

CANO / CAN1 刷写与 FD 5M 实测

P2-1010 临时 USB CAN 板 / 2026-10-29 至 2026-10-30 / Asia/Shanghai

结果：两路均完成原程序备份、新固件烧录和逐字节读回校验；重启后版本查询正常。物理对接的 CAN0 与 CAN1 已通过普通 CAN 1M、CAN FD 1M/2M 和 1M/5M 的双向收发。本次属于原型实机验证，尚未确定满载极限或长期可靠性。

1. 固件与通道身份

原作者 [normaldotcom/canable2-fw](#)，固定提交 `b158aa7ab106bc018715cd4d5f02f544f9de1830`；使用上游源码编译，未修改业务源码。BIN 为 26,468 字节，写入地址 `0x08000000`。每路原主 Flash 均完整备份 128 KiB。

物理通道	USB 路径 / 芯片序列号	刷后接口 / 结果
CANO	5-1.3.3 / 2079367B3945	ttyACM0 / 读回、启动、收发通过
CAN1	5-1.3.4 / 2079367A3945	ttyACM1 / 读回、启动、收发通过
CAN2	5-1.3.5 / 原 <code>gs_usb</code> 固件	未刷写；当前 Linux 名称为 <code>can0</code>

接口编号可能随枚举变化。应按 USB 序列号或物理路径识别通道，不能把刷后剩余的 Linux `can0` 当作物理 CAN0。

2. 实际收发结果

逐帧功能测试共 164 帧，涵盖标准/扩展 ID：普通 CAN 1M 为 0 至 8 字节，FD 2M/5M 为 0 至 8、12、16、20、24、32、48、64 字节，均逐项核对对端实际收到的类型、ID、DLC 和全部载荷。

FD 1M/5M，64 字节	时间	实收 / 实发	实际帧率
CANO → CAN1	10 秒	16,603 / 16,603	1,660.30 帧/秒
CAN1 → CAN0	10 秒	16,615 / 16,615	1,661.50 帧/秒
CANO → CAN1	60 秒	54,929 / 54,929	915.48 帧/秒
CAN1 → CAN0	60 秒	54,929 / 54,929	915.48 帧/秒

连续测试总计 143,076 帧，缺失、重复、内容错误、乱序和串口残帧均为 0；两路每阶段 E 寄存器均为 0。加上功能测试，本次共核对 143,240 帧。

5M 指 FD 数据段位速率，仲裁段仍为 1M。64 字节单向有效载荷约 0.850 Mbit/s；双向每方向约 0.469 Mbit/s。发送脚本请求速率为单向 2,000、双向每方向 1,000 帧/秒，但受脚本节拍和 USB ASCII 链路影响，实际帧率如表；本次没有把 5M 总线跑满，不能据此给出硬件上限。

复测方法与尚待处理的问题

启动、接口切换和测试恢复记录

3. 已定位的串口抢占

首轮出现返回帧截断，保留为失败记录。实查发现 ymbot-web.service 中的 battery 进程以 root 身份自动打开 ttyACMO，读走了 CAN 数据；普通串口独占标记无法阻止 root。临时只在该服务的设备 cgroup 中禁止访问 CANO 串口，待已有句柄自然关闭后复测，164 项功能测试及全部连续测试通过。

测试结束已在 finally 中恢复该设备访问权限，未停止业务服务、未改业务程序或持久服务配置。最终确认 ymbot-web.service 为 active，测试期间无相关 USB 断开/重置日志。正式使用前仍需修正电池程序的设备选择，避免固定打开 ttyACMO；本报告的通过结果是在排除抢读条件下取得。

4. 精简操作流程

- ① 按 USB 路径和芯片序列号确认目标。每次只让准备刷写的一路进入 BOOT/DFU，先备份 0x08000000 起的 128 KiB 原程序。
- ② 将本包 BIN 写入 0x08000000，并读回 26,468 字节与 BIN 比较。取消 BOOT 短接后复位；两路应分别识别为 16d0:117e USB CDC 串口。
- ③ 使用支持 CANable 2.0 FD 扩展的串口工具，按序发送 C、S8、Y5、A1、O，每条以 CR 回车结束；V 查询应返回 b158aa7。
- ④ 使用 b/B 命令发送标准/扩展 FD+BRS 帧；例如 b123F 后接 64 字节数据的 128 个十六进制字符，再接 CR。对端实收逐帧校验，不以 USB 写成功代替总线验证。
- ⑤ 用 C 关闭通道，退出测试。串口波特率 115200 是 USB 虚拟串口设置，不是 CAN 位速率。

这是 SLCAN 固件，两路不会继续作为原生 gs_usb 的 canX 出现。Linux 普通 slcan 绑定程序不保证支持这套 FD 扩展；当前测试直接使用 USB CDC 协议。

5. 校验、来源与验证边界

新固件 SHA-256：

a24290eea6050cf6245ade91353e2f47732f9e570baf21b2a05f0f2a2c8c08f8

工具只通过 DFU 的主 Flash alt 0 备份/烧录/读回，未执行 RDP 解锁、全片擦除或 BOOT0 选项字节改写。另需明确：上游 system.c 启动代码会在 BOR 非 4 级时尝试设为 4 级（约 2.8V）；本次未采集选项字节前后快照，不能声明所有选项字节完全未变。

本次没有示波器位宽测量、长线/终端条件扫描或数小时至数天连续测试。上游 E 寄存器并非完整的控制器 TEC/REC/总线错误统计，E=0 不能代替这些测量。备份二进制留在本地与设备私有状态目录；验证包含烧录日志、通过和失败证据、脚本及校验和。

源码与协议参考：

<https://github.com/normaldotcom/canable2-fw/tree/b158aa7ab106bc018715cd4d5f02f544f9de1830>

关键文件：README.md、src/can.c、src/slcan.c、src/system.c。