TOSUN同星


# GW2112 产品手册

版本：V1.0 | 中文

tosunai.com

## 为什么需要 CAN(FD)中继、报文转换？

CAN 网络可能很复杂，尤其是在大型车辆或工业系统中。中继可以将网络分成多个段，以减少总线负载，提高通信稳定性和可靠性。CAN 有不同的版本，如经典 CAN 和 CAN FD（Flexible Data-rate）。经典 CAN 和 CAN FD 之间的报文格式和数据传输速率有所不同。因此，当网络中存在不同协议版本的设备时，就需要进行报文转换以确保兼容性和数据的正确传输。

CAN FD 相较于经典 CAN 提供了更高的数据传输速率和更大的数据负载。如果需要在同一网络中混合使用经典 CAN 和 CAN FD 设备，中继和报文转换可以帮助在不同带宽要求的设备之间传递数据。在长距离或高干扰环境下，CAN 信号可能会受到影响。中继可以帮助增强信号，保证数据传输的完整性和可靠性。通过 CAN 中继，可以将 CAN 网络扩展到更大的范围，支持更多的设备和更复杂的系统架构。在现代系统中，可能会有不同厂家和型号的设备，这些设备可能使用不同的 CAN 标准或配置。中继和报文转换可以帮助这些不同设备之间无缝通信。

总结来说，CAN 中继和报文转换对于维护系统的稳定性、兼容性和扩展性是非常重要的。它们帮助解决了不同设备、不同协议版本和网络拓扑问题，提高了 CAN 网络的整体性能和可靠性。同星智能推出的 CAN(FD)中继、CAN(FD)报文转换设备 — GW2112 离线网关，正是用于 CAN(FD)中继、报文转换的核心产品。

## GW2112 离线网关能做什么？

- **协议转换**

  CAN/CANFD 网关设备能够实现不同 CAN 协议之间的转换；

- **数据转发**

  CAN 总线上各类信号的转发与处理；

- **报文过滤**

  具有 ID 过滤转换功能，降低 CAN 总线的负荷，去除干扰数据；

- **速率和数据长度适配**

  在 CAN 与 CAN FD 共存的网络中，网关设备可以处理速率和数据长度不同的冲突，例如通过可编程 CAN FD 路由器将 CAN FD 的 64 字节数据拆包后转发给传统 CAN 设备；

- **中继扩容**

  网关设备可以增加负载节点和延长通信距离，实现网络中继扩容的功能。

- ...

目录

# 1.关于手册

## 1.1 免责声明

本文档提供的信息仅供参考，同星智能不构成任何形式的保证或承诺。同星智能保留对文档内容和数据的修改权利，恕不另行通知。同星智能对文档的正确性或因使用文档而产生的损害不承担任何责任。同星智能非常感激您指出错误或提出改进建议，以便我们能够在未来为您提供更加高效的产品。

## 1.2 版权信息

# 2.GW2112

## 2.1 产品概述

GW2112 是同星智能推出的一款 CAN(FD)离线网关设备。它能够增加总线的负载能力和延长通信距离，匹配不同波特率的 CAN(FD)网络，同时可以支持 CAN 和 CANFD 网络的转换。

GW2112 上设有两路 CAN（FD）通道，当 CAN1(CAN2)收到 CAN(FD)报文时从另一路 CAN 通道转发出去。GW2112 提供了多种转换规则供用户选择，用户可以通过配套的软件自由设定转换规则并且配置完的转换规则可以在 GW2112 中持久保存不会因为断电丢失，每次开机都会读取上次储存的转换规则，同时用户配置的转换规则可以转化为配置文件储存在本地，以便直接进行配置导入。

## 2.2 功能特征

- ✓ CAN 通道 DC2500V 隔离
- ✓ CAN 通道波特率 125Kbps—1Mbps 可调；
- ✓ CANFD 通道波特率 125Kbps—8Mbps 可调；
- ✓ 内置 120 欧终端电阻可软件配置；
- ✓ CAN/CANFD 协议转换；
- ✓ CAN/CANFD 报文过滤；
- ✓ CAN/CANFD 报文数据路由；
- ✓ 中继扩容；
- ✓ 支持自定义功能；
- ✓ 内置配置加密，提供多种加密算法。

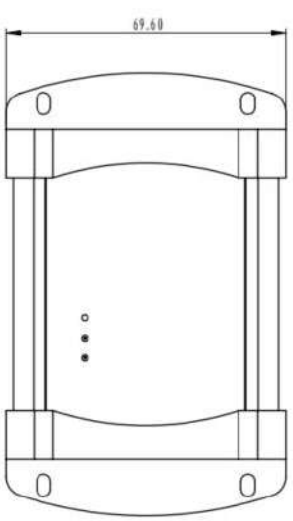
## 2.3 技术参数

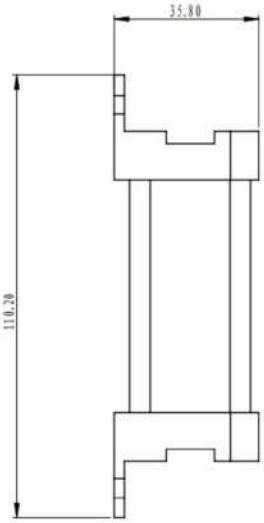
| | |
|---|---|
| 通道 | 2 * CANFD |
| PC 接口 | USB2.0 |
| 驱动 | Windows 系统免驱设计，具备极佳的系统兼容性 |
| 软件 | TSMaster |
| CAN | 支持 CAN2.0A、B 协议，符合 ISO11898-1 规范，波特率 125Kbps-1Mbps |
| CANFD | 支持 ISO 和非 ISO 标准的 CAN FD，波特率 125Kbps-8Mbps |
| 终端电阻 | 内置 120 欧终端电阻可软件配置 |
| 继电器类型 | 磁保持继电器 |
| 报文转发能力 | 20000 帧每秒 |
| 转发延迟 | < 0.5ms |
| 隔离 | CAN 通道 DC2500V 隔离 |
| 供电 | USB 供电，外部 DC 供电(9-36V) |
| 功耗 | 1W |
| 外壳材质 | 金属 |
| 尺寸 | 约 110*70*36mm |
| 重量 | 约 150g（无包装）/约 368g（含包装） |
| 工作温度 | -40℃～80℃ |
| 工作湿度 | 10% ～ 90% （无凝露） |
| 工作环境 | 远离腐蚀性气体 |

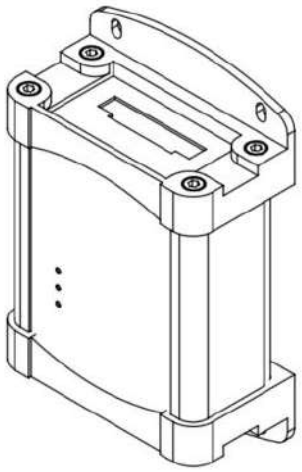
## 2.4 电气参数

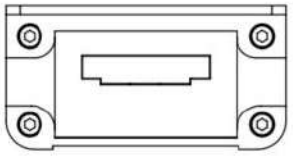
| 参数 | | 测试条件 | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|---|
| 工作电压 | USB 供电 | CAN 转发 | -- | 5 | -- | V |
| | 外部DC供电 | CAN 转发 | 9 | 12 | 36 | V |
| 工作电流 | USB 供电 | CAN 转发 | -- | 0.16 | -- | A |
| | 外部DC供电 | CAN 转发 | -- | 0.06 | -- | A |
| 功耗 | 外部DC供电 | CAN 转发 | -- | 1 | -- | W |
| CAN 接口 | 总线引脚耐压 | CANH、CAHL | -58 | -- | +58 | V |
| | 隔离耐压 | 漏电流小于 1mA | 2500 | -- | -- | VDC |

## 2.5 机械尺寸





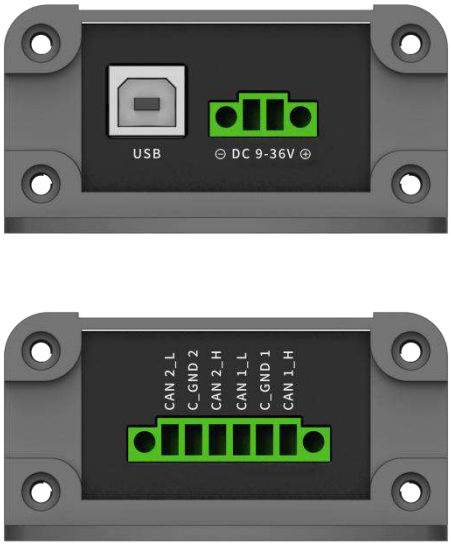
| 材质 Material | | 一般公差 General Tolerances IT12 | 表面处理 Surface Treatment | 工艺tech. |
|---|---|---|---|---|
| A3 | | 设计draw. | 审核audi. | 批准appr. |
| SCALE:1:1 | SHEET 1 OF 1 | | | |
| TOSUN | | | | 重量(g) Weight |
| 上海同星智能科技有限公司 SHANGHAI TONGXING INTELLIGENT TECHNOLOGY CO.,LTD | | GW2112尺寸图 | | 版本 Rev. 00 |

## 2.6 发货清单

✓ GW2112 主设备


✓ USB 连接线

## 2.7 硬件接口说明




- USB2.0 接口；
- DC 供电口；
- 6PIN 凤凰端子：

| 6PIN 凤凰端子 | 引脚 | 定义 |
|---|---|---|
| CANFD 1/2 | PIN1 | CAN2_L |
| | PIN2 | C_GND2 |
| | PIN3 | CAN2_H |
| | PIN4 | CAN1_L |
| | PIN5 | C_GND1 |
| | PIN6 | CAN1_H |

## 2.8 LED 指示灯说明

指示灯实物图：


指示灯说明：

| 指示灯 | 定义 |
|---|---|
| Config | 配置指示灯 |
| CANFD 1 | CANFD 通道 1 指示灯 |
| CANFD 2 | CANFD 通道 2 指示灯 |

指示灯颜色说明：

| 颜色 | 描述 |
|---|---|
| Config 绿灯 | 配置成功后亮起 |
| CANFD 1 绿灯 | CAN FD 通道 1 数据帧发送或者接收正确 |
| CANFD 2 绿灯 | CAN FD 通道 2 数据帧发送或者接收正确 |

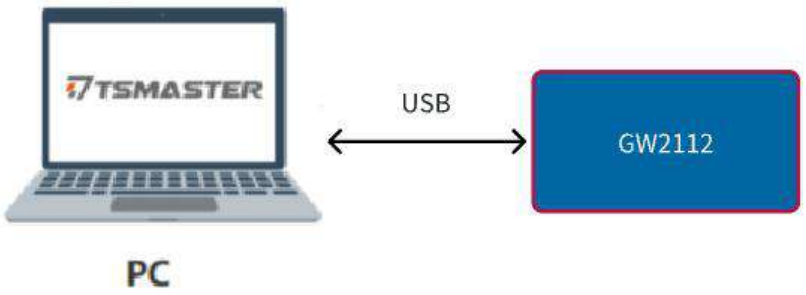
## 2.9 可选配件

无

# 3.使用指南

## 3.1 系统连接

◆ 在线配置




设备需要配置功能时，只需要通过 USB 线与 PC 连接即可，待设备上 Config 灯常亮，配合 TSMaster 软件即可使用配套上位机配置功能。

◆ 离线网关


设备离线使用时，将两路 CAN 接口接入总线，再给设备供电即可，设备上电会按照内部已有的配置进行工作。

## 3.2 驱动安装

TOSUN 硬件均采用免驱设计，具备极佳的系统兼容性，无需安装驱动即可在各种操作系统上（Windows7/8/10/11）直接使用。

## 3.3 软件简介

TSMaster 是一款功能强大的综合工具，可连接、配置并控制所有同星的硬件工具、设备，实现汽车总线嵌入式代码生成、监控、仿真、开发、UDS 诊断、CCP/XCP 标定、ECU 刷写、I/O 控制、测试测量等功能。支持 Matlab Simulink 联合仿真和 CarSim 动力学模型的 ECU 算法仿真测试（软实时 HIL）。它为用户提供了一系列便捷的功能和编辑器，使其能够直接在 TSMaster 中执行 ECU 代码，并且支持 C 脚本和 Python 脚本编辑。同时，TSMaster 还提供了小程序功能，使用户能够自定义仿真测试面板、测试流程、测试逻辑甚至整个测试系统，并自动生成报告。用户基于 TSMaster 编写的代码具有硬件无关性，可方便地分享、引用和在不同硬件平台上使用。

TSMaster 支持多种常用的总线工具，包括 Vector、Kvaser、PEAK，IXXAT,以及市场上主流的仪器（如示波器、波形发生器和数字万用表）和板卡（如 AI、DI、DO 等）。它的设计理念是与测试系统完美结合，实现多硬件、多通道的联合仿真和测试。这使得 TSMaster 能够满足各种汽车电子部件和总成的 PV/DV 测试验证以及产线下线检测的需求。

## 3.4 软件安装

TSMaster 软件下载链接：

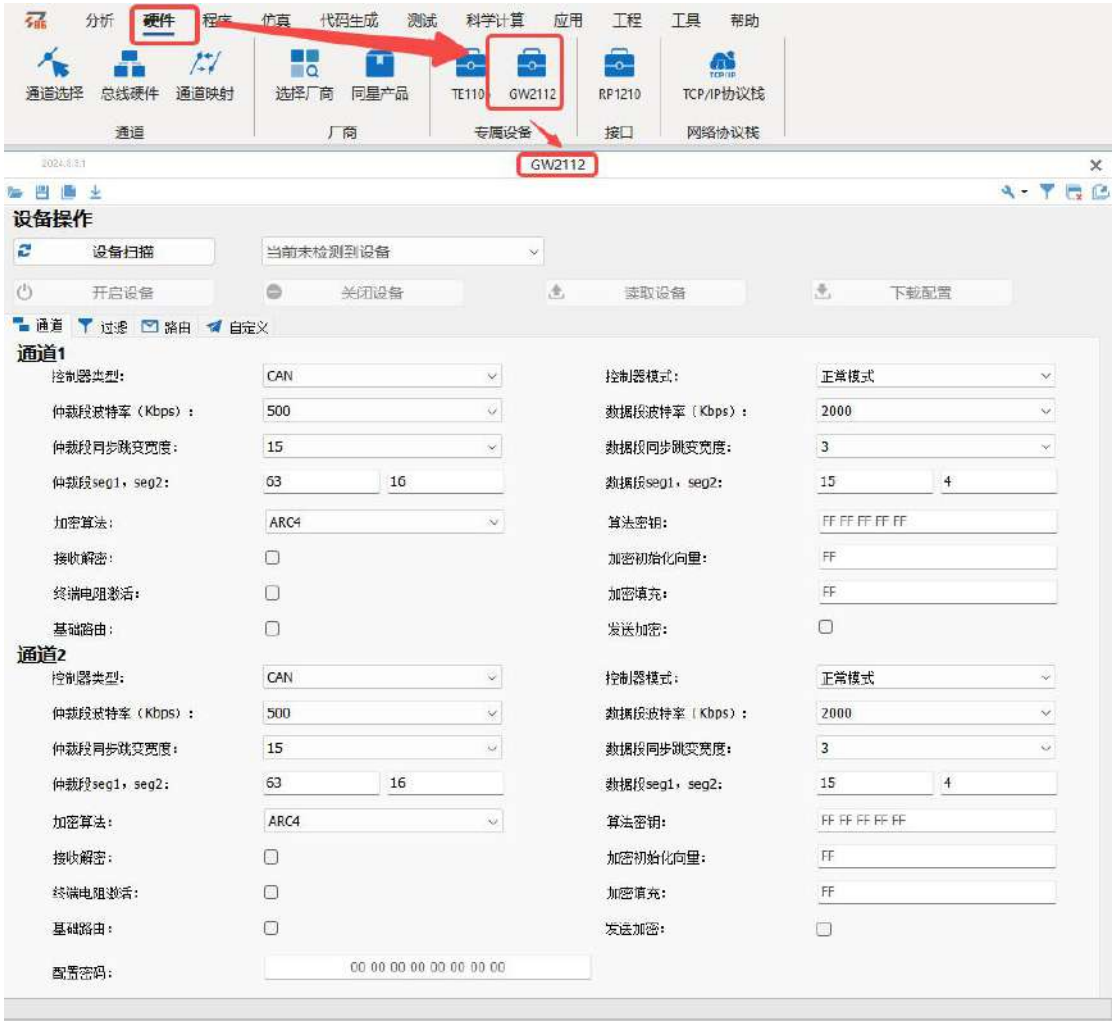
http://download.tosun.tech/TOSUNSoftware/TSMaster_Setup_beta.7z

若无法访问，可联系对应销售人员或登录同星官网获取上位机，亦可扫码关注公众号获取下载链接。

安装完成后，即可在 PC 上看到如下所示软件。

## 3.5 GW2112 工具箱

将 TSMaster 更新到最新版，在 TSMaster 中打开 GW2112 配置页面。



◆ 文件导入导出

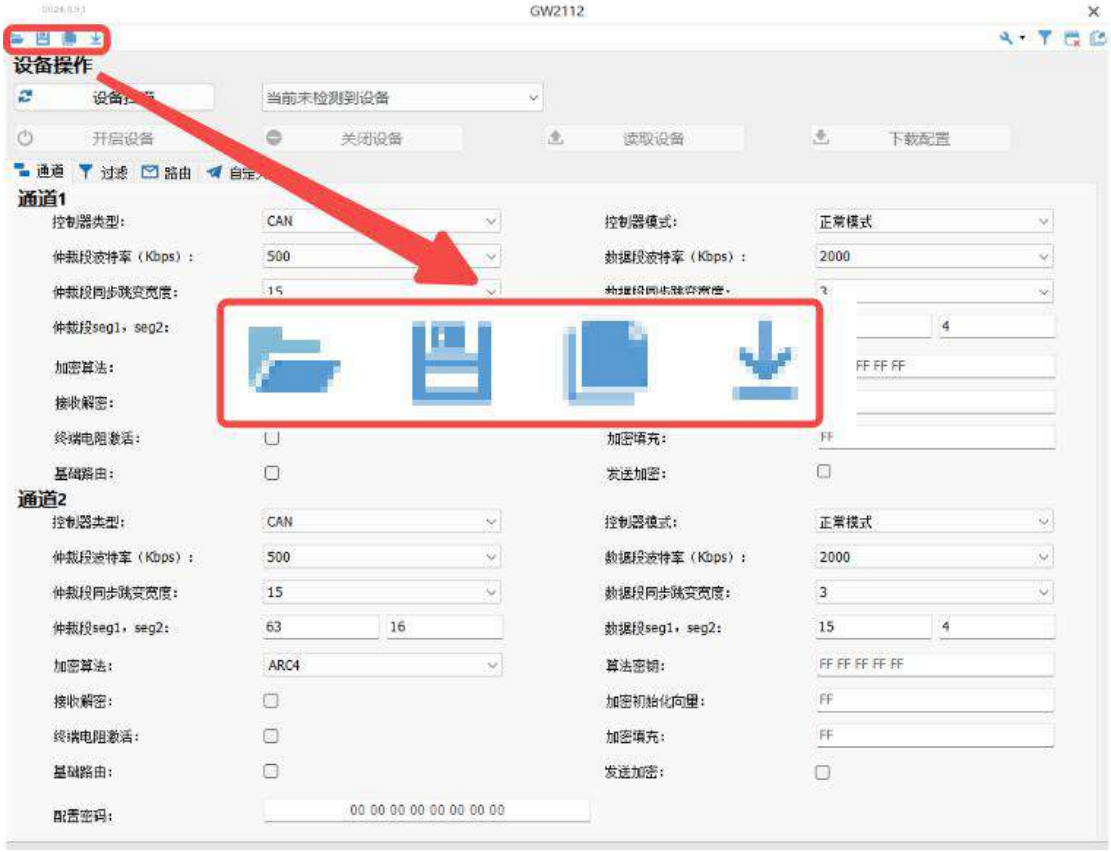
在配置页面的左上角，有如下四个图标：

加载配置：可以加载之前保存的配置 json 文件；

保存配置：将当前配置信息保存为 json 文件，下次可以直接导入 json 文件加载此配置信息；

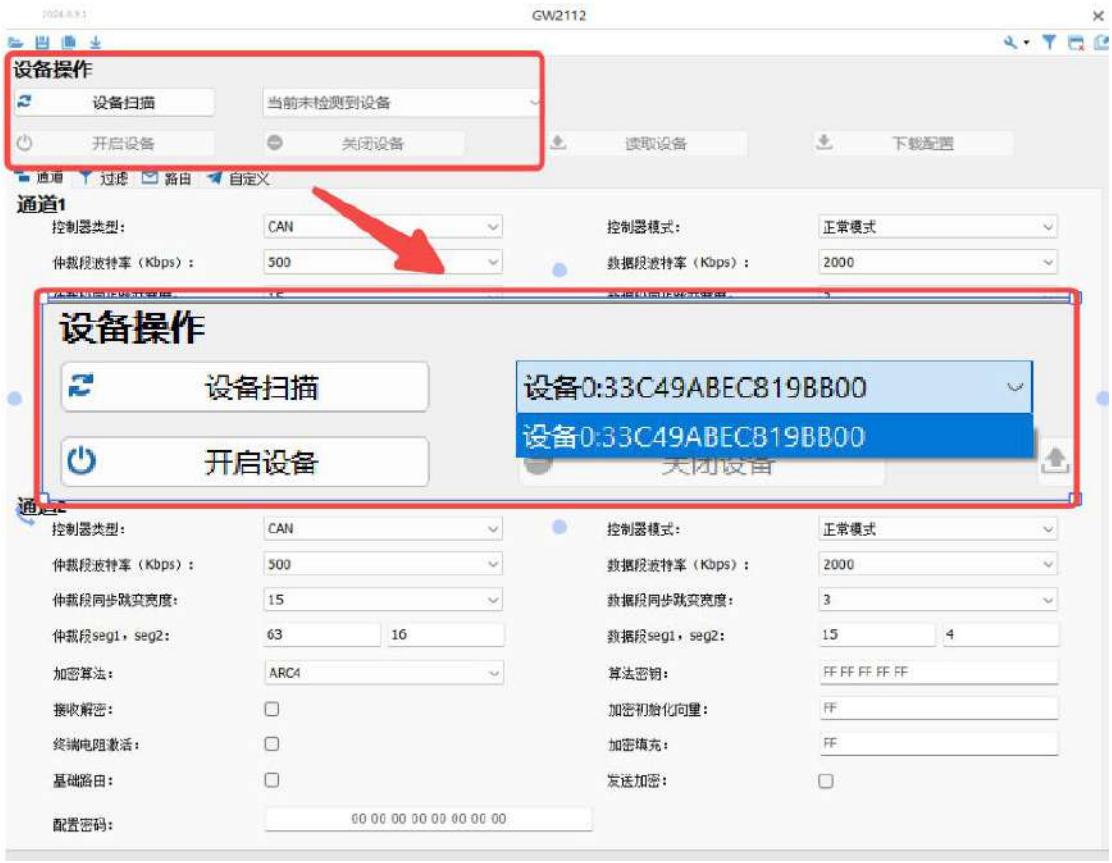
加载自定义 BIN 文件：此按钮功能可以导入自定义配置 bin 文件；

使用说明书下载：下载 GW2112 产品说明书；




◆ 设备连接

点击扫描设备，会检测当前在线设备，可能检测到多个设备，点击下拉框选择设备。“开启设备”按钮用于连接设备，“关闭设备”按钮断开连接。




◆ 读取设备和下载配置

“读取设备”按钮用于读取当前连接设备的配置信息；

“下载配置”按钮用于将配置页面配置的信息下载进设备中，此操作会替换设备中原有配置。

◆ 功能配置

通道：配置两路 CAN 通道参数；

过滤：配置过滤器，屏蔽通道中不需要的报文；

路由：配置转换关系，主要用于在两个 CAN（或 CAN FD）网络之间进行报文的灵活转换和管理；

自定义配置：用户自定义路由规则，用户根据特定需求自定义 GW2112 的报文转换规则和处理逻辑。

## 3.6 功能配置

GW2112 在使用前需要先进行功能配置，通过 USB 线连接电脑和设备后，待设备上状态灯常亮连接设备后即可开始配置。

功能配置分为通道、过滤 、路由、自定义四个部分。

### 3.6.1 设备连接

参考“3.5 GW2112 工具箱”一节进行设备连接。

### 3.6.2 读取配置和下载配置

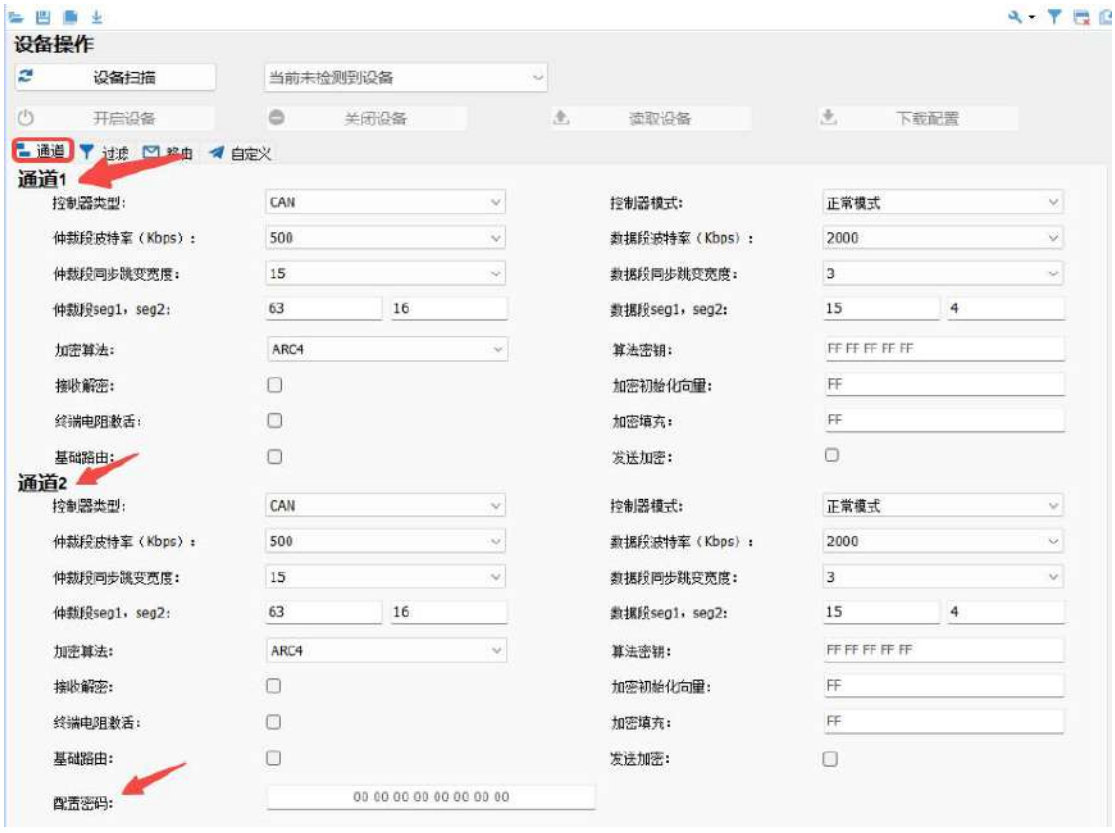
参考“3.5 GW2112 工具箱”一节。

### 3.6.3 通道配置

通道配置可以配置两路 CAN/CAN FD 通道，其中面板上半部分是配置通道 1 信息，下半部分是配置通道 2 信息。

除通道配置外，还可配置密码。

通道配置部分面板布局如下图所示：

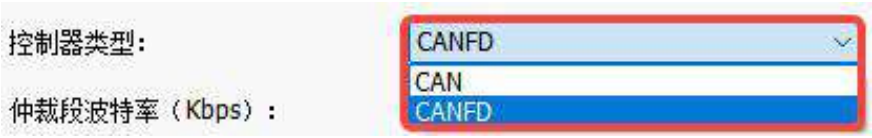


◆ 控制器类型

每条通道可选择配置 CAN 或者 CANFD 类型。

配置为 CAN：防止 CANFD 报文转发到 CAN 总线；

配置为 CAN FD：CAN 报文和 CAN FD 报文都可以转发。




◆ 控制器模式

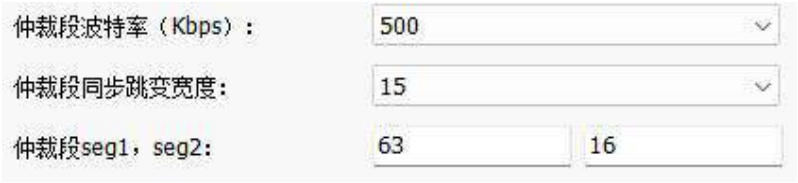
正常模式：不做限制，正常转发；

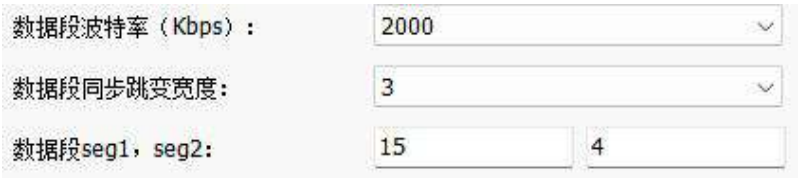
限制监听模式：在该模式下控制器能够接收数据帧和远程帧，确认有效帧，但不支持数据帧、远程帧、活动错误帧、过载帧的发送；

总线监听模式：在该模式下控制器能够接收有效数据帧和有效远程帧，不支持确认有效帧。


◆ 基本参数

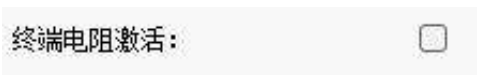
波特率、同步跳变宽度、seg 按需配置。



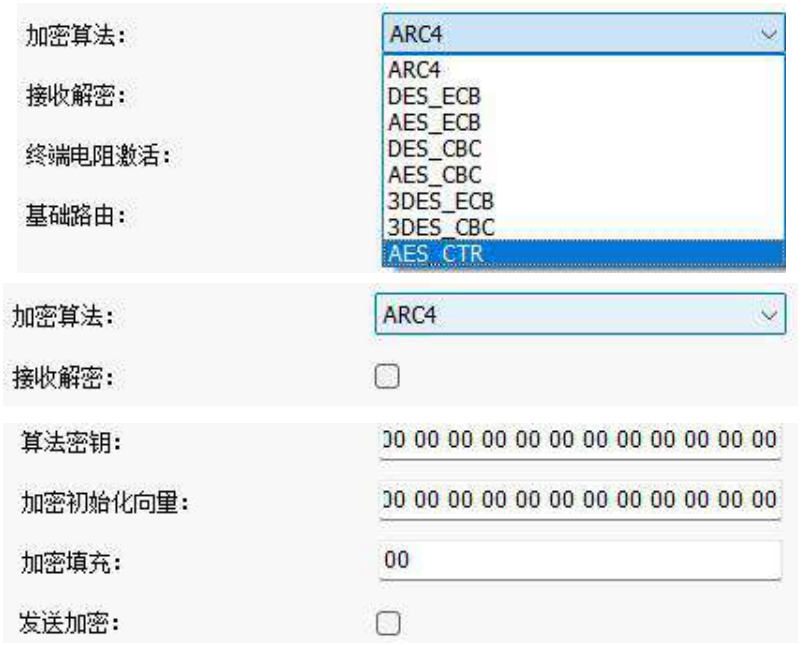




◆ 终端电阻激活

勾选时会激活设备上自带的 120Ω终端电阻。



◆ 加密




加密算法：设备提供多种加密算法，可以在加密算法处自行选择，不同算法对应加解密长度有要求；

| 算法 | 长度 |
|---|---|
| ARC4 | 不做限制 |
| DES_ECB | 16 的倍数 |
| AES_ECB | 8 的倍数 |
| DES_CBC | 16 的倍数 |
| AES_CBC | 8 的倍数 |
| 3DES_ECB | 16 的倍数 |
| 3DES_CBC | 16 的倍数 |
| AES_CTR | 8 的倍数 |

算法密钥：根据算法设置对应密钥；

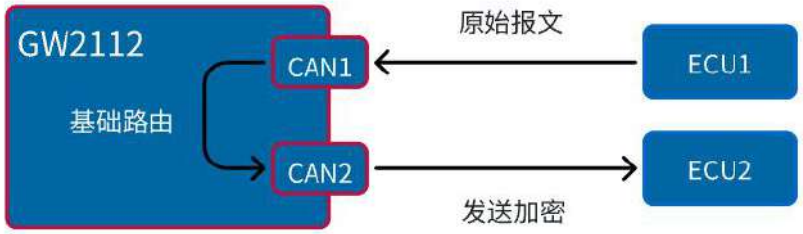
接收解密：当需要对接收的报文进行解密时，勾选此选项；

发送加密：当需要把发送的报文进行加密时，勾选此选项；

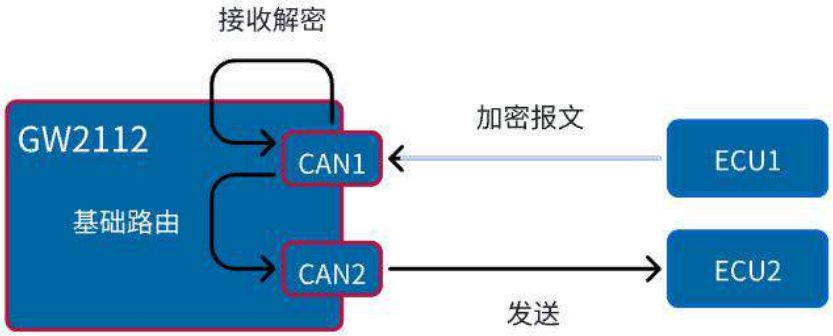
加密初始化向量：用于增加加密数据的随机性，确保相同数据在不同加密操作中产生不同密文；

加密填充：当明文的长度不足以满足加密算法的块大小要求时，填充数据将被添加到明文中，使其长度符合要求。

加密过程：




解密过程：

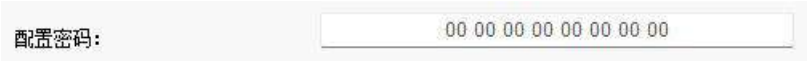



◆ 基础路由

当勾选基础路由时，设备从一个通道收到的数据如果不存在任何处理的情况下会原封不动的从一路通道转出，该功能多数情况下用于测试。

配置密码：

为 8 字节十六进制数字，读取设备时需要输入这里配置的十六进制密码。




### 3.6.4 过滤配置

过滤器用于屏蔽通道中设备所不需要处理的报文，通过点击添加来增加一条过滤项，每条过滤项需要设置用于哪路通道（通道）、过滤格式（分为标准帧和扩展帧），过滤 ID 和掩码 ID，过滤方式采用掩码过滤。


大多数情况下，过滤器会用于过滤单条报文，例：通道 CAN1，过滤格式标准帧，过滤 ID 为 0x1，掩码 ID 为 0x7FF，该条过滤项表示该通道收到标准帧时仅允许 ID 为 0x1 的报文通过，多个过滤项之间为并集关系。

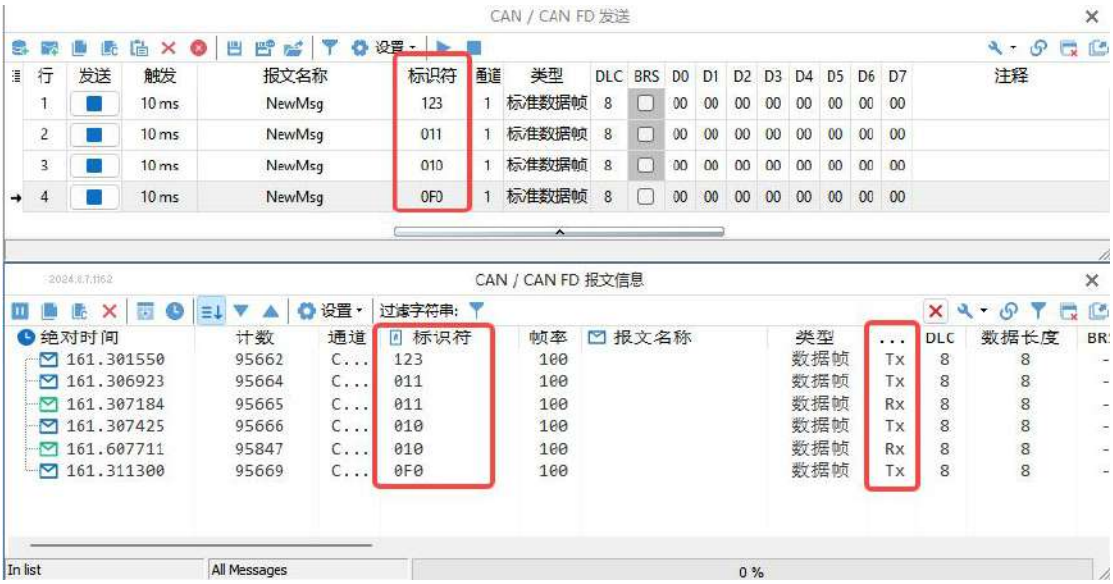
◆ 示例

配置通道为 CAN1、过滤格式为标准帧 ID、过滤 ID 为 0x11、掩码 ID 为 0xF0


结果如下图所示：

使用 TSMaster 的报文发送和报文信息功能，分别通过 CAN1 通道发送 ID 为 0x123、0x011、0x010、0x0F0 的报文，可以看到只有 0x11、0x10 的报文收到了，说明 GW2112 过滤掉了 ID 除 0x11 以及 0x11 与上掩码 ID 以外的报文

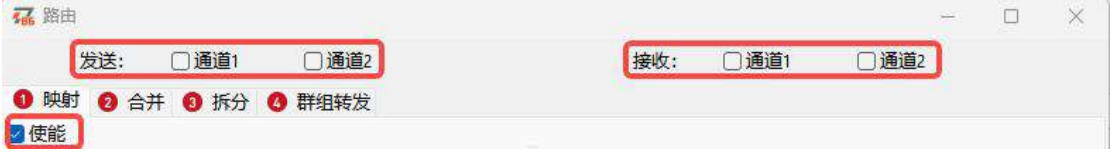
（0x11 & 0xF0 = 0x10）




## 3.6.5 路由配置

路由配置用于配置转换关系，转换关系有映射、拆分、合并、群组转发四种，点击添加可以添加一条关系。

注意：添加任何一种关系时，首先都需要勾选最上方的使能以及接收和发送，判断该规则处理哪路的报文，通过哪路通道发送处理后的报文；勾选上“使能”才可以开始配置。




路由配置功能自带路由，所以要想路由配置生效，需把通道配置中基础路由选项关闭，因为基础路由是将数据不做任何处理的转发。


◆ 映射

接收到指定报文时，将其变为另一条预先配置好的报文，如下图所示，当接收到符合左侧的源报文时，会将报文转变为右侧的目标报文，需注意，仅有左侧打钩的内容会转变，未打钩的部分依旧是原报文的样式。

注意：“加速”选项即 BRS 位，在 CAN 中是没有的，所以这个选项应该选择“不转换”。其余配置也应是符合规范的配置；

➢ 示例

配置如下图所示，

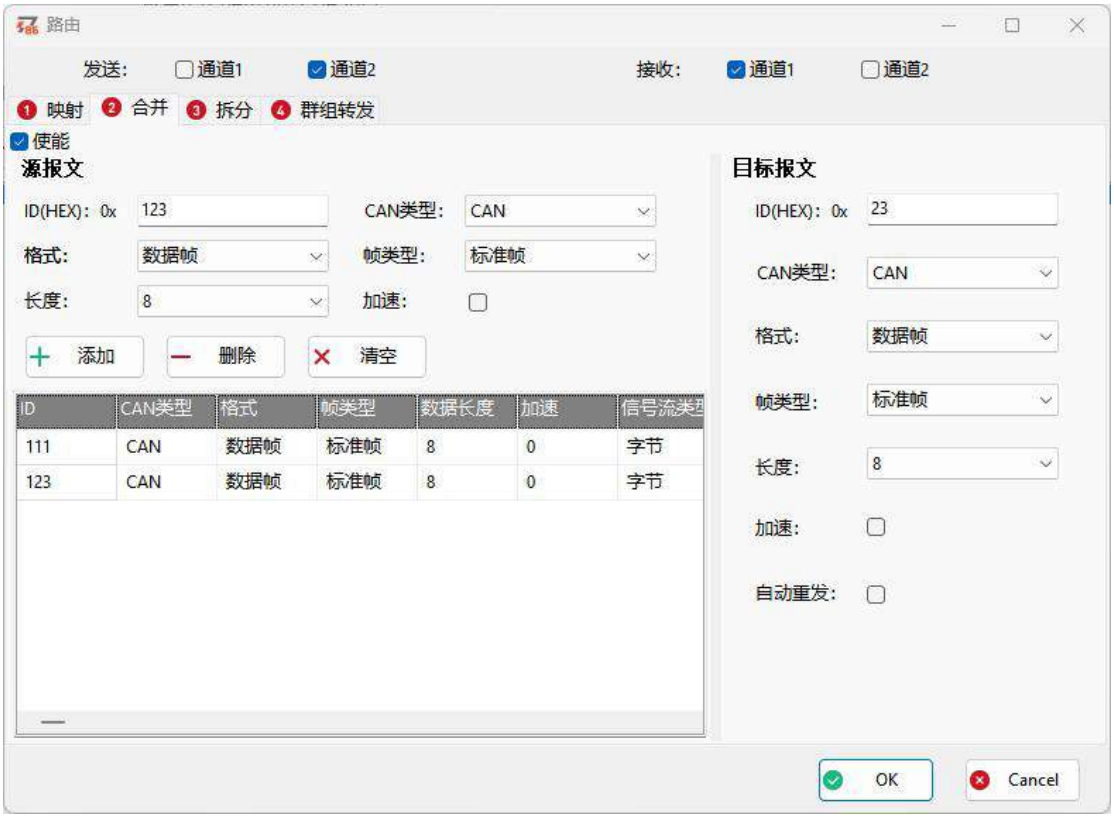

结果演示：如下图所示，发送源报文 ID 为 0x123，数据为 0x12345678，经过 GW2112

转换过后，变成了 ID 为 0x111，数据为 0x87654321。


◆ 合并

接收到指定报文时，将其部分内容缓存起来，当接收到触发发送的报文时（参与合并的最后一条报文）发送合并后的报文。

如下图所示是路由合并配置页面，分为两个部分：源报文和目标报文。

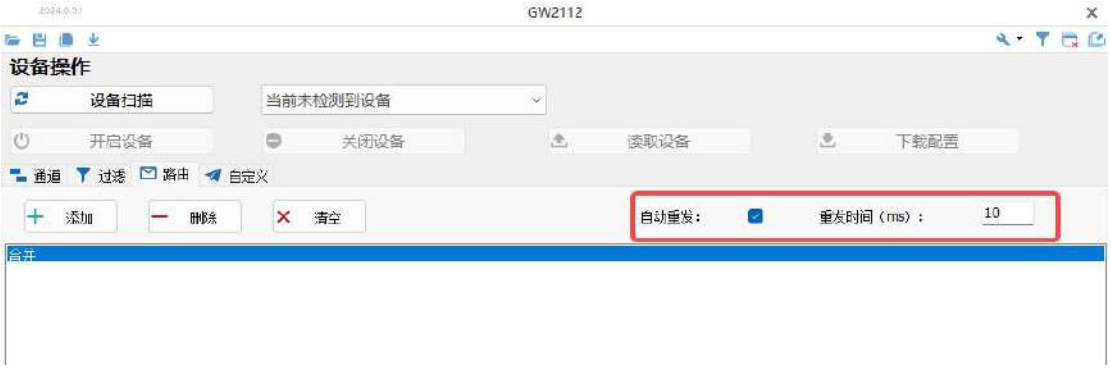



➢ 目标报文

目标报文配置比较简单，直接配置即可，GW2112 会把源报文进行相应处理成为目标报文配置的格式。

自动重发：目标报文中“自动重发”选项勾选上，表示目标报文被触发发送后会自动重发。目标报文自动重发功能还需与上一级页面中的自动重发配合，若需要自动重发，则二者

都得勾选上，并配置重发间隔时间，单位是 ms。



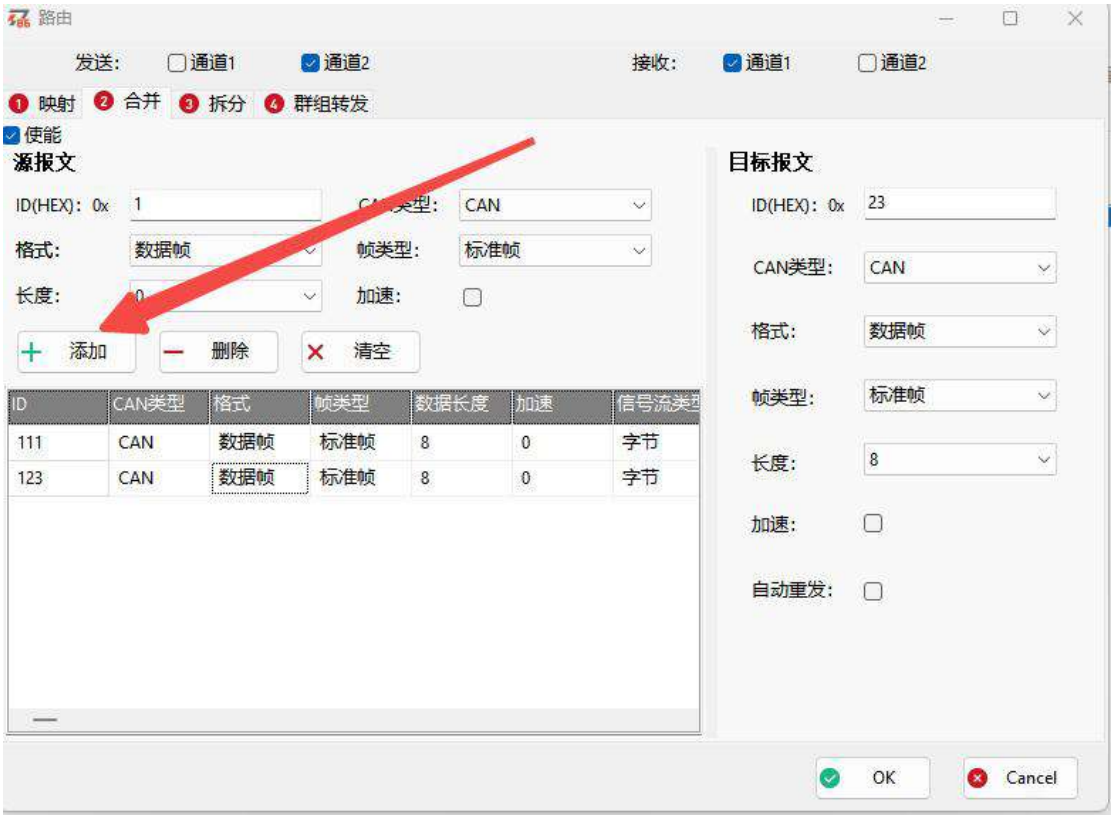

➢ 源报文

勾选“使能”开始配置。可以配置多个不同的 ID。每一个 ID 需添加对应得合并规则。当 GW2112 接收到一条此处配置的报文时，会将其先缓存起来，直到接收到配置的最后一条报文。最后配置的报文会作为触发报文，当 GW2112 收到这条报文后，会将该条报文与之前的报文一起合并为目标报文的数据段，并发送除去。

设置好 ID 等基础信息后，点击“添加”按钮，会弹出合并规则配置窗口；

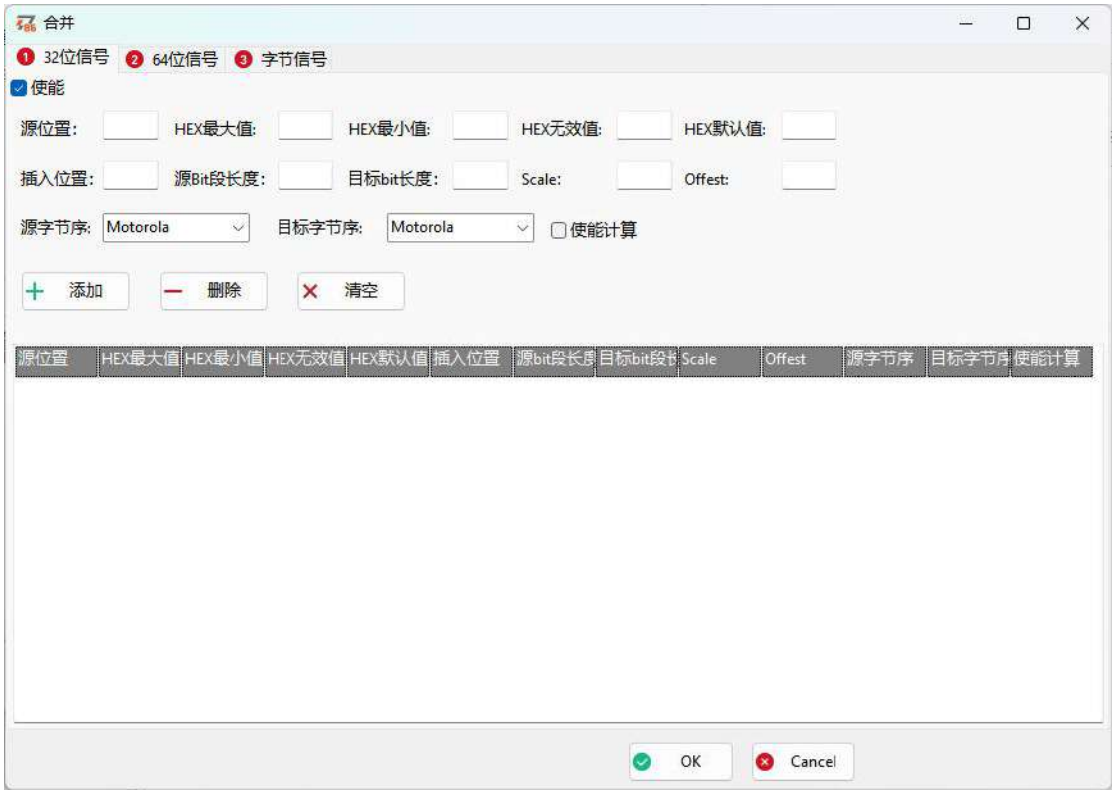
“删除”按钮每点击一次会删除最后一个报文配置；

“清空”按钮会一次清空所有报文配置。



➢ 32 位信号

如下图所示，32 位信号有多个可配置参数，




源位置，源 bit 段长度，源字节序用于确认该条数据的来源；

字节序中，Motorola 代表大端序，Intel 代表小端序；

源位置表示该段信号从原始报文的哪个位置（bit）开始取，加上长度和字节序可以得到一个实际的物理值（设备内部会自己计算），如果勾选使能计算，该值会再*Scale 再加上 Offset 得到一个新值；

HEX 最小值，HEX 无效值，HEX 默认值用于约束这个物理值的意义，当该物理值小于最小值或大于最大值或为无效值时将其变为默认值；

插入位置，目标 bit 长度，目标字节序用于确认该信号在该条拆分子项的数据段中的位置，即将上面得到的 hex 值再转化为 bit 插入拆分子项对应的位置中；

➢ 64 位信号

同 32 位信号，仅 bit 段长度上限不同。

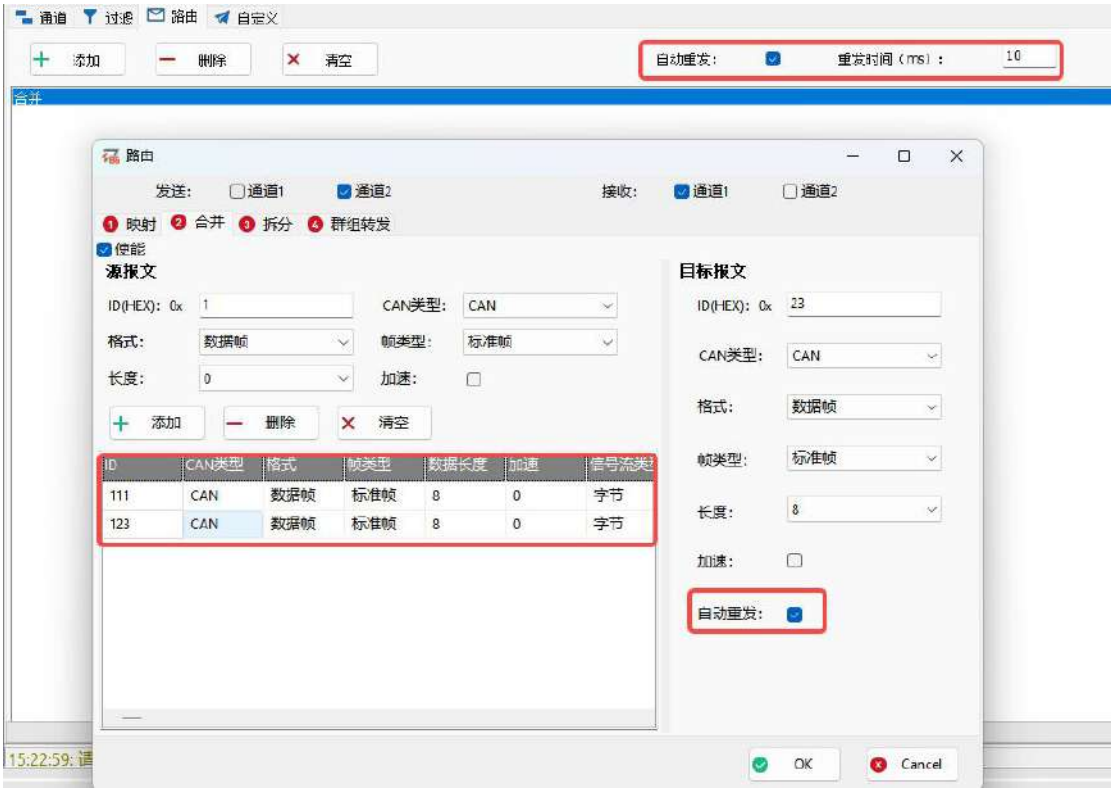
当信号长度大于 32bit 时选择 64bit 信号。

➢ 字节信号

大多数情况下会使用字节信号，仅需要添加源起始字节，长度和目标起始字节即可，将源报文中源起始字节开始的对应长度的数据放入该拆分子项中。


➢ 示例

配置 ID 为 0x111 和 0x123 报文，合并规则为“字节信号”；目标报文 ID 为 0x23, 并设置自动重发，时间间隔为 10ms：




ID 为 0x111 配置如下：

| 目标起始字节 | 长度 | 源起始字节 |
|---|---|---|
| 1 | 1 | 1 |

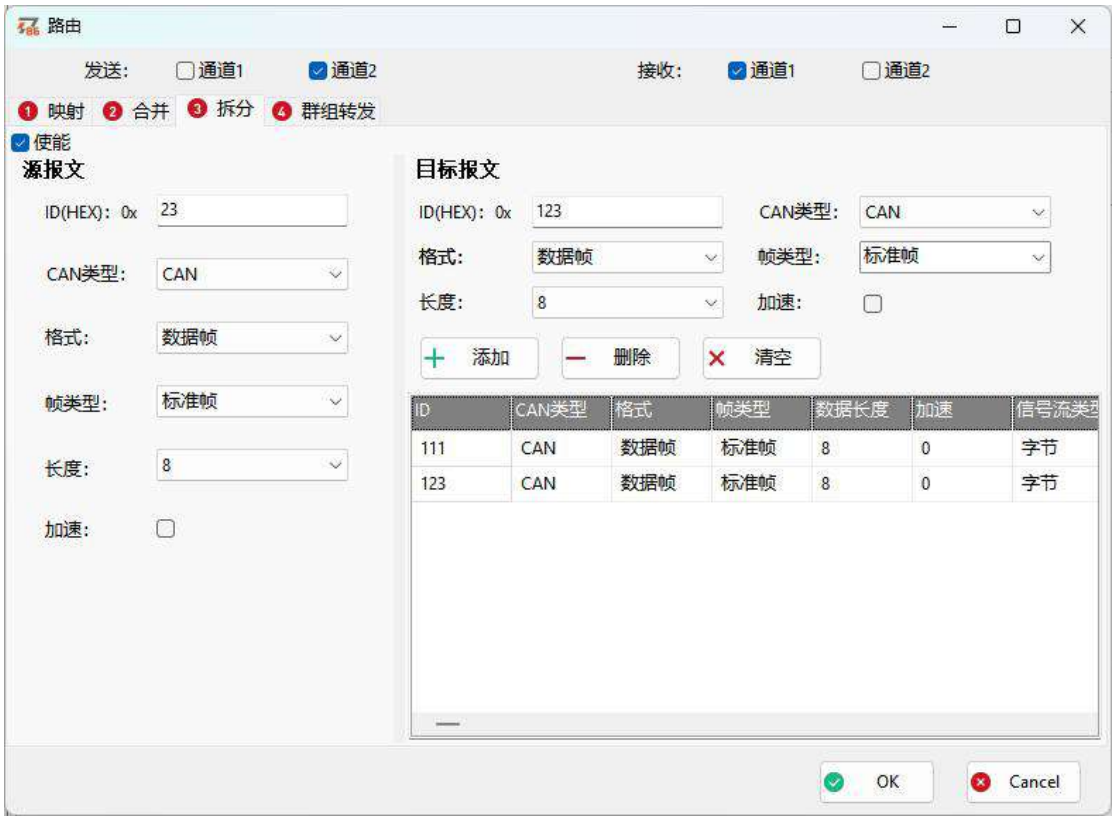
ID 为 0x123 配置如下：

| 目标起始字节 | 长度 | 源起始字节 |
| --- | --- | --- |
| 3 | 1 | 3 |

结果如下：先发送一条 ID 为 0x111 的报文，再发送一条 ID 为 0x123 的报文，发送 0x123 的报文后，会收到 ID 为 0x23 的报文，该条报文是 GW2112 中配置的目标报文。可以看到 ID 为 0x23 的目标报文的数据段内容就是按照配置规则中两个 ID 对应的数据段的对应位和长度合并而来，其余位做补 0 处理。且目标报文是以 10ms 的间隔在接收。


◆ 拆分

可参考“合并”。拆分是把一条报文拆分为多条报文。



➢ 示例

配置源报文 ID 是 0x23，目标报文 ID 是 0x111 和 0x123，目标起始字节分别为 1、3，长度都为 1，源起始字节分别为 1、3，结果如下图所示：

报文经过 GW2112 转发后，将 ID 为 0x23 的报文拆分为了两条报文，数据段也都按照配置规则进行了拆分。



◆ 群组转发

如下图所示，群组转发有三个配置项，配置项只对 ID 起效。


接下来，结合案例对这三个配置项作解释：

将源起始报文 ID 配置为 1，目标起始报文 ID 配置为 8，ID 偏移长度配置为 4，现象如下：

使用 TSMaster 报文发送工具发送了 6 条 ID 从 1 开始的连续 ID 递增报文，使用报文信息工具得知，RX 的报文 ID 从 8 开始，且 RX 的报文从 0x8 开始一直到 0xC 一共只有 5 个报

文被接收到了。但是发出去的一共有 6 条报文，这是因为 ID 偏移长度设置为 4，代表从起始 ID 开始(包括起始 ID)，一共有 4+1 条报文会通过 GW2112 设备转换为起始报文 ID 为 8 发出，也就是在此案例中，ID 为 6 的报文不做转发处理。



### 3.6.6 自定义配置

用户可以自定义配置规则，在自定义配置中，可以配置之前提到的所有规则，可以做到更精细的配置。

◆ 面板说明


CAN 报文启用：勾选“CAN 报文启用”表示接收到 CAN 报文启用的事件；

定时器启用、定时器触发时间：勾选“定时器启用”，搭配“定时器触发时间”(单位：ms 毫秒)，表示每间隔设定好的毫秒时间就触发一次定时器启动事件；

运行空闲启用：在设备主循环程序中每一次都会执行的事件；

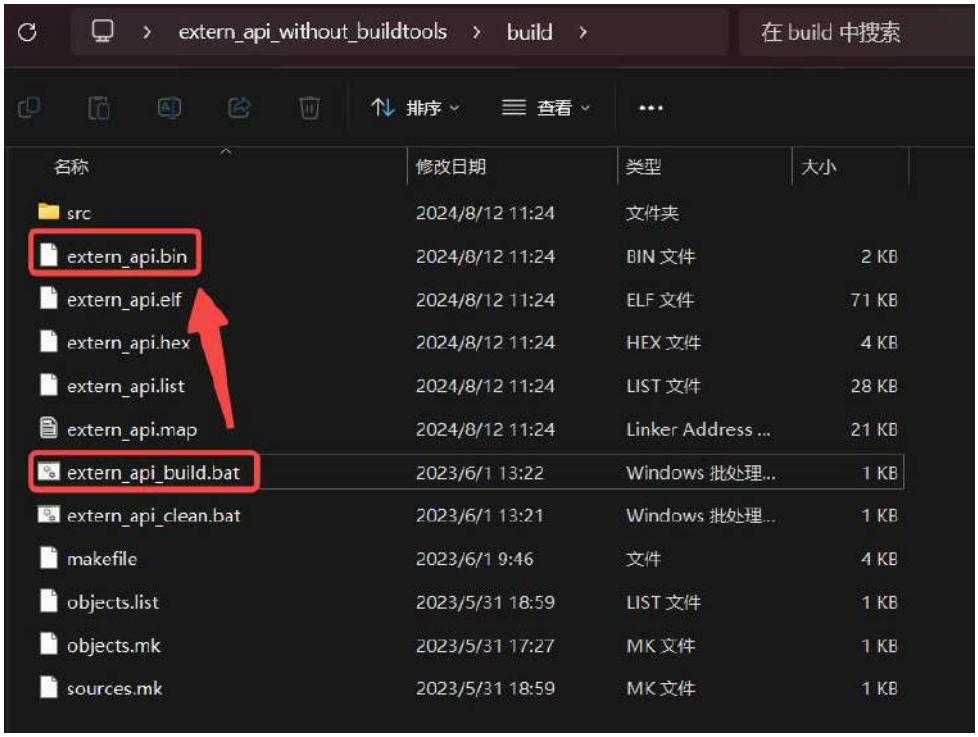
三者是可以同时勾选多个；

◆ 编译下载

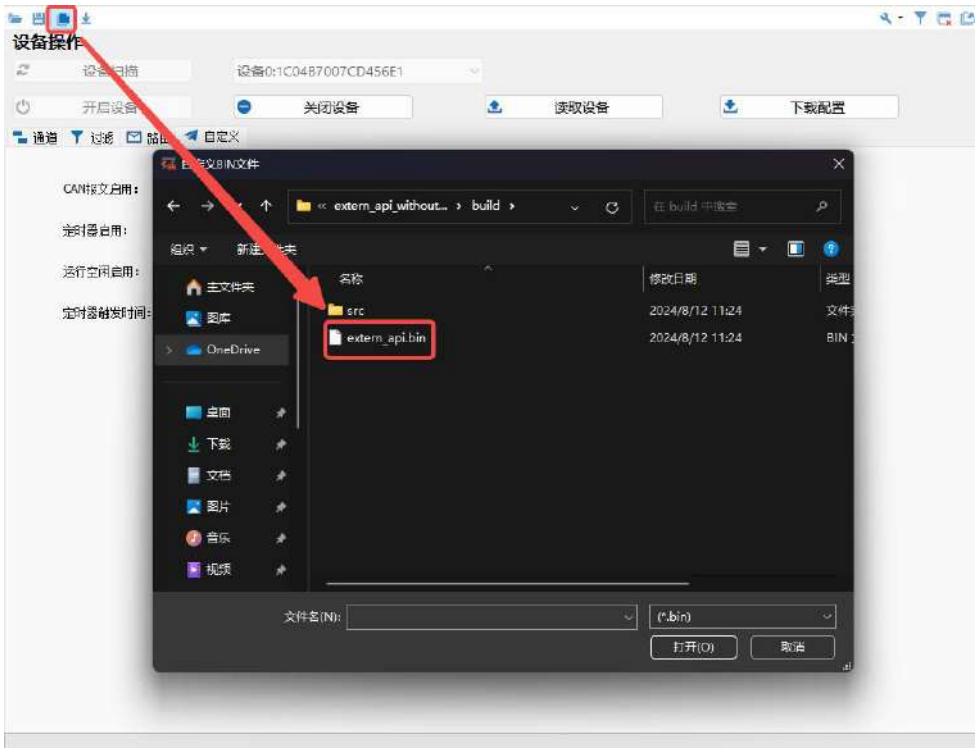
GW2112 自定义函数可以由自己编译也可以是别人编译好的 bin 文件，通过上位机下载到设备中进行使用。

打开附件：注意附件所在目录不要带有中文路径。


进入“build”目录，双击“extern_api_build.bat”文件，会在同级目录下生成/更新 bin 文件，这个 bin 文件就是我们需要下载进设备的文件。




打开 TSMaster 上位机，选择 GW2112 设备，进入配置面板开启设备后，点击导入 bin 文件工具按钮 ，选择对应 bin 文件即可导入 bin 文件。




◆ 自定义函数源代码

打开附件中 src 目录下的“extern_api.c”文件，找到里面的“Event_Handle”函数，这个函数就是自定义函数本体。

函数定义：

```
uint32_t Event_Handle(const Event *Event_p, const Action_Source* Action_p);
void Event_Runtime(const Action_Source* Action_p);
```

函数参数：

➢ Event_p

```
typedef struct {
    Event_Type Event_Source;
    Event_Argument Argument;
} Event;
```

Event_Source：表示自定义函数触发的原因，设备在运行过程中，进入这个自定义函数是有各种各样的触发条件的，一定是在出现了某种情况下才能进入到这个自定义函数中，通过这个 Event_Source 来确定本次进入自定义函数是由于什么原因：

```
typedef enum {
    RECEIVE_CAN = 0x00,
    TIMER_IRQ = 0x01,
    START_IRQ = 0x02
} Event_Type;
```

目前有三种情况，“RECEIVE_CAN”表示接收到了 CAN 报文进入了本次自定义函数，“TIMER_IRQ”表示本次是定时器触发进入了本次自定义函数，“START_IRQ”表示本次是设

备刚开机进入了本次自定义函数，每次进入自定义函数只能有一个触发原因，在不同触发条件下做不同的处理；

```
typedef union {
    Can_Receive_Event Receive_Can;
    Timer_Irq_Event Irq_Timer;
    Start_Irq_Event On_Start;
} Event_Argument;
```

Argument:这个结构体用来提供触发本次自定义函数所对应的资源，如果本次自定义函数是由于接收到了 CAN 报文，那么在这里能够使用的资源只有 Receive_Can：

```
//触发事件，触发原因，操作外设的接口，指令需不需要向上层汇报，用户的空间
typedef struct {
    uint32_t Id;
    uint8_t SourcePort; //来自哪个通道 0为1通道，1为2通道
    uint8_t FrameType; //0为远程帧，1为数据帧
    uint8_t IdType; //0为标准帧，1为扩展帧
    uint8_t CanType; //0为经典CAN，1为FDCAN
    uint8_t BRS; //0为关闭，1为开启
    uint8_t DataLength;
    uint8_t Free[2];
    uint8_t Data[64];
} Can_Receive_Event;
```

在这里可以得到触发本次自定义函数的接收到的报文的所有信息；

如果是由于定时器触发，那么在这里能使用的资源只有 Irq_Timer:

```
typedef struct {
    uint32_t Time;  //Timer记时，叠加，单位为ms

} Timer_Irq_Event;
```

- Action_p

这个结构体也包括两部分，一个发送函数和一个用户自定义空间：

```
typedef struct {
    void(*Can_Transmit)(Transmit_CanTypedef SendCan);

    uint8_t* user_array;
} Action_Source;
```

发送函数 Can_Transmit:用这个函数让设备发送一帧报文，参数 SendCan 内容如下：

```
typedef struct {
    uint32_t Id;
    uint8_t TargetPort; //1为1通道，2为2通道
    uint8_t FrameType; //0为远程帧，1为数据帧
    uint8_t IdType; //0为标准帧，1为扩展帧
    uint8_t CanType; //0为经典CAN，1为FDCAN
    uint8_t BRS; //0为关闭，1为开启
    uint8_t DataLength;
    uint8_t Free[2];
    uint8_t Data[64];
} Transmit_CanTypedef;
```

按照需求填写一个 SendCan，然后调用 Can_Transmit 即可发送一帧自定义的报文；

user_array 为一个指针，指向一块自定义专用的空间，一般的数据，在对应的触发函数调用结束后就会被释放掉，下次再进入自定义函数时都是新的数据，如果需要保留本次自定义处理中的一些数据用于下次再进入自定义函数时使用就需要用到 user_array，可以将数据存入 user_array 中，在自定义空间中保留的数据除非手动在自定义函数中改写，在断电前都会一直保留。

- 运行空闲启用函数

```
/*
示例：用户自定义函数
*/
__attribute__((section(".EXTERN_FUNC")))
void Event_Runtime(const Action_Source* Action_p)
{

}
```

由用户自定义该函数的实现，该函数会在上位机中“运行空闲启用”选项被勾选上后被调用，调用时机为设备内程序主循环的每一次循环被掉用。

**示例：**

附件内会自带一个非常详细的例程，这里对例程进行一个详细的介绍：

```
Transmit_CanTypedef TxCAN = {0};
uint32_t Counter = 0;
```

首先，定义了两个变量，TxCAN 是发送函数的参数，Counter 是用来计数的，下面会用到。

```
if (Event_p->Event_Source == ON_START) {
    TxCAN.Id = 0x1CCAB21;
    TxCAN.FrameType = 1;
    TxCAN.IdType = 1;
    TxCAN.CanType = 1;
    TxCAN.BRS = 1;
    TxCAN.DataLength = 8;
    TxCAN.TargetPort = 0x02;
    for (int i = 0; i < 8; i++) {
        TxCAN.Data[i] = (i + 2);
    }
    Action_p->Can_Transmit(TxCAN);
}
```

此处表示，刚开机的时候 2 通道会发出一帧 Id 为 0x1CCAB21 的 CANFD 报文，报文长度为 8，内容为 2~9。

先不看定时器触发部分，看 CAN_RECEIVE 部分：

```
else if (Event_p->Event_Source == CAN_RECEIVE) {        //接收报文事件
    if (Event_p->Argument.Receive_Can.SourcePort == 0x00) {
        if (Event_p->Argument.Receive_Can.Id == 0xA1) {
            Action_p->user_array[4] = (Event_p->Argument.Receive_Can.Id >
```

此处表示，仅当 1 通道收到报文时有效，

```
if (Event_p->Argument.Receive_Can.Id == 0xA1) {
    Action_p->user_array[4] = (Event_p->Argument.Receive_Can.Id >> 24) & 0xFF;
    Action_p->user_array[5] = (Event_p->Argument.Receive_Can.Id >> 16) & 0xFF;
    Action_p->user_array[6] = (Event_p->Argument.Receive_Can.Id >> 8) & 0xFF;
    Action_p->user_array[7] = (Event_p->Argument.Receive_Can.Id) & 0xFF;
    Action_p->user_array[8] = (Event_p->Argument.Receive_Can.FrameType);
    Action_p->user_array[9] = (Event_p->Argument.Receive_Can.IdType);
    Action_p->user_array[10] = (Event_p->Argument.Receive_Can.CanType);
    Action_p->user_array[11] = (Event_p->Argument.Receive_Can.BRS);
    Action_p->user_array[12] = (Event_p->Argument.Receive_Can.DataLength);
    for (int i = 0; i < 64; i++) {
        Action_p->user_array[13 + i] = (Event_p->Argument.Receive_Can.Data[i]);
    }
}
```

当 1 通道收到 ID 为 A1 的报文时，将报文内容存入用户自定义空间，注意在自定义空间中存放的位置，自定义空间使用全都由使用者自己控制每个字节的内容。

```
else if (Event_p->Argument.Receive_Can.Id == 0xA2) {
    Action_p->user_array[4 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.Id >> 24) & 0xFF;
    Action_p->user_array[5 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.Id >> 16) & 0xFF;
    Action_p->user_array[6 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.Id >> 8) & 0xFF;
    Action_p->user_array[7 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.Id) & 0xFF;
    Action_p->user_array[8 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.FrameType);
    Action_p->user_array[9 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.IdType);
    Action_p->user_array[10 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.CanType);
    Action_p->user_array[11 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.BRS);
    Action_p->user_array[12 + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.DataLength);
    for (int i = 0; i < 64; i++) {
        Action_p->user_array[13 + i + CAN_OCCUPY] = (Event_p->Argument.Receive_Can.Data[i]);
    }
}
```

当 1 通道收到 ID 为 A2 的报文时，将报文存入用户自定义空间，位置 A1 的不同，CAN_OCCUPY 为一个 CAN 报文所占用的内容大小。

```
else if (Event_p->Argument.Receive_Can.Id == 0xA3) {
    Counter++;
    Action_p->user_array[0] = (Counter >> 24) & 0xFF;
    Action_p->user_array[1] = (Counter >> 16) & 0xFF;
    Action_p->user_array[2] = (Counter >> 8) & 0xFF;
    Action_p->user_array[3] = Counter & 0xFF;
}
```

当 1 通道收到 ID 为 A3 的报文时，Counter 计数+1，将 Counter 存进用户自定义空间，用来控制计数。

接下来我们看刚进入函数时的 Counter 取值：

```
Counter = ((Action_p->user_array[0] << 24) |
           (Action_p->user_array[1] << 16) |
           (Action_p->user_array[2] << 8) |
           (Action_p->user_array[3]));
```

此处表示我每次进入函数时会先将 Counter 的值取出来，得到计数值，

接下来我们看定时器处理部分：

```
else if (Event_p->Event_Source == TIMER_MS) {       //定时器触发
    if (Counter != 0) {
        TxCAN.Id = ((Action_p->user_array[4] << 24) |
                    (Action_p->user_array[5] << 16) |
                    (Action_p->user_array[6] << 8) |
                    (Action_p->user_array[7]));
        TxCAN.FrameType = Action_p->user_array[8];
        TxCAN.IdType = Action_p->user_array[9];
        TxCAN.CanType = Action_p->user_array[10];
        TxCAN.BRS = Action_p->user_array[11];
        TxCAN.DataLength = Action_p->user_array[12];
        TxCAN.TargetPort = 0x02;
        for (int i = 0; i < 64; i++) {
            TxCAN.Data[i] = Action_p->user_array[13 + i];
        }
        Action_p->Can_Transmit(TxCAN);
        if (Counter > 1) {
            TxCAN.Id = ((Action_p->user_array[4 + CAN_OCCUPY] << 24) |
                        (Action_p->user_array[5 + CAN_OCCUPY] << 16) |
                        (Action_p->user_array[6 + CAN_OCCUPY] << 8) |
                        (Action_p->user_array[7 + CAN_OCCUPY]));
            TxCAN.FrameType = Action_p->user_array[8 + CAN_OCCUPY];
            TxCAN.IdType = Action_p->user_array[9 + CAN_OCCUPY];
            TxCAN.CanType = Action_p->user_array[10 + CAN_OCCUPY];
            TxCAN.BRS = Action_p->user_array[11 + CAN_OCCUPY];
            TxCAN.DataLength = Action_p->user_array[12 + CAN_OCCUPY];
            TxCAN.TargetPort = 0x02;
            for (int i = 0; i < 64; i++) {
                TxCAN.Data[i] = Action_p->user_array[13 + i + CAN_OCCUPY];
            }
            Action_p->Can_Transmit(TxCAN);
        }
        TxCAN.Id = 0x1EAC882E;
        TxCAN.FrameType = 1;
        TxCAN.IdType = 1;
        TxCAN.CanType = 1;
        TxCAN.BRS = 1;
        TxCAN.DataLength = 8;
        TxCAN.TargetPort = 0x02;
        for (int i = 0; i < 8; i++) {
            TxCAN.Data[i] = (Action_p->user_array[13 + i] + Action_p->user_array[13 + i + CAN_OCCUPY]);
        }
        Action_p->Can_Transmit(TxCAN);
    }
}
```

当 counter 不为 0 时，会根据定时器触发频率周期通过 2 通道发送接收到的 A1 报文，以及 A3 报文，A3 报文内容为 A1 和 A2 的和，当 counter 大于 1 时，发送 A1 报文和 A3 报文的同时还会发送 A2 报文。

# 4.检查和维护

GW2112 产品的主要电气部件是半导体元件，尽管它有很长的寿命，但在不正确环境下也可能加速老化，使寿命大打折扣。因此，在设备使用过程中应该进行定期检查，以保证使用环境保持所要求的条件。推荐每 6 个月到一年，至少检查一次。在不利的环境条件下，应该进行更频繁的检查。 如下表，如果在维护过程中遇到问题，请阅读下面的内容，以便找到问题可能的原因。如果仍无法解决问题，请联系上海同星智能科技有限公司。

| 项目 | 检查 | 标准 | 行动 |
|---|---|---|---|
| 电源供应 | 在电源供应端检查电压波动 | 电源端口+12V DC | 使用电压表在电源输入端检查源。采取必要措施使电压波动在范围之内 |
| 周围环境 | 检查周围环境温度（包括封闭环境的内部温度） | -40℃~+80℃ | 使用温度计检查温度并确保环境温度保持在允许的范围内 |
| | 检查环境湿度（包括封闭环境的内部湿度） | 相对湿度必须在10%~90% | 使用湿度计检查湿度并确保环境湿度保持在允许范围内 |
| | 检查灰尘、粉末、盐、金属屑的积累 | 没有积累 | 清洁并保护设备 |
| | 检查水、油或化学喷雾碰撞到设备 | 没有喷雾碰到设备 | 如果需要清洁保护设备 |
| | 检查在设备区域中易腐蚀或易燃气体 | 没有易腐蚀或易燃气体 | 通过闻或使用一个传感器检查 |
| | 检查震动和冲击水平 | 震动和冲击在规定范围内 | 如果需要，安装衬垫或其它减震装置 |
| | 检查设备附近的噪声源 | 没有重要噪声信号源 | 隔离设备和噪声源或保护设备 |
| 安装接线 | 检查外部接线中的压接连接器 | 在连接器间有足够的空间 | 肉眼检查如果有必要则调节 |
| | 检查外部接线的损坏 | 没有损坏 | 肉眼检查如果有必须则替换接线 |

TOSUN同星　　汽车电子工具链，国产领导品牌

## 软件 TSMASTER

UDS诊断 / ECU刷写 / CCP/XCP标定
嵌入式代码生成 / 应用发布/加密发布 / 记录与回放
图形化编程 / 剩余总线仿真 / C/Python脚本
总线监控/发送 / SOME/IP和DoIP / 自动化测试

扫码关注
获取软件下载链接

## 硬件

1/2/4/8/12通道CAN FD/CAN转USB/PCIe工具
1/2/6通道LIN转USB/PCIe工具
多通道FlexRay/CAN FD转USB/PCIe工具
多通道车载以太网/CAN FD转USB/PCIe工具
车载以太网介质转换工具（T1转Tx）
多通道CAN FD/Ethernet/LIN记录仪
TTS测试系统（通信板卡、数字/模拟量板卡等）

## 解决方案

总线一致性 / 网络自动化测试系统 / 充电测试系统
EMB标定测试设备 / 信息安全解决方案
FCT/EOL测试设备 / 线控底盘测试解决方案
汽车“四门两盖”试验解决方案
电机性能 / 耐久试验解决方案

## 关于我们

同星智能的核心软件TSMaster及配套硬件设备，
具备嵌入式代码生成、汽车总线分析、仿真、测试及诊断、标定等核心功能，
覆盖了汽车整车及零部件研发、测试、生产、试验、售后全流程。

国际组织
ASAM,CiA
质量保证
ISO9001:2015
CE认证

### 愿景

解决一切工程难题！


联系我们
021-59560506
marketing@tosunai.cn
访问官网
www.tosunai.com


上海 | 广州 | 北京 | 长春 | 成都 | 重庆 | 长沙 | 台北 | 斯图加特