UCAN-Mini 技术手册 V2.6

LM 协议、Linux SocketCAN 驱动、Python 自动化脚本与 UCAN-Analyzer

项目	内容
文档名称	UCAN-Mini 技术手册 V2.6
文档版本	V2.6
日期	2026-05
适用对象	上位机开发、Linux 驱动集成、FAE、测试与系统集成人员
适用产品	UCAN-Mini USB to Dual CAN/CAN-FD Interface
重点范围	LM 协议、UCAN LM Linux 内核驱动、Python CLI/API、UCAN-Analyzer、预约发送

目录与阅读路径

本手册面向需要把 UCAN-Mini 接入工程系统的用户。UCAN-Mini 运行 LM 协议；LM 协议既可以通过 Linux 内核驱动呈现为 SocketCAN 接口，也可以通过 Python 脚本直接访问 USB TLV 协议。

基础收发流程

#	用户要完成的事	手册入口	完成标志
1	识别设备与 USB 枚举	1、2、4	设备枚举为 `34B7:E471`
2	选择接入路径	3	确定使用内核驱动、Python CLI/API 或 UCAN-Analyzer
3A	Linux SocketCAN 接入	6	`ip link` 中出现 `can0` / `can1`
3B	Python 脚本接入	7	`python ucan_mini_cli.py info` 返回设备信息
3C	桌面上位机接入	8	UCAN-Analyzer 识别设备并打开通道
4	配置 CAN/CAN FD 通道	6.5、7.4、8.4、10	通道启动，位时序与对端一致
5	完成收发或预约发送	6.6、7.5、8.5、9	总线上看到立即发送或预约发送帧
6	现场维护和异常定位	11、12、13	能区分枚举、驱动占用、位时序、总线和固件版本问题

按任务阅读

任务	优先阅读	目标
快速理解产品能力	1、2、3	理解 LM 主协议、双通道、驱动和脚本边界
Linux 应用接入	6	安装 `ucan_lm` 驱动并使用 SocketCAN
自动化测试脚本	7、9	使用 Python CLI/API 配置通道、收发和预约发送
桌面调试分析	8	使用 UCAN-Analyzer 完成配置、收发、过滤、追踪、分析和固件更新
LM 协议说明	5、7	理解 TLV 消息类型，并使用随包提供的 Python API
现场排查问题	12、13	定位驱动占用、USB 句柄、接口状态和 CAN 总线错误

1. 产品定位

UCAN-Mini 是一款 USB 转双通道 CAN/CAN FD 接口设备。设备运行 LM 协议，面向以下工程需求：

需求	推荐路径	说明
现有 Linux CAN 应用	UCAN LM 内核驱动	设备注册为 SocketCAN 网络接口，直接使用 `can0` / `can1`
自动化测试和产测	Python CLI / API	通过 PyUSB 访问 LM TLV 协议，覆盖设备管理、通道配置、收发、预约发送
桌面可视化调试	UCAN-Analyzer	多平台图形工具，覆盖配置、收发、过滤、追踪、数据分析、波形和固件更新
协议/API 集成	Python API / LM TLV 协议	随包提供命令行和 Python API，其他语言可按 TLV 协议接入
设备通过 LM 协议提供驱动、脚本和桌面上位机三种接入方式。		

2. 核心特性

项目	说明
CAN 通道	2 路 CAN/CAN FD
主协议	LM 协议
Linux 支持	out-of-tree `ucan_lm` SocketCAN 内核驱动
Python 支持	`private_proto/ucan_mini_cli.py` 和 `ucan_mini_proto.py`
上位机工具	UCAN-Analyzer 多平台桌面软件
预约发送	`CMD_SET_SCHEDULE`，slot 0..31，支持单次预约发送
时间戳	LM 协议支持设备时间戳同步；内核驱动可暴露 RX 硬件时间戳
终端电阻	短按按键切换双通道 120R 终端电阻，配置持久化
固件更新	仅在 UCAN-Analyzer 闭源上位机中提供

3. 接入路径选择

接入路径	适合场景	优点	边界
Linux 内核驱动	Linux 控制器、采集程序、已有 SocketCAN 软件	复用标准 CAN 网络接口、can-utils 和现有应用	不提供固件更新功能
Python CLI/API	自动化测试、产线脚本、FAE 诊断、专项功能验证	命令覆盖完整，调试快，能访问预约发送和管理命令	不提供固件更新功能
UCAN-Analyzer	研发调试、现场诊断、可视化分析、固件维护	图形化配置、收发、过滤、追踪、分析、波形、触发和升级	面向交互式调试，不替代批量脚本和内核 SocketCAN 应用

4. USB 枚举与端点

4.1 LM 模式

项目	值
VID	`0x34B7`
PID	`0xE471`
Interface 0	`UCAN Mini LM mgmt`，管理命令
Interface 1	`UCAN Mini LM can0`
Interface 2	`UCAN Mini LM can1`

通道	方向	端点	说明
CMD	Host -> Device	`0x01`	广播/设备级命令
CMD	Device -> Host	`0x81`	命令响应 / 事件
CAN0	Host -> Device	`0x02`	CAN0 数据与通道命令
CAN0	Device -> Host	`0x82`	CAN0 接收 / TX Echo / 通道响应
CAN1	Host -> Device	`0x03`	CAN1 数据与通道命令
CAN1	Device -> Host	`0x83`	CAN1 接收 / TX Echo / 通道响应

5. LM TLV 协议概览

5.1 TLV Header

所有 LM 消息均以 8 字节 TLV Header 开头，后跟可变长度载荷，总长度 4 字节对齐。

```
offset  size  field      description
0x00    2     type       消息类型
0x02    2     size       整条消息总长度，含头部，4B 对齐
0x04    1     channel    CAN 通道号，0xFF 表示设备级/广播
0x05    1     flags      消息级标志
0x06    2     reserved   保留，置 0
0x08    ...   payload    可变长度载荷
```

Wire 格式为 little-endian：`<HHBBH`。

5.2 常用消息类型

类型码	名称	方向	说明
`0x0002`	`CMD_START`	Host -> Device	启动通道
`0x0003`	`CMD_SET_BITTIMING`	Host -> Device	设置仲裁段位时序
`0x0004`	`CMD_SET_DATA_BITTIMING`	Host -> Device	设置 CAN FD 数据段位时序
`0x0006`	`CMD_SET_TERMINATION`	Host -> Device	设置终端电阻
`0x0012`	`CMD_SET_SCHEDULE`	Host -> Device	配置预约发送
`0x0013`	`CMD_CLR_SCHEDULE`	Host -> Device	清除预约发送
`0x0018`	`CMD_TIMESTAMP_SYNC`	Host -> Device	设备时间戳同步
`0x001B`	`CMD_GET_STATUS`	Host -> Device	获取通道状态
`0x1000`	`DATA_CAN_TX`	Host -> Device	CAN/CAN FD 发送帧
`0x2000`	`DATA_CAN_RX`	Device -> Host	CAN/CAN FD 接收帧
`0x2001`	`DATA_CAN_TX_ECHO`	Device -> Host	TX Echo

5.3 能力位重点

能力	含义
`CAN_FD` / `CAN_FD_BRS`	支持 CAN FD 与 BRS
`HW_TIMESTAMP`	支持设备侧时间戳
`TERMINATION`	支持终端电阻控制
`TX_ECHO`	支持发送回显
`SCHEDULE_TX`	支持预约发送
`AUTO_BUSOFF_REC`	支持 Bus-Off 自动恢复

6. Linux 内核驱动

UCAN-Mini 的 LM 模式提供 out-of-tree Linux SocketCAN 驱动，发布包目录为：

```
kernel_driver
```

驱动文件：

文件	说明
`ucan_lm.c`	USB DATA-interface + SocketCAN 驱动实现
`ucan_lm.h`	协议常量和 packed TLV 结构
`Makefile`	外部内核模块构建入口
`dkms.conf`	DKMS 安装模板

6.1 环境准备

Debian / Ubuntu 示例：

```
sudo apt update
sudo apt install -y build-essential linux-headers-$(uname -r) can-utils
```

如果系统没有 `ip` 或 CAN 工具，补充安装：

```
sudo apt install -y iproute2 can-utils
```

6.2 手动编译安装

```
cd kernel_driver
make
sudo make install
sudo depmod -a
sudo modprobe ucan_lm
```

确认模块加载：

```
lsmod | grep ucan_lm
modinfo ucan_lm
```

插入 UCAN-Mini 后查看内核日志：

```
dmesg | tail -80
ip link
```

正常情况下，每个 CAN 通道会注册为一个 SocketCAN 网络接口，例如 `can0`、`can1`。

6.3 DKMS 安装

DKMS 适合随内核升级自动重建模块：

```
sudo mkdir -p /usr/src/ucan-lm-0.1.0
sudo cp -a kernel_driver/. /usr/src/ucan-lm-0.1.0/
sudo dkms add -m ucan-lm -v 0.1.0
sudo dkms build -m ucan-lm -v 0.1.0
sudo dkms install -m ucan-lm -v 0.1.0
sudo modprobe ucan_lm
```

查看 DKMS 状态：

```
dkms status ucan-lm
```

卸载：

```
sudo dkms remove -m ucan-lm -v 0.1.0 --all
sudo modprobe -r ucan_lm
```

6.4 模块参数

参数	默认值	说明
`clock_hz`	`80000000`	SocketCAN 位时序计算使用的 CAN 控制器时钟
`tx_window`	`5`	每通道 in-flight TX echo 槽数量
`rx_hwts`	`true`	暴露由设备时间戳换算的 RX 硬件时间戳
`time_sync_interval_ms`	`1000`	设备/主机时间同步刷新间隔
`time_sync_max_rtt_us`	`5000`	丢弃 RTT 过大的时间同步样本

临时加载示例：

```
sudo modprobe ucan_lm tx_window=8 rx_hwts=1 time_sync_interval_ms=500
```

需要把参数持久化时，可创建 `/etc/modprobe.d/ucan_lm.conf`：

```
options ucan_lm tx_window=8 rx_hwts=1 time_sync_interval_ms=500
```

注意：`rx_hwts=1` 时，驱动会绑定 MGMT 接口用于时间同步。此时用户态 Python 管理工具可能无法同时打开 MGMT 接口。Linux 内核驱动不提供固件更新功能。

6.5 SocketCAN 使用

Classic CAN：

```
sudo ip link set can0 down 2>/dev/null || true
sudo ip link set can0 up type can bitrate 500000
candump can0
cansend can0 123#11223344
```

CAN FD：

```
sudo ip link set can0 down 2>/dev/null || true
sudo ip link set can0 up type can bitrate 1000000 dbitrate 5000000 fd on
candump can0
cansend can0 123##1001122334455667788 99AABBCCDDEEFF
```

双通道：

```
sudo ip link set can0 up type can bitrate 1000000 dbitrate 5000000 fd on
sudo ip link set can1 up type can bitrate 1000000 dbitrate 5000000 fd on
candump can0 can1
```

6.6 时间戳与 TX 窗口

`tx_window` 限制每个通道同时等待 TX echo 的发送槽数量。控制类业务建议保持较小窗口，减少旧命令排队；吞吐类测试可适当增大。

`rx_hwts=1` 时，驱动使用 MGMT 命令端点做设备/主机时间同步，并将设备 RX 时间戳换算到 Linux skb 硬件时间戳路径。若现场只需要普通 SocketCAN 收发，可用 `rx_hwts=0` 简化接口占用。

---

7. Python LM 协议脚本

Python 脚本发布包目录为：

```
private_proto
```

文件	说明
`ucan_mini_cli.py`	命令行工具，适合调试、自动化和产测
`ucan_mini_proto.py`	Python API，适合业务脚本复用

7.1 安装依赖

```
cd private_proto
python -m pip install pyusb
```

Linux 需要 libusb：

```
sudo apt install -y libusb-1.0-0-dev
```

Windows 下需要给 LM 接口安装 WinUSB / libusb 驱动，例如使用 Zadig。若同一台电脑上还有内核驱动或其他工具占用设备，先关闭占用进程。

7.2 基础命令

```
python ucan_mini_cli.py list
python ucan_mini_cli.py info
python ucan_mini_cli.py status --ch 0
python ucan_mini_cli.py identify
```

`info` 走 EP0 vendor request，可用于确认设备在线、通道数量、固件版本和能力信息。

7.3 通道配置

设置 Classic CAN 仲裁段位时序：

```
python ucan_mini_cli.py set-bittiming --ch 0 --brp 10 --tseg1 13 --tseg2 2 --sjw 1
python ucan_mini_cli.py start --ch 0
```

CAN FD + BRS：

```
python ucan_mini_cli.py set-bittiming --ch 0 --brp 10 --tseg1 13 --tseg2 2 --sjw 1
python ucan_mini_cli.py set-data-bittiming --ch 0 --brp 2 --tseg1 13 --tseg2 2 --sjw 1
python ucan_mini_cli.py start --ch 0 --mode fd fd_brs tx_echo
```

终端电阻：

```
python ucan_mini_cli.py set-termination --ch 0 --enable 1
python ucan_mini_cli.py set-termination --ch 1 --enable 1
```

停止通道：

```
python ucan_mini_cli.py stop --ch 0
```

7.4 立即发送与接收

发送标准帧：

```
python ucan_mini_cli.py send --ch 0 --id 0x123 --data "01 02 03 04"
```

发送扩展帧：

```
python ucan_mini_cli.py send --ch 0 --id 0x12345678 --ext 1 --data "AA BB"
```

发送 CAN FD + BRS：

```
python ucan_mini_cli.py send --ch 0 --id 0x100 --fd 1 --brs 1 --data "01 02 03 04 05 06 07 08 09 0A"
```

批量发送：

```
python ucan_mini_cli.py send --ch 0 --id 0x200 --data "FF" --count 100 --interval-ms 10
```

接收：

```
python ucan_mini_cli.py receive --ch 0
python ucan_mini_cli.py receive --ch 0 --count 10 --rx-timeout-ms 1000
```

7.5 过滤、选项与状态

过滤器：

```
python ucan_mini_cli.py set-filter --ch 0 --type range --index 0 --id1 0x100 --id2 0x1FF
python ucan_mini_cli.py set-filter --ch 0 --type exact --index 0 --id1 0x123
python ucan_mini_cli.py set-filter --ch 0 --type mask --index 0 --id1 0x7F0 --id2 0x100
python ucan_mini_cli.py clear-filter --ch 0
```

运行时选项：

```
python ucan_mini_cli.py set-option --ch 0 --options tx_echo error_report bus_load
python ucan_mini_cli.py clear-option --ch 0 --options bus_load
python ucan_mini_cli.py set-auto-recover --ch 0 --enable 1
python ucan_mini_cli.py set-iso-mode --ch 0 --iso 1
```

状态查询：

```
python ucan_mini_cli.py status --ch 0
```

返回信息中重点关注 `state_name`、`started`、`termination`、`tx_count`、`rx_count`、`tx_err_count`、`rx_err_count`、`overrun_count` 和 `busload`。

7.6 Python API

`CanClient` 适合需要设备级管理命令的脚本：

```python
from ucan_mini_proto import CanClient, MODE_FD, MODE_FD_BRS, MODE_TX_ECHO

cli = CanClient()
info = cli.get_device_info()
print(info)

cli.set_bittiming(channel=0, brp=10, tseg1=13, tseg2=2, sjw=1)
cli.set_data_bittiming(channel=0, brp=2, tseg1=13, tseg2=2, sjw=1)
cli.start(channel=0, mode_flags=MODE_FD | MODE_FD_BRS | MODE_TX_ECHO)
cli.send(channel=0, can_id=0x123, data=b"\x01\x02\x03", is_fd=True, brs=True)
```

```
frame = cli.receive(channel=0, timeout_ms=1000)
print(frame)

cli.stop(channel=0)
cli.close()
```

`ChannelClient` 适合两个独立进程分别占用一个 CAN DATA interface。它只暴露通道级命令；设备信息、设备复位、用户 Flash、设备 ID、timestamp sync 等设备级操作仍通过 `CanClient` 完成。Python CLI/API 不提供固件更新功能。

```
from ucan_mini_proto import ChannelClient, MODE_FD, MODE_FD_BRS

can0 = ChannelClient(0)
can0.set_bittiming(brp=10, tseg1=13, tseg2=2, sjw=1)
can0.start(mode_flags=MODE_FD | MODE_FD_BRS)
can0.send(can_id=0x123, data=b"\x01\x02")
can0.close()
```

8. UCAN-Analyzer 多平台上位机

UCAN-Analyzer 是 UCAN-Mini 的配套桌面分析工具，面向 CAN / CAN FD 设备配置、收发测试、帧监控、过滤、数据追踪、曲线分析和固件更新，适合交互式调试、现场诊断和总线可视化分析。

8.1 支持平台与发布形态

平台	架构	发布包
Windows	x86_64	`UCAN-Analyzer-*-windows-x86_64.zip`
Linux	x86_64	`UCAN-Analyzer-*-linux-x86_64.tar.gz`
macOS	ARM64	`UCAN-Analyzer-*-macos-arm64.tar.gz`

按主机系统选择对应发布包，启动后连接 UCAN-Mini 即可进行通道配置、收发测试、总线分析和固件维护。

8.2 功能模块

功能模块	说明
设备连接与状态监控	识别 UCAN-Mini，显示通道状态、帧计数、错误计数和总线状态
CAN / CAN FD 通道配置	配置仲裁位率、数据位率、采样点、终端电阻、TX FIFO/QUEUE、CAN FD、BRS 等
接收与追踪	实时显示 RX、TX Echo 和错误相关帧，支持跟踪、清空和导出
发送与周期发送	支持单帧发送、批量发送、循环发送、发送条目管理和定时发送视图
过滤与触发	支持表达式过滤、设备端过滤配置、触发规则和调试定位
数据分析	按设备、通道、方向、CAN ID、位偏移、位长度和字节序提取数值
波形与信号	支持数据波形、信号查看、DBC 解析和信号曲线
总线诊断	显示总线负载、帧率、ID 分布、错误计数和告警信息
固件更新	仅在 UCAN-Analyzer 闭源上位机中提供

8.3 典型调试流程

```
1. 启动 UCAN-Analyzer
2. 连接 UCAN-Mini
3. 选择 CAN0/CAN1，设置仲裁位率、数据位率、CAN FD、BRS 和终端电阻
4. 启动通道
5. 在接收/追踪窗口观察实时帧
6. 在发送窗口构造单帧、批量帧或周期发送条目
7. 使用过滤、触发、DBC、信号曲线或数据分析定位问题
8. 需要维护时使用 UCAN-Analyzer 的固件更新功能
```

8.4 机器人本体调试建议

UCAN-Analyzer 适合机器人本体内部 CAN/CAN FD 网络的交互式调试：

场景	建议用法
关节模组启动调试	用发送窗口构造控制帧，用接收/追踪窗口确认状态帧和错误帧
驱动器参数验证	过滤目标 CAN ID，观察配置命令、响应和状态变化
传感器与 I/O 调试	用数据分析条目提取位域、字节序和工程值，配合波形查看变化趋势
整机总线诊断	观察总线负载、帧率、ID 分布、错误计数和 Bus-Off 告警
产线辅助测试	保存发送条目、配置和分析视图，用于复测和问题复现

9. 预约发送

预约发送用于减少主机调度抖动对发送时刻的影响。主机先同步设备时间，再把待发送帧和目标时间下发到设备；设备按本地 PTPC 时间触发发送。

9.1 能力与限制

项目	说明
命令	`CMD_SET_SCHEDULE` / `CMD_CLR_SCHEDULE`
槽位	`slot_index = 0..31`
模式	单次预约
时间基准	设备 PTPC 绝对时间，单位 us
最小提前量	固件当前要求至少提前 200 us；工程上建议留出更大余量
周期参数	不支持周期预约，周期请求返回不支持
次数参数	不支持周期重复
数据长度	Classic CAN 最大 8B，CAN FD 最大 64B

9.2 CLI 单次预约

CLI 在未提供 `--start-time-us` 时会自动执行 `timestamp-sync`，并使用当前设备时间加 `--delay-us`。

```
python ucan_mini_cli.py set-schedule \
  --ch 0 \
  --slot 0 \
  --enable 1 \
  --delay-us 10000 \
  --can-id 0x100 \
  --data "01 02 03"
```

这表示约 10 ms 后发送一次 ID `0x100`。

9.3 CLI CAN FD + BRS 预约

```
python ucan_mini_cli.py set-schedule \
  --ch 0 \
  --slot 1 \
  --enable 1 \
  --delay-us 20000 \
  --can-id 0x123 \
  --fd 1 \
  --brs 1 \
  --data "00 01 02 03 04 05 06 07 08 09 0A 0B"
```

清除预约：

```
python ucan_mini_cli.py clear-schedule --ch 0
```

9.4 API 预约发送

```python
import time
from ucan_mini_proto import CanClient, TX_FLAG_FD, TX_FLAG_BRS, len_to_dlc

cli = CanClient()
dev_us, _ = cli.timestamp_sync(int(time.time() * 1_000_000))
data = bytes.fromhex("01 02 03 04")

cli.set_schedule(
    channel=0,
    slot_index=0,
    enable=True,
    start_time_us=dev_us + 10_000,
    can_id=0x100,
    dlc=len_to_dlc(len(data)),
    tx_flags=0,
    data=data,
)
```

9.5 排查建议

现象	检查项
命令返回 `INVALID_PARAM`	槽位、DLC、CAN FD 数据长度、时间提前量
命令返回 `NOT_SUPPORTED`	是否请求了周期预约
没有看到发送帧	通道是否启动、位率是否一致、终端电阻和线序是否正确
预约发送不稳定	主机是否太晚下发、是否频繁重配同一 slot、总线是否接近满载
清除后仍有帧	确认清除的是正确通道，必要时重置通道

10. 通道配置建议

10.1 Classic CAN

项目	建议
位率	与对端一致，常见为 125k、250k、500k、1M
终端电阻	总线两端各 120R，UCAN-Mini 可通过短按或命令切换
验证方式	先单通道 loop 或对端工具收发，再双通道并发

10.2 CAN FD

项目	建议
仲裁段位率	常见 500k 或 1M
数据段位率	常见 2M、4M、5M
BRS	双端配置保持一致
数据长度	Classic CAN 最大 8B，CAN FD 最大 64B
TDC	高数据段位率建议启用并验证采样点
排查顺序	位率、采样点、BRS、终端电阻、线序和对端 FD 支持

11. 按键、LED 与固件更新

11.1 按键功能

操作	功能
短按	切换双通道 120R 终端电阻开关

11.2 LED 指示

状态	指示含义
上电启动	蓝色常亮
LM 运行	紫色常亮
终端电阻开启	在当前协议颜色基础上叠加红色提示
故障状态	红色闪烁

通道灯表现	含义
TX/RX 灯灭	通道未启动
TX 低亮绿，RX 低亮蓝	正常运行空闲
TX 高亮闪烁	发送数据
RX 高亮闪烁	接收数据
黄色	Warning
粉色	Error Passive
红色	Bus-Off

11.3 固件更新

固件更新仅在 UCAN-Analyzer 闭源上位机中提供。Linux 内核驱动和 Python CLI/API 不提供固件更新功能。

12. 调试与诊断

12.1 USB 或驱动异常

现象	可能原因	处理
`ip link` 没有 can 接口	未加载 `ucan_lm`、驱动编译失败或接口被其他进程占用	检查 `lsmod`、`dmesg`、VID/PID
Python 打不开设备	内核驱动或其他进程占用接口	卸载驱动、关闭工具或改用未占用的 DATA interface
开启 `rx_hwts` 后 MGMT 不可用	驱动绑定 MGMT 做时间同步	用 `rx_hwts=0` 重载驱动，或卸载驱动后运行管理工具

12.2 CAN 无法通信

优先检查：

- CANH/CANL/GND 和屏蔽连接。
- 总线两端是否各有 120R。
- 两端仲裁段位率、数据段位率、BRS 和 CAN FD 使能是否一致。
- 通道是否已启动，是否处于 Warning、Error Passive 或 Bus-Off。
- 是否把 CAN0/CAN1、端点或 SocketCAN 接口用反。

12.3 预约发送异常

优先检查：

- 设备能力是否包含 `SCHEDULE_TX`。
- 通道是否已经启动。
- `start_time_us` 是否基于设备时间，不是主机系统时间。
- 下发时间是否太晚，至少预留 200 us，工程测试建议预留数 ms。
- 是否误用了已移除的周期预约参数。

13. 技术规格

参数	规格
产品	UCAN-Mini
接口	USB 转双通道 CAN/CAN FD
CAN 通道	2
CAN 协议	CAN 2.0 / CAN FD
LM VID/PID	`0x34B7 / 0xE471`
LM 接口	mgmt、can0、can1
LM 数据端点	CAN0 `0x02/0x82`，CAN1 `0x03/0x83`
LM 管理端点	CMD `0x01/0x81`
Linux 驱动	`kernel_driver/ucan_lm.c`
Python 工具	`private_proto/ucan_mini_cli.py`、`ucan_mini_proto.py`
上位机工具	UCAN-Analyzer
上位机平台	Windows x86_64、Linux x86_64、macOS ARM64
预约发送	slot 0..31，单次预约
本地按键	短按切换双通道终端电阻
配置存储	NVS 持久化
固件维护	仅 UCAN-Analyzer 闭源上位机提供

14. 修订历史

版本	日期	说明
V2.6	2026-05	移除 GS 兼容模式说明，更新 LM 单次预约发送、固件维护边界和按键行为
V2.5	2026-05	增加 GS 兼容模式、协议切换和周期预约发送说明
V2.4	2026-05	清理修订历史对外表述，更新发布版技术手册与 PDF
V2.3	2026-05	增加 UCAN-Analyzer 上位机使用说明与多平台发布说明
V2.2	2026-05	增加 Linux 驱动、Python CLI 与周期发送说明
V2.1	2026-05	拆分产品介绍与技术手册，统一文档结构和发布格式
V2.0	2026-04	完成双通道 CAN / CAN FD 接口技术手册基础整理