

# 第 14 节 RRC 与上位机通信协议分析

## 1.本节简介

本章节主要是分析 RRC 开发板与上位机（ROS 功能包）的通讯协议，为后续的上位机学习提供基础。

## 2.通讯协议

常用的指令例子可参考：[常用通信指令例子](#)

指令是由十六进制编写，若不熟悉计算方法，可以参考：[使用计算器工具进行进制转换](#)，关于负数和浮点数换算成十六进制，请搜索网上教程。

指令格式：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	(uint8_t) Function	(uint8_t) Length	Data	(uint8_t) CRC

- 帧头：连续收到 0xAA、0x55，表示有数据包到达；
- 功能码：用于标明一个信息帧的用途；
- 数据长度：参数个数值；
- 参数：除功能指令以外需要补充的控制信息；
- 校验：检验该数据是否正确，采用 CRC 校验方式，（CRC 计算 Function、Length、Data 的值，取低 8 位）（具体计算方法见拓展 2）。

### 2.1 用户主动给控制板发送数据部分

开发板上已有专门的 UART 转 USB 电路，只需用数据线将 UART3 口连接到上位机即可通信。

### 1、LED 灯控制：指令名 PACKET\_FUNC\_LED，数值 1

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_LED	7	参数 1: (uint8_t) led_id 参数 2: (uint16_t) 亮的时间(ms) 参数 3: (uint16_t) 灭的时间(ms) 参数 4: (uint16_t) 循环次数	CRC

举例：

①控制 LED 灯闪烁 5 次，每次亮 100ms 灭 100ms：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_LED	7	参数 1: 0x01 (1) 参数 2: 0x64 0x00 (100) 参数 3: 0x64 0x00 (100) 参数 4: 0x05 0x00 (5)	CRC

注意：这里是小端模式，如 uint32\_t 的 5（十进制）写为 0x00 0x05，小端模式是低位在前，所以发送数据时应该写为 0x05 0x00。

②控制 LED 灯闪烁 10 次，每次亮 500ms 灭 300ms：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_LED	7	参数 1: 0x01 (1)	CRC

			参数 2: 0xF4 0x01 (500) 参数 3: 0x2C 0x01 (300) 参数 4: 0x0A 0x00 (10)	
--	--	--	--	--

## 2、蜂鸣器控制：指令名 PACKET\_FUNC\_BUZZER，数值 2

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUZZER	8	参数 1: (uint16_t) 频率(Hz) 参数 2: (uint16_t) 响的时间(ms) 参数 3: (uint16_t) 不响的时间(ms) 参数 4: (uint16_t) 循环次数	CRC

举例：

①控制蜂鸣器鸣响 5 次，以 1400Hz 频率，每次鸣 100ms 停 100ms：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUZZER	8	参数 1: 0x78 0x05 (1400) 参数 2: 0x64 0x00 (100) 参数 3: 0x64 0x00 (100) 参数 4: 0x05 0x00 (5)	CRC

②控制蜂鸣器鸣响 10 次，以 1000Hz 频率，每次鸣 500ms 停 300ms：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUZZER	8	参数 1: 0xE8 0x03 (1000) 参数 2: 0xF4 0x01 (500)	CRC

			参数 3: 0x2C 0x01 (300) 参数 4: 0x0A 0x00 (10)	
--	--	--	---	--

### 3、编码器电机控制：指令名 PACKET\_FUNC\_MOTOR，数值 3

#### (1) 单个电机控制运动

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_MOTOR	6	参数 1: (uint8_t) 0x00 (子命令) 参数 2: (uint8_t) motor_id 参数 3: (float) 速度值 (有正负)	CRC

(注意: 1.float 类型为 4 个字节 2.电机 id 从 0 开始, 对应电机 1, 往后以此类推)

举例：控制电机 1 以 -1r/s 速度转动：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_MOTOR	6	参数 1: 0x00 参数 2: 0x00 参数 3: 0x00 0x00 0x80 0xBF (-1)	CRC

#### (2) 多个电机运动

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_MOTOR	5N+2	参数 1: (uint8_t) 0x01 (子命令) 参数 2: (uint8_t) 电机数量 参数 3: (uint8_t) motor_id_1	CRC

			参数 4: (float) speed_1 . . . . . 参数 2N+1: (uint8_t) motor_id_N 参数 2N+2: (float) speed_N (格式参考参数 3、4)	
--	--	--	---	--

(注意: float 类型为 4 个字节)

举例:

控制电机 1 和电机 2 分别以 -1r/s、2r/s 的速度转动:

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_MOTOR	12	参数 1: 0x01 (子命令) 参数 2: 0x02 参数 3: 0x01 参数 4: 0x00 0x00 0x80 0xBF (-1) 参数 5: 0x02 参数 4: 0x00 0x00 0x00 0x40 (+2)	CRC

### (3) 单个电机停止

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_MOTOR	2	参数 1: (uint8_t) 0x02 (子命令) 参数 2: (uint8_t) motor_id	CRC

举例:

控制电机 1 停止:

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_MOTOR	2	参数 1: 0x02 (子命令) 参数 2: 0x01	CRC

#### (4) 多个电机停止

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_MOTOR	2	参数 1: (uint8_t) 0x03 (子命令) 参数 2: (uint8_t) 电机掩码	CRC

举例：

控制电机 0 和电机 2 停止：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_MOTOR	2	参数 1: 0x03 (模式) 参数 2: 0x05 (二进制 00000101)	CRC

### 4、PWM 舵机控制：指令名 PACKET\_FUNC\_PWM\_SERVO，数值 4

#### (1) 多个 PWM 舵机控制

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_PWM_SERVO	3N+4	参数 1: (uint8_t) 0x01 (子命令) 参数 2: (uint16_t) 运动时间(ms) 参数 3: (uint8_t) 舵机数量	CRC

			参数 4: (uint8_t) servo_id_1 参数 5: (uint16_t) 脉冲宽度 . . . . . 参数 2N+2: (uint8_t) servo_id_N 参数 2N+3: (uint16_t) 脉冲宽度	
--	--	--	---	--

举例：控制舵机 1 和舵机 2 分别转到 90°、180°，分别对应脉宽为 1500、2500，用时 2s：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_PWM_SERVO	10	参数 1: 0x01 (子命令) 参数 2: 0xD0 0x07 (2000) 参数 3: 0x02 (需控制的舵机数量) 参数 4: 0x01 (1) 参数 5: 0xDC 0x05 (1500) 参数 6: 0x02 (2) 参数 7: 0xC4 0x09 (2500)	CRC

## (2) 单个 PWM 舵机控制

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_PWM_SERVO	6	参数 1: (uint8_t) 0x03 (子命令) 参数 2: (uint16_t) 运动时间(ms) 参数 3: (uint8_t) servo_id 参数 4: (uint16_t) 脉冲宽度	CRC

脉冲宽度为[ 500 , 2500 ]，对应的是[ 0° , 180° ]

举例：

控制舵机 1 通过 1s 旋转到 90° 位置，其脉冲宽度为 1500：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_PWM_SERVO	6	参数 1: 0x03 (子命令) 参数 2: 0xE8 0x03 (1000) 参数 3: 0x01 (1) 参数 4: 0xDC 0x05 (1500)	CRC

### (3) 读取 PWM 舵机位置

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_PWM_SERVO	2	参数 1: (uint8_t) 0x05 (子命令) 参数 2: (uint8_t) servo_id	CRC

发送读取位置命令后，控制板会发送对应 ID 的 PWM 舵机的位置数据给上位机。

### (4) 设置 PWM 舵机偏差

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_PWM_SERVO	3	参数 1: (uint8_t) 0x07 (子命令) 参数 2: (uint8_t) servo_id 参数 3: (uint8_t) 偏差参数 (转换为 int8_t 类型，有效范围为 -100 ~ +100)	CRC

举例：

设置舵机 2 的偏差为+10：

帧头	功能码	数据	参数	校验
----	-----	----	----	----



		长度		
0xAA 0x55	PACKET_FUNC_PWM_SERVO	3	参数 1: 0x07 (子命令) 参数 2: 0x02 (2) 参数 3: 0x0A (10)	CRC

#### (5) 读取 PWM 舵机偏差

帧头	功能码	数据 长度	参数	校验
0xAA 0x55	PACKET_FUNC_PWM_SERVO	2	参数 1: (uint8_t) 0x09 (子命令) 参数 2: (uint8_t) servo_id	CRC

发送读取位置命令后，控制板会发送对应 ID 的 PWM 舵机的偏差数据给上位机。

### 5、总线舵机控制：指令名 PACKET\_FUNC\_BUS\_SERVO，数值 5

#### (1) 控制总线舵机转动

帧头	功能码	数据 长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	3N+4	参数 1: (uint8_t) 0x01 (子命令) 参数 2: (uint16_t) 运动时间(ms) 参数 3: (uint8_t) 舵机数量 参数 4: (uint8_t) servo_id_1 参数 5: (uint16_t) 脉冲宽度 . . . . .	CRC

			参数 2N+1: (uint8_t) servo_id_N 参数 2N+2: (uint16_t) 脉冲宽度	
--	--	--	---	--

举例：

控制总线舵机的 1 号、2 号舵机分别转到 200°、240°，对应的脉宽为 833、1000，用时 1s：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	10	参数 1: 0x01 (子命令) 参数 2: 0xE8 0x03 (1000) 参数 3: 0x02 (2) 参数 4: 0x01 (1) 参数 5: 0x41 0x03 (833) 参数 4: 0x02 (2) 参数 5: 0xE8 0x03 (1000)	CRC

## (2) 读取位置命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x05 (子命令) 参数 2: (uint8_t) servo_id	CRC

发送读取位置命令后，控制板会发送对应 ID 的总线舵机的位置数据给上位机。

## (3) 读取输入电压命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x07 (子命令) 参数 2: (uint8_t) servo_id	CRC

发送读取输入电压命令后，控制板会发送对应 ID 的总线舵机的输入电压数据给上位机。

#### (4) 读取温度命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x09 (子命令) 参数 2: (uint8_t) servo_id	CRC

发送读取温度命令后，控制板会发送对应 ID 的总线舵机的温度数据给上位机。

#### (5) 电机掉电命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x0B (子命令) 参数 2: (uint8_t) servo_id	CRC

发送卸载动力命令后，控制板会将对应 ID 的总线舵机的动力卸载。

#### (6) 电机上电命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x0C (子命令) 参数 2: (uint8_t) servo_id	CRC

发送加载动力命令后，控制板会使对应 ID 的总线舵机加载动力。

### (7) ID 写入命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	3	参数 1: (uint8_t) 0x10 (子命令) 参数 2: (uint8_t) servo_id 参数 3: (uint8_t) save_id	CRC

发送 ID 写入命令后，控制板会改写对应 ID 的总线舵机的 ID 号。

### (8) ID 读取命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x12 (子命令) 参数 2: (uint8_t) 0xFE (查询广播)	CRC

发送 ID 读取命令后，控制板会将总线舵机的 ID 号上传给上位机。

---

**注意：**这里的总线舵机只能有一个，否则多个总线舵机同时返回数据会造成总线冲突。

---

### (9) 偏差调整命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	3	参数 1: (uint8_t) 0x20 (子命令) 参数 2: (uint8_t) servo_id 参数 3: (uint8_t) 偏差调整值	CRC

发送偏差调整命令后，控制板会将对应 ID 的总线舵机的偏差值设置为参数 3 的值。

### (10) 偏差读取命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x22 (子命令) 参数 2: (uint8_t) servo_id	CRC

发送偏差读取命令后，控制板会读取对应 ID 的总线舵机的偏差值并上传上位机。

### (11) 偏差保存命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x24 (子命令) 参数 2: (uint8_t) servo_id	CRC

发送偏差保存命令后，控制板会使能对应 ID 的总线舵机保存其当前的偏差值。

### (12) 位置限制设置命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	6	参数 1: (uint8_t) 0x30 (子命令) 参数 2: (uint8_t) servo_id 参数 3: (uint16_t) 低限位 参数 4: (uint16_t) 高限位	CRC

限位的数值范围在[ 0 , 1000 ]。

发送位置限制设置命令后，控制板会为对应 ID 的总线舵机设置限位参数。

### (13) 位置限制读取命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x32 (子命令) 参数 2: (uint8_t) servo_id	CRC

发送位置限制读取命令后，控制板会将对应 ID 的总线舵机的限位参数上传给上位机。

### (14) 电压限制设置命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	6	参数 1: (uint8_t) 0x34 (子命令) 参数 2: (uint8_t) servo_id 参数 3: (uint16_t) 低限位 参数 4: (uint16_t) 高限位	CRC

电压低限位数值应大于 4500，电压高限位数值应小于 14000。

发送电压限位设置命令后，控制板会设置对应 ID 的总线舵机的电压限位参数。

### (15) 电压限制读取命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x36 (子命令) 参数 2: (uint8_t) servo_id	CRC

发送电压限位读取命令后，控制板会将对应 ID 的总线舵机的电压限位参数上传给上位机。

### (16) 温度限制设置命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	3	参数 1: (uint8_t) 0x38 (子命令) 参数 2: (uint8_t) servo_id 参数 3: (uint8_t) 高温阈值	CRC

高温阈值参数值应该小于 100。

发送温度限位设置命令后，控制板会设置对应 ID 的总线舵机的高温阈值参数。

### (17) 温度限制读取命令

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	2	参数 1: (uint8_t) 0x3A (子命令) 参数 2: (uint8_t) servo_id	CRC

发送温度限位设置命令后，控制板会将对应 ID 的总线舵机的高温阈值参数上传给上位机。

## 2.2 控制板主动给用户发送数据部分

### 1、总线舵机数据上传：指令名 PACKET\_FUNC\_BUS\_SERVO，数值 5

#### (1) 总线舵机位置上传

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	5	参数 1: (uint8_t) servo_id 参数 2: (uint8_t) 0x05 (子命令) 参数 3: (int8_t) 是否读取成功 (0: 成功; -1: 失败) 参数 4: (int16_t) 舵机位置	CRC

当接收到读命令后，读取相应舵机的位置参数，上传给上位机。

举例：

上传 5 号舵机的角度为  $30^{\circ}$ ，对应的脉冲宽度为 833：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	5	参数 1: 0x05 (5) 参数 2: 0x05 (子命令) 参数 3: 0x00 (0) 参数 4: 0x41 0x03 (833)	CRC



### (2) 总线舵机电压上传

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	5	参数 1: (uint8_t) servo_id 参数 2: (uint8_t) 0x07 (子命令) 参数 3: (int8_t) 是否读取成功 (0: 成功; -1: 失败) 参数 4: (uint16_t) 舵机电压	CRC

当接收到读命令后，读取相应舵机的电压参数，上传给上位机。

### (3) 总线舵机温度上传

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	4	参数 1: (uint8_t) servo_id 参数 2: (uint8_t) 0x09 (子命令) 参数 3: (int8_t) 是否读取成功 (0: 成功; -1: 失败) 参数 4: (uint8_t) 舵机温度	CRC

当接收到读命令后，读取相应舵机的温度参数，上传给上位机。

#### (4) 总线舵机 ID 号上传

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	4	参数 1: (uint8_t) 0xFE (广播指令) 参数 2: (uint8_t) 0x12 (子命令) 参数 3: (int8_t) 是否读取成功 (0: 成功; -1: 失败) 参数 4: (uint8_t) 读取到的 servo_id	CRC

当接收到读命令后，读取舵机的 ID 号，上传给上位机。

#### (5) 总线舵机偏差参数上传

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	4	参数 1: (uint8_t) 0xFE (之前的广播指令) 参数 2: (uint8_t) 0x22 (子命令) 参数 3: (int8_t) 是否读取成功 (0: 成功; -1: 失败) 参数 4: (uint8_t) 舵机偏差	CRC

当接收到读命令后，读取对应舵机的偏差参数，上传给上位机。

#### (6) 总线舵机位置限制参数上传

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	7	参数 1: (uint8_t) servo_id 参数 2: (uint8_t) 0x32 (子命令) 参数 3: (int8_t) 是否读取成功 (0: 成功; -1: 失败) 参数 4: (uint16_t) 低限位 参数 5: (uint16_t) 高限位	CRC

当接收到读命令后，读取对应舵机的位置限制参数，上传给上位机。

#### (7) 总线舵机电压限制参数上传

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	7	参数 1: (uint8_t) servo_id 参数 2: (uint8_t) 0x36 (子命令) 参数 3: (int8_t) 是否读取成功 (0: 成功; -1: 失败) 参数 4: (uint16_t) 低限位 参数 5: (uint16_t) 高限位	CRC

当接收到读命令后，读取对应舵机的电压限制参数，上传给上位机。

### (8) 总线舵机温度限制参数上传

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_BUS_SERVO	4	参数 1: (uint8_t) servo_id 参数 2: (uint8_t) 0x3A (子命令) 参数 3: (int8_t) 是否读取成功 (0: 成功; -1: 失败) 参数 4: (uint8_t) 温度限制	CRC

当接收到读命令后，读取对应舵机的温度限制参数，上传给上位机。

### 2、上传按键消息：指令名 PACKET\_FUNC\_KEY，数值 6

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_KEY	2	参数 1: (uint8_t) button_id 参数 2: (uint8_t) 按键事件	CRC

部分按键事件回调值如下：

BUTTON\_EVENT\_PRESSED = 0x01, /\* 按钮被按下 \*/

BUTTON\_EVENT\_LONGPRESS = 0x02, /\* 按钮被长按 \*/

BUTTON\_EVENT\_CLICK = 0x20, /\* 按钮被点击 \*/

BUTTON\_EVENT\_DOUBLE\_CLICK = 0x40, /\* 按钮被双击 \*/

举例：

发送按键 1 被按下的消息：

帧头	功能码	数据长度	参数	校验
----	-----	------	----	----

0xAA 0x55	PACKET_FUNC_KEY	2	参数 1: 0x01 参数 2: 0x01	CRC
-----------	-----------------	---	--------------------------	-----

### 3、上传 IMU 数据：指令名 PACKET\_FUNC\_IMU，数值 7

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_IMU	24	参数 1: (float) accel_x 数据 参数 2: (float) accel_y 数据 参数 3: (float) accel_z 数据 参数 4: (float) gyro_x 数据 参数 5: (float) gyro_y 数据 参数 6: (float) gyro_z 数据	CRC

### 4、上传手柄数据：指令名 PACKET\_FUNC\_GAMEPAD，数值 8

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_IMU	7	参数 1: (uint16_t) 按键 参数 2: (uint8_t) hat 值 参数 3: (int8_t) 左摇杆 x 值 参数 4: (int8_t) 左摇杆 y 值 参数 5: (int8_t) 右摇杆 x 值 参数 6: (int8_t) 右摇杆 y 值	CRC

### 5、上传 SBUS 航模遥控器数据：指令名 PACKET\_FUNC\_SBUS，数值 9

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_SBUS	36	参数 1: (int16_t[16])16 个通道值 参数 2: (uint8_t)ch17 参数 3: (uint8_t)ch18 参数 4: (uint8_t)signal_loss 参数 5: (uint8_t)fail_safe	CRC

航模的摇杆、拨钮总共 16 个通道的数据都在参数 1 中。

## 6、PWM 舵机控制：指令名 PACKET\_FUNC\_PWM\_SERVO，数值 4

### (1) PWM 舵机位置上传

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_PWM_SERVO	4	参数 1: (uint8_t)servo_id 参数 2: (uint8_t)0x05（子命令） 参数 3: (uint16_t)脉冲宽度	CRC

当接收到读命令后，读取对应舵机的脉冲宽度，上传给上位机。

### (2) PWM 舵机偏差上传

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_PWM_SERVO	3	参数 1: (uint8_t)servo_id 参数 2: (uint8_t)0x05（子命令） 参数 3: (int8_t)舵机偏差	CRC

当接收到读命令后，读取对应舵机的脉冲宽度，上传给上位机。

## 常用的通信协议例子

准备：

打开串口助手，使用的 type-c 接入 UART2，波特率是 1000000，选择 HEX（16 进制发送）。

例子：

1. 控制蜂鸣器鸣响 5 次，以 1400Hz 频率，每次鸣 100ms 停 100ms：

```
AA 55 02 08 78 05 64 00 64 00 05 00 F0
```

2. 控制蜂鸣器鸣响 10 次，以 1000Hz 频率，每次鸣 500ms 停 300ms：

```
AA 55 02 08 E8 03 F4 01 2C 01 0A 00 8B
```

3. 控制 LED 灯闪烁 10 次，每次亮 500ms 灭 300ms：

```
AA 55 01 07 01 F4 01 2C 01 0A 00 04
```

4. 控制 LED 灯闪烁 5 次，每次亮 100ms 灭 100ms：

```
AA 55 01 07 01 64 00 64 00 05 00 37
```

5. 单个电机控制运动：控制电机 1 以-1r/s 速度转动：

```
AA 55 03 06 00 01 00 00 80 BF DA
```

6. 控制电机 1 停止：

```
AA 55 03 02 02 01 08
```

7. 控制电机 1 和电机 2 分别以 -1r/s、2r/s 的速度转动：

```
AA 55 03 0C 01 02 01 00 00 80 BF 02 00 00 00 40 FB
```

8. 控制 0 号、2 号电机停止：

```
AA 55 03 02 03 05 AD
```

9. 控制舵机 1 和舵机 2 分别转到 90°、180°，分别对应脉宽为 1500、2500，用时 2s：

```
AA 55 04 0A 01 D0 07 02 01 DC 05 02 C4 09 C8
```

10. 控制 PWM 舵机 1 运动到 1000 位置：

```
AA 55 04 06 03 e8 03 01 e8 03 e4
```

11. 设置 PWM 舵机偏差：设置舵机 2 的偏差为+10：

```
AA 55 03 03 07 02 0A 53
```

12. 控制总线舵机转动：控制总线舵机的 1 号、2 号舵机分别转到 200°、240°，对应的脉宽为 833、1000，用时 1s：

```
AA 55 05 0A 01 E8 03 02 01 41 03 02 E8 03 9F
```

13. 控制总线舵机转动：控制总线舵机的 1 号、2 号舵机都转到 0°，对应的脉宽为 0，用时 1s：

```
AA 55 05 0A 01 E8 03 02 01 00 00 02 00 00 D2
```

14. 控制 ID 为 1 的总线舵机掉电：

```
AA 55 05 02 0B 01 B3
```

15. 控制 ID 为 1 的总线舵机上电：

```
AA 55 05 02 0C 01 DD
```

16. 将 ID 为 1 的总线舵机 ID 号更改为 2：

```
AA 55 05 03 10 01 02 68
```



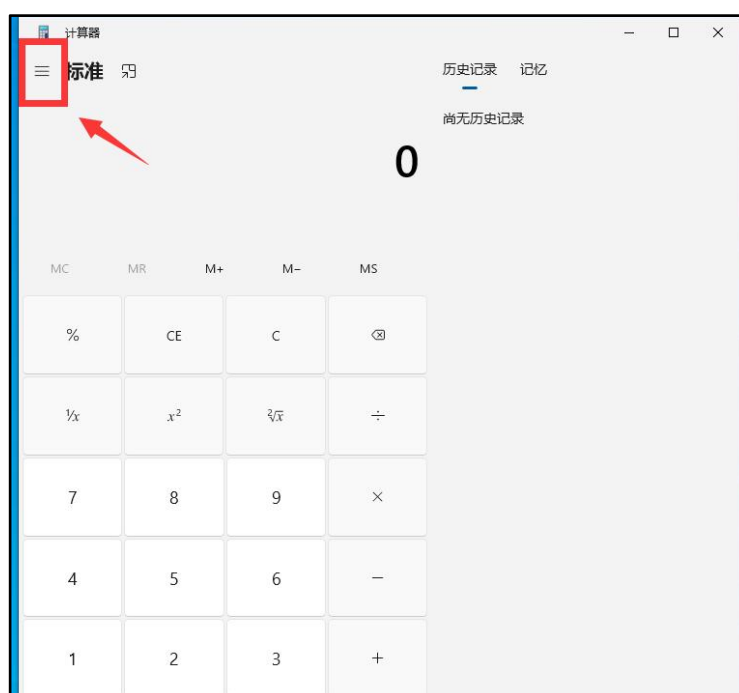
## 拓展 1—使用计算器工具进制转换

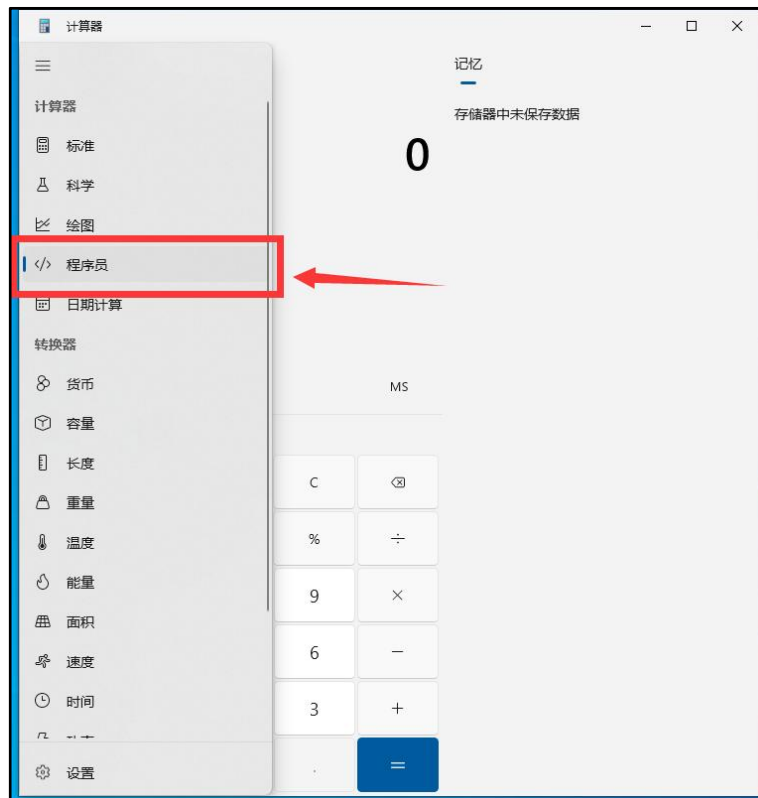
使用 win 系统的计算器进行进制转换

①：打开《开始》，搜索《计算器》，打开计算器。

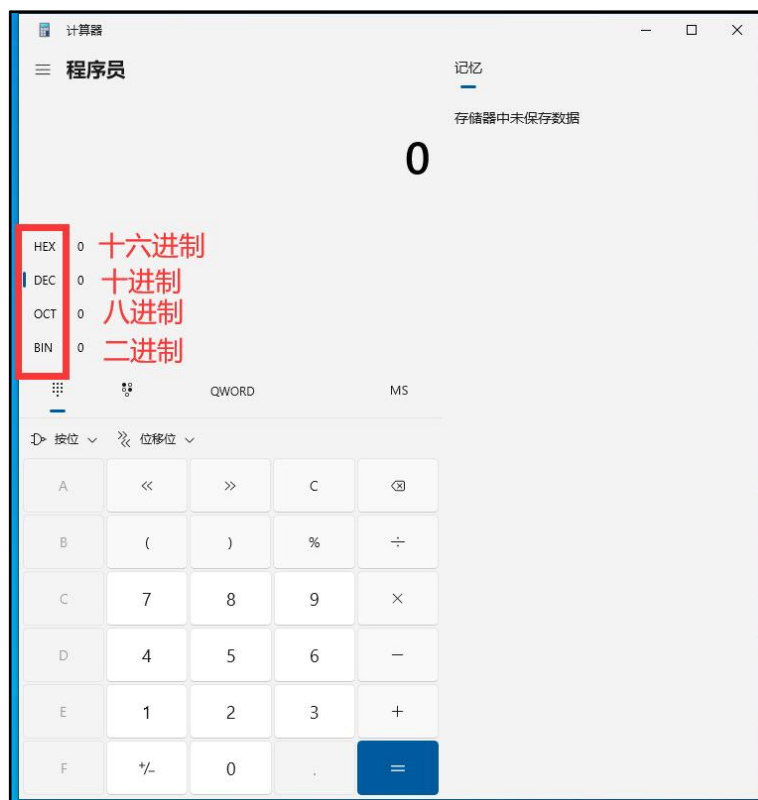


②：点击左上角的《打开导航》，选择《程序员》。





③：有四种进制可选。选中其中一个进制，输入数值，另外三个进制数就会显示出来。





## 拓展 2-CRC 码的计算方法

### 1. 简介

我们在串口指令的校验，选择了 CRC-Maxim 算法。CRC8-Maxim 是一种循环冗余检查 (CRC) 算法，用于数据完整性校验。CRC8-Maxim 是一种 8 位 CRC 算法，由 Maxim Integrated (现在是 Analog Devices 的一部分) 提出。它用于检测数据传输或存储过程中的错误。

算法基本特点包括：

- ① 多项式：CRC8-Maxim 使用的是多项式  $x^8 + x^5 + x^4 + 1$ ，它在二进制中表示为 0x31。这个多项式定义了如何生成 CRC 值。
- ② 初始化值：初始值通常为 0xFF。
- ③ 输入数据处理：数据在计算 CRC 时会逐字节处理，每个字节都参与 CRC 计算。

④ 异或操作：计算过程中涉及到的关键操作是异或（XOR），这是 CRC 计算中的核心操作之一。

⑤ 输出反转：CRC8-Maxim 的计算结果在最终输出时通常需要进行位反转（bitwise inversion）。

## 2. 操作流程及示例

我们在此处为您提供两种计算 CRC8-Maxim 校验码的方法：

方法 1：借助互联网的在线 CRC 计算器计算（下文有详细介绍）

方法 2：代码实现计算（此处不展开）

推荐使用方法 1，若您需要进行批量化、自动化测试，也可以查看方法 2 的 [CRC8\\_MAXIM.c](#) 提供的 C 语言实现 DEMO，封装到您的脚本中。

### 步骤 1：计算串口指令数据

我们先求取想要实现的指令的帧头、功能码、数据长度、参数信息。以“控制四路电机，以-1 的速度转动”为例子：

帧头	功能码	数据长度	参数	校验
0xAA 0x55	PACKET_FUNC_MOTOR	5N+2	参数 1：（uint8_t）0x01（子命令） 参数 2：（uint8_t）电机数量 参数 3：（uint8_t）motor_id_1 参数 4：（float）speed_1 ..... 参数 2N+1：（uint8_t）motor_id_N 参数 2N+2：（float）speed_N （格式参考参数 3、4）	CRC

功能码：PACKET\_FUNC\_MOTOR=0x03

数据长度：（参数的字节数）我们要同时控制 4 路电机，N=4，5N+2=22，换算成十六

进制：0x16

参数：参数 1：0x01

参数 2：0x04(四个电机)

参数 3：0x00(第 1 个电机 id)

参数 4：0x00 0x00 0x80 0xBF (-1)

参数 5：0x01(第 2 个电机 id)

参数 6：0x00 0x00 0x80 0xBF (-1)

参数 7：0x02(第 3 个电机 id)

参数 8：0x00 0x00 0x80 0xBF (-1)

参数 9：0x03(第 4 个电机 id)

参数 10：0x00 0x00 0x80 0xBF (-1)

最后，帧尾的 CRC 校验码与功能码、数据长度、参数三部分的信息有关，这就是我们需要校验的数据，我们将其摘取出来进行下一步计算：

03 16 01 04 00 00 00 80 BF 01 00 00 80 BF 02 00 00 80 BF 03 00 00 80 BF

## 步骤 2：计算 CRC 校验码

1) 您可以点击此处超链接访问 [CRC（循环冗余校验）在线计算 ip33.com](http://www.ip33.com/crc.html)，也可以浏览器访问网址：<http://www.ip33.com/crc.html>，进入到在线 CRC 计算器。

ip33.com

搜索其他工具

搜索

## CRC (循环冗余校验) 在线计算

☒ Hex
 ☐ Ascii
 

校验文件

需要校验的数据:

输入的数据为16进制, 例如: 31 32 33 34

参数模型 NAME: 自定义

宽度 WIDTH: 1

多项式 POLY (Hex): 例如: 3D65

初始值 INIT (Hex): 例如: FFFF

结果异或值 XOROUT (Hex): 例如: 0000

☐ 输入数据反转 (REFIN)
 ☐ 输出数据反转 (REFOUT)

计算

清空

校验计算结果 (Hex): 

复制

高位在左低位在右, 使用时请注意高低位顺序!!!

校验计算结果 (Bin): 

复制

首页

吐槽

2) 选择输入的编码格式为 HEX, 将需要校验的数据粘贴在“需要校验的数据”输入框中, 选择“参数模型”为“CRC-8/MAXIM”, 之后点击计算即可得到 CRC 校验码输出。

CRC (循环冗余校验) 在线计算

1

☒ Hex
 ☐ Ascii
 

校验文件

需要校验的数据:

2

 03 16 01 04 00 00 00 80 BF 01 00 00 80 BF 02 00 00 80 BF 03 00 00 80 BF

输入的数据为16进制, 例如: 31 32 33 34

参数模型 NAME: CRC-8/MAXIM x8+x5+x4+1

宽度 WIDTH: 8

多项式 POLY (Hex): 31 例如: 3D65

初始值 INIT (Hex): 00 例如: FFFF

结果异或值 XOROUT (Hex): 00 例如: 0000

☒ 输入数据反转 (REFIN)
 ☒ 输出数据反转 (REFOUT)

4

计算

清空

校验计算结果 (Hex): 2A
 

得到的CRC校验结果

复制

高位在左低位在右, 使用时请注意高低位顺序!!!

校验计算结果 (Bin): 00101010
 

复制