FIRMWARE MANUAL

D2842-400-HB

D2842-400-HB · 出厂固件 FOC_Double_C · 使用说明书

副标题	D2842-400-HB · 出厂固件 FOC_Double_C · 使用说明书
品牌	Daliang Auto
产品型号	D2842-400-HB（硬件 B 版）
固件名称	FOC_Double_C
固件版本	0.4.24（构建标识 `fc9016f`，2026-09-10 发布；镜像 235524 字节，CRC32 `4792a233`）
文档版本	Rev. 1.6
发布日期	2026-09-10
控制台	USB Type-C 虚拟串口，115200 8N1，无流控
对外接口	控制台命令行 · =tlm 遥测 · 整车 CAN · VGUS 触摸屏

本书与规格书的分工

- **《D2842-400-HB 产品规格书》描述硬件**：电气参数、引脚、连接器、功率能力、机械尺寸。 硬件不改版，规格书就不改。
- **本书描述出厂固件**：命令、标定流程、控制逻辑、保护动作、通信协议、升级方式。 **固件每发布一版，本书跟着走一版**，两者互不牵动。
- 两本书里都出现的量（例如母线电压范围、霍尔极性），**以规格书为准**， 因为那是硬件事实；本书只说固件怎么使用它。

本书对应的是固件 0.4.24。 板子上跑的是哪一版，用 `ver` 命令问它，不要从出厂日期推。 ⚠ Rev 1.4 及更早这一句写的是 0.4.17，而同一页的表头写的是 0.4.20 —— **同一本书里两个答案。** 2026-09-10 一并订正。以后这两处必须一起改。 不同版本之间的差异见 §12 版本历史，其中标注了哪些功能是从哪一版才有的。

---

1. 这套固件是什么

出厂固件把这块板变成一台**可以直接使用的双路电机驱动器**：接上电机、霍尔和电源， 用一根 USB 线做一次标定，之后踏板、触摸屏或整车 CAN 都能驱动它。客户不需要写一行代码。

板卡同时也支持客户自行开发固件（SWD 与 USB 一键下载均已就绪，硬件资源见《产品规格书》的引脚分配表与连接器一览表）。 **本书只描述出厂固件。**

1.1 功能地图

层	固件提供什么
电机控制	双路独立三相 FOC：电流（转矩）、速度、位置、开环电压 / 开环转速五种模式
自学习标定	一条命令测出这台电机的接线、电阻电感、霍尔换相表与机械增益，存在这块板自己的 Flash 里
整车通路	踏板 / 拨杆 / 触摸屏 / CAN / 控制台五个指令源，固定优先级仲裁，挡位、限速、档位带、刹车驱动
极简模式	没有电脑、没有触摸屏、没有挡位开关的整车，靠一次踏板手势使能（0.4.15 起，**出厂默认开**，见 §6）
保护	23 个故障码，每条分「切桥 / 降额 / 仅报告」三档动作；切桥一律锁存
对外接口	人用的命令行、程序用的 `=tlm` 遥测、控制板用的整车 CAN、仪表用的 VGUS 屏
升级	独立 Bootloader + 单个应用区，串口升级，镜像自带版本与校验。⚠ **0.4.18 起没有回滚，而且升级会清空标定**——升级前先按 §11.3 备份

1.2 四条对外通路

通路	对面是谁	物理层	本书章节
控制台命令行	人，坐在终端前	USB Type-C（USART1），115200 8N1	§2 – §7
`=tlm` 遥测	程序：上位机、测试脚本、SDK	同上，与命令行共用一条串口	§10.1
整车 CAN	上位控制板	经典 CAN，≤ 1 Mbps	§10.4
VGUS 触摸屏	仪表	外部 TTL 串口 CN36	§10.5

板上每个连接器出厂固件拿来做什么、收的是什么信号，见 §1.3。
这四条路是四个受众，不是四种格式偏好。人要的是句子，程序要的是不会因为多一个字段就崩的结构， 控制板要的是双方约定好的字节，屏要的是它自己那一套。

1.3 接口功能定义：出厂固件把每个口用来做什么

《产品规格书》回答的是「这个连接器接到哪个引脚、针序是什么」，本节回答两件事： **出厂固件拿它做什么**，以及**这个口收发的是什么信号、什么格式。**

针序、间距、电平耐压、功率能力一律看规格书（§9 通信接口、§10 遥控输入、§11 控制 I/O、 §16 连接器一览表），本节不重复。

> 换一版固件，本节就可能变；规格书那几节不会变。这正是两本书分开的原因。

1.3.1 一眼看完：哪个口做什么

指令输入

连接器	引脚	出厂固件的用途	收什么
CN26	PB1	**模拟油门**（`thr_src 0`，出厂默认的油门源）	**0 – 5 V 直流模拟量**，见 §1.3.2
CN2	PA3 / PB10	PWM 油门 ×2（`thr_src 3` / `4`）	**占空比方波，不是舵机脉宽**；⚠ 只能接 5 V 源，见 §1.3.2
U6 · RC1	PB9	遥控通道 1	舵机 PWM，1000 – 2000 µs
U6 · RC2	PD3	遥控通道 2	舵机 PWM
U6 · RC3	PD2	遥控串行链路	**S.BUS 或 iBUS**（可切换），见 §1.3.2
CN4	PC3	**模式**（挡位的一半）	干接点，低有效
CN6	PF9	**方向**（挡位的另一半）	干接点，低有效
CN3	PF10	⚠ **急停输入**（**不是刹车开关**，见 §1.3.4 第一条）	**常闭**干接点：合上 = 运行，开路或断线 = 停
CN5 / CN40	CAN1	整车 CAN，与上位控制板对接	经典 CAN ≤ 1 Mbps，见 §10.4
CN36	USART3（PD8 / PD9）	**VGUS 触摸屏**	A5 5A 变量帧，波特率自动搜索，见 §1.3.2
USB1	USART1	控制台命令行 + `=tlm` 遥测	ASCII 文本行，115200 8N1，无流控
SW2	PF2	**本地锁**：长按一次切换（0.4.11 起）	板载按键，接不了线
板载电位器 ×2	PE14 / PE15	台架旋钮（`pot_bench_ma`）或当油门源（`thr_src 1` / `2`）	⚠ **出厂默认关**（`pot_debug`），见 §9.4

功率与反馈

连接器	出厂固件的用途	说明
P4 / P6	电机 1 / 电机 2 三相线	相序不用人管：`learn` 自己测出来
U15 / U9	电机 1 / 电机 2 霍尔（5P：+5 V、GND、HA、HB、HC）	六路霍尔都经板上反相施密特缓冲，**固件已经把这层反相还原**；霍尔表由 `learn` 实测，不用人工换算
P5	有刷 H 桥输出：刹车电缸 / 执行器	⚠ **出厂默认不驱动**，要 `brake on` + `save`，见 §5.3
P7	母线泄放电阻	**硬件比较器自治**，44.1 V 开 / 43.1 V 关。固件只能读状态、只能强制关，**开不了它**
J1 / F1 / CN1	主电源、保险、外接电源开关	固件不参与
P1 / P2 / P3 / CN7 / U2	+24 V / +12 V / +5 V / +3.3 V 输出	固件不参与
H1	SWD 调试	见规格书 §14

1.3.2 每一路收的是什么信号

模拟油门（CN26）：0 – 5 V 直流模拟量

项目	内容
接口收什么	**0 – 5 V 直流电压。**霍尔式踏板或电位器踏板都行
连接器	2.0 mm 3P：**1 = GND / 2 = THR / 3 = +5 V**（可直接给踏板供电）
板上前端	分压 ×0.667 + RC 滤波 ⇒ 5 V 输入落到 ADC 上约 3.33 V
固件怎么读	**固件里的单位是「连接器上的毫伏」**，分压已经在驱动层还原。命令与参数里看到的 mV 就是万用表在 CN26 第 2 脚量到的那个数
两个端点	出厂默认 **800 mV（松开） / 4200 mV（踩到底）**，按常见霍尔踏板取值。两端都可改，也有命令直接从踏板本身量出来
有效带	**300 – 4800 mV。**落在带外的读数**不是踏板位置**，这一路直接判死，不产生任何油门
为什么两头都要卡	信号线断了读数掉到 0，那是「松开」，本身无害；**但信号短到自己的电源上读出来是「踩到底」**，传感器坏在高位也一样。所以两头都封，带外一律不当需求
确认要多久	连续 **100 ms** 一致才改判，一次毛刺不会把踏板判死

PWM 油门（CN2）：占空比，不是舵机脉宽

项目	内容
接口收什么	**一路方波，需求编码在占空比里。**给的是控制器输出、PLC 通道、手持盒子这类东西
⚠ **只能接 5 V 源**	CN2 前端是 2 kΩ 串入 3.3 kΩ 下拉（62.3%）。5 V 源到 MCU 是 3.11 V，读得对；**3.3 V 源只到 2.06 V，而这颗芯片的 VIH 是 2.31 V —— 高电平永远看不见，表现是「这一路是死的」。**这是板子的事实，不是可以在固件里绕过去的
别和遥控口搞混	RC1 / RC2 / RC3 的前端是 1 kΩ 串入 100 kΩ（99%），3.3 V 和 5 V 接收机都能接
允许的周期	**50 Hz – 20 kHz。**之外的算毛刺，不计
超时	**60 ms** 没有新边沿就不再相信这一路
⚠ 恰好 0% 和恰好 100%	这两个值**不产生边沿**，和「线没插」分不开，固件按「没接」处理。⇒ 静止时输出平直 0% 的源，要把端点挪进行程里，不要指望固件区分

遥控（U6 的 RC1 / RC2 / RC3）

项目	内容
RC1 / RC2	常规接收机的**舵机 PWM**，1000 – 2000 µs
RC3	**串行链路**，两种协议可选： **S.BUS** —— 100000 baud，8E2，**反相**，25 字节帧，16 个 11 位通道，值域 172 – 1811 对应别处说的 1000 – 2000 µs **iBUS** —— 115200 baud，8N1，不反相，32 字节帧，14 个 16 位通道，单位就是微秒，帧尾 16 位校验和
RC3 为什么不用外接反相器	它落在 MCU 的 UART 接收端，芯片内置硬件电平反相，S.BUS 直接收，iBUS 就是同一路把反相关掉
只收不发	那一路 UART 的发送端被电机 PWM 占用了。需要回传的协议（S.BUS2 遥测、F.Port 双向部分）只能用到接收方向
电平	三路都是 1 kΩ + 100 kΩ 下拉 + 100 pF，**3.3 V 与 5 V 接收机都能直接接**
超时	**60 ms**（约三个帧周期）没动静就不再相信这一路

项目	内容
三态	「从来没接过」和「刚刚断了」不是一回事：前者是接线问题，后者是链路问题，`rc` 命令把两者分开报，还报边沿数、帧数和被拒的帧数
⚠ 坏帧不会被悄悄用掉	两种协议都是整帧校验（S.BUS 验帧尾，iBUS 验校验和），过不了的帧重新对齐，并且计数

模式 / 方向 / 急停（CN4 / CN6 / CN3）：干接点

项目	内容
接口收什么	**无源触点**。板上每一路都是 100 Ω 串阻 + **10 kΩ 上拉到 +3.3 V** + 10 nF 对地
有效电平	**低有效**（触点闭合到 GND = 有效）。⚠ 不要从外部灌电压
挡位怎么解出来	CN4「模式」断开 ⇒ **N**；CN4 合上而 CN6「方向」断开 ⇒ **D**；两个都合上 ⇒ **R**。**P 从开关上到不了**（没有驻车制动可命令，P 和 N 做的是同一件事）
不接线是什么状态	三路都读高 ⇒ **N**，也就是不出力。这是刻意选的映射：分不清的那个状态是安全的那个
去抖	**20 ms**。一台在 20 kHz 开关几十安的机器上，「一次采样就换挡」等于「电机一起转就换挡」
极性可改	出厂按低有效。客户往端子上接什么是客户的决定，两种极性的「没接线」状态都安全
⚠ 检测不到「没插」	没接线的输入和接了线但打在 N 的输入，电气上是同一件事。要问这个问题用 `io` 命令：它主动驱动这条线看它动不动

急停（CN3）单独说

项目	内容
接什么	**常闭**到 GND 的急停回路：合上 = 运行，**断开或者线断了 = 停**
为什么是常闭	断线和按下故意是同一个结果。故障码 `estop` 在这两种情况下报的是同一条
⚠ 出厂默认没有布防	`estop_armed` 出厂是关的，而且是存盘的。原因很直接：**没接 CN3 就是开路，开路正是断线的样子** —— 要是它自己布防，一块什么都没插的开发板会永远停在急停里
⇒ 装车时	接好 CN3 之后**必须显式布防并存盘**，否则这条回路是摆设。请把它列进整车出厂检查项
它不是控制源	它不产生指令、不参加优先级、也不能被任何东西盖掉 —— 它抬一条故障，那是另一套机制

触摸屏（CN36）：VGUS 变量帧

项目	内容
串口	CN36：**1 = GND / 2 = 板侧 RX / 3 = 板侧 TX / 4 = +5 V**。⚠ 信号线**只有 3.3 V 电平，不耐 5 V**
波特率	**固件自己搜出来**：依次试 9600、115200、19200、38400、57600、128000，哪一个能回出结构正确的应答就用哪一个。屏换了配置也能自己找回来
帧格式	`A5 5A <LEN> <CMD> <ADDR_H> <ADDR_L> <数据…>`。`LEN` 从 `CMD` 起算（不是整帧，也不是净荷）；地址与数据**大端**；**没有任何校验和**
⚠ 帧头是 `A5 5A`，不是 `5A A5`	网上讲 DGUS 的教程用的是另一个顺序。照 DGUS 手册写的驱动发出的帧，这块屏**静默丢弃**
命令字	`0x82` 写变量，`0x83` 读变量

项目	内容
屏 → 板	模式 `0x05A1`、速度档 `0x05C1`、挡位 `0x05E1`
板 → 屏	车速 `0x02E5`、连接指示 `0x05E6`（也是心跳）、电量 `0x05EB`、母线电压 `0x05EC`（0.1 V/字）、母线电流 `0x05F0`（0.1 A/字，含符号）、使能指示 `0x0608`（GBK **文本**控件）、录制灯 `0x0618`、保存灯 `0x0619`
节奏	链路建立后每 **100 ms** 探一次；**500 ms**（五次）没有回应判丢
丢屏会怎样	抬 `hmi-lost`（仅报告），屏这个控制源失效 ⇒ 挡位被收回 ⇒ **车辆滑行停下**，并且要求踏板先松开才会再接受挡位。**不切桥** —— 在没有固件可控行车制动的车上，切桥和滑行是同一个行为，而保住电流环的那个还能被叫停

整车 CAN（CN5 / CN40）与控制台（USB1）

口	收发什么	详见
CN5 / CN40	经典 CAN ≤ 1 Mbps，两个口可级联，SW4 投切 120 Ω 终端。本板收 `0x210`，发 `0x211`（20 ms）与 `0x212`（100 ms）。**没被搭话之前一个字都不发**	§10.4
USB1	ASCII 文本：一行一条命令；回复是文本行，机器要读的走 `=tlm` / `=param` / `=calib` 等前缀行	§2、§10.1

1.3.3 出厂固件不使用的接口

位置	引脚	状态
U6 第 4 列	PB11 / PB7 / PB6	扩展 GPIO，固件不配置也不读
U6 第 5 列	PE6 / PD1 / PD0	扩展 GPIO，固件不配置也不读

> 这六个脚是留给二次开发的（规格书 §19.4）。列在这里是为了**不让人对着一个接了线却没有反应的脚找原因。**

1.3.4 两条容易踩的

一、CN3 现在是急停输入，不再是刹车开关。 规格书 §11 / §16 把 CN6 / CN4 / CN3 写成「方向 / 模式 / **刹车**」，出厂固件里 **CN3 是急停。** 换掉的理由是那一路数字刹车开关本来就没人会用 —— 一个没有线性度的刹车就是一个按钮，而没有人拿按钮当刹车；而这只脚的形状（10 kΩ 上拉、低有效、**在连接器上**）正好是常闭急停要的。

> ⇒ **接线以本节为准。** 另外注意 §1.3.2 里那条：`estop_armed` 出厂是关的，接好 CN3 之后不显式布防并存盘，这条回路不起作用，而且不起作用的时候**不会有任何报错。**

二、CN2 的两路 PWM 油门只能接 5 V 源，遥控口才是 3.3 V 也能接的那个。 两者前端分压差了一个数量级（CN2 是 62.3%，U6 是 99%）。把 3.3 V 的信号接到 CN2 上，现象是「这一路完全没反应」，而线、参数、协议全都是对的。

---

2. 接上控制台

2.1 接线与串口参数

物理连接	板上 USB Type-C，内置 USB-UART 桥接，插上电脑就是一个 COM 口
串口参数	**115200 8N1，无流控**
⚠ DTR / RTS	**必须拉低。**这两根线在桥接芯片一侧可能被接到复位或下载引脚；一个只是想读日志的终端软件，不应该重启一个正带电的功率级

上电后板子**每秒自己打一行心跳**，不需要先敲任何命令。健康的样子：

```
[123] vbus=38010 mV loop=20000 Hz isr=1207/1213 cyc drv=000/000 i1=7/6/35 mA
```

字段	应该是什么
`vbus=`	母线电压，单位 mV（`38010` = 38.0 V）
`loop=20000 Hz`	电流环在跑。这个数不对就不要往下做
`drv=000/000`	两个桥都是放倒的。上电就该是这样

心跳会把正在敲的字冲掉，这是有意的：一个不说话的固件更难查。

2.2 每条命令都标了它会不会让机器动

本书所有命令都带这个标记，按回车之前先看它：

标记	含义
🟢	**机器不会动。** 查询、设置、存盘，功率级不导通
🟡	**机器不会自己动，但功率级带电了。** 手推轮子可能顶回来
🔴	**机器会动。** 动之前确认轮子悬空，或者车辆周围无人

同样的分级也在机器可读的命令表 `COMMANDS.json` 里（字段 `danger`：`none` / `gates` / `motion`），上位机据此给按钮上锁，见 §10.3。

2.3 任何时候都可以打断的一条

```
stop                    🟢  它让机器停，所以它自己是安全的
```

`stop` 不取参数、不检查任何前提，直接放倒两个桥、清零所有需求、两路回 idle，并锁存一个 `user-stop` 故障。**锁存是故意的**：它要求你显式 `clear` 才能继续，这样「我按过急停」不会被下一条命令悄悄抹掉。

```
clear                   🟢  重新武装：驱动器、电流零点，然后清故障
```

2.4 最先该敲的三条

```
ver                     🟢  固件版本、镜像长度、CRC、MCU 唯一 ID
stat                    🟢  每个通道现在在做什么
setup                   🟢  这块板还缺哪几件事
```

`ver` 应该说 `crc ok`。如果它说 `NOT STAMPED`，那是一个**开发构建**，不是坏镜像—— 但它不应该出现在产线或客户手里。

`id`（0.4.13 起）用一行说清这块板是什么，给上位机自动识别用：

```
=id 1 board model=FOC_DOUBLE_C hw_rev=C board=foc_double_c uid=<24hex> fw_ver=0.4.17
```

3. 让这块板认识这台车

这块板出厂时不知道它将来带的是哪台电机、装在哪台车上。**这些信息分两个流程录进去**， 两个流程互相独立，可以分开做：

流程	内容	怎么进去
一、测量	接线相序、绕组电阻电感、霍尔换相表、机械增益	板子给电机通电自己测（`learn`）
二、录入	极对数、铭牌功率、轮周长、转速上限	人照着铭牌和实测填（`param set` / 专用命令）

第二个流程不是第一个的前提：极对数测不出来，所以它不挡在标定流程里。 `calib status` 会同时报「测量做到第几步」和「还缺哪些要人填的值」。

3.1 流程一：测量（`learn`）

四个必需步，一次做完，之后存在这块板自己的 Flash 里。**换电机、换供应商，重跑一遍即可。**

步	测什么	说明
`wiring`	接线是否接反、B/C 是否对调	电压矢量扫一整圈做匹配打分，不含任何电机参数。判据不达标就拒绝执行，**不落默认值**
`rl`	相电阻 R、绕组电感 L、逆变器有效死区	闭环两点法做两次（死区补偿开 / 关），斜率给 R、截距移动给死区
`hall`	本机霍尔安装角，即换相表	电流对齐扫描（288 步/圈 × 双向），量跳变而不是量中心；滞后角过大自动加大电流重扫
`inertia`	机械增益 K	速度环的增益是从它推出来的，不是拧出来的

三个可选步：`pp`（数极对数，见 §3.2）、`window`（分流放大器建立时间）、`flux`（磁链 λ / Ke， **诊断用，不落盘**，断电后需重跑）。

一条命令走完（0.4.15 起）：

```
en 2                  # 功率级带电
learn 2 auto          # 四个必需步一路跑完，桥全程不落
save                  # 写进 flash
```

轮子必须**悬空且周围无障碍**：`inertia` 那一步会把它转起来再让它惯性滑行。

> ⚠ **整串期间控制台不能沉默超过 10 秒。** 会让机器动的标定步骤有一条 dead-man： 控制台安静 10 秒就中止并落桥（理由是「USB 被拔的时候上位机已经不在了」）。 一条 `learn auto` 买到的是四十秒到一分半的无人值守时间，所以上位机**必须在整串期间 保持 `calib status` 轮询**（10 秒内至少一次）。这是接口约定，不是实现细节。

也可以一步一步来（`learn 2 wiring` / `rl` / `hall` / `inertia`）。每一条被拒绝时都会说原因， 最常见的三条：

它说	你要做的
`bridge is off -- 'en <m>' first`	`en <m>`
`motor is not idle -- 'mode <m> idle' first`	`mode <m> idle`
`safety tripped`	`clear`

标定结束后**每一项都会立刻回读校验**（磁链除外），并打印整条派生链供人工核对。 **未标定的通道会拒绝进入转矩以上的模式，并指名缺哪一步**——不落默认值， 因为一个看起来合理的错误答案代价更高。

3.2 流程二：录入（人填的四项）

板子自己会报这张表（`calib list`），每一项带一句「不填会怎样」：

项	每台电机各一份?	不填会怎样
`pole_pairs` 极对数	是	**进不了速度模式**，踏板是死的
`watts_cap_w` 铭牌功率	是	堵转门限和绕组热额定按默认值算，保护点不对
`wheel_mm` 轮周长	否（整车一份）	车速与里程全错，而**读起来完全合理**
`speed_limit_rpm` 转速上限	否	**进不了速度模式**

命令：

```
pp <m> <n>                          🟢 极对数，知道就直接写
drive watts <m> <W>                 🟢 这台电机的铭牌功率
drive wheel <mm>                    🟢 轮周长
param set speed_limit_rpm <rpm>     🟢 转速上限
```

两条容易出错的：

- **轮周长要滚一圈量地面**，不要拿轮胎侧壁上的直径去算。
- **极对数填错没有任何症状**——它只缩放转速和里程读数，电机照转。 不知道就跑 `learn <m> pp`（🟢 桥必须关着，你手转轮子，板子只数）， 并且**用不同圈数跑两次看商一不一致。**

> `learn <m> pp` 是唯一一条要求桥关着的，理由不是小心，是测量本身：手转的电机会发电， 而正在开关的桥给那个电流提供了回路，等于给被测的轮子加制动。桥关着，绕组是开路。

3.3 存盘

```
save                                🟢 写进 flash（要求两个桥都关断），写完回读校验
load                                🟢 从 flash 读回
forget                              🟢 恢复出厂值
```

`save` 之前先 `param get` 一遍要留的量：存进去的是**板子现在真正持有的值**， 不是你以为你设过的值。

---

4. 运行模式

三级串级结构，每一级都可以单独作为对外接口：

```
轨迹发生器 → 位置环 → 速度环 → 电流环(FOC) → SVPWM → 三相桥
                ↑        ↑         ↑
            位置指令  转速指令  转矩指令      任一级都可直接下命令
```

模式	命令	说明
转矩（电流）	`mode <m> torque` + `iq <m> <mA>`	d-q 解耦 PI，20 kHz 电流环
速度	`mode <m> speed` + `sref <m> <elec rad/s>`	四象限，可回馈制动
位置	`mode <m> position` + `pref <m> <deg>` + `plim <m> <rad/s>`	位置环 + 加加速度限幅轨迹
开环电压 / 转速	`mode <m> ol` + `volt` / `spd`	调试与自学习用

模式	命令	说明
空闲	`mode <m> idle`	松开（滑行，不是刹停）

双路完全独立：两台电机各有独立的参数、限幅、故障与铭牌，互不干扰。

三种模式的前置条件层层叠加（torque ⊂ speed ⊂ position），拒绝信息会告诉你缺的是哪一层：

模式	额外要求	拒绝时它说
`torque`	绕组与增益	`not commissioned -- 'learn <m> wiring', then 'rl', then 'hall'`
`speed`	+ 机械增益 K	`no mechanical plant -- run 'learn <m> inertia' first`
`position`	+ 极对数	`pole pairs not counted -- 'learn <m> pp'`

4.1 三件第一次用会意外的事

一、 `sref` 的单位是电角速度 rad/s，不是 rpm。 机械 rpm = `sref × 60 / (2π × 极对数)`。 控制台会替你换算并印出来，按回车后核对它印的那个 rpm，那是最便宜的一次防呆。

二、 `iq` / `sref` / `pref` 转的方向可能和挂 D 挡相反。 双轮车必有一侧反装—— 两台相同的电机装在车的两边，要让车直行它们必须朝相反方向转。整车通路（油门 / 挡位） 知道这件事并替你处理，而**控制台的直接命令绕过它**。控制台会当场提示这一条。

三、转矩命令里没有任何东西限制转速，这是它的本意。 对着一个空载轮子给电流， 它会一直加速到母线没有电压可用为止；这时得到的电流比要的少，那是电压包络保护在报告 「没电压了」，不是调节器坏了。**第一次给电流请从小开始。**

4.2 位置模式的静止分辨率

仅有霍尔反馈时，转子停住就不产生边沿，位置无法内插，估计值被量化到扇区中心。 **静止分辨率是 ±30° 电角**（15 对极即 ±2.0° 机械）。位置环据此设有分辨率死区， 以免在扇区边界上做粘滑极限环。

这是规格不是缺陷，任何增益都买不过去。 需要更高静止精度，请选用带编码器的机型。

4.3 控制台直接命令的安全上限

控制台的上限比整车通路低，而且是故意的：坐在板子旁边的手，和坐在车上的脚，风险不是一回事。

	控制台闸	整车通路
电流 `iq`	**3 A**	`drive imax` 可到 20 A
电压 `volt`	**6 V** / 轴	—
开环转速 `spd`	600 rad/s	—

5. 整车通路

§4 是「直接命令电机」，本节是「让各个控制源的需求到达电机」，两条不同的路。

```
drive          ●  每个源、合并结果、以及真正到达电机的东西
drive on       ●  让合并后的需求可以到达电机
drive off      ●  切断（源仍在被读，但到不了电机）
```

`drive on` 只是打开闸门，不决定用哪个环——决定用哪个环的是 `mode <m> speed`。

5.1 控制源与优先级

板上同时接受多个指令源，**优先级固定**：

- 油门形式：模拟电压、PWM ×2、SBUS、CAN、串口
- 开关量：方向（**CN6**）与模式（**CN4**），低有效干接点，两条合起来解出挡位；⚠ **CN3 是急停输入，不是刹车开关**（见 §1.3.4 第一条）
- 档位选择器即钥匙：**空挡 = 不出力**
- **任何「拒收」都会出声**：被优先级更高的源盖掉、或指令不合当前模式，都会明确报出， 不做静默忽略

> 台架上留下的控制台断言会跟着走：`thr 0` / `gear p` 不是「归零」，而是**控制台在断言**， 而控制台优先级最高。台架留下它们等于车上踩油门没反应。 把车交给踏板之前跑一次 `road` （§5.4）。

5.2 限速：两个上限取小的赢

参数	决定什么
`drive limit <rpm>`	转速上限（0 = 关）
`drive band <kph>`	满档（100%）等于多少 km/h
档位（`band <percent>`）	在上面两个上限上再乘一个百分比

`drive` 会把真正生效的那一个印出来，**不要自己算**：

```
band : 35% selected.  IN FORCE: 87 rpm = 4.1 km/h, set by 'drive limit'
       (the two: 'drive limit' 250 rpm x 35% = 87 rpm; band 25 km/h x 35% = 182 rpm -- the lower wins)
```

档位只缩放转速上限，永远不削扭矩。 最低档和最高档一样有劲，只是顶速低。

档位带是被保持的，它活得比给它的源长。 控制板说过 60% 然后被拔掉，油门交还给踏板， 但天花板停在 60%，要重新上电才回满档。这是故意的：弹回满档等于仪表坏掉的那一刻车自己加速， 而司机的脚还在原处。 `drive` 会说出这个档位是谁给的、多久以前给的。

5.3 刹车驱动

板载有刷 H 桥用来驱动刹车电缸或执行器。**出厂默认是关的：**

```
brake on                🟡  使能刹车驱动
save                    🟢
```

装有刹车执行器的整车，**必须在调试时执行 `brake on` 并 `save`**， 否则每次上电后本板都不驱动它，且 CAN 上 `0x211[0]` bit7 报 0。

这个默认值是刻意的：一块 BRK 上什么都没接、却照样驱动 H 桥的板子，等于在声称自己正在刹车。 但正因为它是「安静地不动作」，**调试时漏掉这一步不会有任何报错**——请把它列进整车出厂检查项。

5.4 把车交给踏板之前：`road`

```
road                    🟢
```

下面每一条，单独一条，都足以造出一台「看起来武装好了、而踩着踏板的人怎么也开不动」的机器： 控制台还占着油门（哪怕值是 0）、控制台从来没释放过挡位、手设的速度带还钉在那里、 某次测试留下的 rpm 上限、为了让空载轮子安静而调低的电流上限、上一次 `stop` 留下的锁存。

`road` 一次性全部撤销，并打开自动武装。

> ⚠ `road` **是「要把车交出去」时跑的，不是「台架收工」时跑的。** 它会打开自动武装，并清掉 `drive limit` ——而 `drive limit` 正是档位的缩放基准，台架的限速会跟着一起没。
>
> **台架收工用这三条：** `stop`（或 `dis 1 + dis 2`） 🟢 `drive off` 🟢 `drive auto off` 🟢 别把一台会自己武装的机器留在桌上

5.5 ⚠⚠ 纯 CAN 控制的车：两件会让「一切正常但车不动」的事

这一节写的是两条真实发生过的事。一个客户为它们各丢了一整天，而那一整天里 **两块板的健康位全是绿的**：CAN 通、使能合、挡位显示 D、标定做完、母线电压两边一致。 ⇒ 下面这两条，看不出来是它们的时候最像「板子坏了」。

一、挡位要在油门抬起的时候请求

本板有一条互锁：**踩着油门不许从空挡挂进 D 或 R。** 理由是防一台车在油门已经踩下去的状态下被挂上挡直接窜出去。

⚠ 所以控制器发 `0x210` 时，**挂挡那一帧的油门字节必须是 0。** 挡位进去之后再给油门，正常。

现象长这样 —— 油门进得去，挡位进不去：

```
canreq: thr=890  gear=1(F)  flags=121(allow_drive=1)   ← 你发的
veh:    throttle=890  gear=2(N)  armed=0  demand=0     ← 板子合并出来的
```

同一个源、同一帧，油门赢了合并、挡位没赢 —— 这一对数字就是换挡互锁的签名。 0.4.18 起板子会把理由直接放进遥测： `veh` 记录里 `gearblk=1` （ `drive` 命令则印一句 `HELD: release the pedal` ）。

二、起桥：0.4.18 起不需要任何设置

「武装」是让电流真正到达电机的那道门。

0.4.18 起：控制板在总线上说「允许驱动」时，驱动板会自己起桥， **不需要另外设置任何参数。**条件是三条同时成立 —— 挡位归 CAN 所有、链路是活的、 `ALLOW_DRIVE` 置位。这三条每个周期重新推导，控制板一不说它就不成立。

> ⚠ 它只对**总线握着挡位**的车生效。控制台或触摸屏压过挡位时不算 —— 一块屏握着挡位，就仍按屏那套规矩来。

为什么这条在 0.4.18 之前是个死结（还在用旧固件的话要知道）：起桥那扇门 出厂默认关着，而整车 CAN 又要求这种车关掉极简模式（§6）—— **两条各自都对的指示，合起来让那台车没有任何途径起桥**，而板子不会说为什么。 ⇒ 遇到这个现象，先看固件版本：**升到 0.4.18 或更高。**

⇒ 0.4.18 起， `armed=0` 一定带着理由：遥测 `veh` 记录的 `armwhy=` 是一个码， `stat` 会把它翻成人话。**不要把 `armed=0` 读成「还没轮到它」，它一定有原因。**

---

6. 极简模式与踏板使能手势（0.4.15 起）

面向没有电脑、没有触摸屏、也没有挡位选择器的整车。 这种车上没有任何东西可以告诉板子 「现在可以起桥了」，于是用一次不可能被意外凑出来的踏板动作来代替。

6.1 ⚠ 出厂默认是开的

`simple_mode` 的出厂默认值是 **1（开）**。也就是说：**一块没有被专门设置过的板子， 只要有人踩着踏板上电再松开，它就会自己使能。**

挡在前面的判据一条都没松（见 6.3），台架上踏板松着或者踏板根本没接，都做不出这个手势。 但这是一次行为默认的反转，**装机流程要按「默认开」来写。**

不需要它的整车（有屏、有挡位选择器、或者由 CAN 控制的）请显式关掉：

```
param set simple_mode 0
save
```

关掉这件事会被记成「有人明确关过」，将来即使默认值再变，也不会把这块板悄悄重新打开。

6.2 手势本身

上电之前把踏板踩到底，上电，保持，然后松开。

阶段	判据	参数	默认
稳定	上电后的前 50 ms 什么都不判（模拟前端在醒）	定值，不可改	50 ms
到底	行程 ≥ `arm_full_pct`	`arm_full_pct`	90 %
保持	到底状态持续 `arm_hold_100ms`	`arm_hold_100ms`	500 ms
松开	行程 ≤ `arm_rest_pct`	`arm_rest_pct`	5 %
静止	松开后静止 `arm_settle_100ms`	`arm_settle_100ms`	500 ms
整个手势	必须在上电后 `arm_window_100ms` 内完成	`arm_window_100ms`	15 s

手势成立之后板子做的**只有一件事**：请求一个 D 挡。它不发油门（踏板那个源本来就在发），也不越过任何一道原有的闸——母线电压窗口、通道是否可武装、是否已标定、电流零点标定，全部照旧。`stop`**、急停、跳闸、dead-man 四条停机路径一条都没动。**

6.3 三重防线，以及为什么需要三重

一根断了的信号线、或者一次对电源的短路，读数会撞到轨到轨——那正好长得像「踩到底」。

	做法
有效带	踏板读数必须落在 0.3 – 4.8 V 带内（霍尔踏板规格 0.8 – 4.2 V）。**任何一次出带就当故障，整个手势当场作废**，不是「不计入」
人做得到的过渡时长	从「满」到「松」中间如果没有行程（小于 30 ms），判为线掉了不是脚。脚离开踏板要扫过中段一百来个采样；接插件断开在两个采样之间就完成了
一次转换不算数	每个判据都是「电平 + 持续时长」，另有 50 ms 的开机宽限

油门源不是踏板时，手势整个不跑，并且出声说明。 板载电位器的滑动端本来就该扫满整条轨，它能产生的每一个读数都是合法读数，包括它坏掉时产生的那些——建在有效带上的整套防线在电位器上 **不存在**，而形状看上去和踏板一模一样。

6.4 踏板零点

踏板松开时的读数通常不是零（实测有 3% 左右）。手势的「静止」阶段测到的读数**就是这台车的零点**，使能之后按它重新映射，低于零点一律当 0。否则桥一起来机器就在出力，而没有人碰过踏板。

零点**只用不写**：它不会被当成一次标定写进 flash。一个信号线半掉的踏板做出的手势，如果被写进标定，会静默地毁掉这台机器的整个行程标定。行程测量结果只报告，要不要采纳由人用 `thr cal` 决定。

6.5 出门之前先在台架上读一遍

户外那个人只有一声蜂鸣，它只说「失败了」，不说为什么。所以整句解释必须在出门之前读到—— `simple_mode` 开着时 `drive` 会多打三行：

```
simple  : ON -- pedal to the floor BEFORE power, then let go
          gesture dead -- the pedal was not at the floor when the board came up
          floor>=90% rest<=5% hold=500 ms settle=500 ms window=15000 ms
```

6.6 一条要知道的代价

一台**既有**挡位开关**又**开了极简模式的车上，手势完成之后，把开关拨到 N 是不起作用的 （触摸屏和 CAN 仍然起作用）。原因是这块板上「开关没接线」和「接了但在 N」电气上分不开， 手势要是排在开关之下，就会在它唯一有意义的那种车上永远输给一个恒定的 N。

两样都接的整车，请关掉 `simple_mode`。

---

7. 蜂鸣器：七个音型

板载有源蜂鸣器是这台机器在没有屏幕时唯一的输出。**装机时可以先用 `beep <名字>` 把七个都听一遍**， 不要等到真响那天第一次听见。

音型	听起来	什么时候响	听到了该做什么
`ok`	一短（60 ms）	`save` 写进 flash 并回读成功	不用做什么，存住了
`lock`	两短	这块板不再相信总线	你要的就不用管
`unlock`	一中（220 ms）	又开始相信总线了	同上
`trip`	**一长（600 ms）**	**功率级被切**（任何锁存型故障）	**停下。** `fault` 说为什么；关电或 `clear`
`armed`	短 短 长	手势成立，**所有已标定通道都起来了**	可以走
`partial`	短 短 长 短	同上，但**有通道被落下**（通常是没标定）	可以走，但牵引力少一半
`refused`	三短	手势做成了，但机器**没**起来	看 `drive`；关电重来

`trip` 是户外那个人最需要认出来的一声，所以它是唯一的「单独一个长音」。 `armed` 和 `partial` 故意共享前三声：它们是同一个事件， `partial` 只是多一条尾巴—— 听不出区别的人拿到的结论仍然是对的。

---

8. 保护与故障

8.1 三档动作

统一故障总线，**23 个故障码**，每条分三档动作：

动作	含义
`kill`	**切桥**，并且**锁存**——必须显式 `clear` 才能恢复
`derate`	需求被压低，电机继续转
`report`	记录并显示，不改变任何行为

```
fault          ● 出了什么事，以及该怎么办
fault codes    ● 完整的编号故障表（板上那一份是权威）
```

板上的 `fault codes` 是权威。 下表是 0.4.17 的一份抄本，供不在板子跟前时查阅； 两者不一致时以板子为准。

码	名字	动作	含义
1	`post`	切桥	上电自检未通过，任何门都不许开

码	名字	动作	含义
2	`drv`	切桥	栅极驱动器报故障：过流、过温、栅驱异常或供电欠压
3	`overcurrent`	切桥	某一相电流越过软件跳闸点
4	`bad-measure`	切桥	三相电流之和不为零——采样链有一路是错的
5	`bad-vbus`	切桥	母线读数超出这块板可能的范围
6	`envelope`	降额	没电压了：调节器长时间顶到上限，需求正被压低
7	`envelope-lost`	切桥	没电压且需求已到底，反电动势本身超过桥能给出的电压
8	`hardfault`	切桥	处理器硬故障，已切栅并复位
9	`nmi`	切桥	不可屏蔽中断（本芯片上指 Flash ECC 不可纠错）
10	`watchdog`	仅报告	上一次运行停止喂狗，被独立看门狗复位
11	`user-stop`	切桥	有人故意放倒了桥（`stop`）
12	`deadman`	切桥	桥带电、机器不转、无人操作达 5 分钟（见 8.2）
13	`overvoltage`	切桥	母线超过平台额定，泄放没跟上
14	`undervoltage`	切桥	桥在开关时母线跌到栅驱需要的电压以下
15	`chopper`	切桥	泄放电路与母线电压互相矛盾
16	`stall`	切桥	有电流、转子不动、电流矢量也不动——一相和一颗管子在扛全部电流
17	`no-sync`	仅报告	开环扫场而转子没跟上
18	`hall`	切桥	霍尔码非法（三线全高或全低不是一个位置）
19	`hmi-lost`	仅报告	触摸屏心跳中断，它刚才在下的命令现在没人在下
20	`estop`	切桥	急停输入有效：有人按了，或者它的线断了（这两件事故意是同一件）
21	`winding-hot`	降额	绕组热估计到达连续额定，需求被压住维持在那里
22	`regen-full`	降额	母线没有地方再放能量，回馈制动正在退出——下坡时表现为车不再被拖住
23	`vbus-sense`	仅报告	母线电压的快速读数不再是一次测量。**没有切桥**：保护和电流环切到轮询读数继续跑；但停下之后不再起桥，要断电重上。这是板子的事，不是电源的事（0.4.16 起）

三条对使用者最要紧的：

- `winding-hot` **是降额不是跳闸。** 长时间低速大电流爬坡时车会「变软」，那是保护在起作用。
- `regen-full` **在下坡时意味着电子制动正在消失**，请使用机械刹车。电池满或 BMS 停止受流时， 固件没有任何办法把能量放到别处。
- `estop` **的断线与按下是同一个结果**，这是设计：急停回路是常闭对地，断线即停。

8.2 三个自动停机计时器

计时器	时长	条件
独立看门狗（IWDG）	500 ms	调度器被阻塞就复位

计时器	时长	条件
整车 dead-man	**5 分钟**（4 分钟先警告）	只在**三条同时成立**时计时：没有任何需求、机器静止、且没有任何控制源见过人（按键、挡位、开关变化、踏板移动）。**它不可能在行驶中触发**
标定 dead-man	10 秒	只在会让机器动的标定步骤期间：控制台沉默 10 秒就中止并落桥

> 触摸屏回应心跳**不算**「见过人」——那只证明仪表有电，不是这里要问的问题。

8.3 固件做不到的那一件

本板不提供硬件级急停通路。 板上没有任何一条从输入到栅极、不经过 CPU 的关断通路 （栅极驱动器自身的 VDS 过流除外）。固件急停是 1 kHz 轮询的软件实现，快且可靠，但它仍是软件； **换任何自研固件，这条硬件事实不变。**

整机级的硬件急停能力须由系统设计者在板外实现。本板只提供急停**输入接口**。详见《产品规格书》Rev. 1.2 §19.2 与「已知限制」一章。

8.4 ⚠ 换完电池、或者主电断过之后，敲一次 `clear`

主电断了又回来，而板子的逻辑电源一直活着（调试时 USB 插着、或者整车上换电池）， 栅极驱动会掉电复位，而那一路的电流测量会带上噪声。

现象很不像故障：**除了噪声，每个数都好看** —— 零点在中点、开路直流是几毫安、 故障寄存器干净。唯一的表现是 `learn <m> rl` 每次都失败，而板子在 0.4.18 之前 会把原因说成「转子不自由」，那两个原因都不成立。

```
clear                          🟢  0.4.19 起，它会把电流测量链一并重启
```

- **0.4.19 及以后**：敲一次 `clear` 就行。回执里会多一句 `measurement chain restarted`； 恢复不成功时它**报失败而不是报成功。**
- **0.4.18 及更早**： `clear` 救不回来。它会回「成功」、零点和直流读数都正常， **而通道仍然是坏的** ⇒ 那些版本上只能**复位整块板。**

> ⚠ 这条对整车尤其要紧：**电池是要换的**，这不是异常工况。 如果换完电池之后 `learn` 失败或者电流读数不对，先 `clear`（或复位），再查别的。

8.5 换完电池不用有人去敲 `clear` （0.4.23 起）

上一节那条规矩在整车上有一个现实问题：**现场没有人会去连串口敲 `clear`。**

0.4.23 起，上电自检因为**供电原因**失败时，板子自己重试一次：

什么情况会自动重试	母线电压、栅极驱动、栅极驱动报故障、电流零点、电流采样初始化——这五条都是"主电没到位"的表现
什么时候重试	母线电压回到 **10–45 V 并稳住 1 秒**之后
重试几次	**每次供电回来只试一次。**一直不成功就一直是失败，不会反复空转
哪些不自动清	霍尔、斩波这一类。它们不是供电问题，自动清等于把一个真故障擦掉

控制台上会看到这几行，前缀是固定的，程序可以直接认：

```
POST retry: <结果>
re-armed: drivers reconfigured, measurement chain restarted, offsets re-zeroed, trip cleared
cannot re-arm: <原因>
```

⇒ 换电池的正常流程因此变成：**换上，等电压稳住，板子自己回来。** 串口 `clear`、CAN 清故障、这条自动重试走的是**同一个函数**，做的事完全一样。

8.6 司机清故障：`0x210` 挡位字节 bit7（0.4.22 起）

控制板在 `0x210` 的挡位字节最高位（bit7）举一个**上升沿**，本板就做串口 `clear` 的全部事。 挡位仍然在低 3 位，两者互不影响。

⚠ **判据是这一位自己的上升沿，不是电平。** `0x210[6]` 序列号每帧自增，所以一个盯着 序列号变化的边沿检测器等于电平检测器——一个一直把 bit7 举着的控制器会每 20 ms 清一次故障。

司机那一侧要做什么动作才触发，由控制板决定（连拨两次 P / 长按 N / 屏幕按钮）， 见控制板说明书 §4.6。本板只认这一位的沿。

---

9. 参数

9.1 板子自己是参数表的真源

```
param list                  ●  每一个可设的量：名字、单位、范围、默认值、当前值、能不能存盘
param get <name>            ●
param set <name> <v>        ●
```

0.4.17 的表是 **47 行。**本书**不抄一份参数表出来**：抄出来的那份第一次改动就过期， 而它看起来和真的一模一样。板上 `param list` 里的 `min` / `max` / `default` 和板子真正执行的是同一行代码，所以它不可能对不上。

9.2 三条使用规矩

一、单位写在名字里，值只是整数。 `vbus_mv=40152`，不是 `vbus=40.152`。 `vbus_mv` 不可能被读成伏特。有几个参数的单位是缩放过的（`%`、`100ms`、`kph10`、`10w`）， `param list` 会报出来。

二、命令回显成功不等于命令生效。 每次改完都读回来——`param set` 回的是板子写完之后 真正读到的值，不是你发过去的值。

三、`save` 要求两个桥都关断。 存的是板子现在真正持有的值。

9.3 两类特殊的行

- **静默参数（silent）**： `simple_mode`、`estop_armed`、`pot_debug` 一类。它们设错了 第一次使用就暴露，而**没设**则什么都不说，所以上位机应当把它们明确显示给操作员。
- **不存盘参数**：`pot_bench_ma`（台架旋钮，见 9.4）复位即 0， `param set` 成功时回 `code=not-persisted`。这是设计：一块上电就带着两个已授权旋钮的板子，正是这个功能不允许出现的东西。

9.4 板载电位器：默认关（`pot_debug`）

板上两个电位器是给调试用的，**出厂默认关闭。**关着的时候：

写什么	回什么
`param set thr_src 1`（或 2，选电位器当油门）	`code=wrong-state`
`param set pot_bench_ma 500`（台架旋钮当 iq 命令）	`code=wrong-state`

要用先 `param set pot_debug 1`。一个开关管两扇门，因为它们是同两个元件、同一个意图。

关掉的时候会**立刻**把油门从旋钮上拿回踏板，并在控制台说出来，不悄悄改。

台架旋钮 `pot_bench_ma` 的开关和量程是同一个数（`param set pot_bench_ma 800` = 开， 且旋钮拧到底等于 800 mA）：**打不开它而不说明授权多大电流。** 它绕开整车仲裁 （挡位、速度带、整车限幅都不参与），但**电机层的保护一条都没少**——功率上限、电压包络、 堵转计时、热降额、过流跳闸全在下面。

---

10. 给程序用的接口

10.1 `=tlm` 遥测

```
tlm                                  ●
```

形状：`=tlm <版本> <记录> <键>=<值> ...`，以 `=tlm end` 结束。全部整数，单位写在名字里。

当前版本号是 2。 版本规则：

- **加一个字段不动版本号。** 这个固件几乎每一版都长出新的测量值， 一个在客户现场跑着的解析器不该因为我们多测了一个东西就坏掉。
- **改一个字段的含义必须动版本号。** 只知道版本 1 的客户端必须拒收版本 2，不许猜。
- 反过来不成立：不许因为版本号陌生就拒收一个更低的版本。

记录：板级 `sys` / `veh`，CAN 四条 `can` / `canlink` / `cantx` / `canreq`， 每台电机各三行 `m1` / `m2`。

0.4.18 给 `veh` 加了五个键，用来回答「车为什么不动」（追加，不动版本号）：

键	是什么
`gearblk`	**挡位是不是被拒了。**1 = 请求的挡位被互锁挡住（见 §5.5 第一条）
`gearsel`	请求的是哪个挡，和实际生效的 `gear` 分开
`gearsrc` / `thrsrc`	挡位和油门**此刻归哪个源**（`ctrl_src_t` 的枚举值）
`armwhy`	**没有武装的原因**，一个码； `stat` 会翻成人话。 `0` = 已武装

⚠ 这五个键是给程序读的那条路补的。在它们之前，一个客户端能读到 `gear=2`， 却读不到「2 是被拒出来的」——于是**一次拒绝会被读成一次正常的挡位状态**， 而板子其实一直知道答案。 ⇒ `armwhy` 的码只增不重编号；不认识的码按「有原因但本客户端不认识」处理， 不要当成 0。

0.4.24 给 `sys` 追加了三个键：上一次是怎么结束的（同样是追加，不动版本号）：

键	是什么
`rst_wdog`	1 = 上一次运行是被**看门狗**打断的（程序卡死）
`rst_cpu`	1 = 上一次是**处理器故障**（HardFault 那一族）
`rst_trip`	1 = 上一次是**带着跳闸**结束的

⚠ **它活过复位，活不过断电。** 三个都是 0 有两种意思：正常关机，或者刚断过电。

⇒ 在它之前，这件事只写在开机横幅和 `fault` 的散文里 —— 一个没看见横幅的客户端 **分不出"刚开机"和"看门狗刚把它重启了"**，而板子一直知道答案。

两个坐标系的约定：

字段	坐标系
`rpm_veh` / `iq_veh_ma`	**整车坐标系**：正 = 向前。要判断「加速还是制动」必须用这两个
`rpm` / `iq_ma`	**电机自己的坐标系。**双轮车必有一侧反装，在那一侧它们和整车速度每一个样本都符号相反

一份遥测快照**不是一个瞬间**（写者是 20 kHz 中断和 1 kHz 槽，取快照时不加锁）， 所以不要拿其中两个字段去算需要同时性的量，比如电压乘电流。真要功率，用环里算好并上报的那个。

10.2 `=param` 与 `=calib`

配置界面用 `param list` 拿参数表，标定向导用 `calib list` / `calib status` 拿两张表 （测量 7 步 / 录入 4 项，当前帧版本 **2**）。

`calib status` 每个通道带一个 `needs=`，列出这个通道还缺哪些要人填的值。 **这一行一定要念给用户听**：一张写着「4/4 完成」、对缺失的极对数只字不提的卡片， 是在告诉客户他做完了，而机器其实还进不了速度模式。

10.3 命令表 `COMMANDS.json`

机器可读的命令清单**随发布包一起发**，不在板上。59 条命令，每条带：

字段	说明
`id`	只追加、永不复用。上位机的中文解释按它挂
`name`	线上真正要发的词
`args`	参数形状
`scope`	`board` / `motor`
`needs`	`none` / `idle` / `bridge` / `torque` / `commissioned`，界面据此置灰
`danger`	`none` 只读 / `gates` 改板子状态但动不了机器 / `motion` 能让机器动

一条命令有多种形式时， `danger` 取其中最坏的那个。

10.4 整车 CAN

经典 CAN，与上位控制板对接。**协议由控制板一侧定义**，本板按它实现。 本板为**拓扑 A（一板双电机）**。

ID	方向	周期	内容
`0x210`	控制板 → 本板	20 ms	油门‰、转向‰、allow_drive / brake 标志、挡位、速度档位、序列号、超时请求
`0x211`	本板 → 控制板	20 ms	health、两轮车架机械转速、挡位回显、指令有效位
`0x212`	本板 → 控制板	100 ms	母线电压 mV、母线电流（10 mA/LSB）、车速（0.1 km/h）

链路健康判据不是「有没有帧进来」，而是序列号有没有动：应用挂死但邮箱里压着一帧的对端， 不会被误判为健康。链路超时下限取 3 个帧周期、上限 500 ms。

没被搭话之前一个字都不发。 总线上没有控制板时，本板不会主动往上灌帧。

三条需要控制板一侧知道的约定：

- `0x211[0]` **bit7 = 刹车驱动已使能。** 它是板上存着的一个决定，**不是一次测量**： 执行器脱落、线断、卡死，这一位仍然是 1。它只说明「板子收到的 brake 位会被送到 H 桥」， **不构成「本车可以停下」的证据。**反过来 0 有两个意思（没装执行器 / 装了但没使能）， 总线上分不开，所以不要把 0 直接显示成「刹车故障」。
- `0x211[7]` **bit6 = 本地锁生效中**（0.4.11 起）： 这块板暂时不接受总线指令。
- `0x210` **挡位字节的 bit7 = 清故障**（0.4.22 起，**上升沿**触发，见 §8.6）： 挡位本身仍在低 3 位。控制板举一帧就够，不要一直举着。
- **转向字段本板解析并回报，但不执行。** 两轮差速转向是一条需要标定的混合律， 未经实测不会接进电机。
- **车速由本板换算后发出**（`0x212`），不建议消费端自行用转速乘轮周长——轮周长存在本板上， 让每个消费者各算一遍，同一个物理量就会在总线上出现多份且互不相等。

10.5 VGUS 触摸屏

仪表显示与本地操作走外部 TTL 串口。屏是被「告知」状态的，不是被「询问」的： 本板按固定节奏推送，屏侧只管显示与按键。屏的心跳中断会产生 `hmi-lost`（仅报告）， 车辆滑行停下，并要求踏板先松开才会再接受挡位。

> ⚠ 外部串口 CN36 的信号线**不耐 5 V**，接 5 V TTL 设备须在板外加电平转换。见《产品规格书》的通信接口与「已知限制」两章。

11. 固件升级

11.1 单槽，没有回滚（0.4.18 起）

Bootloader 占独立分区，应用占**一个**区（464 KB）。

- **镜像自带自述头**：版本、构建、校验，出厂可追溯。
- **升级通道：串口。** CAN 升级协议开发中，本版没有。

> ⚠ **0.4.17 及更早是 A / B 双槽，本书上一版描述的就是那个。** 0.4.18 把两个槽合成一个，换来的是应用区从 232 KB 变成 464 KB。 代价说清楚：**没有回滚了** —— 一次中途断掉的升级，留下的是一块没有可用应用的板子。

那块板不是砖。**加载器和 ROM 引导两条通路在没有应用的时候都能工作**， 所以恢复的办法是把镜像重写一遍，而不是返修。上位机走的正是加载器那条通路。

11.2 发布包内容

每个发布版本是一个目录，包含：

文件	说明
`FOC_Double_C-app.bin` / `.hex`	应用镜像。⚠ 0.4.17 及更早这里是 `-slotA` / `-slotB` 两个文件
`bootloader.bin` / `.hex`	Bootloader
`MANIFEST.txt`	版本、提交号、各镜像长度与 CRC32、工具链、构建时间、主机测试结果
`SHA256SUMS`	上面每个文件的校验和
`COMMANDS.json`	机器可读命令表（见 §10.3）
`BOARD_TEST.txt`	板上验收记录
`source.zip`	这一版的完整源码

0.4.20 的 `MANIFEST.txt` 摘要：

```
name:        FOC_Double_C
version:     0.4.20
loader:      bootloader.bin  5364 bytes, crc32 d0d25cfe, load 0x08000000
application: FOC_Double_C-app.bin  233540 bytes, crc32 ..., load 0x08008000
host tests:  244246 checks, 0 failure(s)
```

11.3 ⚠⚠ 升级会清空标定 —— 升级之前先备份

本书上一版这里写着「升级不会擦掉标定」。那句话现在是错的，而且是这一章里 最要紧的一处更正。

上位机只有一条写入通路：进 ROM 引导 → **整片擦除** → 写入 → 逐块回读校验。 ⇒ **对你来说「更新固件」和「整片重刷」是同一件事**，板上存的标定记录会一起没。

被擦掉的东西分两类：

	能不能敲回去
R、L、电流环增益、极对数、轮周长、额定功率、`auto_arm` 等设置	✅ 能。 `param` / `mot` 收，照着旧记录敲
霍尔扇区表（两路各六个边界）	❌ **敲不回去。** 写它的只有 `learn <m> hall` 扫描，而那要**把驱动轮架空**

⇒ 在一台已经装好的车上，重跑霍尔扫描不是一分钟的事。**所以升级之前先备份。**

升级前（0.4.20 起）

```
report                          整条换算链，存一份
hall dump 1                     第 1 路的霍尔表
hall dump 2                     第 2 路的霍尔表
```

`hall dump` 会印出一行**可以原样粘回去的命令**，形如：

```
put it back with:  hall set 1 15462343a96e139880c385ee3119ce f99b599c
```

把两路的这两行连同 `report` 的输出存成文本文件。

> ⚠ **这两行只对这块板的这一路有效。** 末尾那八位是校验，它覆盖板子的 UID 和电机号 —— 把一块板的表灌到另一块，或者把第 1 路的表灌到第 2 路，会被拒绝。 这道检查不是形式：**一张属于别的电机的霍尔表，别的检查全都会放行，机器照样转**， 只是换相偏几度、同样力矩多吃电流，从外面看不出任何异常。 ⇒ 备份文件请按**板子 UID + 电机号**归档，不要只按板型号。

升级后

```
hall set 1 <粘回第 1 路那一行>
hall set 2 <粘回第 2 路那一行>
save                            ⚠ 不 save 就只在内存里
```

然后照 `report` 那份把 R、L、增益、极对数、轮周长、额定功率、 `auto_arm` 敲回去， 再 `save` 一次。

> ⚠ **`hall set` 只恢复霍尔这一档。** 它成功不等于「标定回来了」—— 接线方向、R/L、增益是各自独立的记录，缺哪个就还缺哪个，用 `setup` 核对。

⚠⚠ **敲回去的数是数，不是「这一步做过了」。** 这一整套走完之后 （2026-09-08 在台架上真的擦一遍验过），`setup` 会说三项都 SET， 而 `calib status` 仍然把 **`wiring`** 列成下一步 —— 因为接线方向那一档是 **测**出来的， `mot` 只是把它的数值写进去，没有把那一步标成做过。

⇒ 两句话分清楚：

问题	谁回答
板子现在拿什么数在算?	`setup` / `report` —— 敲回去之后它们是对的
哪一步还没做过?	`calib status` —— 它记的是**做没做**，敲数不会让它变

⇒ **能开的车不必为此重跑** `learn`：数都在，电机照常出力。 但如果你要一份「标定流程走完」的记录（出厂验收、交付单）， 那就得让 `learn <m> wiring` 真的跑一次。

⚠ 还有两处敲不回原样的小地方，一并记在这里： - `dtc` **（死区补偿）按千分之一周期设，而** `report` **按纳秒显示。** 149 ns 和 144 ns 敲回去都会变成 150 ns（3‰）。差几纳秒不影响使用， 但别以为它会一模一样。 - `speed_limit_rpm` 之类的**限速项**不在上面的清单里，它们是整车配置， 按你自己的车重新设。

0.4.19 及更早的板子怎么办：那些固件上 `hall dump` 印出来的那行**超过控制台 能接收的长度，敲不回去**（0.4.20 修的就是这条）。⇒ 从 0.4.19 或更早升上来的这一次， 霍尔表只能重扫；升到 0.4.20 之后，后面每一次升级都可以备份了。

11.4 升级之后先做的几件事

`ver`	版本号和 `crc ok`
`setup`	标定与录入项是否还在

- 按 §11.3 把标定灌回去。
- 核对一遍 `param get simple_mode`：0.4.15 把它的默认值改成了「开」， 一块被擦干净的板子会带着这个默认值回来。
- 整车用 CAN 控制的，核对 `param get auto_arm`（见 §5.5）。

---

12. 版本历史

12.1 固件

版本	日期	内容
0.4.24	2026-09-10	`=tlm` 的 `sys` 行追加三个键 `rst_wdog` / `rst_cpu` / `rst_trip`：上一次运行是被看门狗打断的、还是处理器故障、还是带着跳闸结束的（§10.1）。追加，不动遥测版本号
0.4.23	2026-09-10	**换完电池不用有人去敲 `clear`**：上电自检因供电原因失败时，母线回到 10–45 V 稳 1 秒后自动重跑一次 re-arm，每次供电回来只试一次；霍尔 / 斩波那一类不自动清。控制台多三行固定前缀 `POST retry:` / `re-armed:` / `cannot re-arm:`（§8.5）
0.4.22	2026-09-10	`0x210` **挡位字节 bit7 = 清故障**（上升沿触发），收到就做串口 `clear` 的全部事，让司机能在车上清故障而不必连串口（§8.6）
0.4.21	2026-09-10	flash 地图：CMake 传进来的槽地址和 `boot_map.h` 那张表用静态断言真的拴在一起了——两者不一致以前只会在烧录时表现为"烧不进去"；开机横幅改成读镜像自述头里的板型和硬件版本，不再写死自称 `FOC_DOUBLE_B`
0.4.20	2026-09-08	修 `hall dump` 印出来的还原命令**比控制台能接收的行还长**（78 字符对 64，16 个参数对 6）⇒ 0.4.19 上那条命令敲不进去，标定备份实际上做不了。改成一个 30 位的十六进制串，五个参数约 47 字符
0.4.19	2026-09-08	**主电断过一次再回来（换电池、或调试时 USB 供着逻辑电），栅极驱动掉电复位而板子不重启电流测量链** ⇒ 那一路带约 900 mA 峰峰噪声，`learn <m> rl` 必失败，而板子把原因说成「转子不自由」。⚠ 关键是 `clear` **以前救不回来**：它报成功、零点和直流读数都正常，通道却仍然坏着 —— 只有复位能。现在 `clear` 会一并重启测量链（本书 §8.4）
0.4.18	2026-09-07	⚠⚠ **应用区改单槽**，没有回滚（§11.1）；`learn <m> rl` 的拒绝改成**报出它拒绝时用的那个数**并指向复核用的命令，不再只给一张「可能的原因」清单；`=tlm veh` 追加 `gearblk` / `gearsel` / `gearsrc` / `thrsrc` / `armwhy` 五个键（§10.1）；纯 CAN 的车多一扇窄的自动起桥门（§5.5）
0.4.17	2026-09-03	**背景 ADC 采样改 DMA**：三路背景转换器启动一次之后再不停止，母线注入组从此没有任何一条路径能把它关掉（0.4.16 修的是那条路径本身，这一版是让它不存在）；`learn <m> rl` 的点间等待从 60 ms 提到 250 ms，测量前把转子等停——那个偶发的「两次电阻拟合不一致」由此消失，代价是每次 `rl` 多约 0.8 秒；相电流零点测不到样本时改为**一句拒绝**，不再是一次看门狗复位
0.4.16	2026-09-02	⚠ **母线保护失效的修复**：背景采样每 5 ms 关一次 ADC5，把母线注入组一并关掉且再未启动，`JDR1` 自开机起冻结——行驶中母线真的塌下来时欠压保护不会动。**此前所有已发布固件都有这条**；新故障码 `[23] vbus-sense`（本书 §8.1）；`safety` 行加 `conv` / `bridges` 两个字段；栅驱关闭时心跳不再把量程底当电流印
0.4.15	2026-09-02	**极简模式与踏板使能手势**（默认开）；踏板零点重映射；`learn <m> auto` 一条命令走完测量流程；标定拆成「测量 / 录入」两张表（`=calib` 帧版本 2，极对数不再占必需步）；板载电位器总开关 `pot_debug`（默认关）；新增 `armed` / `partial` / `refused` 三个蜂鸣音型
0.4.14	2026-09-01	修复发布包校验清单漏项——0.4.12 / 0.4.13 的客户端下载不完整

版本	日期	内容
0.4.13	2026-09-01	`id` 命令：一行说清这块板是什么
0.4.12	2026-09-01	遥测中板子自报板型，上位机不必再问人
0.4.11	2026-08-29	台架旋钮 `pot_bench_ma`；本地锁（`lock`，并上报到 CAN `0x211[7]` bit6）；随包发布机器可读命令表 `COMMANDS.json`

> ⚠ **0.4.12 与 0.4.13 已从发布服务器索引中摘除**（这两版的发布包校验清单不完整，客户端下载会失败）。请使用 **0.4.14 或更高**。

功能与版本的对应关系（上位机与整车文档写前置条件时按这张表）：

功能	从哪一版起可用
标定备份 `hall dump` / `hall set`	**0.4.20**。⚠ 0.4.19 也有这两条命令，但 `dump` 印出来的那行敲不回去 ⇒ **在 0.4.20 之前，标定备份是做不了的**
`clear` 会重启电流测量链	**0.4.19**。更早的版本上，主电断过之后只能复位整块板（§8.4）
单槽、没有回滚	**0.4.18**。⚠ 从 0.4.17 及更早升上来的那一次是整片重刷
`=tlm veh` 的 `gearblk` / `gearsel` / `gearsrc` / `thrsrc` / `armwhy`	**0.4.18**。⚠ 更早的固件不是「没有被拒」，是**不会说它被拒了**
纯 CAN 的车自动起桥（挡位归总线 + 链路活 + `ALLOW_DRIVE`）	**0.4.18**。更早的固件上这种车起不了桥，也不会说为什么 ⇒ 升级（§5.5）
故障码 `[23] vbus-sense`	**0.4.16**。⚠ 更早的固件不是"没有这个故障"，是**没有这项检查**——母线快速读数停掉时它们不会说
极简模式 / 踏板手势 / `learn auto` / `calib` v2 / `pot_debug`	**0.4.15**
`id` 命令	0.4.13
`COMMANDS.json` / 本地锁 / 台架旋钮	0.4.11
CAN `0x211[0]` bit7 刹车使能位	早期固件该位恒为 0，与「未使能」同形。接进安全判据之前先确认固件版本

12.2 本书

版本	日期	说明
Rev. 1.6	2026-09-10	跟上 0.4.24：§10.1 补 `sys` 那三个复位原因键
Rev. 1.5	2026-09-10	跟上固件 0.4.21–0.4.23（书此前停在 0.4.20，而扉页那句「本书对应的是固件 0.4.17」和表头的 0.4.20 是**同一本书里的两个答案**，一并订正）。新增 §8.5 换完电池自动重跑一次自检、§8.6 `0x210` 挡位字节 bit7 清故障；§10.4 补上那一位
Rev. 1.4	2026-09-08	仍对应固件 **0.4.20**。§11.3 补三条**在台架上真擦一遍才发现**的事：敲回去的数是数不是「这一步做过了」（`setup` 会说齐了，而 `calib status` 仍把 `wiring` 列成下一步）；`dtc` 按千分之一周期设而 `report` 按纳秒显示，149/144 ns 回来都是 150 ns；限速项不在备份清单里

版本	日期	说明
Rev. 1.3	2026-09-08	对应固件 **0.4.20**。⚠⚠ **更正上一版 §11.3 那句「升级不会擦掉标定」——它现在是错的**：上位机唯一的写入通路是整片擦除，所以每次升级标定都会没，而霍尔表敲不回去。整章 §11 重写：单槽没有回滚、新的包内容、升级前后的备份与灌回步骤。新增 §5.5（纯 CAN 的车两件会让「一切正常但车不动」的事：挡位要在油门抬起时请求、`auto_arm`）、§8.4（换完电池敲一次 `clear`）；§10.1 补 `veh` 的五个新键；版本历史补 0.4.18 / 0.4.19 / 0.4.20
Rev. 1.2	2026-09-04	新增 §1.3 接口功能定义：逐个连接器写出出厂固件的用途，以及每一路收的是什么信号与什么格式（模拟油门 0 – 5 V 与有效带、CN2 的占空比油门与「只能接 5 V 源」、S.BUS / iBUS 帧、干接点与挡位解码、VGUS 帧格式与全部 VP 地址）；写明 **CN3 是急停输入而不是刹车开关**，并修正 §5.1 里沿用旧说法的那一行
Rev. 1.1	2026-09-03	对应固件 **0.4.17**。补上 0.4.16 与 0.4.17 两条版本历史（0.4.16 发布时本书没跟着走，这里一并补齐）；故障码从 22 条改为 **23 条**，新增 `[23] vbus-sense` 的表行与「从哪一版起可用」条目
Rev. 1.0	2026-09-02	首版。对应固件 0.4.15。此前固件内容以《产品规格书》Rev. 1.1 §19「出厂固件功能（预览）」形式发布，自本书起独立成册

13. 已知限制（固件侧）

硬件侧的限制见《产品规格书》「已知限制与使用须知」一章，本表只列固件。

#	项	说明
A	位置模式静止分辨率 ±30° 电角	霍尔固有，非增益可改善。见 §4.2
B	持续低速大电流爬坡会被热模型降额	车会「变软」，这是保护在起作用，不是误动作
C	CAN 转向字段解析并回报，但不执行	差速转向混合律未标定
D	CAN 升级协议开发中	当前升级走串口
E	磁链 λ / Ke 不落盘	诊断用，断电后需重跑
F	绕组热模型的时间常数是估计值	它只决定到达速度，不决定落点（落点由实测电阻和铭牌功率决定）。**不要从它推出「峰值能持续 X 秒」的数字**，那需要仪器实测，而板上没有温度传感器
G	一台既有挡位开关又开了极简模式的车，手势完成后开关拨 N 不起作用	见 §6.6。这类整车请关掉极简模式

14. 附录：命令一览

板上的 `help` 是权威，它和代码在同一个文件里。下表按用途分组，标记含义见 §2.2。

看（全部 🟢）

命令	作用
`help` / `?`	全部命令
`ver`	固件版本、镜像长度、CRC、MCU 唯一 ID
`id`	这块板是什么，一行
`stat`	每个通道在做什么
`report`	整条换算链，每一级带来源
`tlm`	同样的东西，键=值，给程序读
`fault` / `fault codes`	出了什么事 / 完整故障表
`hall`	霍尔码、扇区、转速、错误计数

命令	作用
`phase` / `io` / `chop` / `can` / `slim` / `prof`	相电流、开关量、泄放、CAN、限幅、性能剖面
`dump` / `hdump`	录波结果

设与存（全部 🟢）

命令	作用
`setup`	还缺哪几件事
`param list` / `get` / `set`	参数表
`pp <m> <n>`	极对数
`trip` / `stall` / `therm` / `slim` / `plim` / `dtc` / `env` / `smp`	保护门限与环路设置
`save` / `load` / `forget`	存 / 读 / 恢复出厂
`beep [名字]`	试听蜂鸣音型
`estop` / `lock` / `rc` / `hmi` / `boot`	急停、本地锁、遥控协议、屏、引导

标定（除 `pp` 外全部 🔴）

命令	作用
`learn <m> auto`	四个必需步一次跑完
`learn <m> wiring | rl | hall | inertia`	单步测量
`learn <m> pp`	数极对数。🟢 桥必须关着，你手转轮子
`learn <m> window | flux`	可选步
`calib list` / `calib status`	两张表与进度
`cal`	电流零点
`thr cal`	踏板行程标定

开关

命令	作用	标记
`en <m>`	这一路功率级带电	🟡
`dis <m>`	放倒这一路的桥	🟢
`stop`	两桥立刻放倒，不取参数、不检查前提	🟢
`clear` / `rearm`	重新武装	🟢

直接命令电机（全部 🔴）

命令	作用
`mode <m> idle | ol | torque | speed | position`	选环
`iq <m> <mA>`	转矩
`sref <m> <elec rad/s>`	转速（控制台会换算成 rpm 印出来）
`pref <m> <deg>`	位置（机械角）
`volt` / `spd`	开环电压 / 开环转速

命令	作用
`step` / `rec`	阶跃响应与录波

整车

命令	作用	标记
`drive`	每个源、合并结果、真正生效的限速	🟢
`drive on` / `off`	需求能否到达电机	🟡 / 🟢
`drive wheel | watts | limit | band | imax | invert | auto | regen`	整车设置	🟢
`gear` / `band` / `thr`	控制台代替挡位 / 档位 / 油门	🔴
`brake on | off | apply | release`	刹车驱动	🟡
`road`	撤销台架留下的东西，把车交给踏板	🟢