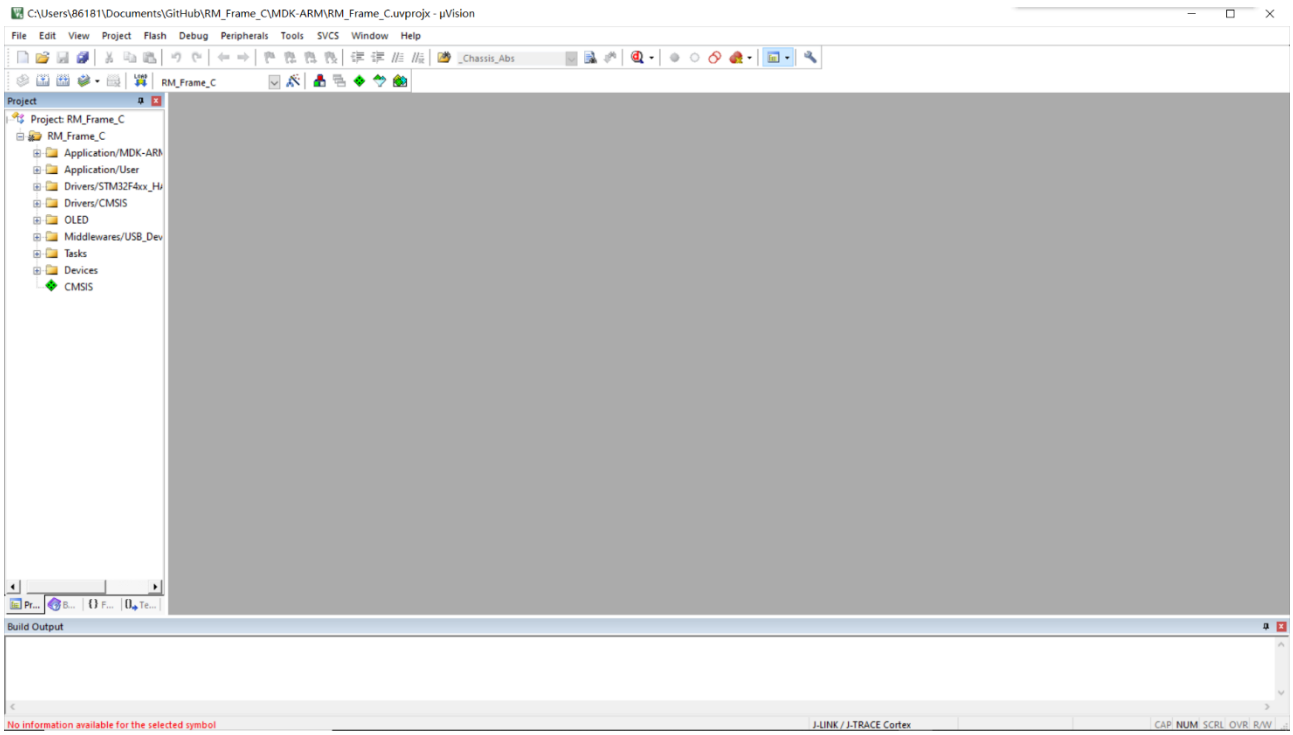
目录

1. 框架的使用、编译及烧写

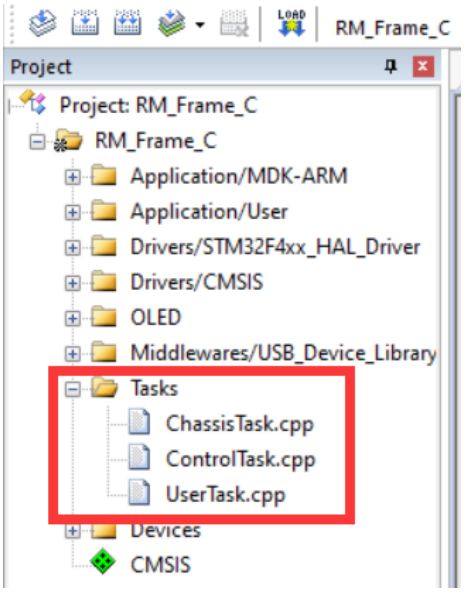
1.1 如何打开校内赛项目

下载校内赛框架，打开 MDK-ARM 文件夹，打开 RM_Frame_C.uvprojx 文件，如果已经配置好 keil 环境，该文件可以正常打开。

打开界面如图：



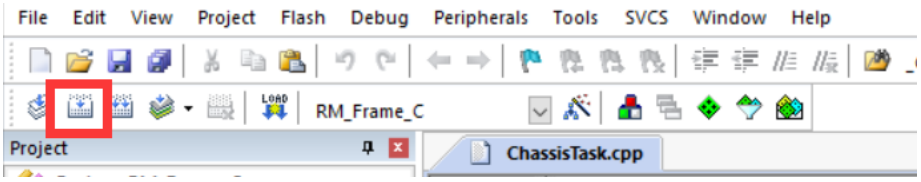
点击左侧项目文件目录中的 Tasks 目录，用户基本上都在本目录下修改&添加代码（具体修改方法后续章节会说明，此处仅简述几个初始代码文件的作用）。其中 ChassisTask.cpp 为底盘控制任务代码，未完成，需要参赛队伍自行补全。ControlTask.cpp 为遥控器控制任务代码，根据参赛队伍各自需求自行完善。双击可打开对应文件。



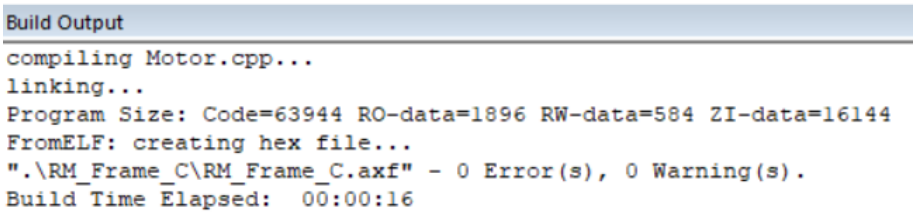
1.2 如何编译及烧写框架

1.2.1 编译

点击左上角工具栏中 Build 按钮编译

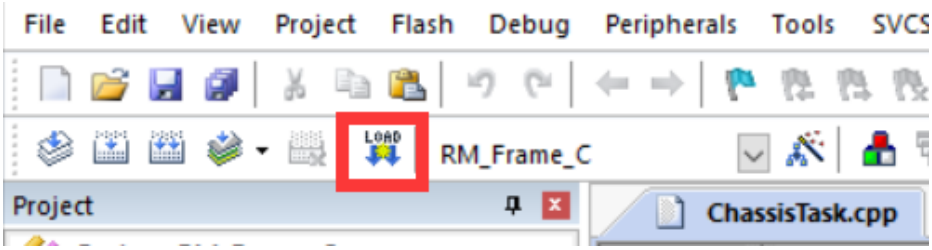


下方状态栏显示".\RM_Frame_C\RM_Frame_C.axf" - 0 Error(s), 0 Warning(s)即为编译成功

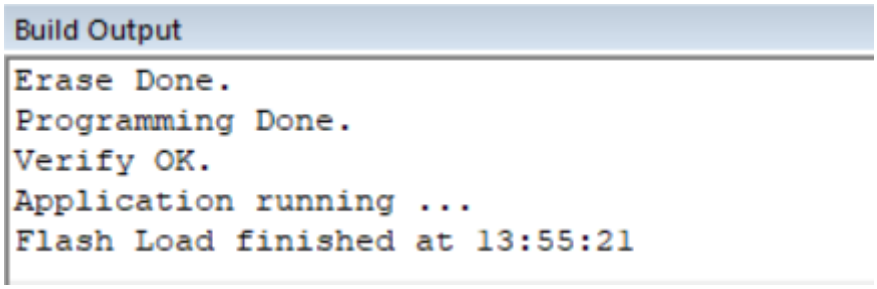


1.2.2 烧写

将发放的 ST-Link 的 USB 端插入电脑，4pin 端插入 C 板烧写口，点击左上角 Download 按钮烧写程序。



下方状态栏显示 Flash Load finished 即为烧写成功



烧写成功后可以看到主控板 C 板 led 小灯亮起，并周期性变色，可以用于验证是否烧写成功。

2. ChassisTask:底盘控制任务

2.1 功能简介

本部分代码负责控制参赛队伍机器人底盘运动，本赛事推荐使用麦克纳姆轮底盘（后简称麦轮），发放的官方物资中也含有 4 个麦轮及 4 个 3508 电机负责驱动底盘运动。麦轮得益于其独特的设计可实现不改变轮子方向的情况下实现快速的全向平移以及自旋。更多关于 4 麦轮底盘运动解算的内容在校内赛培训中已经说明，读者也可以自行搜索进一步理解学习。

本部分代码任务少，完成度较高，但重要性很高，决定了参赛机器人的下限（基本的运动），建议参赛队伍先完成此部分任务。

2.2 底盘运动控制

参赛队伍需要完成 WheelsSpeedCalc 函数中四个轮子速度解算部分，CMFLSpeed,CMFRSpeed,CMBLSpeed,CMBLSpeed 分别代表左前、右前、左后、右后轮，fbVelocity ,lrVelocity , rtVelocity 为要用到的三个变量，分别代表前后平移速度、左右平移速度及自旋速度。三个速度分量的大小均与遥控器摇杆推动距离成正比，从遥控器到这三个变量部分代码已在框架中，不需要参赛队伍自己完成，直接使用变量即可。

```
1.  //四个轮子线速度，单位：m/s
2.  CMFLSpeed = 0;
3.  CMFRSpeed = 0;
4.  CMBLSpeed = 0;
5.  CMBRSpeed = 0;
```

2.3 底盘电机 PID 调节

底盘电机使用 pid 单环控制，使用者可在 pidRegulator 结构体初始化中修改 pid 的三个系数以及 p,i,d 控制量的上限，直接修改数值即可。PID 调试内容在校内赛培训中已经说明，建议参赛队伍调参时逐步缓慢增大，尽可能避免超调，注意安全。

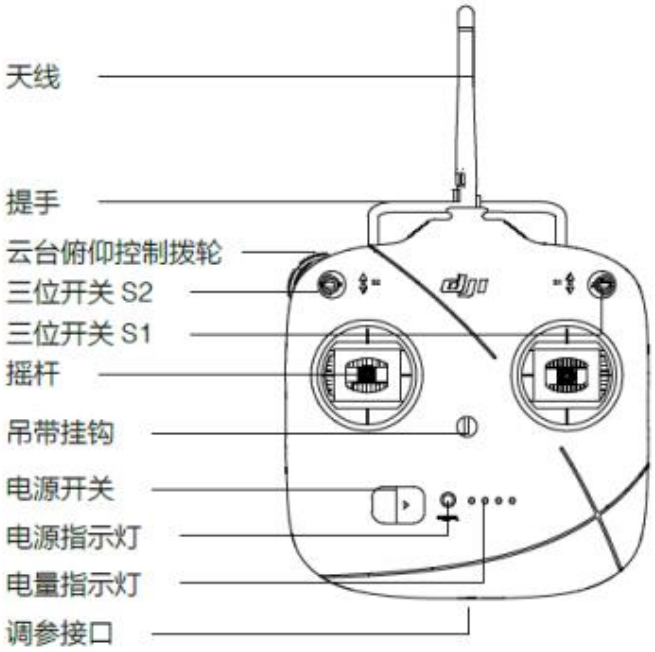
```
1.  PID_Regulator_t pidRegulator = {
2.      .kp = 1,
3.      .ki = 0,
4.      .kd = 0,
5.      .componentKpMax = 5000,
6.      .componentKiMax = 0,
7.      .componentKdMax = 0,
8.      .outputMax = 5000
9.  };
```

3. ControlTask:遥控器控制任务

3.1 遥控器的物理档位

首先介绍遥控器上的物理档位，遥控器示意图如下图所示。在本次校内赛中，电控框架给出了如下几种控制方式的编写接口：

1. 三位开关 S1
2. 三位开关 S2
3. 左右遥杆
4. 云台俯仰控制拨轮



3.2 代码介绍

在介绍完以上几种控制按键后，下面来介绍这几种控制方式的代码，注意，该代码存在于 RM_Frame_C\userCode\tasks\Src 的 Control.cpp 文件中，通过在该程序中对已经写好的其他功能接口（如底盘运动、控制舵机）进行调用即可实现通过遥控器对车辆和其功能的控制：

3.2.1 三位开关 S1

位于遥控器右上角，该开关对应变量：RemoteControl::rcInfo.sRight

该变量的值包括 UP_POS,MID_POS,DOWN_POS 分别对应了开关的上中下三档。

注意，该变量一般不用更改，仅仅明白这个变量的意义即可

3.2.2 三位开关 S2

位于遥控器左上角，该开关对应变量：RemoteControl::rcInfo.sLeft

该变量的值包括 UP_POS,MID_POS,DOWN_POS 分别对应了开关的上中下三档。

注意，该变量已经在框架里面给出：

```
switch (RemoteControl::rcInfo.sLeft) {
    case UP_POS://左侧一档
        break;
    case MID_POS://左侧二档
        break;
    case DOWN_POS:default://左侧三档
        break;
}
```

3.2.3 左右摇杆

左右遥杆分别位于遥控器中间部分的左右两侧。

左摇杆:

```
float RemoteControl::rcInfo.left_col;//上下值，向上返回值大于 0、向下小于 0
float RemoteControl::rcInfo.left_rol;//左右值，向右返回值大于 0、向左小于 0
```

右摇杆:

```
float RemoteControl::rcInfo.right_col;//上下值，向上返回值大于 0、向下小于 0
float RemoteControl::rcInfo.right_rol;//左右值，向右返回值大于 0、向左小于 0
```

以上值均返回一个-1~1 的 float 类型数。

如以下示例：

这里的 ChassisSetVelocity()为控制底盘速度的接口（具体实现见本教程相关内容），以下函数实现了右摇杆上下拨动控制前后移动（RemoteControl::rcInfo.right_col）、右摇杆左右拨动控制左右移动（RemoteControl::rcInfo.right_rol）、左摇杆左右拨动控制左右旋转（RemoteControl::rcInfo.left_rol）。（注意这里的 4.2 是一个系数，表明移动的速度，需要调一个合适的值）

```
ChassisSetVelocity(RemoteControl::rcInfo.right_col*4.2,RemoteControl::rcInfo.right_rol*4.2
,RemoteControl::rcInfo.left_rol*60);
```

3.2.4 云台俯仰控制拨轮

控制拨轮位于遥控器左上角。

```
float RemoteControl::rcInfo.dialWheel;//前后拨，往后大于 0，往前小于 0
```

返回一个-1~1 的 float 类型数，注意拨轮往后拨动返回值大于 0，往前拨动返回值小于 0。

具体使用方法与左右遥杆类似，这里不再赘述。

4. UserTask:使用者自定义任务

本部分为使用者自定义任务实现区域，底盘之外的独立设计如夹爪、转矿机构和爬升机构等。

本框架已含有电机、舵机类供使用者直接使用，UserTask.cpp 中已定义一个 2006 电机供使用者参考，官方物资中也含有一个 2006 电机。电机和舵机的具体使用说明见下文 4.3 及 4.4 节。

4.1 UserHandle:自定义任务主循环

自定义任务请写在 Handle 函数中，Handle 函数以 1ms 的频率反复调用，请注意算法，不要使用运行时间过长的算法，不要使用 HAL_Delay 等使程序停止运行的代码。

```
1.  /***
2.   * 用户自定义任务主循环
3.   */
4.  void UserHandle(){
5.      UserMotor.Handle();
6.  }
```

4.2 UserInit:自定义初始化

本函数为提供给使用者的一个初始化函数，用于参赛队各自机构的初始化部分，初始化任务可以理解为仅在开机时执行一次的任务，例如舵机、电机转到初始角度，或者电机以某规律转动来检查电机状态是否正常。直接在函数内写入代码即可，每次开机会自动调用一次本函数。

```
1.  /***
2.  * 在此函数中写入初始化内容
3.  */
4.  void UserInit(){
5.
6.  }
```

4.3 舵机的使用

舵机的具体实现在 RM_Frame_C\userCode\devices\Inc 的 Servo.h 和 RM_Frame_C\userCode\devices\Src 的 Servo.cpp 中。感兴趣同学自行阅读代码，本教程只对这两个源文件中对应的接口使用进行解释。

在本框架中，由于已经对舵机该设备进行了封装，所以在校内赛中，只用对以下提到的代码在 RM_Frame_C\userCode\tasks\Src 中的 C++源文件中相应进行调用即可。

一个舵机的使用流程：定义舵机->填写 handle 函数->用 settarget 函数赋值使其运动，除此之外还要使舵机在遥控器急停挡位断电。

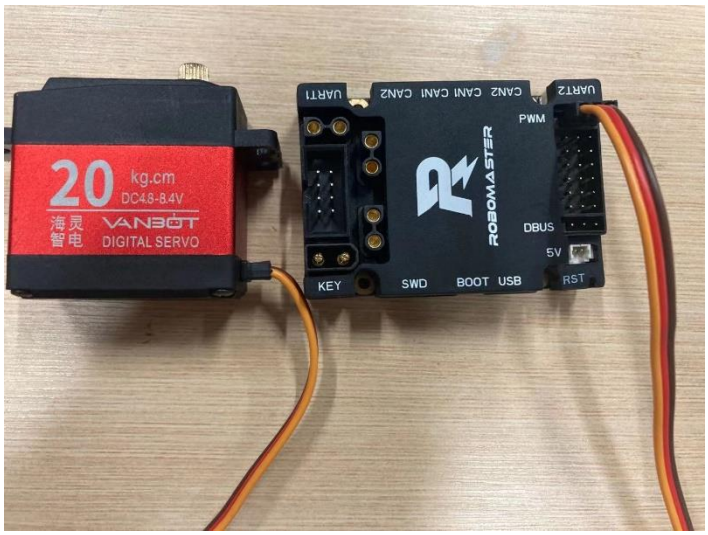
4.3.1 如何定义舵机

可以对以下参数进行调整来实现舵机的定义：

```
SERVO_INIT_T Servo_Test_Init ={      //初始化函数名称
    .servoType = ,          //舵机类型：POSITION_180：180°舵机（舵机类型自行搜索）
                            //POSITION_360：360°舵机（舵机类型自行搜索）
    .servoID =    ,          //舵机 ID：由 SERVO_ID_1~SERVO_ID_7 组成，与硬件接线有关，不可重复
    .firstAngle = ,          //开机角度：舵机初次上电时转到的角度
    .angleLimit_Min = ,     //最小角度：舵机可以转到的最小角度
    .angleLimit_Max =       //最大角度：舵机可以转到的最大角度
};
Servo TestServo(&Servo_Test_Init);    //声明舵机，调用先前的初始化函数
```

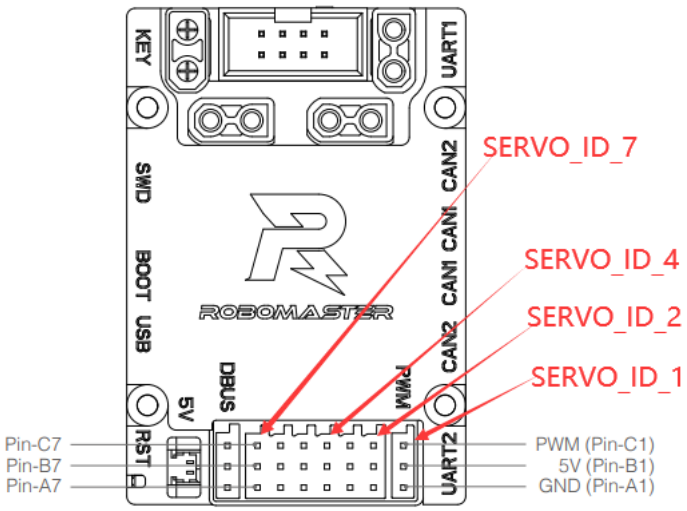
以下代码为定义名为 ClawServo 和 TurnLServo 的舵机的示例代码：

```
/*定义舵机 ClawServo*/
SERVO_INIT_T Claw_Servo_Init ={
    .servoType = POSITION_180,     //舵机类型：180°舵机（舵机类型自行搜索）
    .servoID = SERVO_ID_1,         //舵机 ID：由 SERVO_ID_1，接线如下图所示
    .firstAngle = 0,               //开机角度：舵机初次上电时转到的角度
    .angleLimit_Min = 0,           //最小角度：舵机可以转到的最小角度
    .angleLimit_Max = 180          //最大角度：舵机可以转到的最大角度
};
Servo ClawServo(&Claw_Servo_Init);    //声明舵机
/*定义舵机 TurnLServo*/
SERVO_INIT_T TurnL_Servo_Init ={
        .servoType = POSITION_180,
        .servoID = SERVO_ID_2,
        .firstAngle = 0,
        .angleLimit_Min = 0,
        .angleLimit_Max = 180
};
Servo TurnLServo(&TurnL_Servo_Init);
```



4.3.2 硬件部分

这里介绍舵机有关硬件接法，由于其和初始化代码中的.servoID 有关，所以这里在单独介绍一下，注意接线时舵机和舵机的 ID 相对应，并且按照线序正确链接（线序可以参考上图，按颜色连接）。



4.3.3 如何驱动舵机

将如下代码放入该舵机所定义 cpp 文件中的 Handle()函数中。

```
1.  void ServoTestHandle(){
2.  //在这里放入 TestServo.Handle()即可使舵机正常运行
3.  }
```

依然用之前的 ClawServo 舵机举例，将 ClawServo.Handle()函数放入本框架中预先提供的整体 Handle 函数即可使用，如果使用了其他舵机，均可以放在这个地方来驱动舵机。

```
1.  void ClawHandle(){
2.      ClawServo.Handle();
3.      TurnLServo.Handle();
4.  }
```

4.3.4 如何使舵机转到相应位置

可以对以下参数进行调整来实现舵机的转动：

```
1.  TestServo.SetTargetAngle(float _targetAngle); //可以将转到_angle 位置
```

以下为之前定义示例舵机 ClawServo 的驱动程序

```
1.  ClawServo.SetTargetAngle(10);                    //将舵机转到 10°
2.  ClawServo.SetTargetAngle(170);                   //将舵机转到 170°
```

注意该驱动程序可以放在遥控器的功能模块里，有关遥控器的功能模块会讲到在本文档的其他位置讲到。

4.3.5 如何使舵机断电

该部分为通过检录而准备在遥控器急停档位使的舵机断电的程序。

将如下代码放入该舵机所定义 cpp 文件中的 stop()函数中即可实现该功能。

```
1.  void ServoTestStop(){
2.  //在这里放入 TestServo.stop()即可使舵机在遥控器急停档位断电
3.  }
```

依然用之前的 ClawServo 舵机举例，将 ClawServo.stop()函数放入本框架中预先提供的整体 Stop 函数即可使用，如果使用了其他舵机，均可以放在这个地方来使其他舵机断电。

```
1.  void ClawStop(){
2.      ClawServo.stop();
3.      TurnLServo.stop();
4.  }
```

4.4 电机的使用

此处的电机特指官方发放的 M3508，M2006 电机，需配合 C620 和 C610 电调使用。控制这两种电机时即可使用 Motor 类。

4.4.1 如何定义电机

定义电机类对象时，需同时调用其构造函数

```
1.  /**
2. * @brief Motor 类的构造函数
3. * @param _init 类的初始化结构体指针
4. */
5. Motor::Motor(MOTOR_INIT_t* _init)
```

以及

```
1. /**
2. * @brief Motor 类的构造函数另一重载，可以方便地定义参数相同，ID 不同的电机
3. * @param _id 电机 ID
4. * @param _init 电机初始化结构体指针
5. */
6. Motor::Motor(uint32_t _id, MOTOR_INIT_t* _init)
```

其中涉及到电机类的初始化结构体

```
1. typedef struct {
2.     PID_Regulator_t* speedPIDp;//速度环 pid 参数结构体指针
3.     PID_Regulator_t* anglePIDp;//角度环 pid 参数结构体指针
4.     uint32_t _motorID;//电机 ID
5.     float reductionRatio;//减速比
6.     MOTOR_CTRL_TYPE_e ctrlType;//控制类型
7. }MOTOR_INIT_t;
```

特别需要注意的是，其中电机 ID 的取值，应为下列宏定义之一。电机 ID 前八个对应 C 型开发板 can1 上 1 到 8 的 ID，9 到 16 对应 C 型开发板 can2 上的 1 到 8。具体 ID 设置参见电调说明书。

```
1. /**
2.  * @defgroup motor_IDs
3. * @brief 电机 ID 前八个对应 C 型开发板 can1 上 1 到 8 的 ID，9 到 16 对应 C 型开发板 can2 上的 1 到 8
4.  */
5. #define MOTOR_ID_1 0x00000001u
6. #define MOTOR_ID_2 0x00000002u
7. #define MOTOR_ID_3 0x00000004u
8. #define MOTOR_ID_4 0x00000008u
9. #define MOTOR_ID_5 0x00000010u
10. #define MOTOR_ID_6 0x00000020u
11. #define MOTOR_ID_7 0x00000040u
12. #define MOTOR_ID_8 0x00000080u
13.
14. #define MOTOR_ID_9 0x00000100u
15. #define MOTOR_ID_10 0x00000200u
16. #define MOTOR_ID_11 0x00000400u
17. #define MOTOR_ID_12 0x00000800u
18. #define MOTOR_ID_13 0x00001000u
19. #define MOTOR_ID_14 0x00002000u
20. #define MOTOR_ID_15 0x00004000u
21. #define MOTOR_ID_16 0x00008000u
```

举例：chassisTask.cpp 中对几个底盘电机的声明

```
PID_Regulator_t pidRegulator = {//此为储存 pid 参数的结构体，四个底盘电机共用
    .kp = 60,
    .ki = 0,
    .kd = 0,
    .componentKpMax = 20000,
    .componentKiMax = 0,
    .componentKdMax = 0,
    .outputMax = 20000
};

MOTOR_INIT_t chassisMotorInit = {//四个底盘电机共用的初始化结构体
    .speedPIDp = &pidRegulator,
    .anglePIDp = nullptr,
    ._motorID = MOTOR_ID_1,
    .reductionRatio = 19.0f,
    .ctrlType = SPEED_Single,
};
Motor CMFL(MOTOR_ID_1,&chassisMotorInit);//定义左前轮电机
Motor CMFR(MOTOR_ID_2,&chassisMotorInit);//定义右前轮电机
Motor CMBL(MOTOR_ID_3,&chassisMotorInit);//定义左后轮电机
Motor CMBR(MOTOR_ID_4,&chassisMotorInit);//定义右后轮电机
```

4.4.2 如何驱动电机

将该电机的 Handle 函数放入某一个任务的 handle 中，比如 chassisHandle，UserHandle

```
/**
* @brief 底盘任务的处理函数，定时执行
* @callergraph void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) in Device.cpp
*/
void ChassisHandle() {
    if(ChassisStopFlag == 0) {
        WheelsSpeedCalc(FBVelocity, LRVelocity, RTVelocity);
    }
    CMFL.Handle();
    CMFR.Handle();
    CMBL.Handle();
    CMBR.Handle();
}
```

4.4.3 如何控制电机

电机有两种控制方式，需要在初始化结构体中确定。

```
/**
* @enum 控制电机的方式
* @example SPEED_Single 单环电机，控制速度
* @example POSITION_Double 双环电机，控制角度
*/
typedef enum{
    SPEED_Single,POSITION_Double
}MOTOR_CTRL_TYPE_e;
```

两种控制方式分别有一个控制函数和对应的控制逻辑

```
/**
* @brief 用于设置电机速度，只用当电机控制类型对应时才有效
* @param _targetSpeed 目标速度
```

```
 */
void Motor::SetTargetSpeed(float _targetSpeed) {
    stopFlag = 0;
    targetSpeed = _targetSpeed;
}
/**
* @brief 用于设置电机角度，只用当电机控制类型对应时才有效
* @param _targetAngle 目标角度
*/
void Motor::SetTargetAngle(float _targetAngle) {
    stopFlag = 0;
    targetAngle = _targetAngle;
}
```

例如在底盘控制四个轮子的速度时

```
CMFL.SetTargetSpeed(RADpS2RPM(CMFLSpeed));
CMFR.SetTargetSpeed(RADpS2RPM(CMFRSpeed));
CMBL.SetTargetSpeed(RADpS2RPM(CMBLSpeed));
CMBR.SetTargetSpeed(RADpS2RPM(CMBRSpeed));
```

4.4.4 如何使电机断电

该部分为通过检录而准备在遥控器急停档位使的电机断电的程序。

将如下代码放入该电机所定义 cpp 文件中的 stop()函数中即可实现该功能。

```
/**
* @brief 执行急停模式的底盘任务处理
*/
void ChassisStop(){
    ChassisStopFlag = 1;
    CMFL.Stop();
    CMFR.Stop();
    CMBL.Stop();
    CMBR.Stop();
}
```

云汉交龙战队
工作地点:学生创新中心B203、学生创新中心C104
微信公众号：上海交大交龙机器人俱乐部
Bilibili:RM云汉交龙战队