

噪声污染 Sync 测量，错误波特率被接受，FE 自锁

问题描述

如下图所示，在 GPIO 捕获 Sync（0x55）期间若 LIN 总线混入毛刺边沿，由于当前识别逻辑仅校验**边沿个数与总宽度**，额外毛刺会使捕获在主机 Sync 尚未发完时即凑满边沿，且 4 段双位间隔总和一旦落入 ±10.5% 容差窗即被提前判为 `LIN_SYNC_OK`，写入错误波特率并将 `s_isAutobaudMode` 清零；切回 UART（状态已置为 `RECV_PID`）后，UART 会将总线上残余的半截 Sync（及后续波形）误当作 PID 字节错位接收。**只有 PID 阶段产生帧错误**（`UART_FRAME_ERR`，**[T3]**）时，驱动在 Header 阶段不调用 `Callback` 直接退回 `IDLE`，导致 `s_isAutobaudMode` 无法恢复、错误波特率持续驻留，**且后面的帧最终在 [T5~T6] 陷入持续 FE，才会一直维持不再进入自适应模式，产生自锁**：

代码块

```
[A] 毛刺致 Sync 半途切回，PID 段再遇噪声才触发 FE (1=隐, 0=显):
            |<-- GPIO Capture(d1..4) ->|<- UART(RECV_PID/BadBAUD) >|
RX=1 --+    +--+      +-+ +-+     +--+ +-+   +--+    +--------+        +----+----
       |Brk|D| d1   | | | | d3 |   | | |   |   | S |SyncRem|Stop=0?|PID |Idle
RX=0   +---+ +-----+ +-+ +----+   +-+ +--+   +---+-------+-------+----+
              ^       ^   ^       ^   ^   ^                    ^
              |       |   |       |   |   |                    +- T3: 遇噪声才FE
              |       |   |       |   |   +- T2: r0提前SYNC_OK,切UART,Mode=0
              |       |   |       |   +- T1: e4满5沿仅检总宽 -> 错写BAUD
              |       |   |       +-- 毛刺混入: 3280/516/1361/186(Sync未完)
```

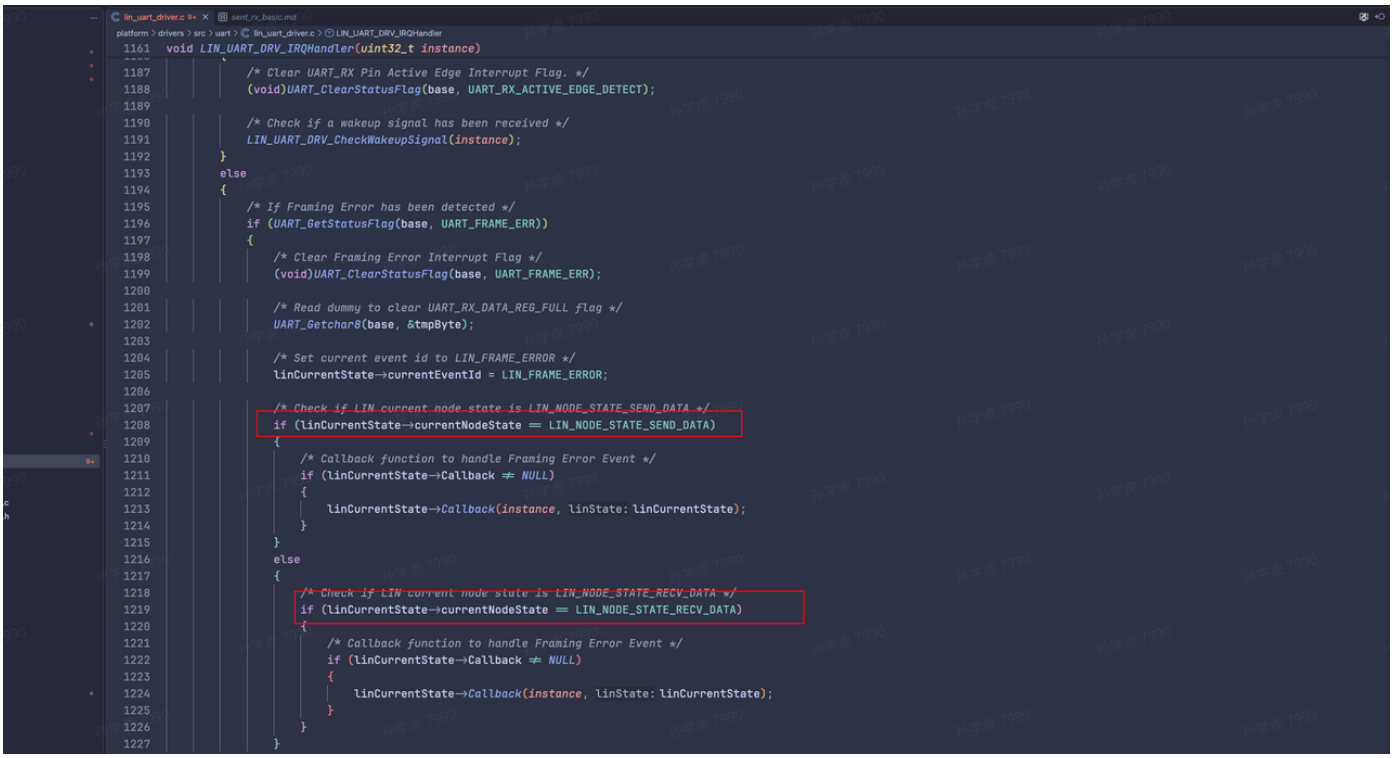
```
            +- T0: e0 起始沿 (Break/Del毛刺或起始沿, MUX=GPIO, Mode=1)

[B] 跨帧状态演进与 FE 静默吞没自锁时序 (Frame N -> Frame N+1..n):
  App: LinSync (Mode=s_isAutobaudMode) | SDK: lin_uart_driver
  -------------------------------------+------------------------------
  [T0] Break OK (Mode == 1):           |
   |   EnableGpioCapture() (MUX=GPIO)  | State = RECV_SYNC (开边沿捕获)
  [T1] GPIO ISR (e0..e4, count == 5):  |
   |   sum=5343 -> Baud=17964 (-6.4%)  | 仅检边沿数+总宽, 放过畸变间隔!
   |   SetBaudRate(17964) ------------>| UART->BAUD 写入错误波特率
  [T2] GPIO ISR (r0 rising -> SYNC_OK):| (此时主机 Sync 实际尚未发完!)
   |   ExitAutobaudMode():             |
   |   Mode = 0, Restore MUX = UART -->| State = RECV_PID (提前切回UART)
  [T3]                                 | 残余当PID收, PID段遇噪声才FE!
  [T4] (Callback NOT called!)          | IRQHandler(L1208): RECV_PID
   |   RearmForNextHeader() skipped!   | 非 DATA 阶段 -> 跳过 Callback!
   |   Mode stays 0!  <----------------| GotoIdleState() 静默退回 IDLE
  -------------------------------------+------------------------------
  [T5] Next Break OK (Frame N+1..n):   | 检出新帧 Break, 调 Callback()
   |   if (Mode) == false (Mode == 0)  |
   |   EnableGpioCapture() skipped! -->| State = RECV_SYNC (仍错 BAUD!)
  [T6] (Callback STILL skipped!)       | 收 Header 再触发 FE -> 不回调!
   v   Mode locked at 0 forever!  <----v GotoIdleState() -> 死循环自锁!
  ======================================================================
  [自锁终态] s_isAutobaudMode=0(拒入自适应) + BAUD=错误值(持续触发FE)
             + Header阶段FE不调Callback ==> 帧帧静默回IDLE, 持续无响应!
  ======================================================================
```

主要矛盾

- **Header 阶段（RECV_PID / RECV_SYNC）触发帧错误时驱动不调用 Callback，阻断 s_isAutobaudMode 恢复闭环：**

  ① 驱动层 Header 阶段唯独 FE（及伴随 FE 的噪声）静默吞没回调：



```
lin_uart_driver.c 9+ ×  sent_rx_basic.md
platform > drivers > src > uart > lin_uart_driver.c > LIN_UART_DRV_IRQHandler
1161  void LIN_UART_DRV_IRQHandler(uint32_t instance)
1187          /* Clear UART_RX Pin Active Edge Interrupt Flag. */
1188          (void)UART_ClearStatusFlag(base, UART_RX_ACTIVE_EDGE_DETECT);
1189
1190          /* Check if a wakeup signal has been received */
1191          LIN_UART_DRV_CheckWakeupSignal(instance);
1192      }
1193      else
1194      {
1195          /* If Framing Error has been detected */
1196          if (UART_GetStatusFlag(base, UART_FRAME_ERR))
1197          {
1198              /* Clear Framing Error Interrupt Flag */
1199              (void)UART_ClearStatusFlag(base, UART_FRAME_ERR);
1200
1201              /* Read dummy to clear UART_RX_DATA_REG_FULL flag */
1202              UART_Getchar8(base, &tmpByte);
1203
1204              /* Set current event id to LIN_FRAME_ERROR */
1205              linCurrentState->currentEventId = LIN_FRAME_ERROR;
1206
1207              /* Check if LIN current node state is LIN_NODE_STATE_SEND_DATA */
1208              if (linCurrentState->currentNodeState == LIN_NODE_STATE_SEND_DATA)
1209              {
1210                  /* Callback function to handle Framing Error Event */
1211                  if (linCurrentState->Callback != NULL)
1212                  {
1213                      linCurrentState->Callback(instance, linState: linCurrentState);
1214                  }
1215              }
1216              else
1217              {
1218                  /* Check if LIN current node state is LIN_NODE_STATE_RECV_DATA */
1219                  if (linCurrentState->currentNodeState == LIN_NODE_STATE_RECV_DATA)
1220                  {
1221                      /* Callback function to handle Framing Error Event */
1222                      if (linCurrentState->Callback != NULL)
1223                      {
1224                          linCurrentState->Callback(instance, linState: linCurrentState);
1225                      }
1226                  }
1227              }
```

次要矛盾

- **Sync 识别仅校验“边沿个数 + 总宽度”，导致在 Sync 半途提前切回 UART 并将残余 Sync 误当 PID 接收：**应用层在 `LinSync_ExitAutobaudMode()` 中将 `s_isAutobaudMode` 清为 `0`，而 `LIN_RECV_BREAK_FIELD_OK` 分支未重算该标志，完全依赖后续回调调用 `LinSync_RearmForNextHeader()` 置回 `1`。一旦 `RECV_PID` 阶段静默 `FE`，`s_isAutobaudMode` 永久锁死为 `0`，后续每帧跳过自适应并沿用错误波特率反复触发静默 `FE`。同理，关联工程 `Lin_AutoBaud_Dma_Demo` 仅在回调 `Rearm` 路径重装 `LinSync_ArmDmaCaptureIdle()`，遭遇 `RECV_PID` 静默 `FE` 漏掉回调后下一帧因 DMA 请求未重装且 `LBKDIE=0` 也将陷入永久死锁。。
- **Sync 识别仅校验“边沿个数 + 总宽度”，畸变间隔可被接受：** `LinSync_HandleCapture()` 凑满 5 个下降沿即取 4 段平均，只检 ±10.5% 窗口就写入 BAUD，随后第一个上升沿即判 `LIN_SYNC_OK` 并切回 UART。毛刺补出的边沿让捕获在 Sync 半途凑满，写入错误 BAUD，并把残余 Sync 当作 PID 接收而触发 FE。它是写入错误 BAUD 的诱因，自锁本身由主要矛盾造成。

板级复现

- **方法一（总线间断接地 / 毛刺注入复现）：**主机按标称波特率周期发送 LIN 帧，对 LIN 总线进行多次间断对地擦碰或注入随机脉冲干扰。当毛刺边沿混入 Break/Delimiter 或 Sync（0x55）捕获区间，使 5 个下降沿在 Sync 半途提前凑满且 4 段总和落入 ±10.5% 窗口时，从机提前切回 UART（状态置为 `RECV_PID`）并将残余 Sync 波形错位当作 PID 字节接收。**只有 PID 阶段**（错位字节的停止位采样点）触发 `UART_FRAME_ERR` 时，**且后面的帧最终在 [T5~T6] 陷入持续 FE，**才陷入持续无响应。

复现记录

在实体板上（LE1 中断版）对帧头逐点注入毛刺，每帧后用 J-Link 非停机读回 `UART0->BAUD`、`s_isAutobaudMode`、节点状态与 `s_twoBitTimeLength[]`（单毛刺扫描 93 点 + 错误 BAUD 注入 18 档 + 双毛刺端到端 3 次），实测结果如下：

代码块

```
[C] 单毛刺 (10 us) 位置 -> 帧后 J-Link 读回 (19,200 bps, 93 点):
Glitch region       Edge(us)   BAUD read(bps)   Mode  | 本帧结果
-------------------------------------------------------+--------------------
Brk gap 440~574     450~584    19231 keep       1     | 通过(偶发漏检 Break)
Brk gap 576         586 (~T1)  19231            1     | 2/3 进入旧竞态
Brk gap 578~580     588~590    17647 (-8.1%)    0/1   | 第 2 个干净帧恢复
Brk gap 582~650     592~660    17857~20979      1     | 丢 1 帧
Brk gap 652~654     662~664    21127 (+10.0%)   1     | 丢 1 帧
Brk gap 655~665     665~675    reject >21216    1     | 越界拒绝
Delim / S / b0~b5   -          reject >=+15%    1     | 拒绝
Sync b6 1106~1125   1106~1125  21127~20270      1     | 丢 1 帧
Sync b7 gap 1160    1170       19231            0     | 提前切回 UART, 丢 1 帧
Sync b7 1165~, Stop -          19231            1     | 通过
-------------------------------------------------------------------------------
 s_twoBitTimeLength @578us = [1696, 1248, 1250, 1251]  (normal ~1250)
 accept window [17184, 21216] bps; model vs. read-back <= 0.5%

[D] 错误 BAUD + Mode=0 注入 (J-Link 写 SBR, 之后 5 个干净帧):
SBR    BAUD(bps)    Dev        F1 F2 F3 F4 F5  | x=无回复 P=回复 数字=帧后 Mode
-----------------------------------------------+----------------------------
70     21429        +11.6%     x0 x0 x0 x0 x0  | 自锁(超出接受窗)
71     21127        +10.0%     x0 x0 x0 x0 x0  | 自锁
72     20833        +8.5%      x0 x0 x1 P1 P1  | 丢 3 帧后恢复
73     20548        +7.0%      x1 P1 P1 P1 P1  | 丢 1 帧
74~82  20270~18293  +5.6~-4.7  P1 P1 P1 P1 P1  | 不丢帧(回复速率偏差 <=5.6%)
83~87  18072~17241  -5.9~-10.2 x1 P1 P1 P1 P1  | 丢 1 帧(含 SBR 85, -8.1%)

[E] 双毛刺端到端: 毛刺1 = Break 缺口 653~663 us, 毛刺2 = 1612~1647 us:
Run  After glitch frame     Clean x6     Each snapshot    | 结果
-------------------------------------------------------+------------
1    21127 / Mode0 / IDLE   0/6 reply    same, LBKDIE=1   | 永久失联
2    21127 / Mode0 / IDLE   0/6 reply    same, LBKDIE=1   | 永久失联
3    21127 / Mode0 / IDLE   0/6 reply    same, LBKDIE=1   | 永久失联
ctl  glitch1 only           next frame   recovers 3/3     | 对照
-------------------------------------------------------------------------------

[实测结论] 自锁需两个条件同时成立，缺一则 1~3 帧内自行恢复:
 - 毛刺帧以静默 FE 结束
 - 错误的baudrate导致后面的帧在sync和pid阶段持续产生FE
```

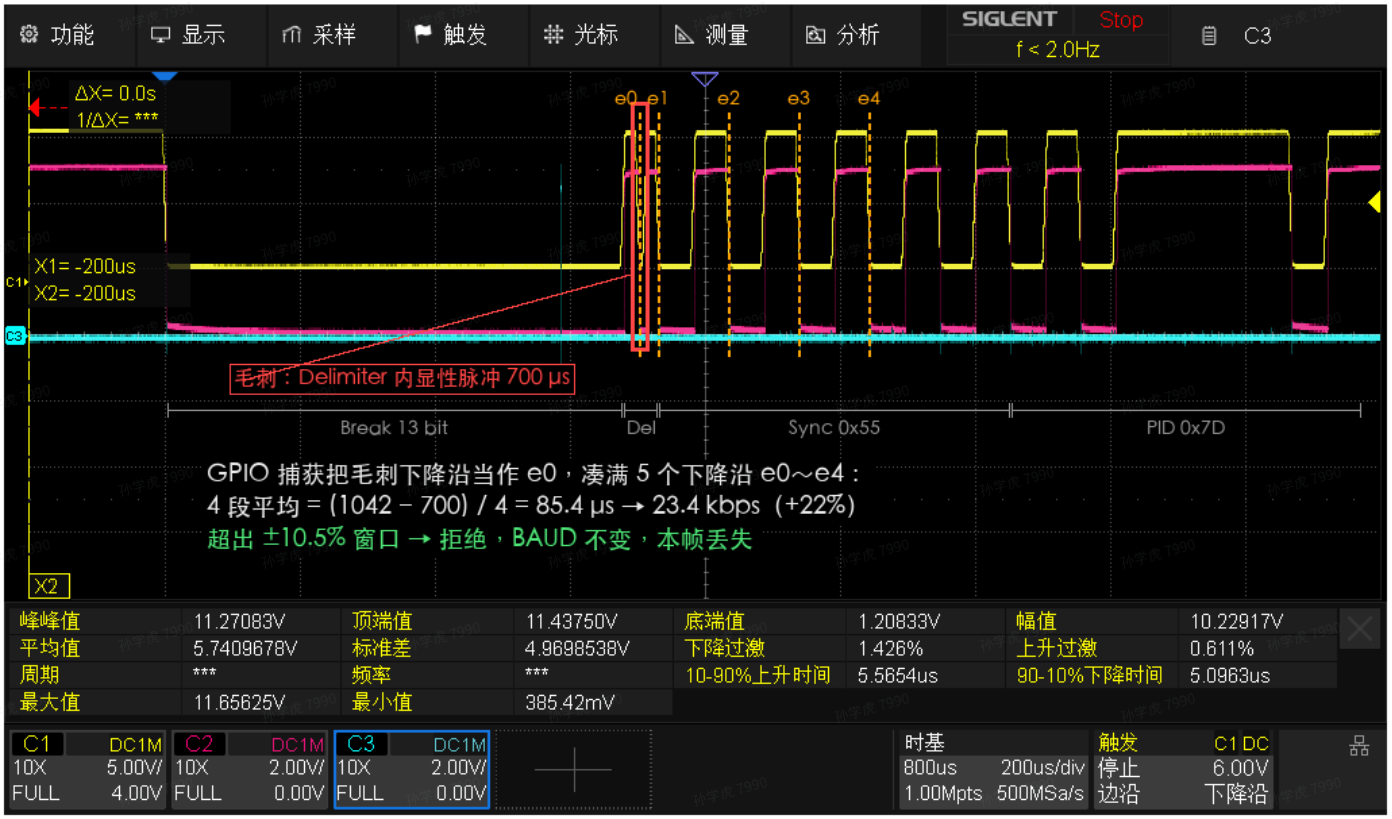
```
 - 自锁终态(3/3): BAUD=21127 s_isAutobaudMode=0 State=IDLE LBKDIE=1
──────────────────────────────────────────────────────────────────────────────
```

实测波形实证（图 1～图 3）

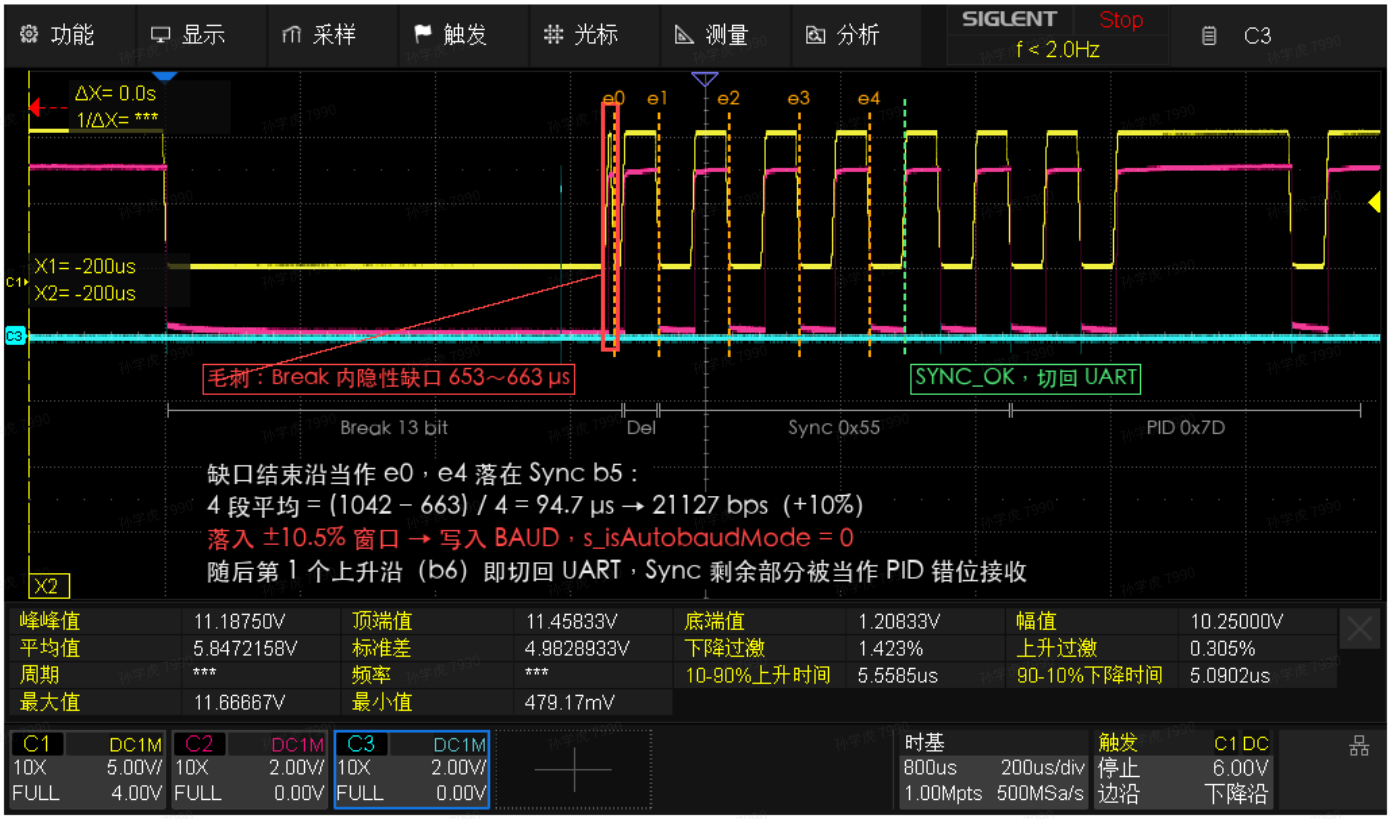
C1 黄 = LIN 总线，C2 粉 = LE1 RX，C3 青 = PTC14 标记；触发 C1 下降沿 6 V，Break 起点在左起第 1 格。图中标注：**红框** = 注入的毛刺；**橙色虚线 e0～e4** = GPIO 捕获计入的 5 个下降沿；**绿色虚线** = 判 LIN_SYNC_OK 并切回 UART 的位置；底部标尺为帧头各段的理论位置。

图 1（错误波特率被接受）：Break 检出后打开 GPIO 捕获，把遇到的前 5 个下降沿记为 e0～e4，只用 4 段间隔的平均值判断波特率。(a) Delimiter 内毛刺让平均值偏 +22%，越界拒绝；(b)(c) 同样 10 µs 的毛刺落在 Break 缺口或 Sync bit6 时，平均值偏 +10%，落入 ±10.5% 窗口，21127 bps 被写入 BAUD，并在 Sync 尚未发完时切回 UART，本帧丢失。

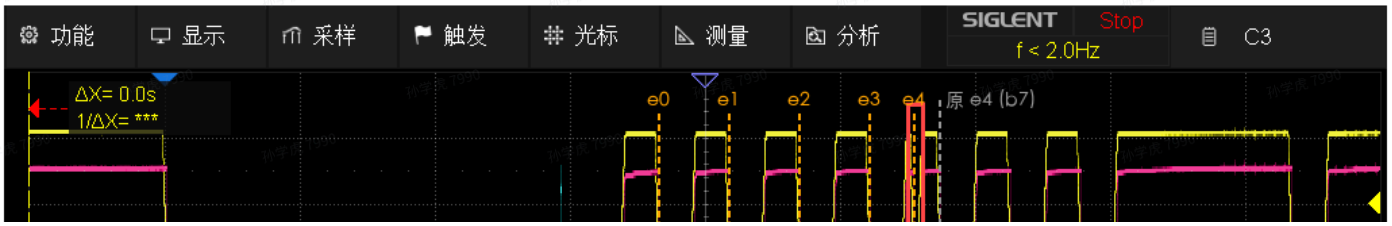
(a) 对照：Delimiter 内毛刺（700 µs）→ 平均 +22%，越界拒绝



(b) Break 内隐性缺口（653～663 µs）→ 21127 bps 被接受



(c) Sync bit6 内显性毛刺（1107 µs）→ 第 5 沿提前，21127 bps 被接受



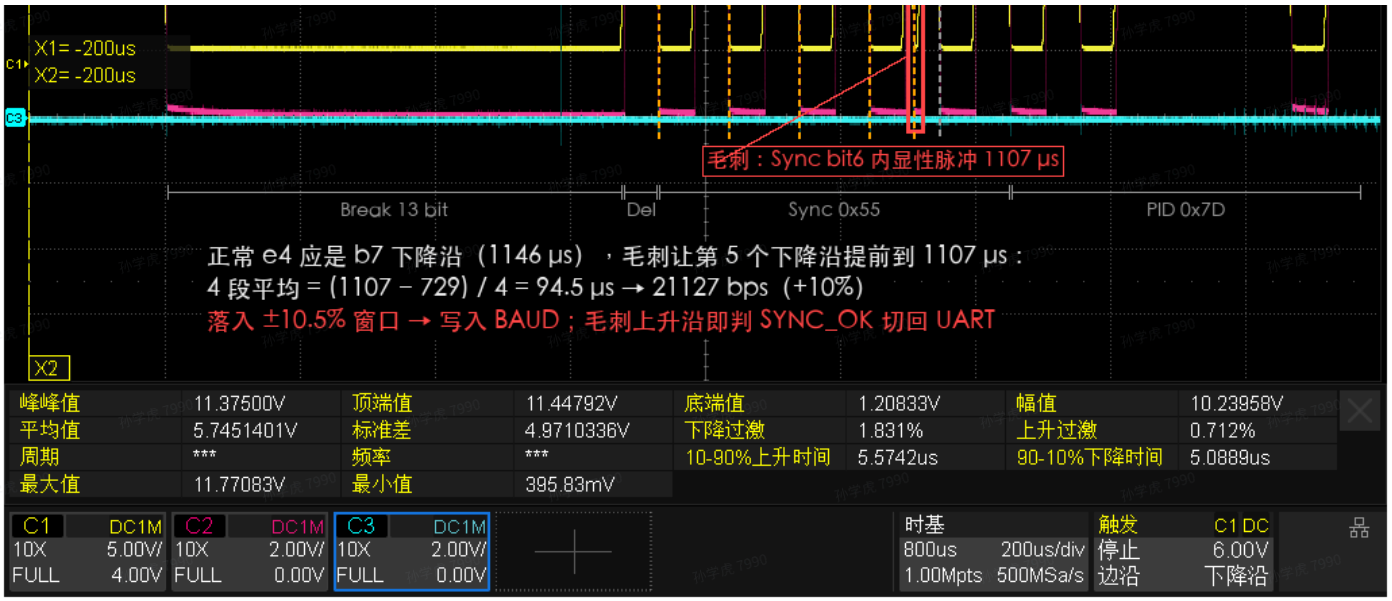
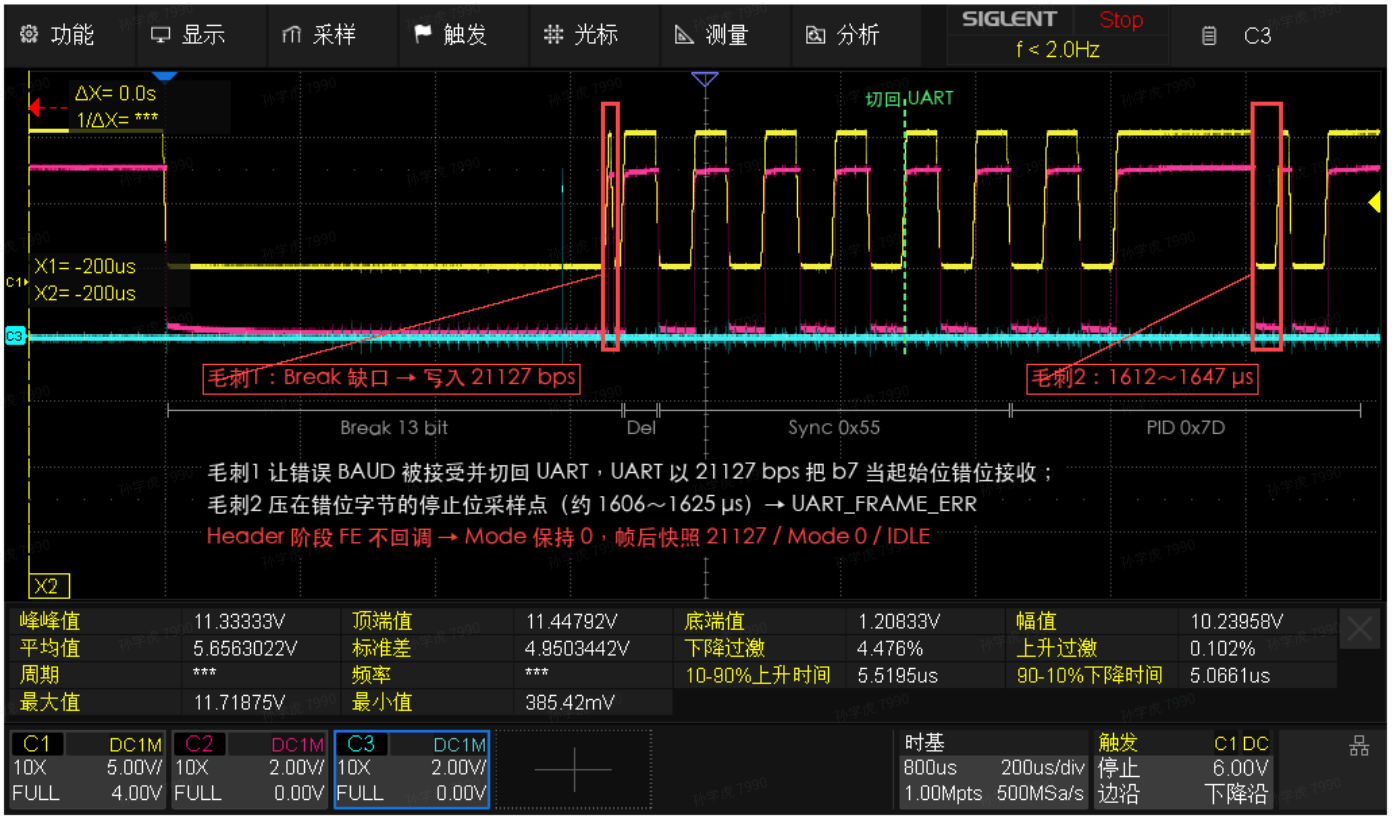
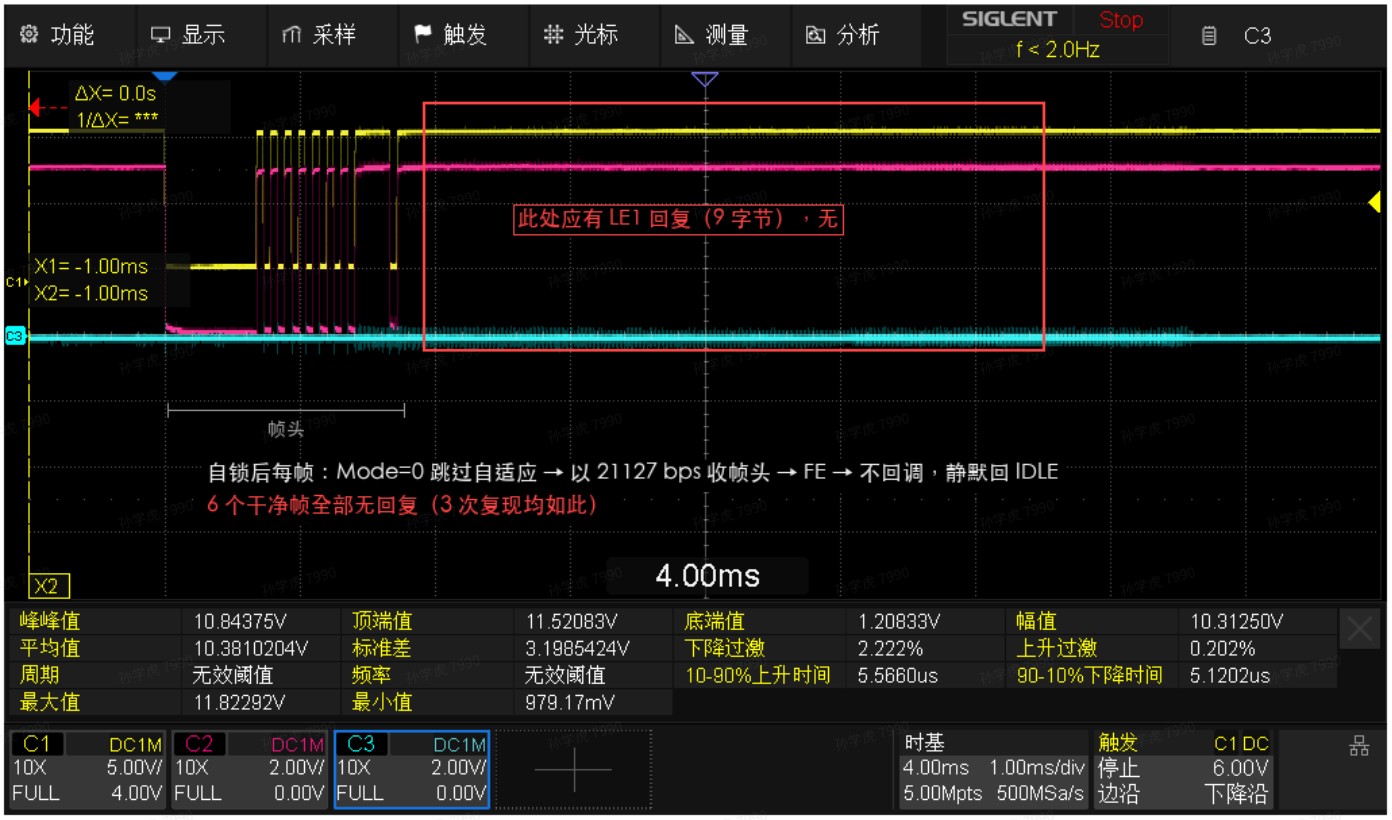


图 2（静默 FE 与自锁）：(a) 在 Break 缺口之外，再加一个落在错位字节停止位采样点（1612～1647 µs）的毛刺，该帧以静默 FE 结束，快照 21127 / Mode 0 / IDLE；(b)(c) 之后的干净帧帧头完好，但从机全部无回复（3 次复现，每次 6 帧）。

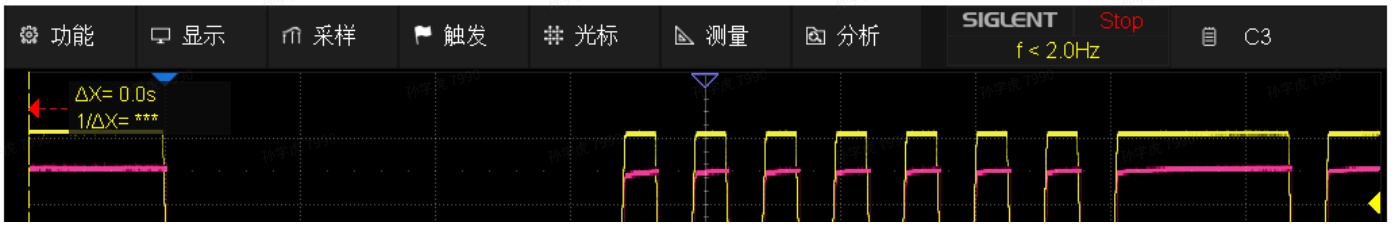
(a) 双毛刺帧：错误 BAUD 被接受 + 停止位处第二毛刺 → 静默 FE

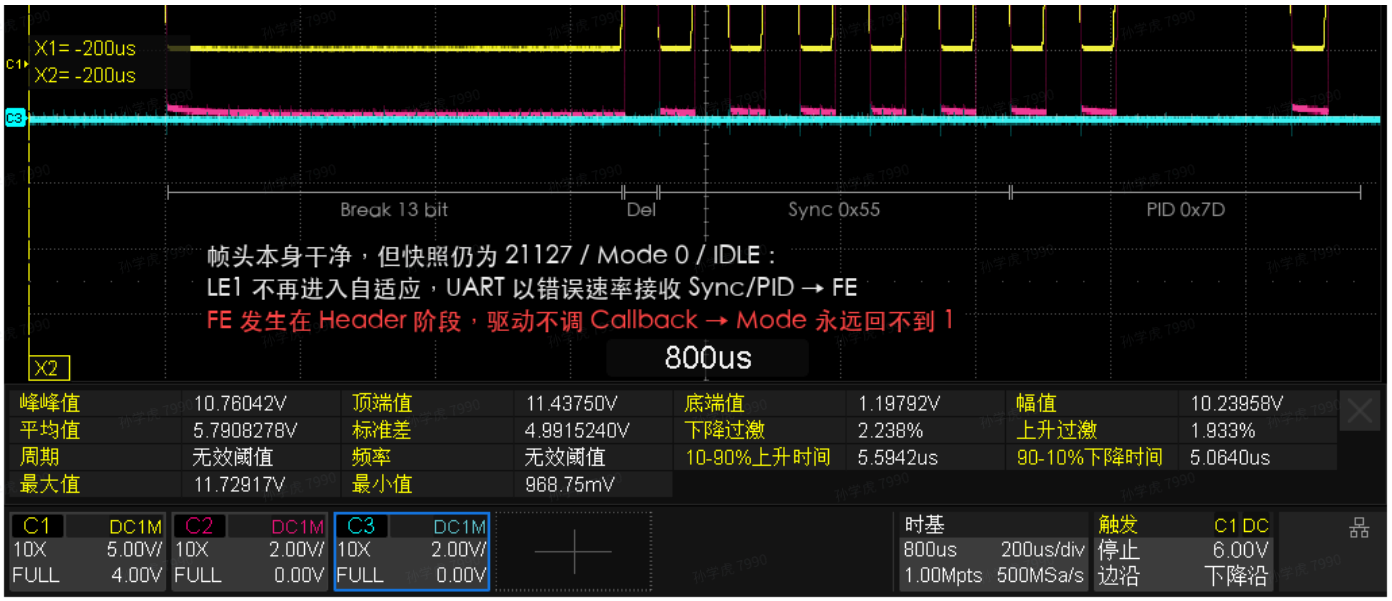


(b) 自锁后的干净帧（1 ms/div）：只有帧头，从机无回复



(c) 同一干净帧头放大（200 µs/div）

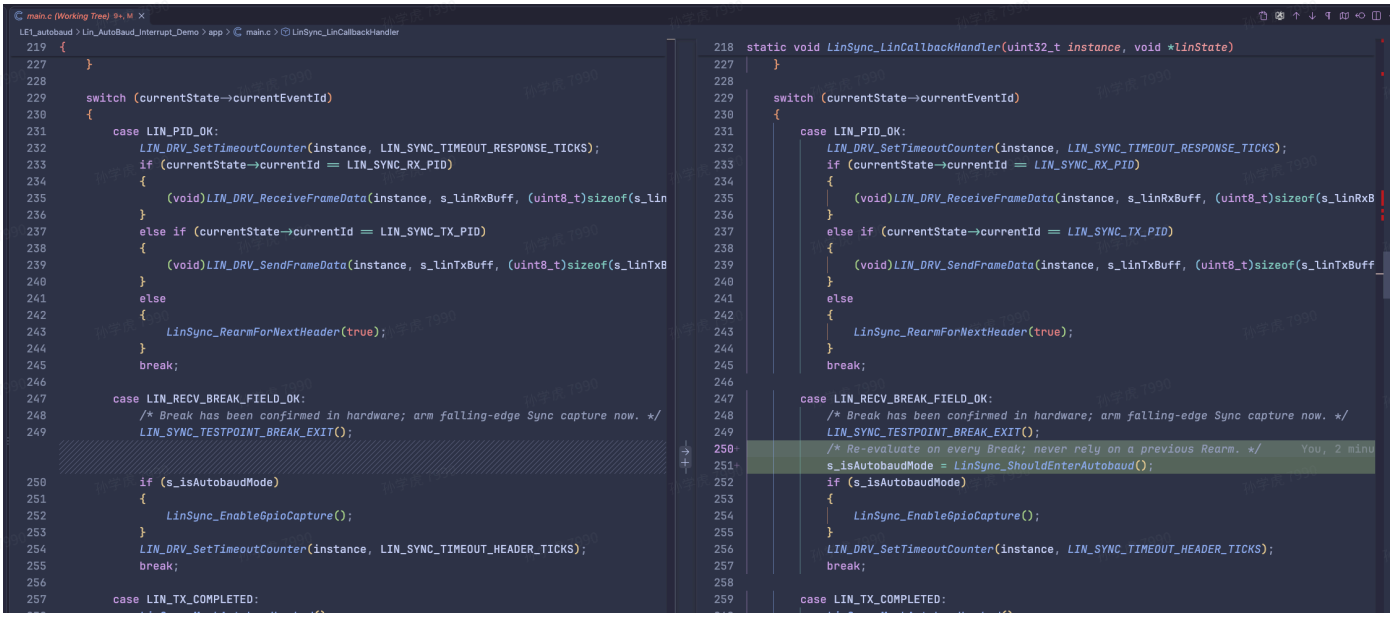




修复方式

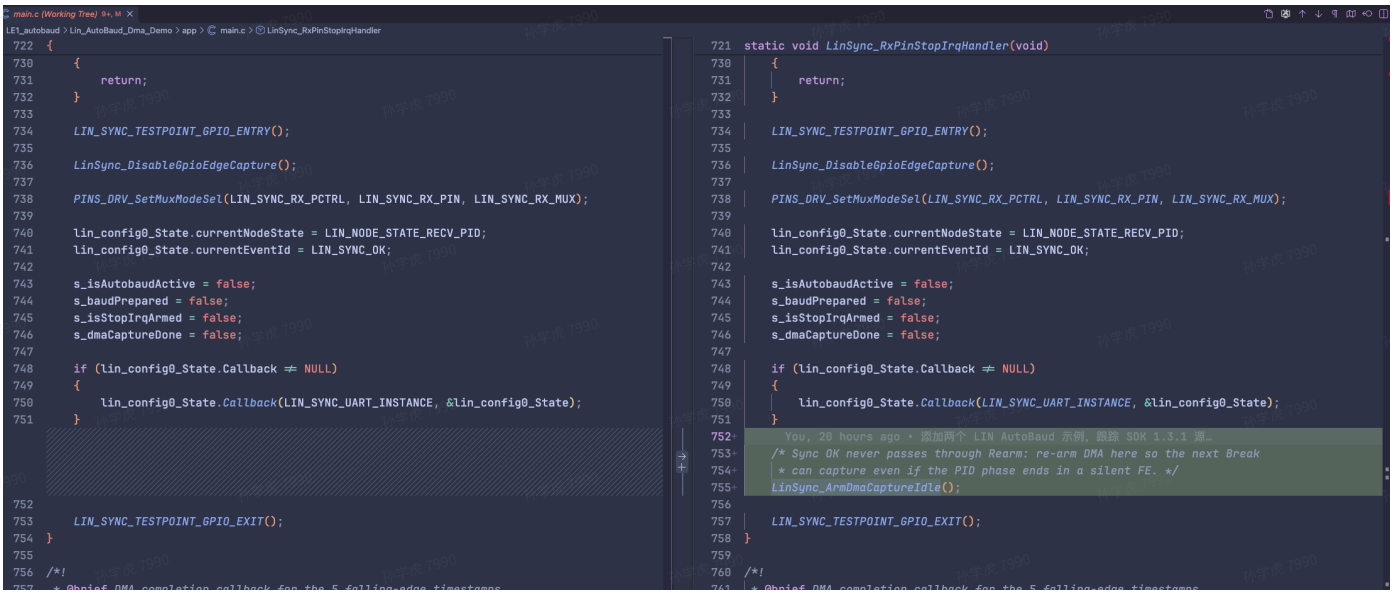
修复只改两个 Demo 的 `main.c`（USER CODE 区，共 6 行），**不修改 SDK 驱动**：

- **① 中断版每次 Break 重算入口（对应次要矛盾）**：在 `LIN_RECV_BREAK_FIELD_OK` 分支先执行 `s_isAutobaudMode = LinSync_ShouldEnterAutobaud()`，不再依赖上一帧回调里的 Rearm。Header 阶段 FE 被驱动吞掉后，下一帧 Break 仍会重测 Sync 并覆盖错误 BAUD，**最坏丢 1 帧**。



```
case LIN_RECV_BREAK_FIELD_OK:
    /* Break has been confirmed in hardware; arm falling-edge Sync capture now. */
    LIN_SYNC_TESTPOINT_BREAK_EXIT();
    /* Re-evaluate on every Break; never rely on a previous Rearm. */
    s_isAutobaudMode = LinSync_ShouldEnterAutobaud();
    if (s_isAutobaudMode)
    {
        LinSync_EnableGpioCapture();
    }
    LIN_DRV_SetTimeoutCounter(instance, LIN_SYNC_TIMEOUT_HEADER_TICKS);
    break;
```

- **② DMA 版 Sync OK 后立即重装 DMA（对应 DMA 同类隐患）**：在 `LinSync_RxPinStopIrqHandler()` 回调 `LIN_SYNC_OK` 之后调用 `LinSync_ArmDmaCaptureIdle()`。DMA 版**不需要错误波特率**：PID 字节被噪声打出一次 FE 就会漏掉 Rearm，下一帧 RX 滞留 GPIO 且 `LBKDIE=0`。SDK 1.3.1 的 `LIN_UART_DRV_TimeoutService()` 不处理 Header 阶段，没有超时兜底，因此仅靠 ① 不够。此处 GPIO 请求已关闭、UART 已接管 RX，重装安全，也不占用 Delimiter 到 Sync 起始沿的窗口。



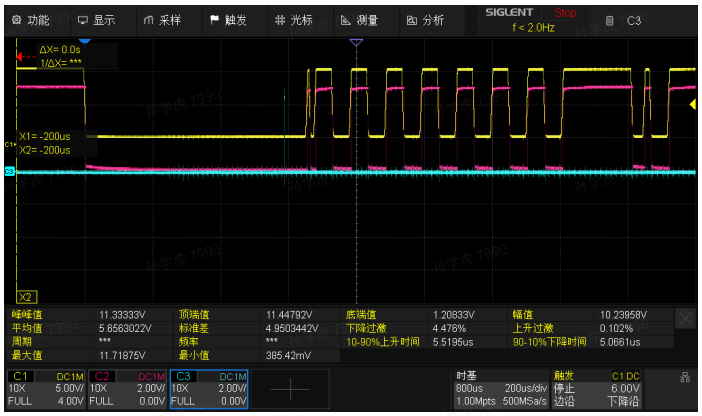
测试结果

中断版修复后，修复前能稳定复现永久失联的激励都降为**最多丢 1 帧、下一帧恢复**。双毛刺帧之后的状态（BAUD 21127、Mode 0、IDLE）与修复前完全相同，说明自锁的触发条件已到达；下一个 Break 重算 `s_isAutobaudMode`，重新测量 Sync 并覆盖了错误 BAUD。

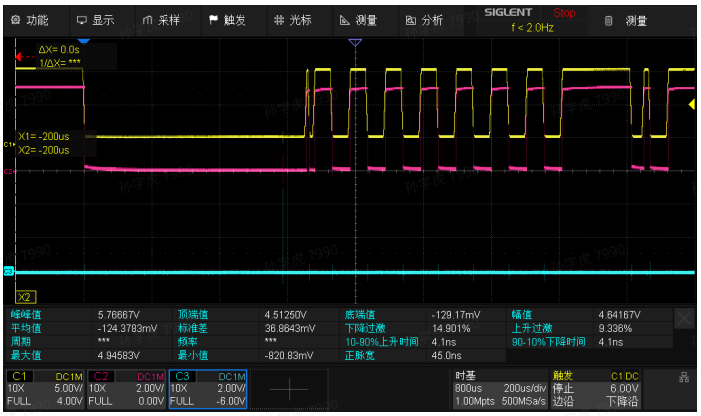
测试	修复前	修复后
双毛刺端到端（方法一）：Break 缺口在 653 µs，第二个毛刺在 1612 µs、宽 35 µs	3/3 永久失联	0/5 失联；6 个恢复帧全部应答，第 1 个干净帧即恢复为 19231 bps / Mode 1
错误 BAUD 注入：J-Link 写 UART0->BAUD.SBR=70～87 并置 Mode=0，每个 SBR 发 5 个干净帧	SBR 70/71（+11.6%/+10.0%）自锁；72 丢 3 帧；73、83～87 丢 1 帧	90/90 帧应答，0 丢帧；文档中的 -8.1%（SBR 85）也不再丢帧
单毛刺全帧头扫描：440～1240 µs，步进 5 µs，93 个有效点	无永久失联	无永久失联，每个点都在第 1 个干净帧恢复

实测波形对比（左：修复前，右：修复后；C1=LIN 总线，C2=LE1 RX）：

① 双毛刺帧（Break 缺口 653 µs + 1612 µs 处 35 µs 毛刺）

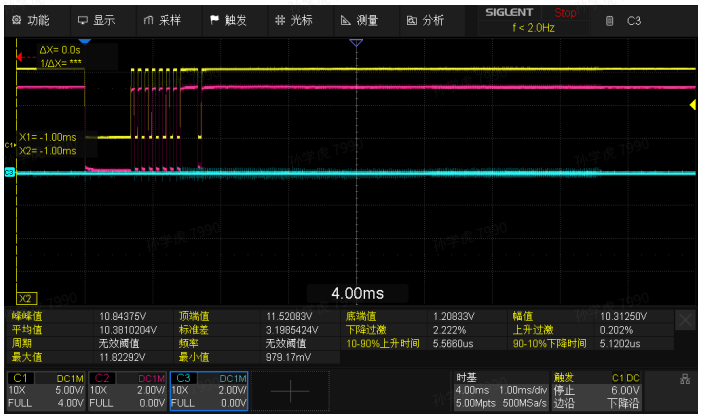


修复前：帧后 BAUD=21127、Mode=0

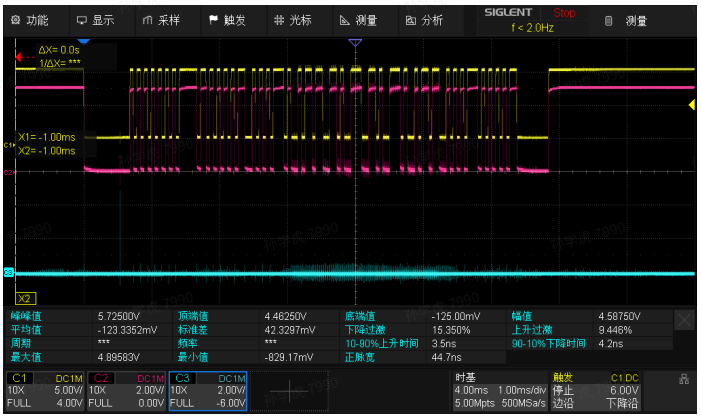


修复后：帧后同样 BAUD=21127、Mode=0（触发条件已到达）

② 双毛刺之后的干净帧

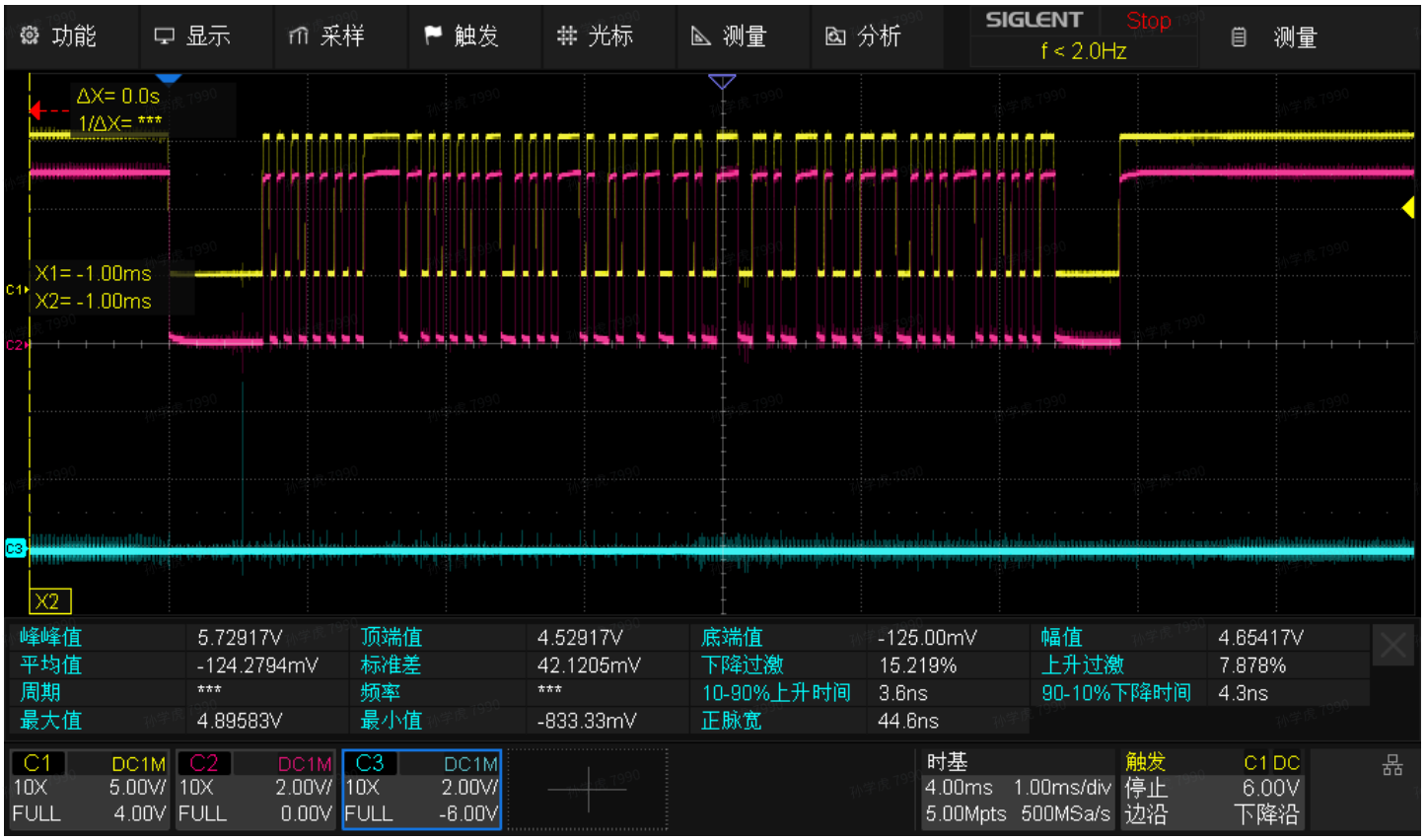


修复前：从机无应答，持续自锁



修复后：第 1 个干净帧即应答，BAUD=19231、Mode=1

③ 修复后：注入 BAUD=21127 + Mode=0



修复前该状态（SBR 71）5/5 帧失联；修复后第 1 个干净帧即正常应答