CAN 通信

从共享总线，到控制 M3508

RoboMaster Embedded Tutorial

这节课聚焦 5 个关键问题

1 **控制器局域网** 8 路点对点，如何变成 2 条共享网络

2 **物理信号** 差分信号与显性 / 隐性电平

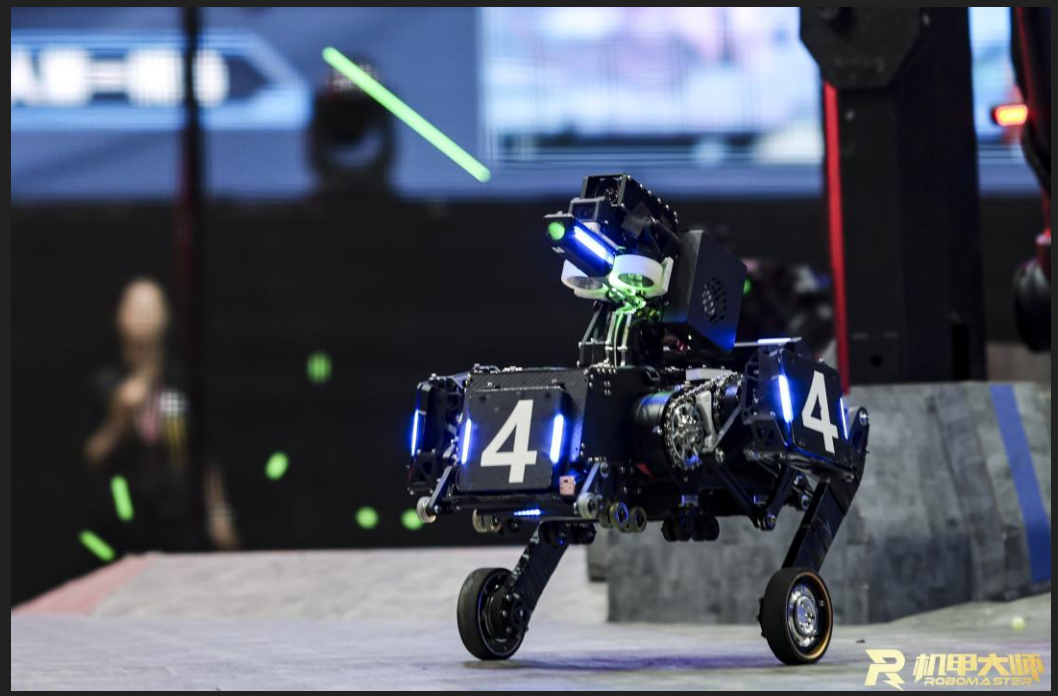
3 **收发机制** CAN ID、数据帧、仲裁与 Filter

4 **电机协议** 电调、编码器、M3508 / GM6020 与 CAN 帧

5 **上机实操** 接线、CubeMX、先读反馈再限速测试

如果只用点对点通信：8 台电调要占 8 路接口

以 RM2026 的串联腿平衡步兵底盘为例

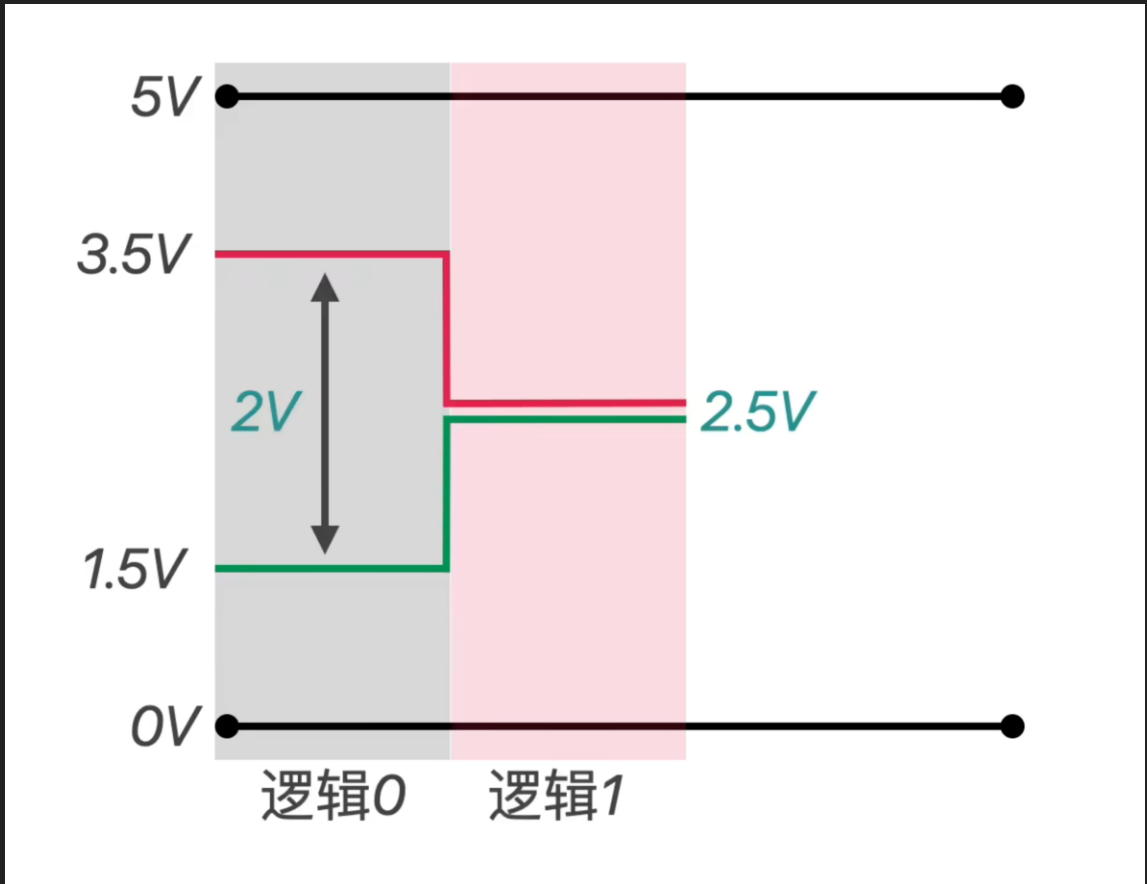


8 路独立通信接口 · 主板外设与引脚资源紧张 · 设备数量增加时，主板侧独立通信接口与独立链路同步增加

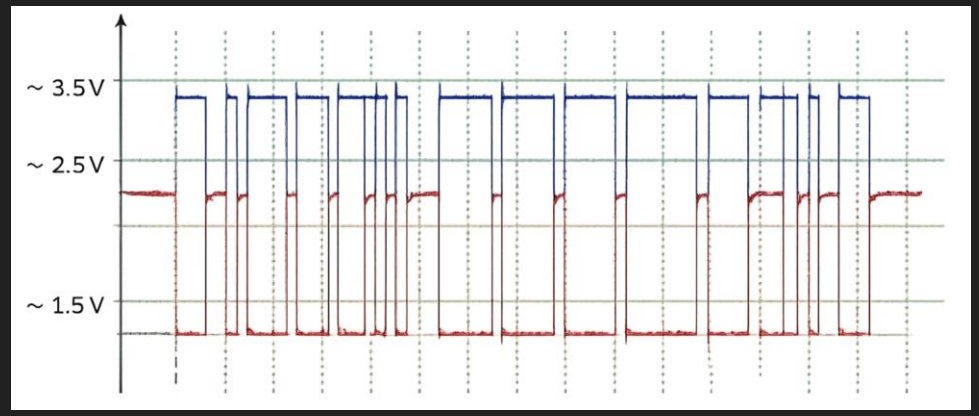
CAN 是控制器局域网：多个节点共享同一条总线

每一路 CAN 都是独立的局域网：节点共享 CAN_H / CAN_L，用 CAN ID 区分消息

差分接收只看 CAN_H - CAN_L：共同干扰会被相减

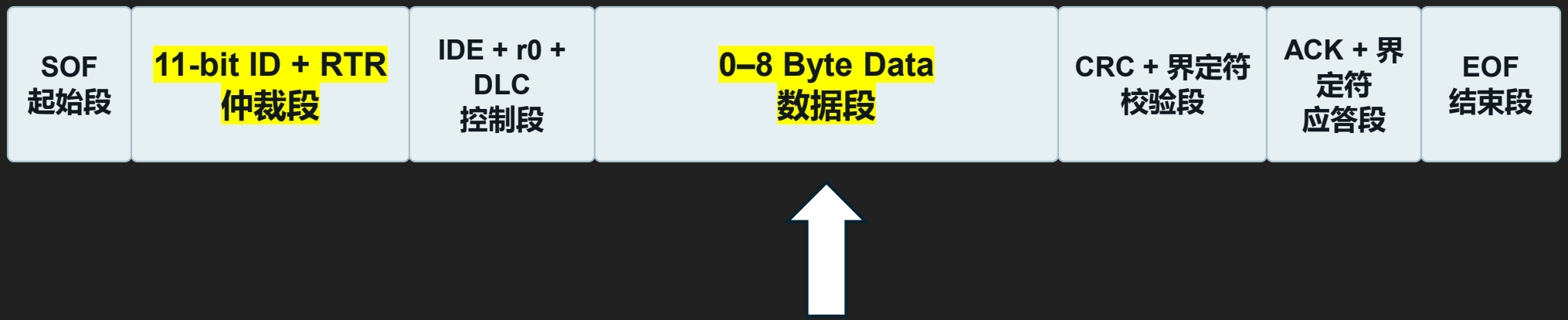


CAN_H / CAN_L 波形



两条线同时受干扰时，做差后共同变化会被抵消

CAN 协议帧，你只需要关注这两部分



真正要传输的数据就放在这

第一个问题

多个节点都想说话，听谁的？？

仲裁段

CAN 协议中定义了一个专门用于决定**哪个节点可以发送数据**的仲裁字段。

仲裁的两个机制

1. 在仲裁段中，只要任意节点发送显性位 0，总线就会呈现显性位 0；只有所有节点都发送隐性位 1 时，总线才会呈现隐性位 1。

2. 节点在发送仲裁位的同时会实时读取总线状态。若节点发送隐性位 1，却检测到总线为显性位 0，则立即停止发送当前帧，转为接收状态

通过前面的两条规则可以保证：

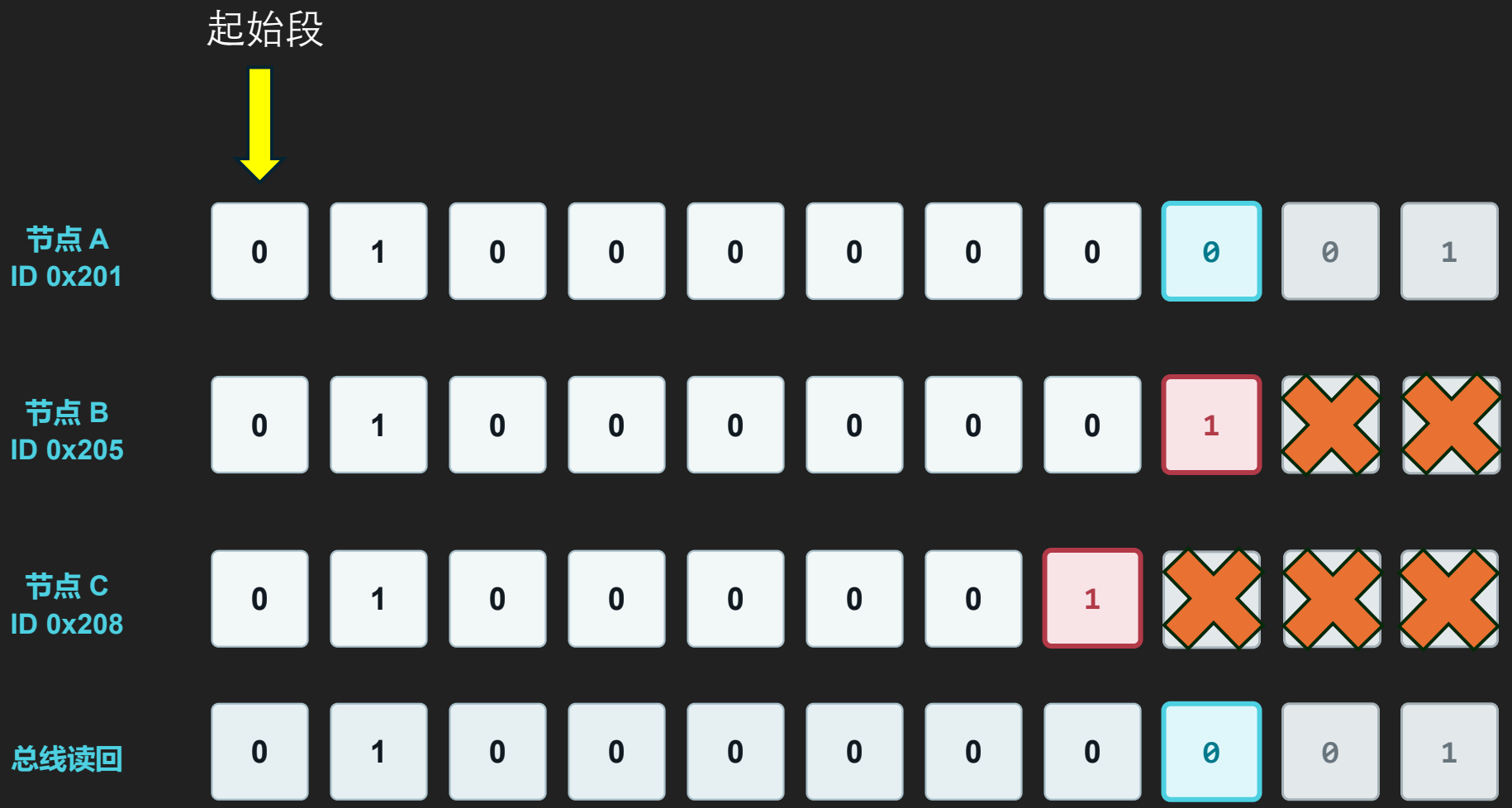
当多个节点同时发送不同 ID 的报文时，经过仲裁后，只有一条报文能够继续发送，其他节点则停止发送并转为接收状态。

也就是说：

在仲裁段的时候，是有可能多个节点同时“说话”的，但是在数据段的时候，只有一个节点能说话

（这有一个前提就是每个节点发送的报文id必须不同！）

举个例子



仲裁段结束后，只剩 A 一个节点继续发送整帧。

报文ID越小，报文的优先级越高，在仲裁中越容易获胜

注意：CAN ID 本身不表示节点地址，其具体含义由通信协议约定；

第二个问题

节点如何“选择性”收听报文?

Filter

共享总线上，所有节点都能看见同一帧。

Filter 决定本节点是否把这条消息放进 **FIFO**。

详细的 ID / Mask 判断放到后面的配置函数。

CAN总体流程回顾

发送方

1. 发送报文id
2. 仲裁成功后发送完整报文

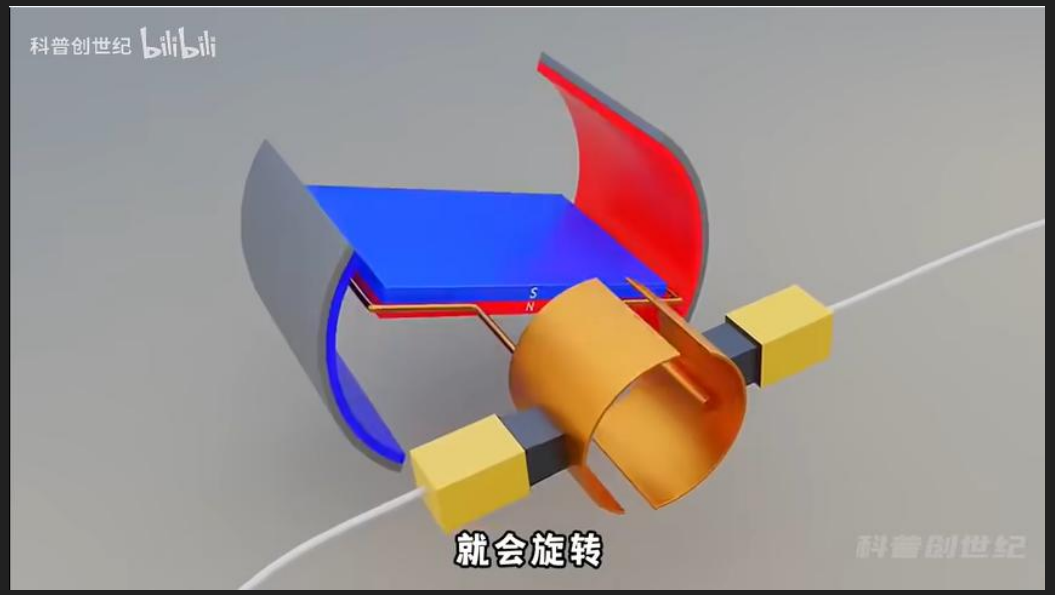
收听方

1. 收听报文ID
2. ID满足filter条件将报文放入FIFO队列，触发中断

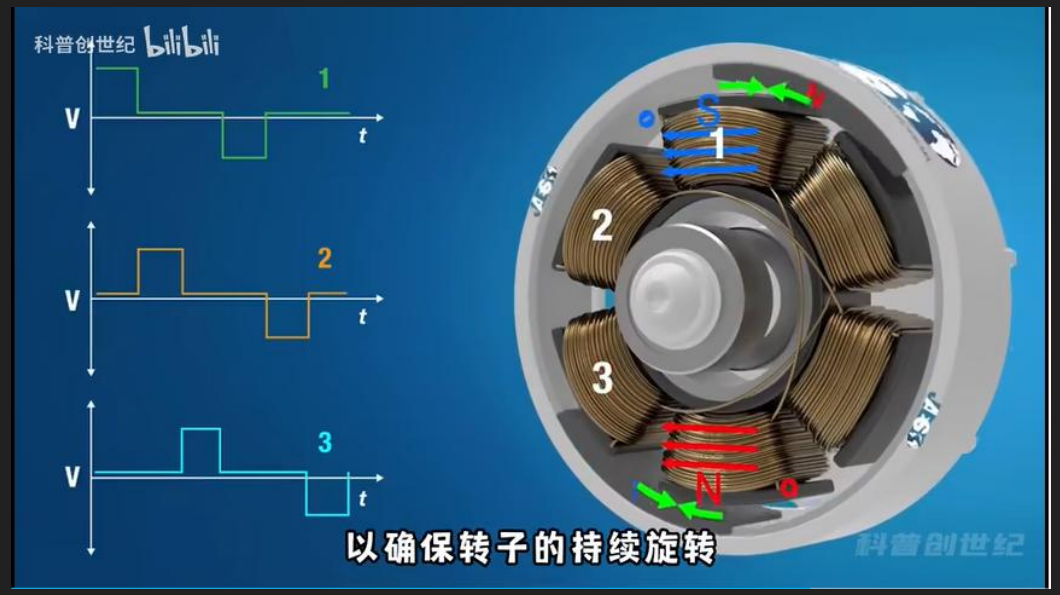
接下来：用 CAN 控制电机并读取反馈

电调（ESC）把控制命令变成三相驱动电流

有刷电机：机械电刷换向



无刷电机：电调电子换向



MCU 只能输出低功率逻辑信号，不能直接驱动 M3508 的三相绕组

C620 电调负责功率放大、电子换向，并把角度/转速等状态打包反馈

编码器跨零：正转与反转都会改变多圈 cnt

单圈反馈 **angle**：0-8191（对应 0-360°）

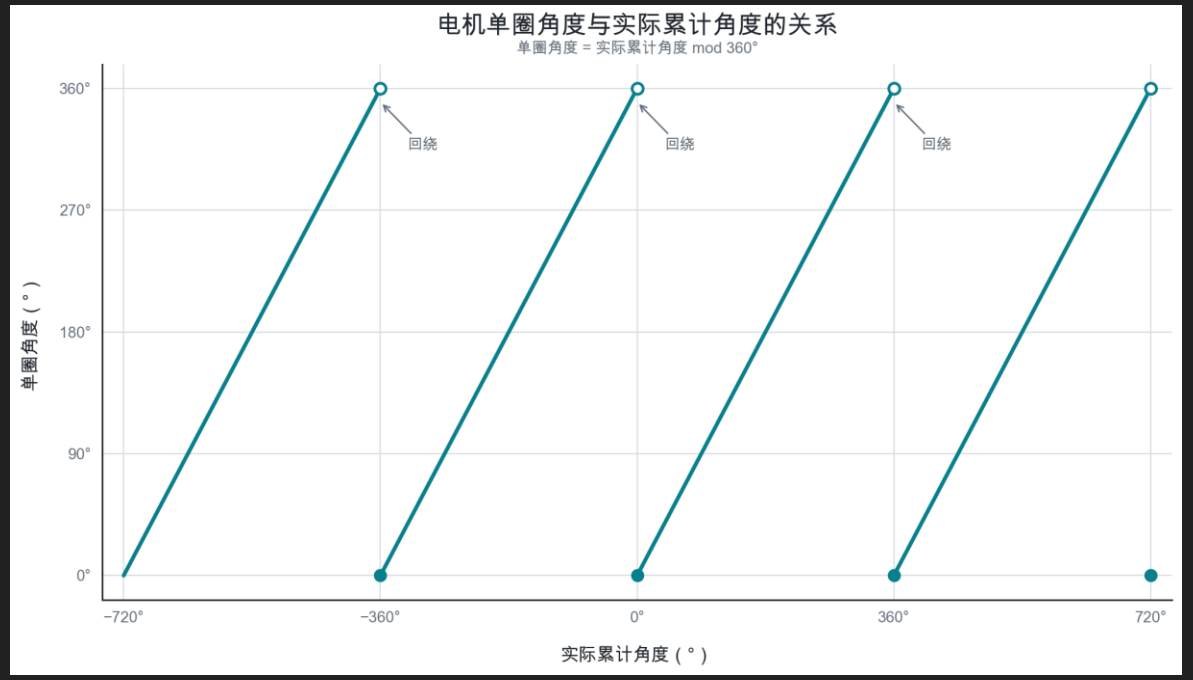
多圈计数 **cnt** 记录跨零方向

正转 8191 → 0：**cnt++**

反转 0 → 8191：**cnt--**

累计角度 = **cnt** × 360° + 单圈角度

实际累计角度 vs 单圈角度



正转跨零：cnt++　　**反转跨零：cnt--**

M3508 对比 GM6020

M3508 + C620
外置电调 · 减速输出

GM6020
驱动器集成 · 中空轴直驱

关键差异

M3508	GM6020
外置 C620	**内置驱动**
减速比 3591:187	低速直驱
底盘 / 轮系	云台 / 关节

本课实操：M3508 + C620

- **M3508**：多圈位置常以上电首帧为零点累计，属于相对位置。
- **GM6020**：云台单圈内可作为绝对角度，断电不丢零位。

M3508 通过 C620 接入 CAN 并反馈状态

位置传感器（编码器功能）测量转子角度；C620 把角度、转速、电流和温度打包成 CAN 反馈。

M3508的控制帧

3508电机的控制报文采用广播模式，即一条报文可以同时控制四个电机，控制帧的帧格式如下

标识符：0x200

帧格式：DATA

帧类型：标准帧

DLC：8 字节

数据域	内容	电调 ID
DATA[0]	控制电流值高 8 位	1
DATA[1]	控制电流值低 8 位	
DATA[2]	控制电流值高 8 位	2
DATA[3]	控制电流值低 8 位	
DATA[4]	控制电流值高 8 位	3
DATA[5]	控制电流值低 8 位	
DATA[6]	控制电流值高 8 位	4
DATA[7]	控制电流值低 8 位	

标识符：0x1FF

帧格式：DATA

帧类型：标准帧

DLC：8 字节

数据域	内容	电调 ID
DATA[0]	控制电流值高 8 位	5
DATA[1]	控制电流值低 8 位	
DATA[2]	控制电流值高 8 位	6
DATA[3]	控制电流值低 8 位	
DATA[4]	控制电流值高 8 位	7
DATA[5]	控制电流值低 8 位	
DATA[6]	控制电流值高 8 位	8
DATA[7]	控制电流值低 8 位	

解读M3508的控制帧协议

电调的ID不是CAN的ID！！！

控制 1-4 号电调

CAN ID：0x200

控制 5-8 号电调

CANID：0x1FF

电调反馈

反馈 ID = 0x200 + 电调 ID

范围：0x201-0x208

方向与 CAN ID

方向	CAN ID
控制 1-4	0x200
控制 5-8	0x1FF
反馈 1-4	0x201-0x204
反馈 5-8	0x205-0x208

电调 5-8 不必改硬件 ID：把代码中的 StdId 改为 0x1FF

控制帧：一帧同时控制 4 台电机

1 号电机		2 号电机		3 号电机		4 号电机	
DATA[0] 电流高 8 位	DATA[1] 电流低 8 位	DATA[2] 电流高 8 位	DATA[3] 电流低 8 位	DATA[4] 电流高 8 位	DATA[5] 电流低 8 位	DATA[6] 电流高 8 位	DATA[7] 电流低 8 位

每台电机占 2 字节：`tx[0] = current >> 8`　　`tx[1] = current & 0xFF`

范围 -16384 … +16384，对应约 -20 … +20 A

M3508的反馈帧

电调反馈报文格式

电调向总线上发送的反馈数据。

标识符：0x200 + 电调 ID（如：ID 为 1，该标识符为 0x201）

发送频率：1KHz （默认值，可在 RoboMaster Assistant 软件中修改发送频率）

帧格式：DATA

帧类型：标准帧

DLC：8 字节

数据域	内容
DATA[0]	转子机械角度高 8 位。角度值范围：0 ~ 8191 （对应 0~360°）
DATA[1]	转子机械角度低 8 位。角度值范围：0 ~ 8191 （对应 0~360°）
DATA[2]	转子转速高 8 位（单位：rpm）
DATA[3]	转子转速低 8 位（单位：rpm）
DATA[4]	实际转矩电流高 8 位
DATA[5]	实际转矩电流低 8 位
DATA[6]	电机温度（单位：℃）
DATA[7]	电机错误码： 0：无异常 1：无法访问电机中的存储芯片（仅开机自检） 2：电调供电电压过高（仅开机自检） 3：电机三相线未接入 4：与电机相连的数据线中位置传感器数据丢失 5：电机温度异常或过高 (≥180℃) 7：电机校准失败 若同时存在多个警告或异常状态，优先反馈级别更高（数值更小）的错误码。 8：电机过热 (≥125℃)

反馈帧: 来自电机的反馈信息

反馈 CAN ID = 0x200 + 电调 ID

编码器角度 0-8191 · 转速 RPM · 电流 · 温度 °C

DATA[0]	DATA[1]	DATA[2]	DATA[3]	DATA[4]	DATA[5]	DATA[6]	DATA[7]
角度高 8 位	角度低 8 位	转速高 8 位	转速低 8 位	电流高 8 位	电流低 8 位	温度	错误码

```
angle = (DATA[0] << 8) | DATA[1]    speed = (DATA[2] << 8) | DATA[3]    error = DATA[7]
```

错误码: 0 正常 · 1 存储芯片 · 2 供电过压 · 3 三相线未接
4 位置数据丢失 · 5 温度异常/过高 · 7 校准失败 · 8 电机过热

实操：接线、配置、收发、验证

开始前准备好设备、软件和资料

硬件

1. Internal 板
2. C620 电调 + M3508 电机
3. 24 V 电源
4. CAN 线

软件

1. CubeMX
2. VSCode
3. Ozone

下载资料

M3508 电机说明书（PDF）

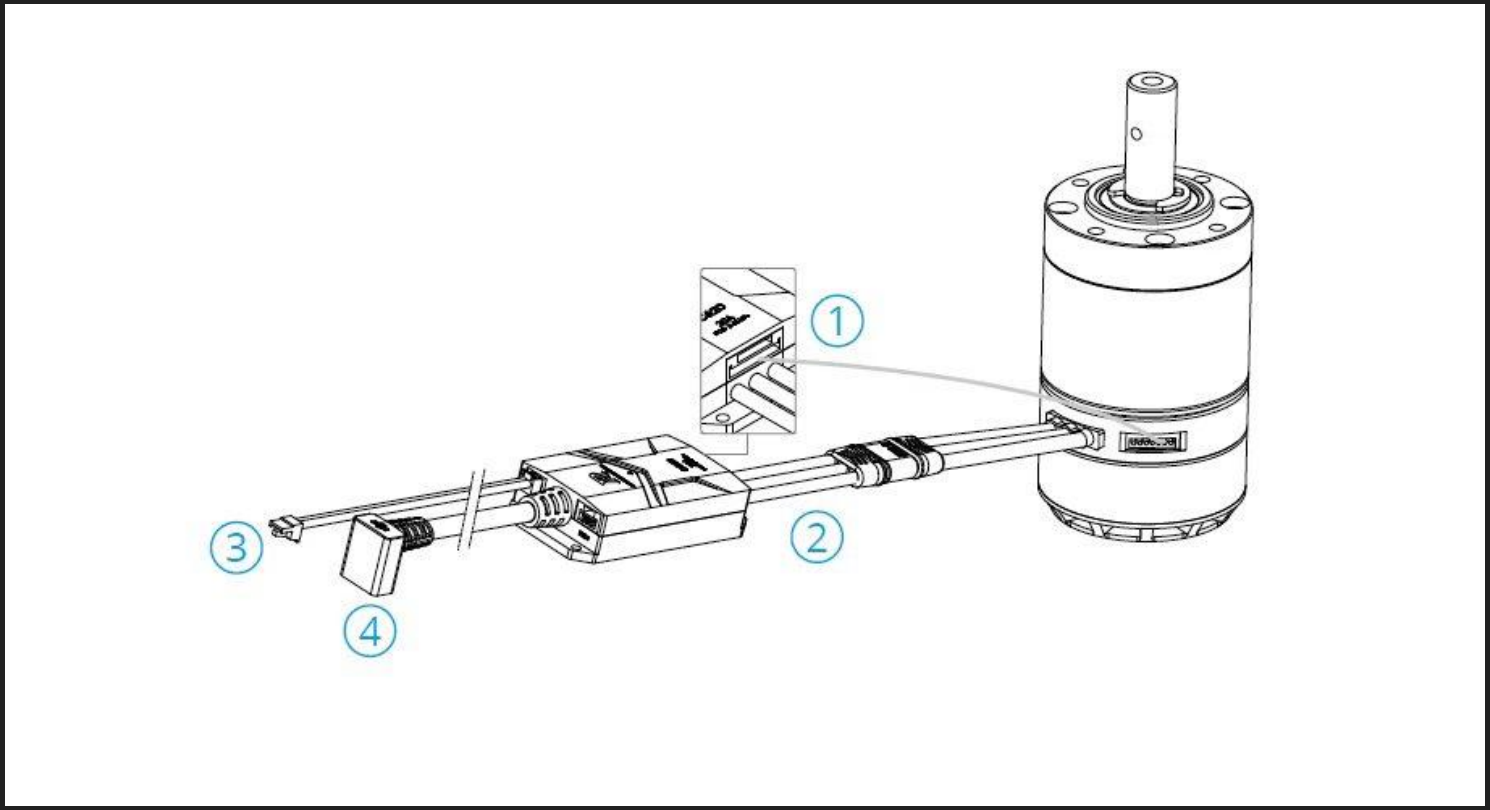
C620 电调说明书（PDF）

WARNING!!!

这是电机的转子

电机转动的时候不要碰！

C620 与 M3508：先认清 4 组线



1 7-Pin 数据线
电机位置传感器 ↔ C620

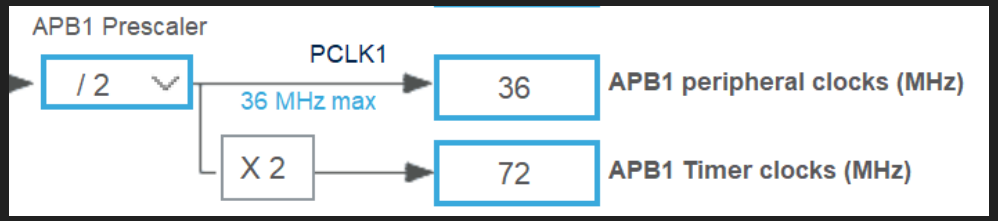
2 三相动力线
C620 驱动电机三相绕组

3 CAN 信号线
接 CAN_H / CAN_L 总线

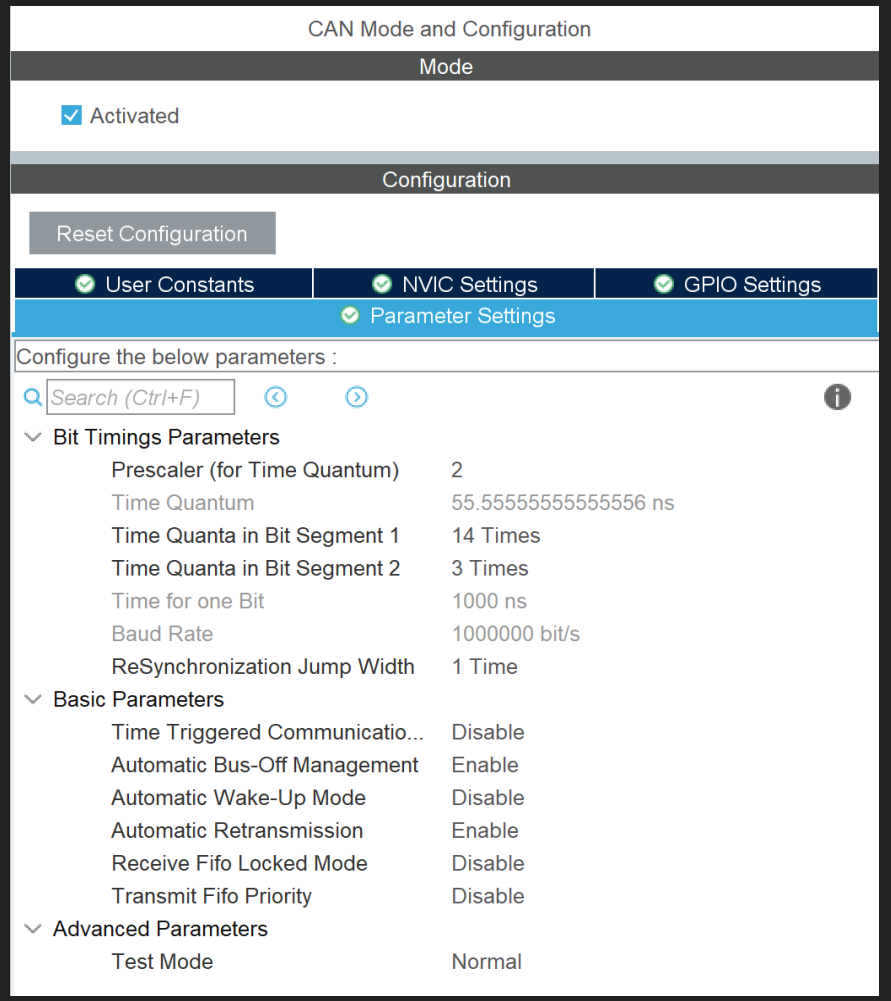
4 24 V 电源线
给 C620 功率级供电

CubeMX：完成时钟、参数与中断配置

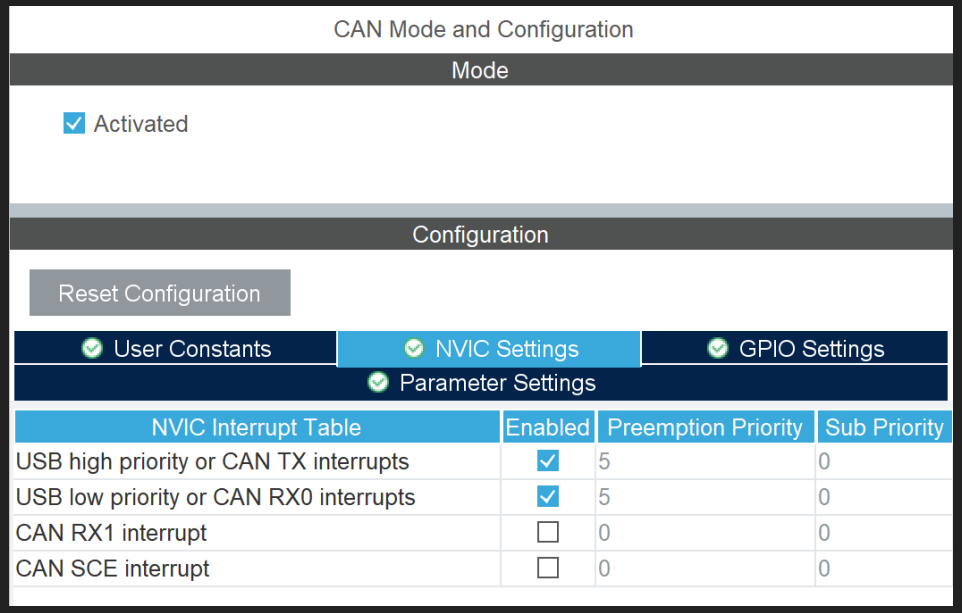
1 APB1 外设时钟 = 36 MHz



2 CAN Activated；参数结果Baud Rate = 1 Mbps



3 NVIC 勾选 USB_LP_CAN1_RX0 interrupt



Filter 判断：（收到 ID & Mask）== Filter ID?

Filter里面有两个东西，一个Mask，一个Filter ID

对于总线上的报文ID，Filter执行如下操作：判断(总线 ID & Mask) == Filter ID?

如果是，则将报文内容放入FIFO，否则“拒收”

Filter 判断：（收到 ID & Mask）== Filter ID?

Mask 中的 1：该位参与比较
Mask 中的 0：该位忽略

判断为真 → **进入 FIFO0**
判断为假 → 直接丢弃

右侧用两组 ID **逐位验证**。

Tips | 数字前缀

前缀	含义
0b	二进制
0o	八进制
无前缀	十进制
0x	十六进制

例 1 | 只接收 0x205

收到的 ID	`0b 010 0000 0101`
Mask	`0b 111 1111 1111`
收到ID & Mask	`0b 010 0000 0101`
Filter ID	`0b 010 0000 0101`

完全一致 -> 进入 FIFO0

例 2 | 接收 0x200-0x20F

收到的 ID	`0b 010 0000 xxxx`
Mask	`0b 111 1111 0000`
收到ID & Mask	`0b 010 0000 0000`
Filter ID	`0b 010 0000 0000`

0x201-0x208 在范围内 -> 接收反馈

Filter 判断：(收到 ID & Mask) == Filter ID ?

```
// TODO: Define the CAN filter for m3508
CAN_FilterTypeDef m3508Filter = {
  .FilterIdHigh         = 0x200 << 5,
  .FilterIdLow          = 0,
  .FilterMaskIdHigh     = 0x7F0 << 5,
  .FilterMaskIdLow      = 0,
  .FilterFIFOAssignment = CAN_FILTER_FIFO0,
  .FilterBank           = 0,
  .FilterMode           = CAN_FILTERMODE_IDMASK,
  .FilterScale          = CAN_FILTERSCALE_32BIT,
  .FilterActivation     = CAN_FILTER_ENABLE,
  .SlaveStartFilterBank = 14
};
```

接收启动：配置 Filter、通知与 CAN

```
// TODO: Start CAN!

// Config the filter for the message to be received
HAL_CAN_ConfigFilter(&hcan, &m3508Filter);
// Activate the notification/interrupt for FIFO 0 new message
HAL_CAN_ActivateNotification(&hcan, CAN_IT_RX_FIFO0_MSG_PENDING);
// Start the CAN peripheral
HAL_CAN_Start(&hcan);
```

接收：从 FIFO0 取 8 字节，并解码 DATA[7] 错误码

```
// M3508 结构体中先新增：uint8_t error;
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
  CAN_RxHeaderTypeDef rxHeader;
  uint8_t CAN_rxBuf[8];

  HAL_CAN_GetRxMessage(
    hcan, CAN_RX_FIFO0, &rxHeader, CAN_rxBuf
  );

  m3508.id          = rxHeader.StdId;
  m3508.angle       = (CAN_rxBuf[0] << 8) | CAN_rxBuf[1];
  m3508.speed       = (CAN_rxBuf[2] << 8) | CAN_rxBuf[3];
  m3508.current     = (CAN_rxBuf[4] << 8) | CAN_rxBuf[5];
  m3508.temperature = CAN_rxBuf[6];
  m3508.error       = CAN_rxBuf[7];
}
```

TX Header：StdId 决定控制哪 4 台电调

```
int16_t current = 0;
uint8_t can_tx_buf[8];
uint32_t txMailbox;

CAN_TxHeaderTypeDef txHeader = {
  // 电调 1-4 用 0x200; 电调 5-8 改为 0x1FF
  .StdId              = 0x200,
  .ExtId              = 0,
  .IDE                = CAN_ID_STD,
  .RTR                = CAN_RTR_DATA,
  .DLC                = 8,
  .TransmitGlobalTime = DISABLE
};
```

发送：先只控制 1 号电机，其余 3 路保持 0

```
// 1 号电机占 DATA[0:1]
can_tx_buf[0] = (current >> 8) & 0xFF;
can_tx_buf[1] = current & 0xFF;

// 2-4 号电机在 bring-up 阶段保持 0
can_tx_buf[2] = 0;
can_tx_buf[3] = 0;
can_tx_buf[4] = 0;
can_tx_buf[5] = 0;
can_tx_buf[6] = 0;
can_tx_buf[7] = 0;

HAL_CAN_AddTxMessage(
  &hcan, &txHeader, can_tx_buf, &txMailbox
);
```

转动前完成 5 项检查，确保接线与操作安全

1 CAN_H 对 CAN_H，CAN_L 对 CAN_L，并且共地

2 7-Pin 数据线、三相线和 24 V 电源插牢

3 电调 1-4：发送 **0x200**；电调 5-8：发送 **0x1FF**

4 CubeMX 确认 **1 Mbps**，并开启 CAN RX0 中断

5 固定电机、手和线远离转子；电流保持 0，**先确认收到反馈**