

MCAL应用_Lin模块配置及实例(一)

版本

Config Tool Version: 2.0.2

ME0 MCAL version: 1.3.2

前言

1. Lin一般依赖Mcu、Platform和Port模块，Mcu模块里面需要配置外设时钟，Platform里面可以配置外设中断，Port配置使用到的端口资源。
2. Port、Mcu和Platform的配置参考《MCAL应用 - 工程建立及工具通用设置》和《MCAL应用 - Mcu模块配置及实例》文档，Port的配置参考《MCAL应用 - Port模块配置及实例》。
3. 配置中包含AutoSar规范标准选项和云途自定义选项，左边树选框中粗体字为AutoSar标准选项，细体字为云途自定义选项。

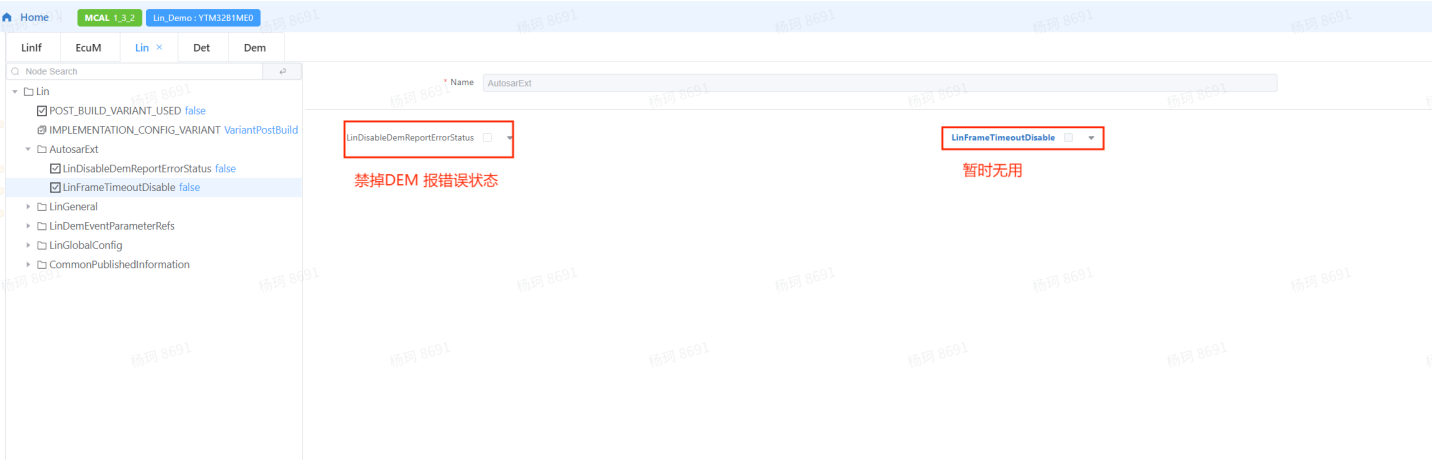
1. Lin配置介绍

1.1 如何配置LIN模块

1.1.1 框架说明



1.1.2 AutoSarExt



1.1.3 LinGeneral



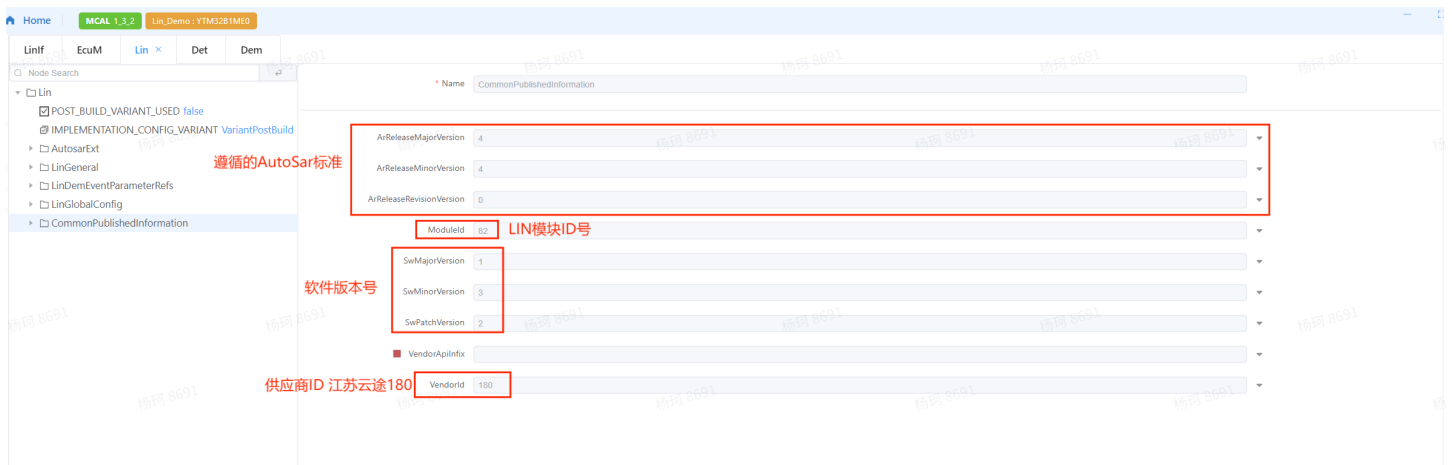
1.1.4 LinDemEventParameterRefs



1.1.5 LinGlobalConfig



1.1.6 CommonPublishedInformation



2. LIN模块应用实例

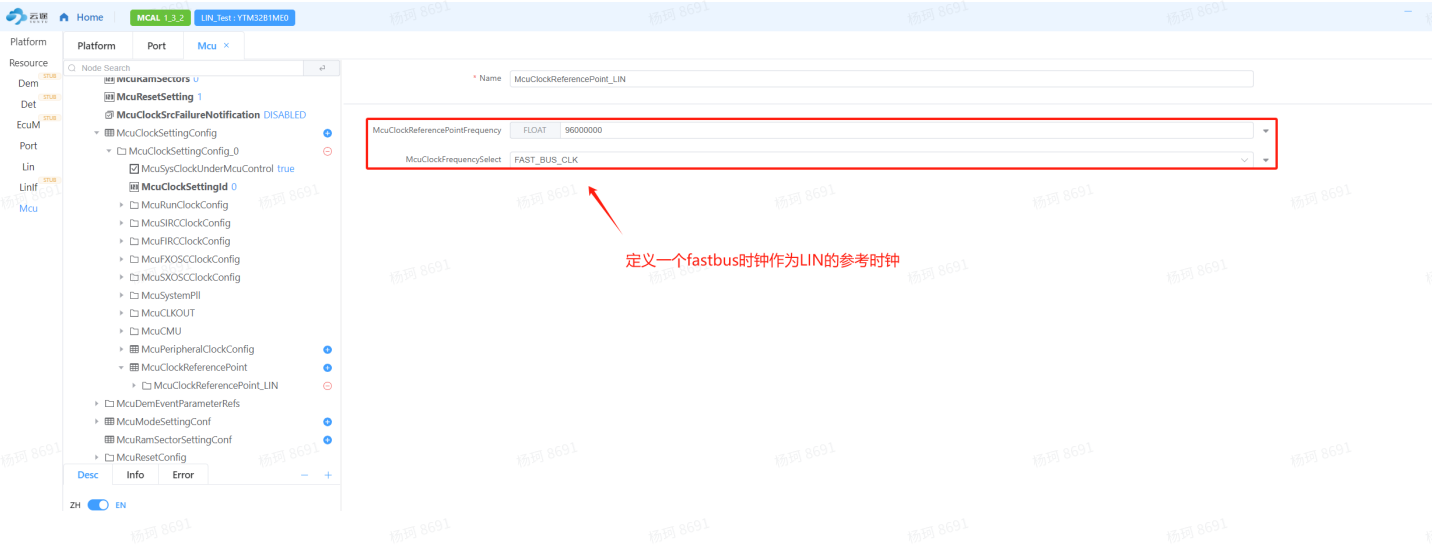
2.1 应用场景

- 基于ME0 Demo板，使用LIN0和LIN1
- 配置波特率为19200
- LIN1做从机，上位机发送 ID为0x31，回复数据为{0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x99, 0x10}
- LIN0做主机，主发帧头+Response ID为0x1F 发送数据为{0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x01}
- 通过LIN上位机查看通信。

2.2 配置步骤

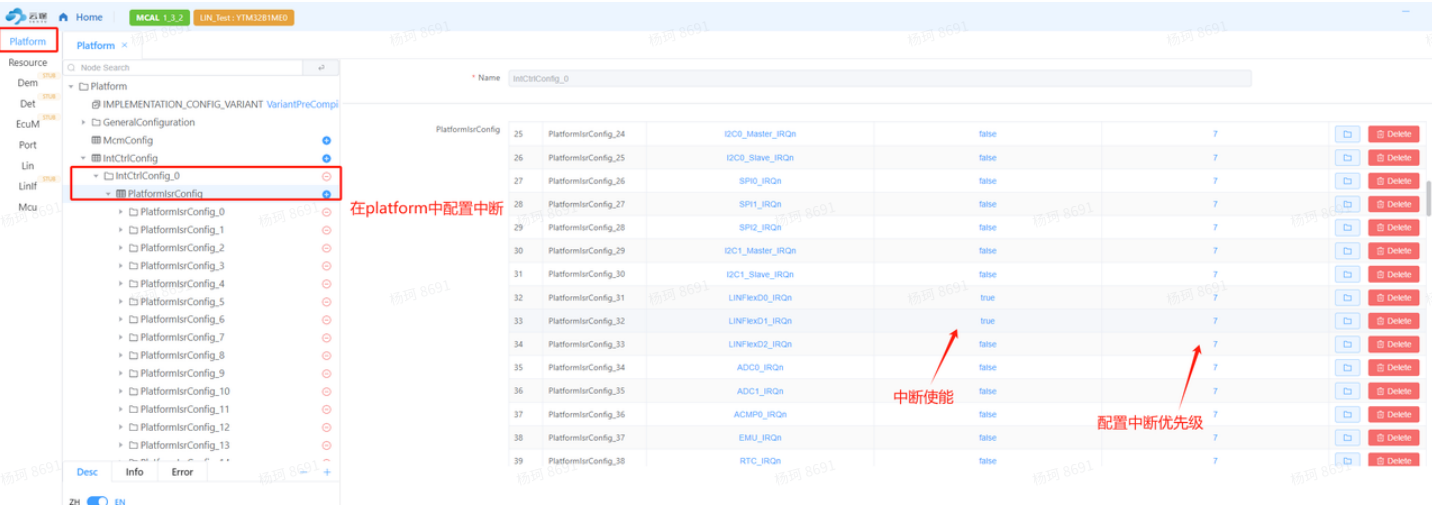
2.2.1 时钟配置

在MCU模块中配置LIN的reference clock。



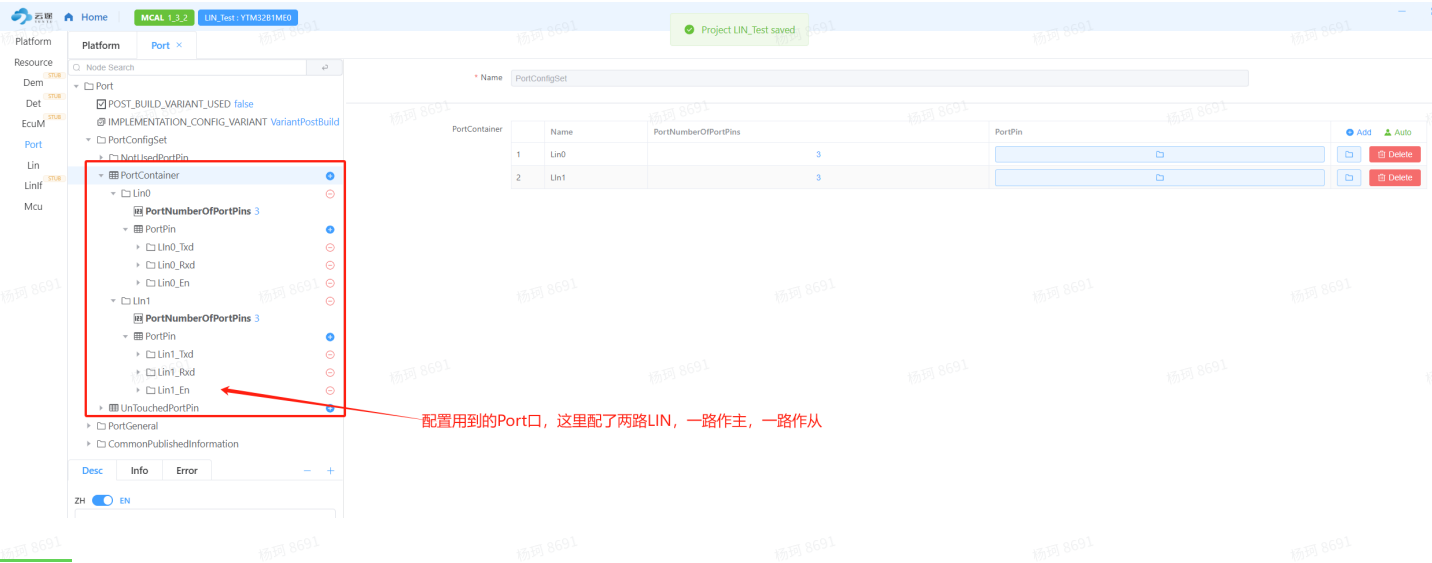
2.2.2 中断配置

根据需求选择是否使能中断，如若需要中断，则在Platform中打开中断，若使用了OS，则需要在OS中配置LIN的外设中断。



2.2.3 Port配置

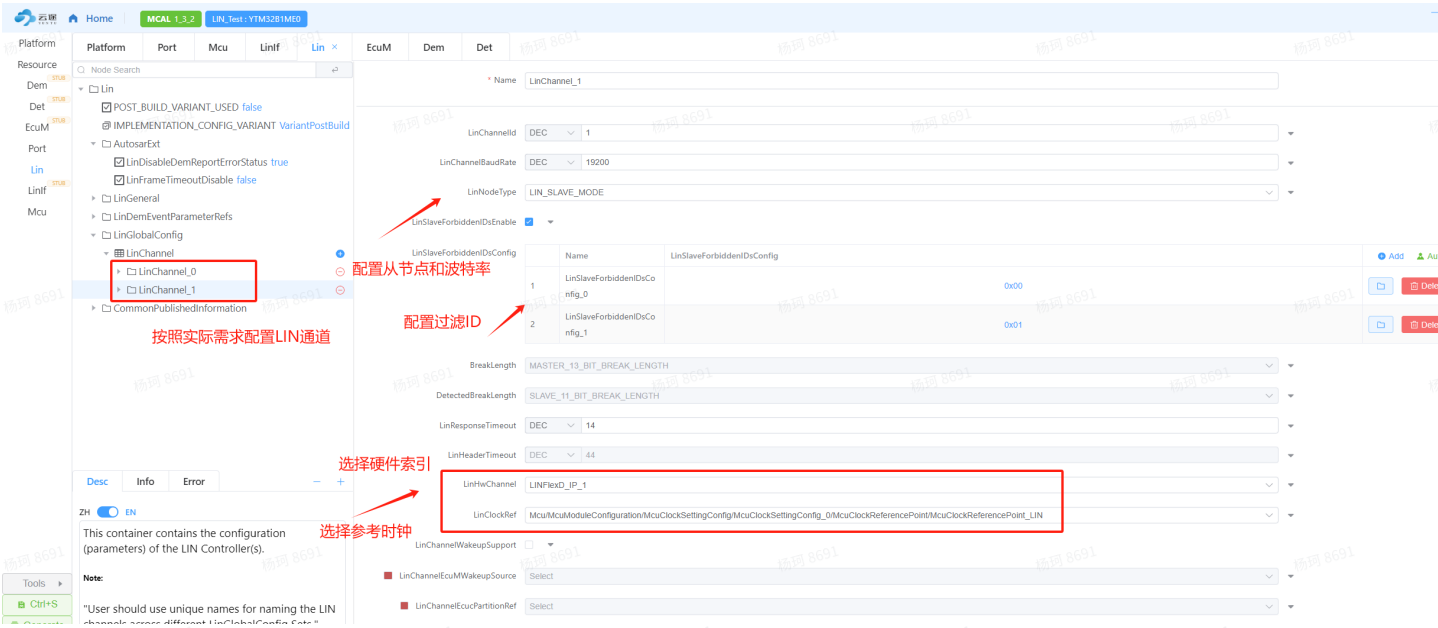
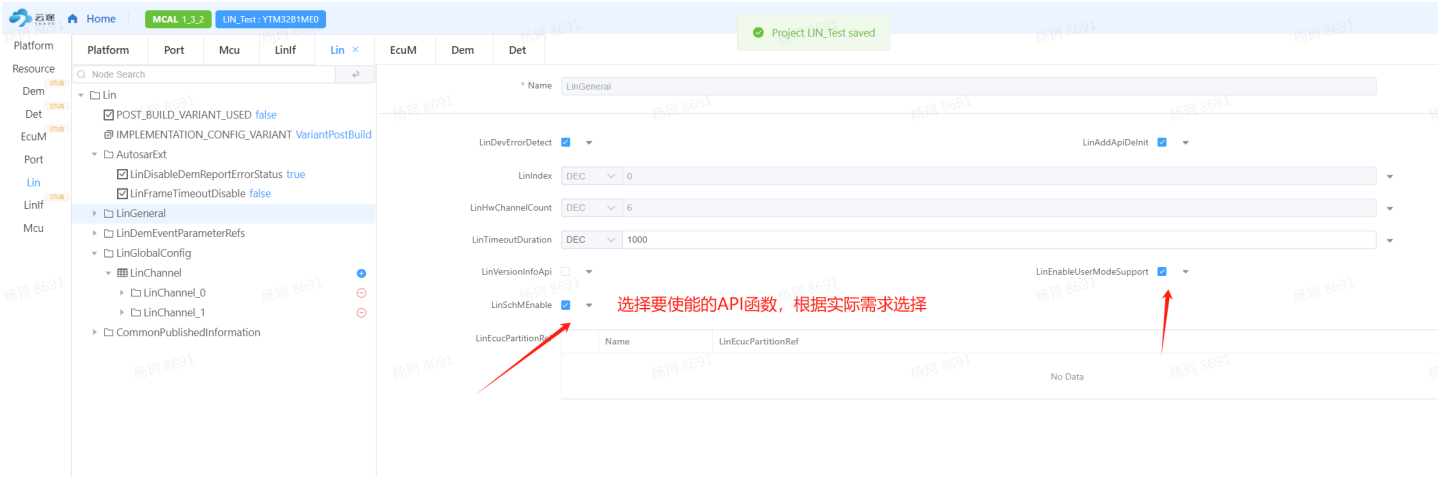
这里配置了两路LIN



Tips:

LIN RX外部电路没有上拉的时候， 需要内部上拉，

2.2.4 配置LIN模块



2.3 API说明

```

298 #endif
299 /* USER CODE END 1 */
300 Board_Init();
301 /* USER CODE BEGIN 2 */
302
303 Lin_Wakeup(LinConf_LinChannel_LinChannel_0);
304 Lin_WakeupInternal(LinConf_LinChannel_LinChannel_1);
305 /* USER CODE END 2 */
306
307 /* Infinite loop */
308 /* USER CODE BEGIN WHILE */
309 while (1)
310 {
311     #if (LIN_MASTER_API_SUPPORT == STD_ON)
312     time++;
313     if ((time > 150000U) || (sign_BusIdle == TRUE))
314     {
315         time = 0;
316         Lin_CheckWakeup(LinConf_LinChannel_LinChannel_0); 唤醒相应LIN通道
317         Lin_SendFrame (LinConf_LinChannel_LinChannel_0, linTxPduMatrix + dataLoop);
318     }
319     if((time % 1000U) == 0U)
320     {
321         LinStatus = Lin_GetStatus(LinConf_LinChannel_LinChannel_0, &lin_rxData);
322         if((LinStatus == LIN_TX_OK) || (LinStatus == LIN_TX_HEADER_ERROR) || (LinStatus == LIN_TX_ERROR))
323         {
324             Lin_SendFrame (LinConf_LinChannel_LinChannel_0, linRxPduMatrix + dataLoop); 发送LIN帧
325             dataLoop++;
326             if(dataLoop >= 10)
327             {
328                 dataLoop = 0;
329             }
330         }
331         else if(LinStatus == LIN_RX_OK)
332         {
333             //sign_BusIdle = TRUE;
334             for(uint8 x = 0; x < 8; x++)
335             {
336                 buffer0[x] = *(lin_rxData + x);
337             }
338             /* here process the data */
339         }
340     }
341 }
342 /* USER CODE END WHILE */
343 /* USER CODE BEGIN 3 */

```

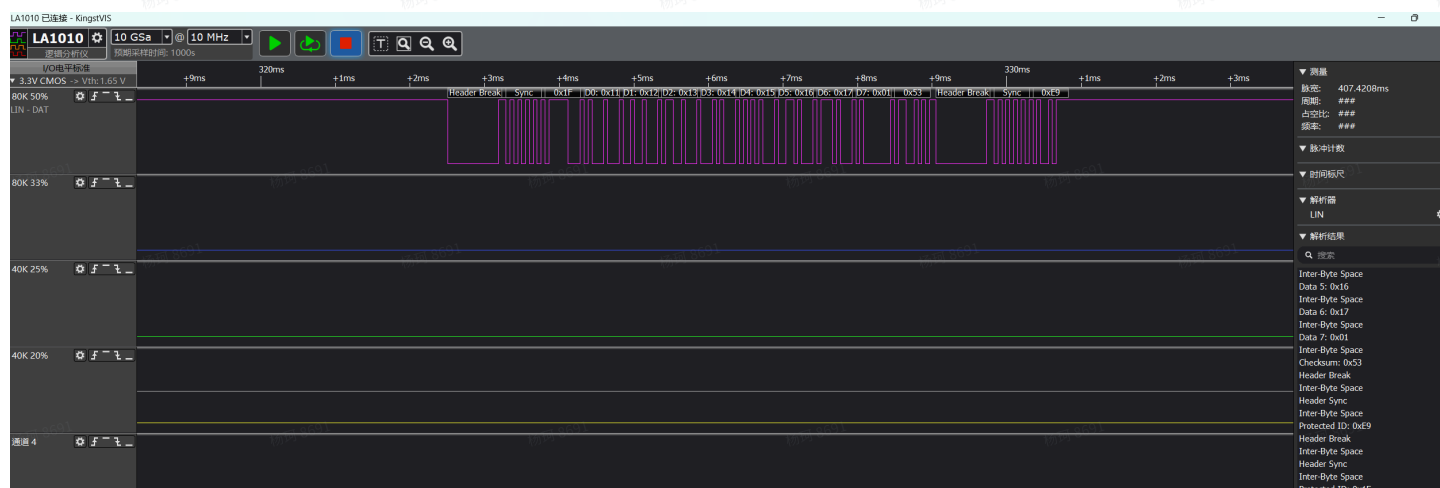
2.4 现象

LIN0做主机

主发帧头+Response ID为0x1F

发送数据为{0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x01}

主发帧头 ID为0x53



LIN1做从机

上位机发送 ID为0x31

回复数据为{0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x99, 0x10}

ZM-USB_LIN-V1.6.0_r

设置

通信与状态

通信口: ZMELEC:SN: F400F300915333393737314 关闭通信口 刷新通信口 设备连接成功

基本设置

过滤设置

高级发送窗口

从机写配置

模式: 主机接收 ID(PID): 49/0x31 (0xB1) 校验: 增强 数据: 0 0 0 0 0 0 0 0 校验: 4E 周期(ms): 1000

模式: ID(PID): 校验: 数据: 11 22 33 44 55 66 77 88 校验: 0 周期(ms): 1000

模式: ID(PID): 校验: 数据: 11 22 33 44 55 66 77 88 校验: 0 周期(ms): 1000

模式: ID(PID): 校验: 数据: 11 22 33 44 55 66 77 88 校验: 0 周期(ms): 1000

模式: ID(PID): 校验: 数据: 11 22 33 44 55 66 77 88 校验: 0 周期(ms): 1000

数据栏

序号	传输方向	时间标识	ID值(PID)	数据长度	数据	校验	状态
16	主接收(循环-1000ms)	23-11-29 17:55:01:323	49/0x31 (B1)	8	10 10 10 10 10 10 99 10	44	增强校验成功
17	主接收(循环-1000ms)	23-11-29 17:55:02:323	49/0x31 (B1)	8	10 10 10 10 10 10 99 10	44	增强校验成功
18	主接收(循环-1000ms)	23-11-29 17:55:03:323	49/0x31 (B1)	8	10 10 10 10 10 10 99 10	44	增强校验成功
19	主接收(循环-1000ms)	23-11-29 17:55:04:324	49/0x31 (B1)	8	10 10 10 10 10 10 99 10	44	增强校验成功
20	主接收(循环-1000ms)	23-11-29 17:55:05:323	49/0x31 (B1)	8	10 10 10 10 10 10 99 10	44	增强校验成功
21	主接收(循环-1000ms)	23-11-29 17:55:06:325	49/0x31 (B1)	8	10 10 10 10 10 10 99 10	44	增强校验成功
22	主接收(循环-1000ms)	23-11-29 17:55:07:324	49/0x31 (B1)	8	10 10 10 10 10 10 99 10	44	增强校验成功
23	主接收(循环-1000ms)	23-11-29 17:55:08:323	49/0x31 (B1)	8	10 10 10 10 10 10 99 10	44	增强校验成功

文档历史

版本号	日期	修订记录
V1.0	2023.11.20	初始版本