ZLG 致远电子

微文摘

ZLG MICRO DIGEST

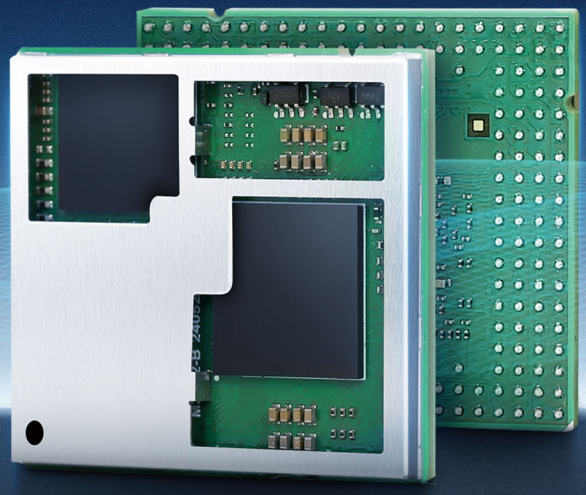
2025/4 第1期

季刊

ZLG 致远电子

M3562 Cortex®-A53核心板

M3562核心板新上市



低成本RK3568方案

致远电子官方网站　致远电子官方微信

选型表

型号	M3562-2GF16GI-B	M3562-4GF32GI-B
处理器	RK3562J	RK3562J
内核	四核 Cortex-A53	四核 Cortex-A53
主频	1.8GHz	1.8GHz
操作系统	buildroot，Debian	buildroot，Debian
内存	2GB	4GB
存储	16GB	32GB
GPU 3D	Mail-G52 核，支持 OpenGL ES 3.2	Mail-G52 核，支持 OpenGL ES 3.2
GPU 2D	支持 ARGB/RGB888 等多种格式，裁剪、旋转等操作	支持 ARGB/RGB888 等多种格式，裁剪、旋转等操作
NPU	1TOPs	1TOPs
视频解码	H.265 4K@30fps	H.265 4K@30fps
视频编码	H.264 1080p@60fps	H.264 1080p@60fps
最高显示分辨率	2048 x 1080@60Hz	2048 x 1080@60Hz
LCD 接口（RGB）	可提供方案支持	可提供方案支持
LVDS/MIPI DSI	1 路 4 lanes	1 路 4 lanes
MIPI CSI	2 路 4 Lanes	2 路 4 Lanes
电阻触摸屏	可提供方案支持	可提供方案支持
电容触摸屏	可提供方案支持	可提供方案支持
USB3.0 OTG	1 路	1 路
USB2.0 HOST	1 路	1 路

型号	M3562-2GF16GI-B	M3562-4GF32GI-B
以太网	1 路千兆，1 路百兆	1 路千兆，1 路百兆
SDIO	2 路	2 路
CAN BUS	2 路	2 路
UART	8 路（最高 10 路）	8 路（最高 10 路）
IIC	3 路（最高 6 路）	3 路（最高 6 路）
IIS	1 路（最高 3 路）	1 路（最高 3 路）
PCIE2.0	最高 1 路	最高 1 路
SPDIF	最高 1 路	最高 1 路
PDM	最高 1 路	最高 1 路
SPI	最高 3 路	最高 3 路
PWM	最高 16 路	最高 16 路
ADC	最高 16 路	最高 16 路
GPIO	最高 108 路	最高 108 路
RTC 时钟	提供接口驱动支持	提供接口驱动支持
独立硬件看门狗	支持	支持
供电电压 (PM362 电源模块)	3.3V	3.3V
机械尺寸	35mm x 40mm	35mm x 40mm
环境测试	-40℃ ~85℃	-40℃ ~85℃
评估底板	M3562-EV-Board （搭载 M3562-4GF32GI-B 核心板）	M3562-EV-Board （搭载 M3562-4GF32GI-B 核心板）
核心板形态	BGA 封装	BGA 封装

目录

工业自动化

新能源及汽车通讯

边缘计算

云平台

互联互通

致远微电子芯片

【产品应用】ZMC600E运动控制器直线和圆弧插补算法详解

ZLG 致远电子 2025-01-20 11:36:33

想深入了解 ZMC600E 运动控制器的插补算法吗？空间直线插补适用于精确定位，圆弧插补提供平滑曲线过渡。合理应用这些算法，能提升多轴协同能力，保障工作精度与可靠性，让工业自动化更高效、更智能。

ZMC600E 是广州致远电子股份有限公司开发的最新一代高性能智能总线型控制器，是面向工厂智能化时代的机器人控制器。ZMC600E 采用多核异构应用处理器为核心，其内核包括 2 个 64 位的 Arm®Cortex®-A53 核，主频 1.0GHz；4 个 Cortex®-R5F 内核，主频 800MHz。同时板载 1GB DDR4、4GB eMMC 以及 32KB FRAM。

ZMC600E EtherCAT 主站控制器为了满足不同的自动化应用需求，在硬件接口上面，精心设计了 1 路 EtherCAT 主站接口，ms 周期任务抖动 ±5μs 以内；3 路以太网接口，其中 1 路是千兆网口；2 路 RS485；2 路 CAN；32 个用户 I/O，其中 16 路 DI，16 路 DO，输入支持最多 2 路正交编码或两路脉冲计数；1 个 USB3.0 接口；支持 TF 卡；系统电源采用高稳定隔离电源，支持掉电检测；提供多种程序加密手段，保护用户应用软件知识产权。

ZMC600E 示意图及接口图如下所示：

图1 产品示意图

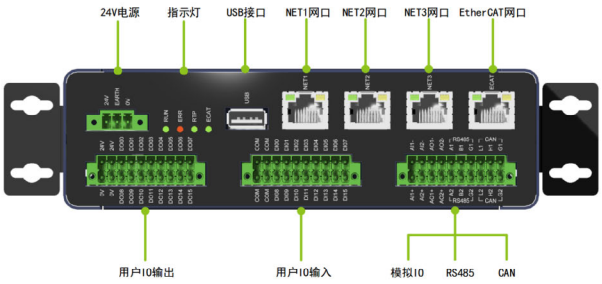


图2 ZMC600E接口图

ZMC600E EtherCAT 主站控制器为设备制造商提供多轴数、多 IO 点数控制的新一代可靠智能的运动控制解决方案，提供空间直线、圆弧插补算法，可以广泛应用于注塑行业、冲压行业、车床行业、搬运码垛、关节机器人、喷涂、玻璃机、压铸机、包装设备、3C 设备、锂电池设备、纺织、流水线工作站、非标自动化装备、特种机床等高端设备应用。

插补的概念

插补是数控机床中的一项关键技术，它通过特定的算法精确确定刀具或机械部件的运动轨迹。这一过程的核心目标是根据预定的运动要求，实时计算出各个坐标轴的运动指令，驱动执行部件协调地运动，确保机械部件能够沿着理想的轨迹和速度精确移动。插补算法的基本原理涉及数字计算，它不断地生成各轴的进给指令，使得在运动过程中各轴能够同步且协调地运动，从而确保设备能够精确地完成预定的路径。

插补通常涉及至少两个轴的协同工作。首先，通过建立坐标系，将运动轴映射到相应的坐标系中。然后，运动控制器依据插补算法来控制各个轴的运动，实现所需的轨迹。在常见的插补类型中，直线插补用于实现两个点之间的直线运动，而圆弧插补则用于实现沿圆弧轨迹的运动。这两种插补都依赖于算法的实时计算，以保证轨迹运动的精度和平稳性。简而言之，插补是数控系统中确保机械部件按照预定轨迹精确、高效运动的一种技术。

直线插补

以二维空间简化说明，如图 1 所示，直线插补运动：由起始点处沿 X 方向走一小段（给一个脉冲当量轴走一段固定距离），发现终点在实际轮廓的下方，则下一条线段沿 Y 方向走一小段，此时如果线段终点还在实际轮廓下方，则继续沿 Y 方向走一小段，直到在实际轮廓上方以后，再向 X 方向走一小段，依次循环类推，直到到达轮廓终点为止。空间直线适用于任意维空间的直线，插补方法也是一样的道理。

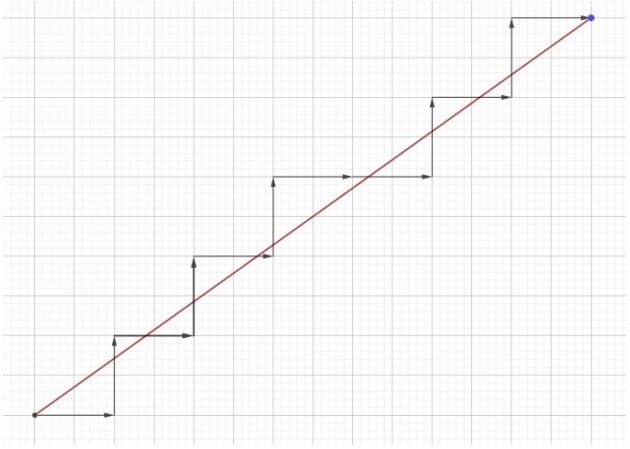
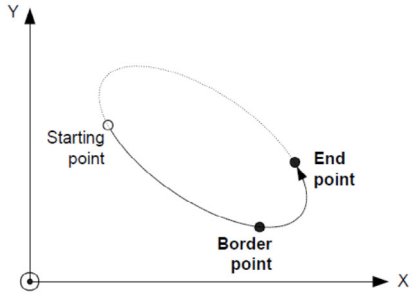
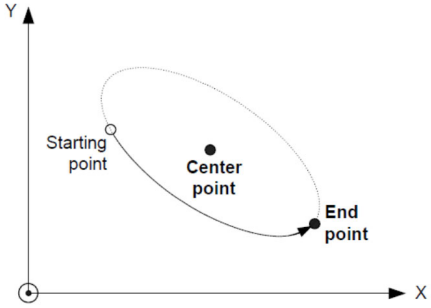


图3 直线插补示意图

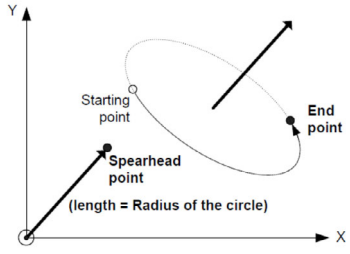
空间圆弧默认为三维，构造圆弧时，方向逆时针为正、除起始点和终点外，辅助参数输入分为三种：



a)辅助点为边上的点



b)辅助点为圆心



c) 辅助点 与原点形成的向量为平面法向量，向量长度为半径坐标

图5 空间圆弧构造

1. 辅助点为圆弧上的点，其优势在于，路径方向唯一，且实际辅助点可以通过示教获得。

2. 辅助点为圆心点，正常情况下是有两个方向的解，因此需要额外指定方向。但与起点、终点的夹角不能是 180 度，否则有无数解；缺点在于圆心处于障碍物区间时无法通过示教获得。

3. 辅助点为与与原点形成的向量为平面法向量，向量长度为半径坐标，方向根据右手原则。但已知半径下，圆心在它们的中垂线上，同样有两个解。算法会选择起点和终点之间距离较短的那个圆，即总的运动角度最多为 180 度。缺点是圆弧角度必须小于或等于 180°。

插补算法API

ZMC600E 运动控制器提供了直线和圆弧插补 API 接口，方便用户进行插补运动的设置和调整。

创建直线轨迹规划

指令原型	ZMC_DLLPORT trajectory_handle trajectory_new_line(uint32_t axis_num)
详细说明	创建轨迹规划——空间直线
参数解释	axis_num 轴的数量，1~N
返回值	返回创建的轨迹句柄

创建圆弧轨迹规划

指令原型	ZMC_DLLPORT trajectory_handle trajectory_new_arc(uint32_t type,double*auxPoint, int isClockwise)
详细说明	创建轨迹规划——圆弧
参数解释	type：0，辅助点为圆弧上的点； 1，辅助点为圆弧的中心点； 2，辅助点，与原点形成的半径长度的法向量的点。 auxPoint:辅助点，默认是 3 维{x,y,z} pathChoice：0 为逆时针，1 为顺时针，仅 type 为 1 时有效
返回值	返回创建的轨迹句柄

删除轨迹规划的句柄

指令原型	ZMC_DLLPORT void trajectory_delete(trajectory_handle handle)
详细说明	删除轨迹规划的句柄
参数解释	handle：轨迹规划句柄
返回值	void

设置起始位置

指令原型	ZMC_DLLPORT int32_t trajectory_set_start_pos(trajectory_handle handle, const double *startPoint)
详细说明	设置起始位置
参数解释	handle：轨迹规划句柄 startPoint：起始点
返回值	返回错误码

设置终点位置

指令原型	ZMC_DLLPORT int32_t trajectory_set_end_pos(trajectory_handle handle, const double *endPoint);
详细说明	设置起始位置
参数解释	handle：轨迹规划句柄 endPoint：终点
返回值	返回错误码

开启轨迹规划

指令原型	ZMC_DLLPORT int32_t trajectory_set_plan(trajectory_handle handle);
详细说明	用于规划，用于建立距离与空间点的关系
参数解释	handle：轨迹规划句柄
返回值	返回错误码

获取当前轨迹规划经过的总距离

指令原型	ZMC_DLLPORT double trajectory_get_plan_distance(trajectory_handle handle);
详细说明	获取当前轨迹规划经过的总距离
参数解释	handle：轨迹规划句柄
返回值	返回规划的总距离

获取某个经过距离点的规划信息

指令原型	ZMC_DLLPORT void velocity_get_plan_slice(velocity_handle handle, double slice, double *pos);
详细说明	通过经过的距离点来获取当前轨迹规划的切片，返回该点瞬时的位置
参数解释	handle：轨迹规划句柄
	slice：切片点，表示相对于起点的距离，有效的范围为：0~规划总距离，当输入超限时，会自动限制在范围内
	pos： 返回瞬时点的位置
返回值	void

示例

1. 直线插补示例：起始点 { 0, 10, 20 }，终点 { 10, 0, -20 }

代码如下：

```
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include<iostream>
#include "trajectory.h"
int main()
{
 double startPos[3] = { 0, 10, 20 };
 double endPos[3] = { 10, 0, -20 };

 trajectory_handle handle = trajectory_new_line(3);
 trajectory_set_start_pos(handle, startPos);
 trajectory_set_end_pos(handle, endPos);
 trajectory_set_plan(handle);

 double dist = trajectory_get_plan_distance(handle);
 for(double d = 0; d <= dist; d+=0.1)
 {
  double pass_dist[3];
  trajectory_get_plan_slice(handle, d, pass_dist);
  std::cout << "pass_dist :" << pass_dist[0] << "," << pass_dist[1]
<< "," << pass_dist[2] << endl;
 }
 trajectory_delete(handle);

 return 0;
}
```

执行结果如图下图所示：

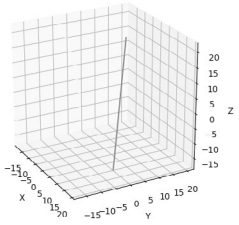


图6 直线插补轨迹

2. 圆弧插补示例：

起始点 { -10,0,10 }，终点 { 10, 0, 5 }, 经过圆弧点 {0, 10, 7.5}

代码如下：

```
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include<iostream>
#include "trajectory.h"
int main()
{
 double startPt[3]= { -10, 0, 10 };
 double auxPt[3] = { 0, 10, 7.5 };
 double endPt[3] = { 10, 0, 5 };

 trajectory_handle handle = trajectory_new_arc(0, auxPt, 0);
 trajectory_set_start_pos(handle, startPt);
 trajectory_set_end_pos(handle, endPt);
 trajectory_set_plan(handle);

 double dist = trajectory_get_plan_distance(handle);
 for (double d = 0; d <= dist; d+=0.1)
 {
  double pass_dist[3];
  trajectory_get_plan_slice(handle, d, pass_dist);
  std::cout << "pass_dist :" << pass_dist[0] << "," << pass_dist[1]
<< "," << pass_dist[2] << endl;
 }
 trajectory_delete(handle);
 return 0;
}
```

执行结果如图下图所示：

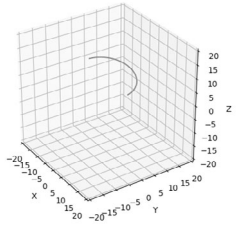


图7 空间圆弧插补示例

ZMC600E 运动控制器 配备了直线和圆弧插补算法，这些算法能够灵活应对各种应用需求，确保轨迹的精确性。掌握这些运动插补技术，有助于提高设备的控制精度和工作效率，为工业生产带来显著的效益提升。

【技术分享】手把手带你入门AWStudio运动控制编程

ZLG 致远电子 2025-02-10 11:35:29

面对一台 ZMC600E 运动控制器与多台伺服电机，我们该怎样让它工作起来？本文带你了解 PLCOPen，并详细讲解如何使用 AWStudio 控制电机运动。

在自动化领域，运动控制是一件很复杂的事情，包含了上位机通讯，工业现场总线通信协议，运动控制算法，伺服电机控制等领域知识。致远电子自主研发的 AWStudio 的 AWPLC 解决方案可以快速实现运动控制编程。

AWStudio 的逻辑组态提供了符合 IEC61131-3 标准的 ST 语言编程环境，该开发环境和运行时均是由 ZLG 致远电子自主研发，提供了带智能提示的代码编辑器，以及多种程序调试手段，让用户可以轻松开发 PLC 程序。

AWPLC 提供了 PLC 的运行时环境，可以运行用户开发的 PLC 程序（开发环境生成的二进制文件），进而控制硬件 IO 或电机伺服器等按程序逻辑进行工作。AWPLC 运行时环境是一个软 PLC 的引擎，需要在一个硬件载体上运行，比如 ZLG 致远电子的运动控制器，或者 PC 机等。

PLCOPen 是总部在欧洲的 IEC61131-3 推广团体，目的是为了使得不同 PLC 厂家的产品应用编程通用化，消除产品技术差异和壁垒，在用户选用不同品牌 PLC 时，不必另外学习相应的编程方法。PLCOPen 对运动控制功能块 MC 的技术规格进行了约定，包括 MC 功能模块的命名、具体功能、输入输出变量定义、相关时序逻辑等，最大限度地保证用户编程技术的互通。

AWStudio 提供了支持 PLCOPen 规范的运动控制接口，包含了 MC_Power、MC_MoveAbsolute、MC_MoveRelative、MC_Reset 等单轴功能块。接下来，我们介绍如何通过 AWStudio 快速地创建 ZMC600E 应用示例：控制 ZMC600E 打印字符，通过 EtherCAT 控制电机转动。

编译环境搭建

首先需要搭建编译环境，准备硬件和软件。

1. 必备硬件

x86 架构 Windows 系统的个人电脑；
ZMC600E 运动控制器；
支持 EtherCAT 协议的电机驱动器和伺服电机。

2. 软件工具

电脑端安装 AWStudio 软件；
ZMC600E 端安装 awplc 运行时。

配置电脑与 ZMC600E 设备的连接：使用网线连接 windows10 电脑端网络端口与 ZMC600E 运动控制器的 NET2 网络端口， 在电脑上配置在同一个网络子网。

配置 ZMC600E 设备与从站的连接，使用网线连接 ZMC600E 运动控制器的 ECAT 网络端口与从站网络端口。

新建工程

双击 AWStudio 图标启动程序。

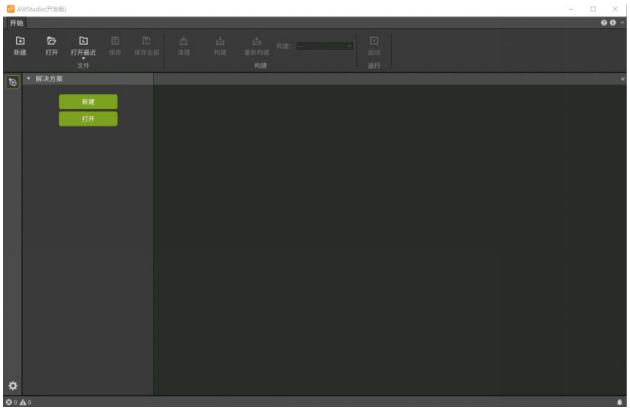


图1 AWstudio界面图

图2 新建解决方案对话框图

1. 单击工具栏的【新建】按钮。
2. 选择【运动控制器】-【AWPLC 编程】解决方案。
3. 修改解决方案名称。
4. 点击【确定】按钮。

图3 新建解决方案对话框图

选择控制器型号为 ZMC600E，点击【确定】按钮。

图4 选择运动控制器型号图

打开了空白项目。

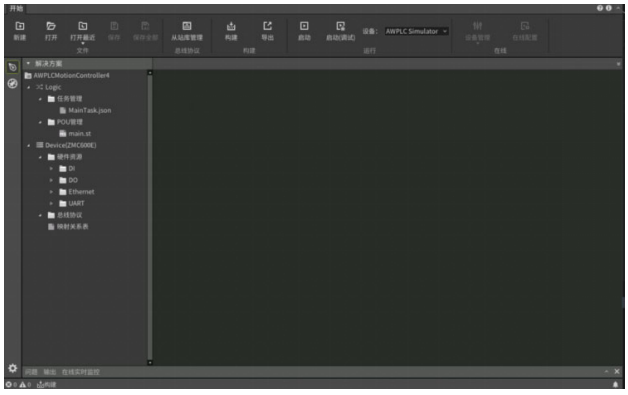


图5 打开空白控制器项目

编写简单脚本

我们尝试编写一个简单的 st 脚本，脚本功能是打印字符串。新建一个整型变量。

1. 单击 main.st 激活编辑页
2. 单击变量新建的【+】按钮
3. 输入变量名
4. 单击类型选择下拉按钮
5. 单击以选择类型
   点击【确定】按钮

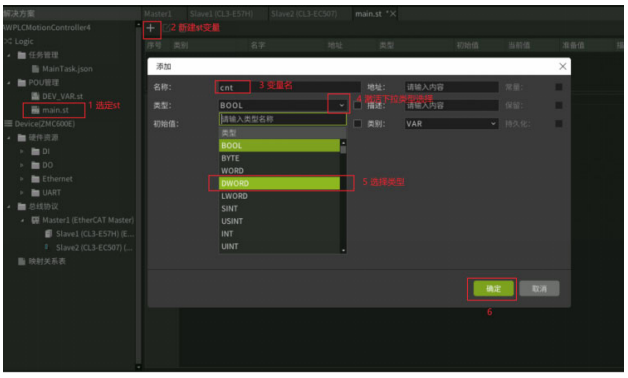


图6 新建变量对话框

编写 st 脚本，包含两条语句：计数递增，打印计数。

```
PRINT_STR('cnt = ' + DWORD_TO_STRING(cnt));
cnt := cnt + 1;
```

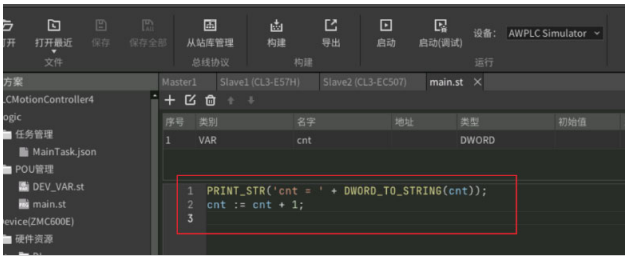


图7 代码编辑图

POU 调度程序会在每个周期调用 main.st 程序。

仿真运行程序

AWStudio 提供了仿真模式，无需连接目标设备，程序可以运行于 windows 系统的仿真器。

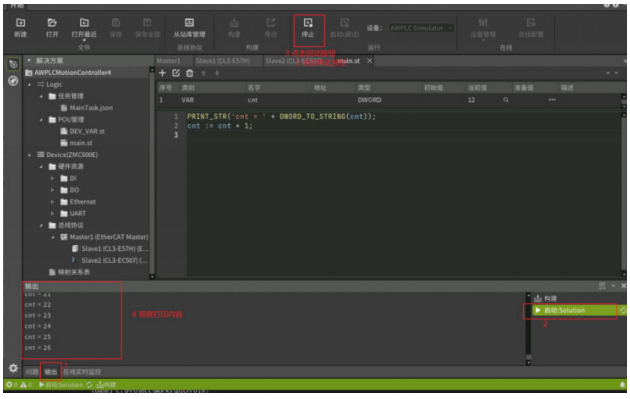


图8 工程启动过程图

1. 点击界面下方的【输出】按键。
2. 选择【输出】-【启动 Solution】子页。
3. 点击工具栏的【启动】按钮，【启动】按钮切换成【停止】按钮。；
4. 观察【输出】文本框，将会显示打印内容。
5. 点击电机工具栏的【停止】按钮。

可以在输出 - 启动区域看到打印结果，计数每次累加 1 并打印到屏幕。

在目标设备运行

连接 ZMC600E 设备运行。

1. 点击设备下拉按钮。
2. 电机刷新搜索设备按钮。
3. 单击选择设备。

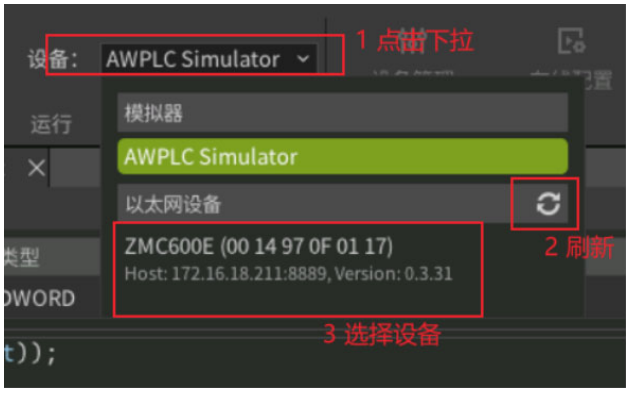


图9 设备连接示意图

输入管理密码，登录到设备。

图10 登录对话框图

登录成功时，下方状态栏会显示连接设备和对应的 MAC 地址。

在设备端运行 st 脚本。

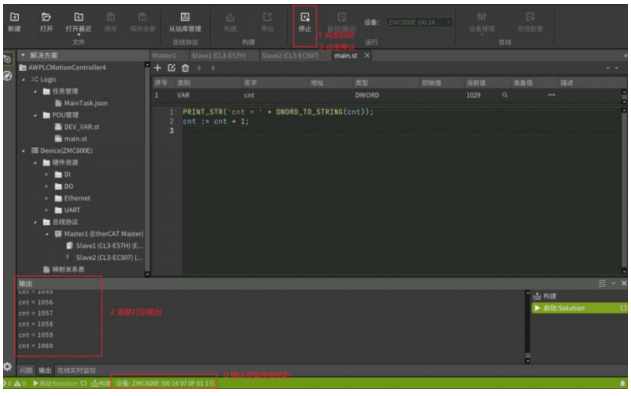


图11 设备端启动过程图

1. 点击工具栏的【启动】按钮，【启动】按钮切换成【停止】按钮。
2. 观察【输出】文本框，将会显示打印内容。
3. 点击电机工具栏的【停止】按钮。

可以在输出文本框区域看到打印结果，计数每次累加 1 并打印到屏幕。

在电脑端编写代码，单击一个按钮就能编译程序并在设备端运行程序，非常方便。实际上 AWStudio 在后台做了编译相关工作，把构建好的 st 程序打包发送到 ZMC600E 设备，并通知守护进程启动 st 脚本程序。

添加主站从站

ZMC600E 控制器通过 EtherCAT 协议连接了从站设备（电机驱动器）。首先添加一个 EtherCAT 主站总线，如下图。

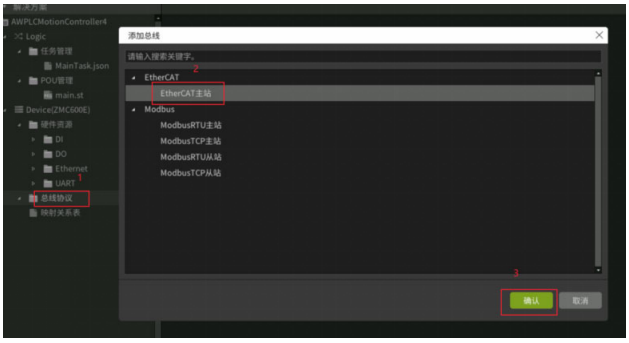


图12 添加总线对话框图

ZMC600E 充当主站控制器，需要配置总线网口。

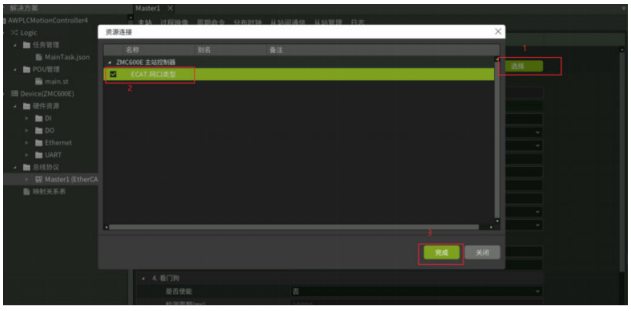


图13 资源连接对话框图

接下来添加从站设备。打开添加从站对话框。

图14 打开添加从站对话框图

搜索并添加从站。

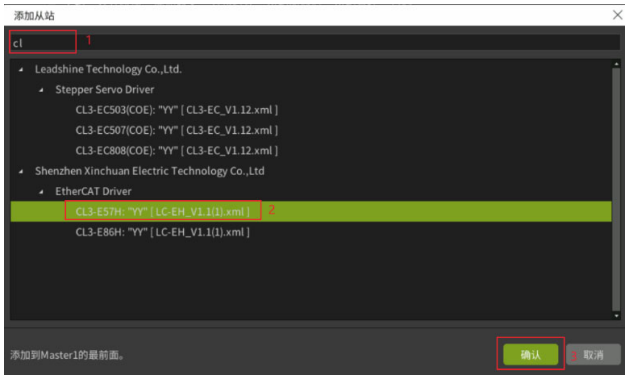


图15 添加从站图

从站添加成功，右边是从站详细配置页面。

图16 从站详细配置图

从站编辑页可以配置从站子页，变量子页，FMMU/SM 子页，邮箱子页，初始化命令子页，分布时钟子页。变量页包含了 PDO 详细信息，是从站的核心配置。AWStudio 提供了自动映射机制，可以自动拆包解析 PDO 变量，避免了用户直接处理繁琐又容易出错的 PDO。这里我们跳过配置流程，直接使用默认配置。

如果有多个从站设备，需要重复上述流程添加从站。

电机配置

打开电机配置页面。

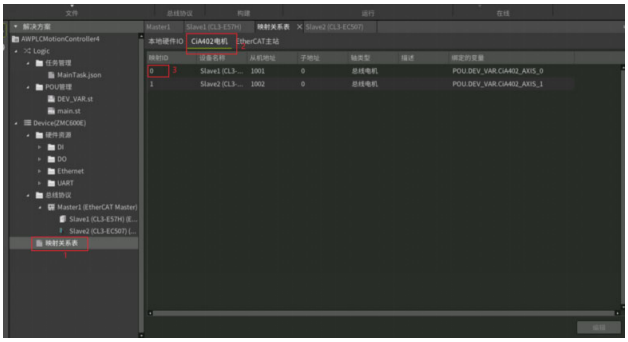


图17 Cia402电机配置图

映射关系页面，可以查看或修改绑定的变量名。第一个从机的电机变量名称默认为 “CiA402_AXIS_0”，该电机变量就是 PLCOPen 中的轴对象，后续我们写代码需要用到该变量。

图18 电机映射关系配置页面

通用页面，可以编辑轴安全配置。

图19 电机通用配置页面

轴的虚拟模式开关，支持在仿真模式下模拟轴运行，虚拟电机的控制模式和位置。仿真模式下能模拟 EtherCAT 主站和从站电机，便于用户不连接设备也能调试脚本。

切换到缩放 / 映射页面。默认脉冲比是 1：1，不需要特别配置。

图20 电机缩放/映射配置页面

如果缩放比分子调的比较大，则后面的代码中的运动功能块的位置和速度和加速度就需要调小。

电机编程

由于 AWStudio 提供了 PDO 映射，所以我们可以免于管理主站和解析变量。此外 AWStudio 提供了 PLCOPen 运动接口，用户可以方便地控制电机运动，避免直接管理电机控制和状态的字典变量。接下来我们使用 PLCOPen 接口编写电机使能代码和电机运动代码。

轴是对伺服电机的高级抽象，避免了直接处理从站电机的状态字、控制字、指令位置等对象字典项，支持轴使能、发布运动、轴状态查询等功能。

以下是轴控制功能块常用接口的简单介绍：

- MC_Power 控制轴使能（核心功能块）。
- MC_Reset 清除轴错误。
- MC_ReadStatus 查询轴状态。
- MC_ReadActualPosition 查询轴实际位置。
- MC_MoveAbsolute 轴运动到指定的绝对位置。
- MC_MoveRelative 轴基于当前位置按照偏移量运动。

操作要点：轴要先使能，才能驱动电机运动。电机存在错误时，要先清除错误，才能使能轴。

电机使能

首先添加多个功能块：轴状态查询，轴位置查询，轴错误复位，轴使能功能块。

然后编写一个简单的脚本，执行上述的功能块。

程序每运行 200 个周期，打印一次当前计数、实际位置、轴使能状态标志。

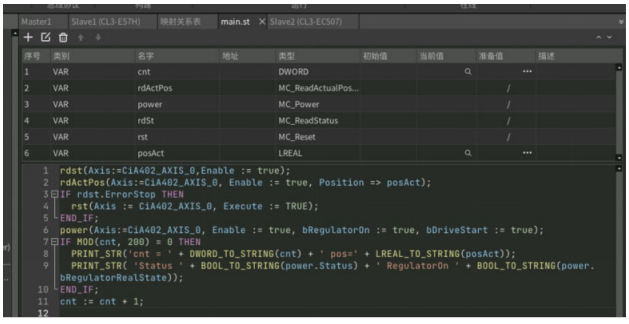


图21 代码编辑框图

脚本代码如下：

```
    rdst(Axis:=CiA402_AXIS_0,Enable := true);
    rdActPos(Axis:=CiA402_AXIS_0, Enable := true, Position => posAct);
    IF rdst.ErrorStop THEN
      rst(Axis := CiA402_AXIS_0, Execute := TRUE);
    END_IF;
    power(Axis:=CiA402_AXIS_0, Enable := true, bRegulatorOn := true, bDriveStart := true);
    IF MOD(cnt, 200) = 0 THEN
      PRINT_STR('cnt = ' + DWORD_TO_STRING(cnt) + ' pos=' + LREAL_TO_STRING(posAct));
      PRINT_STR( 'Status ' + BOOL_TO_STRING(power.Status) + ' RegulatorOn ' + BOOL_TO_STRING(power.bRegulatorRealState));
    END_IF;
    cnt := cnt + 1;
```

查看打印过程。

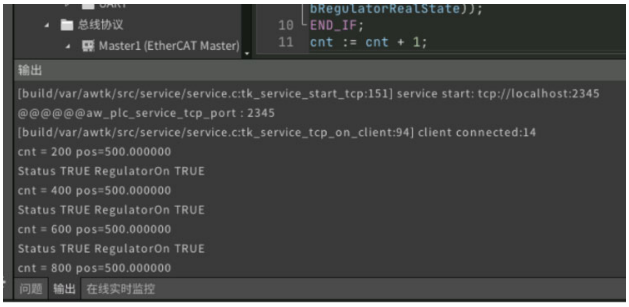


图22 电机使能过程打印图

可以看到，轴使能标志位开始时为假，然后为真，并打印了电机实际位置。

电机运动

电机使能成功，接下来就可以实际驱动电机转动。

首先添加绝对运动功能块，相对运动功能块对应的变量。

然后编写程序：

1. 等待轴使能状态标志为真。
2. 轴执行绝对运动。从当前位置运动到 100 单位的位置。
3. 绝对运动到达目标后，再执行相对运动。从当前位置，往前运动 200 的单位。

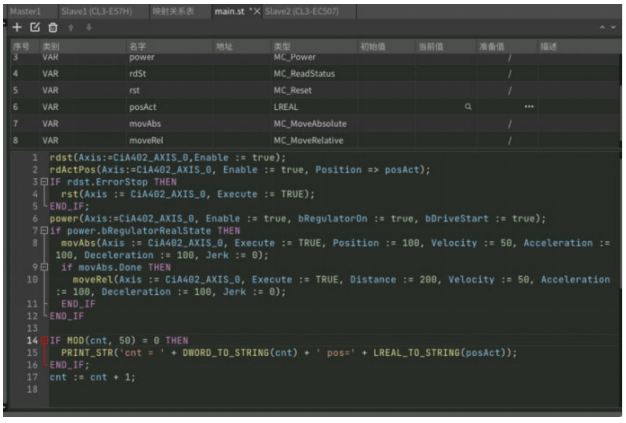


图23 代码编辑框图

代码中的位置，速度，加速度和减速度仅供参考，用户可以根据实际情况自行缩放。

脚本代码如下：

```
    power(Axis:=CiA402_AXIS_0, Enable := true, bRegulatorOn := true, bDriveStart := true);
    if power.bRegulatorRealState THEN
      movAbs(Axis := CiA402_AXIS_0, Execute := TRUE, Position := 100, Velocity := 50, Acceleration := 100, Deceleration := 100, Jerk := 0);
      if movAbs.Done THEN
        moveRel(Axis := CiA402_AXIS_0, Execute := TRUE, Distance := 200, Velocity := 50, Acceleration := 100, Deceleration := 100, Jerk := 0);
      END_IF
    END_IF

    IF MOD(cnt, 50) = 0 THEN
      PRINT_STR('cnt = ' + DWORD_TO_STRING(cnt) + ' pos=' + LREAL_TO_STRING(posAct));
    END_IF;
    cnt := cnt + 1;
```

查看打印过程。

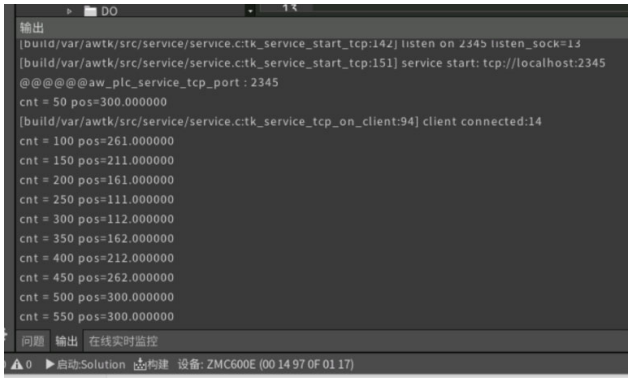


图24 运动轮廓打印图

查看打印过程，可以得到大概的运动轮廓：初始时轴在 300 单位位置，然后轴从 300 单位运动到 100 单位位置，再从 100 单位位置运动到 300 单位位置，然后停留不动。

ZMC600E运动控制器

ZMC600E 是广州致远电子股份有限公司开发的最新一代高性能智能总线型控制器，是面向工厂智能化时代的机器人控制器。ZMC600E 采用多核异构应用处理器为核心，其内核包括 2 个 64 位的 Arm®Cortex®-A53 核，主频 1.0GHz；4 个 Cortex®-R5F 内核，主频 800MHz，同时板载 1GB DDR4、4GB eMMC 以及 32KB FRAM。

ZMC600E 示意图及接口图如下所示：

图1 产品示意图

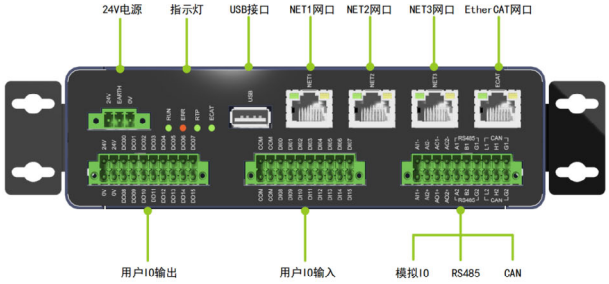


图2 ZMC600E接口图

ZMC600E EtherCAT 主站控制器为设备制造商提供多轴数、多 IO 点数控制的新一代可靠智能的运动控制解决方案，提供空间直线、圆弧插补算法，可以广泛应用于注塑行业、冲压行业、车床行业、搬运码垛、关节机器人、喷涂、玻璃机、压铸机、包装设备、3C 设备、锂电池设备、纺织、流水线工作站、非标自动化装备、特种机床等高端设备应用。

关节电机驱动，解锁智能机器人 的“灵动” 密码

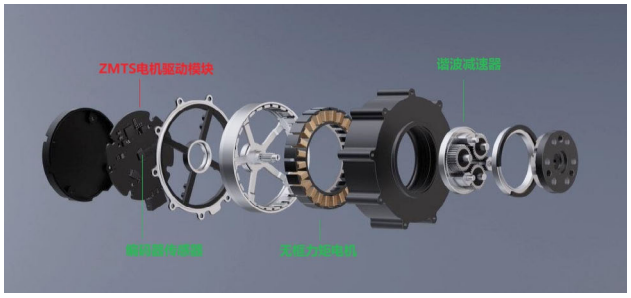
ZLG 致远电子 2025-03-13 11:35:22

在智能机器人与高端设备的世界里，关节电机驱动是赋予它们灵动身姿的关键。它集成电机、减速器、传感器和驱动电路，精准控制运动。本文将深入剖析其核心技术，揭秘两款高性能驱动模块，一同探寻智能运动的奥秘。

前言

什么是关节电机？ 简单来说，它就好像人体的关节一样，可以精确控制机器人的运动姿态，是机器人的核心动力部件。它将电机、减速器、传感器和驱动电路集成在一起，能够实现高精度的旋转或直线运动，是四足机器狗、人形机器人、协作机械臂等设备的动力核心部件。

关节电机：运动设备的“心脏”



关节电机主要由以下四大核心组件构成，既保证了高扭矩密度，又实现了紧凑的空间布局：

1. 无框力矩电机

一般为外转子式无刷直流电机，电机的转子直接嵌入设备关节轴，定子与负载刚性连接，这样就可以省去了传统外壳，大幅减小体积。其扭矩密度高达 35N·m/kg，非常适合仿生机器人、协作机械臂等对空间敏感场景。电机常用电压为 24V~48V，功率范围为 50W~1000W。

2. 谐波减速器

采用柔轮与波发生器的弹性啮合结构，减速比为 30:1~160:1，传动效率超过 85%，实现零背隙动力输出，重复定位精度可达 1 弧分以内。

3. 高精度编码器

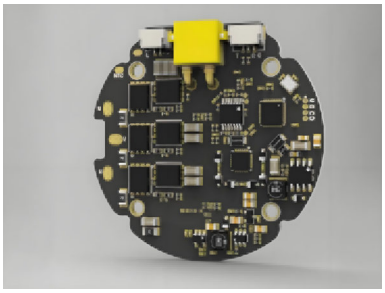
集成 17 位（或更高精度）的绝对值编码器，可实时反馈位置和转速信息，为闭环控制提供精准数据。由于电机通过谐波减速器驱动负载时，电机和负载之间可能存在非线性位置误差，因此部分高精度关节电机会在电机和负载端各安装一个编码器。

4. 电机驱动模块

关节电机一般通过 EtherCAT、CAN 或 RS485 与主站通信。对于协作机器人等对实时性要求高的应用，通常采用 EtherCAT 总线并执行 CIA402 运动控制协议；而对于机器狗和人形机器人等对响应速度要求高的场景，则常采用 MIT 或厂家自定义的简易通讯协议。

驱动模块，精准定位

ZLG 致远电子推出了两款关节电机驱动模块，专为高端运动控制设计，兼容性强、稳定性高，适用于工业和消费级场景，可轻松嵌入关节电机。



1. ZMTS-EB1200-JM（EtherCAT版）

适合高实时性、高位置精度的应用，如协作机器人。

宽压动力：24V~48V 输入，7A 输出，直径 58mm，适配非穿孔电机。

双编码器：内置 21 位绝对值编码器，可扩展 SSI 或正交编码器。

实时通信：工业级 EtherCAT 总线，同步时间≤ 256μs。

协议兼容：支持 CIA402 标准协议（CSP/CSV/CST/PP/PV/HM 模式）。

运动黑科技：双环 ADRC 位置控制，结合“前馈 + 反馈”纠偏，显著提升抗干扰能力和定位精度。

2. ZMTS-CB1200-JM（CAN版）

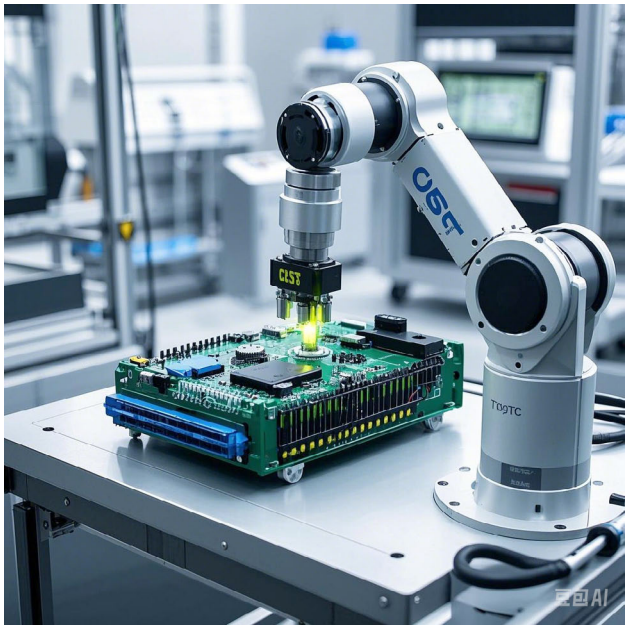
适合响应速度快、位置精度要求不高的应用，如机器狗、人形机器人、摄影云台等。

- 宽压动力：12V~48V 输入，7A 输出，直径 58mm，适配非穿孔电机。
- 单编码器：内置 21 位绝对值编码器。
- 可靠通信：基于 CAN 总线，同步时间≤ 2ms，适应复杂电磁环境。
- 协议兼容：无缝适配 MIT 机器狗协议，支持运控 / 位置 / 速度 / 扭矩模式。
- 温控黑科技：通过电流和电阻实时估算电机温度，响应快、精度高，有效防止电机过热。
- 两款驱动模块各具特色，满足不同场景需求，助力机器人运动控制更高效、更智能。

应用场景，解锁智能未来

1. 工业协作臂 — 转矩模式下的安全交互

ZMTS-EB1200-JM（EtherCAT 版），在 CIA402 的转矩模式（CST）下，驱动模块实时感知外部阻力，遇碰撞时自动降低输出力矩，保护人员与设备安全，适用于精密装配、医疗辅助等场景。



2. 四足机器狗 — 速度模式下的动态适应

ZMTS-CB1200-JM（CAN 版）在废墟救援等复杂环境中表现出色。在运控模式下，机器狗能够快速调整步频，轻松跨越碎石、斜坡等地形。相比传统的三环位置控制，其稳定性大幅提升，让机器狗在复杂环境中行动自如。

3. 超稳云台 — 平顺且灵活

ZMTS-CB1200-JM（CAN 版）在摄影稳拍器中也有出色表现。在运控模式下，它可以灵活配置响应速度和位置精度的关系，确保画面始终保持稳定输出。无论是在连续拍摄还是复杂运动场景中，都能让画面平滑流畅，满足专业影视创作的需求。

技术赋能，驱动无限想象

从工业生产线到家庭服务，从极限救援到影视创作，关节电机驱动板正在不断突破“智能运动”的边界。我们以算法创新为核心，以可靠性为基石，为开发者提供“开箱即用”的解决方案。无论是科研机构、机器人厂商，还是创客团队，都能在这里找到最适合自己的动力驱动方案。

【新品发布】ZIO系列插板式电机驱动&IO模组，按需灵活设计您的IO系统

ZLG 致远电子 2025-03-24 11:32:00

ZIO系列插板式模组

ZIO 系列插板式 I/O 模块是 ZLG 开发的可灵活设计的远程 I/O 扩展模块。该系列产品由耦合器、数字 I/O、电机驱动、模拟量、电源等功能模块组成。

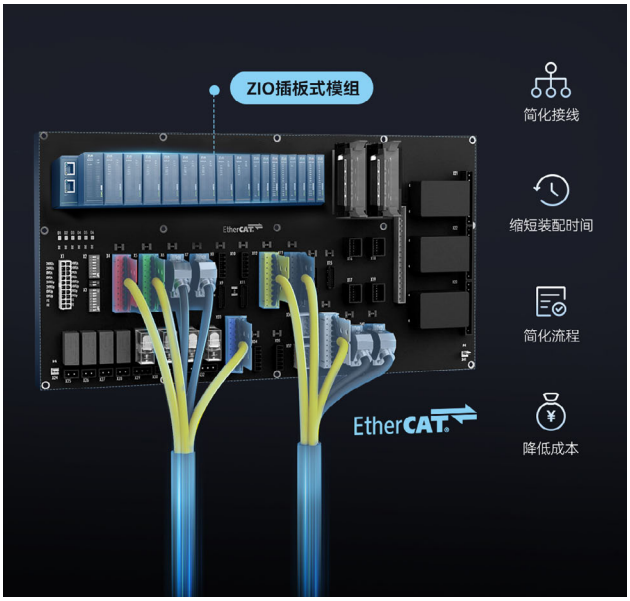
ZIO 系列可以通过定制化的底板集成各类接口，搭配多种功能模块，能够实现小型化的同时通过预制线缆的方式减少接线，从而减少设备安装调试的人工成本。

预制分线底板 即插即用

ZIO 系列模组采用 EtherCAT 总线通讯，通过独特的结构设计，可以插入预制专用的分线底板中，分线底板可按需定制接口种类及数量，并通过编码好的组件实现即插即用。

适用于批量标准化生产

ZIO 插板式模组和传感执行层的连接组件可以灵活布置在分线底板上，用线束避免了接线错误，用防呆接口连接器防止插错，可以简化接线工作，同时极大缩短电气装配时间，简化工作流程，降低批量的安装生产成本。



模块种类丰富

具有 EtherCAT 耦合模组、电机驱动模组、DI\DO 模组、AI\AO 模组以及模组电源等功能模组可供选择。

工业自动化 ▼

结构紧凑、体积小巧

ZIO 插板式模组结构非常紧凑，其体积几乎为常规卡片式模块的一半，可以有效节省设备控制箱的空间。

选型表

模块功能	型号	规格
耦合器	ZCPC-80801	EtherCAT 总线耦合器，IN/OUT 双网口，含 4.5A 电源
分线底板	ZMB-IO-6	EtherCAT 总线耦合器 +6 个单排模组或 3 个双排模组
	ZMB-EF1200-4	EtherCAT 总线耦合器 +4 个 ZMTC-EF1200 步进伺服模组
	ZMB-EF1200-8	EtherCAT 总线耦合器 +8 个 ZMTC-EF1200 步进伺服模组
电机模组	ZMTC-EF1200	EtherCAT 总线型，两相步进，48V/5A, 开环或编码器闭环可配
数字量输入模块	ZIOC-E1600DN	16 通道数字量输入，NPN，0-7V
	ZIOC-E1600DP	16 通道数字量输入，PNP，11-30V
数字量输出模块	ZIOC-E0016DN	16 通道数字量输出，NPN，单路 0.5A
	ZIOC-E0016DP	16 通道数字量输出，PNP，单路 0.5A
模拟量输入模块	ZIOC-E0800AI	8 通道模拟量输入，电流型，0-20mA
	ZIOC-E0800AU	8 通道模拟量输入，电压型，0-10V
	ZIOC-E0800AU1	8 通道模拟量输入，电压型，2x(0-10V) +6x(-10-+10V)
模拟量输出模块	ZIOC-E0008AU	8 通道模拟量输出，电压型，0-10V
通讯电源	ZPWC-240304	EtherCAT 模组电源，3.3V/4.5A

* 提供多种分线底板参考设计，可直接生成，或按需修改

行业应用

如需了解更多产品详情，可填写申请表单，
我们会有专人与您联系。

点击申请

从CANTest到ZCANPRO再到ZXDoc，他们有话说！

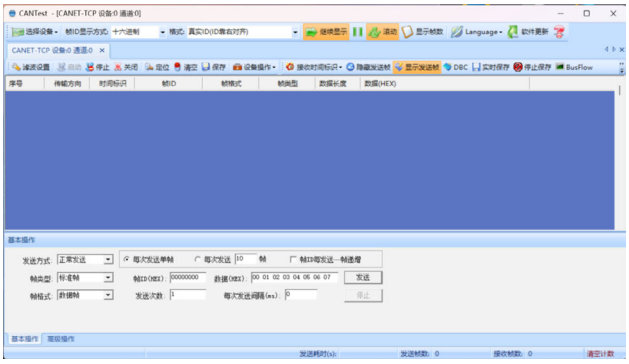
ZLG 致远电子 2025-03-31 11:37:30

横跨二十余载，周立功 CAN 接口卡软件经历了一代又一代的更迭，仿佛青春就在一刹间。

CANTest：余热生辉

我生于 21 世纪初，年轻时候我可厉害了！几乎国内所有工业通讯的工程师都知道我，他们依靠我和冰冷的工业节点沟通交流（CAN 数据收发），心情不好时不想听的内容我也可以给他们屏蔽（滤波），需要时候还能给他们充当翻译（DBC 解析），大家都把我当成朋友，成为了彼此的工作伙伴。

我在 2019 年就正式退休啦！不过承蒙大家厚爱，时至今日，闲暇时我依旧和曾经的老战友们奋斗在一线！但我深知岁月不饶人，时代的洪流急促的推着我这身老骨头前进，是时候退下了，未来属于更有能力的年轻人们。



USB接口CANFD卡

点击购买

ZCANPRO：中流砥柱

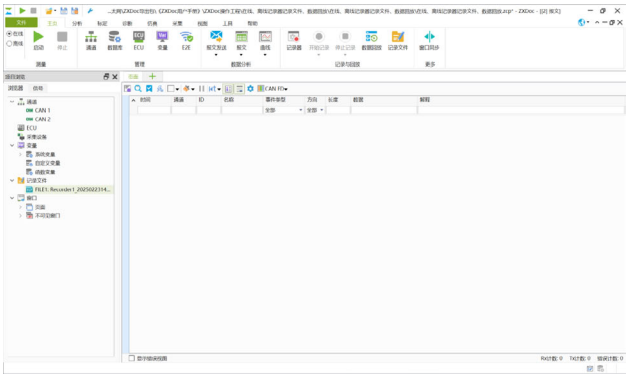
接过前辈 CANTest 的衣钵，我撑起了 21 世纪周立功工业软件的第二个十年。2011 年 10 月我发布了，随着国内工业体系的发展，我支持了更高通讯速率的 CANFD 总线，同时为了满足日益增长的总线升级功能需求，应用层的 UDS 诊断和 ECU 刷写等更是不在话下，逐渐我的风头超过了前辈，成为了彼时的中流砥柱，也是从我开始，二次开发接口 zlgcan 的适配性和稳定性得到了进一步加强。

迈入 21 世纪的第三个十年，汽车时代袭来，我的时代也将终结，但我会继续坚守岗位，服役着万千并肩的伙伴！我也会将圣火继续传递下去！

ZXDoc：未来可期

我生于“软件定义汽车”时代，再加上家里多年的硬件基本功，组成了新时代“硬件 + 软件”的双杰，背负着汽车总线测试的使命。我继承了前辈们的优良基因，并且不断开拓创新，攻克了 XCP/CCP 标定、仿真、SOME/IP、DoIP 等多种长期被国外厂商垄断的软件功能，且完全免费开放至各位用户，我深知，十年磨一剑，属于我的时代悄然来临了！

站在时代的潮头回望，每一次技术革新都如星辰点亮长夜，即便当下或许面临挑战与未知，但正是这些探索的足迹，铺就了通往更广阔世界的道路。青年一代以知识与热忱为帆，在数字浪潮中破浪前行，用创新重新定义边界，那些奋斗的日夜都在诉说着同一个信念：未来可期。



总结

ZLG 致远电子致力于全球车载总线发展，提供极致的产品和服务，构建车载总线安全保障体系。欢迎持续关注我们，接下来会连载汽车软件 ZXDoc 的功能说明。更多产品了解：

https://www.zlg.cn/carbustools/carbustools/product/id/382.html

边缘计算 ▼

ZLG嵌入式笔记(连载16) | CAN通信节点多时，如何减少寄生电容和保障节点数量？

ZLG 致远电子 2025-01-03 11:39:48

在汽车电子与工业控制等领域，CAN 通信至关重要。本文围绕 CAN 通信，阐述节点增多时如何减少寄生电容的策略，同时从发送、接收节点等方面，讲解保障节点数量及通信可靠性的方法。

如何减少寄生电容？

增加节点就会带来寄生电容的增加，节点增加到一定数量，波形严重失真，导致数据接收错误。

硬件设计 CAN 电路时，需要总线抗受电磁兼容同时需要寄生电容小，直接给总线并联 TVS 瞬态抑制二极管，这些就可以？答案肯定是不行的，因为 TVS 瞬态抑制二极管的结电容高达 1000pF 以上，完全可以使波形变形。

既要使用 TVS 瞬态抑制二极管，又不能使结电容很大，怎么解决？我们知道电阻并联阻值减少，电容并联容值增大。那么根据电容串联容值减少这一思路进行硬件设计，参考设计如图 1 所示。

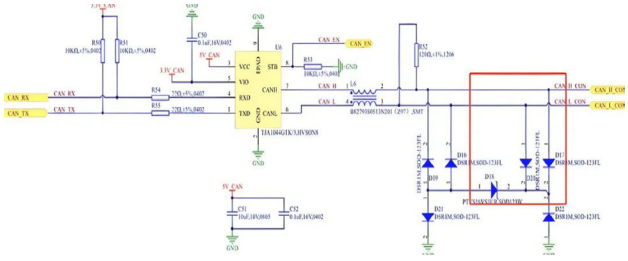


图1 CAN电路设计1

如图 1 所示的 CAN 电路设计，二极管和 TVS 瞬态抑制二极管都是串联的关系，利用的是二极管单向导电性和小结电容特性。如果觉得分立元器件太多，可以参考如图 2 所示的电路设计，同等效果。

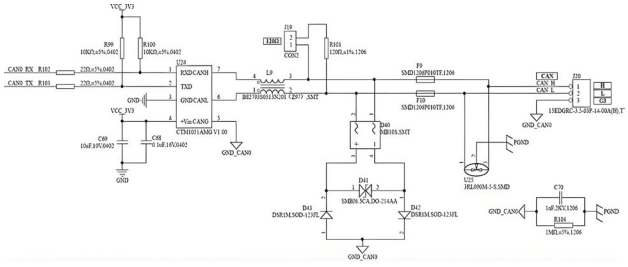


图2 CAN电路设计2

如何保障节点数量？

在 CAN-bus 电路设计中，理论上收发器支持节点数最多可做到 110 个，但实际应用中往往达不到这个数量。今天我们就来谈谈如何通过合理的 CAN-bus 总线设计，保证 CAN 网络中的通讯的可靠性和节点数量。

影响总线节点数的因素有多种，我们从满足接收节点的差分电压幅值方面来讨论，只有满足了这个前提条件，我们才能考虑总线的其他因素如寄生电容、寄生电感对信号的影响。

1. 发送节点的CAN接口负载，为何考虑CAN接口负载？

CAN 接口负载即为 CANH、CANL 之间的有效电阻值大小，该电阻会影响发送节点输出的差分电压的幅值，组网后网络中各个节点的负载电阻 RL 接近，如图 3 所示结果是我们测试了 CTM1051M 小体积 CAN 隔离模块在不同负载下的输出差分电压幅值。

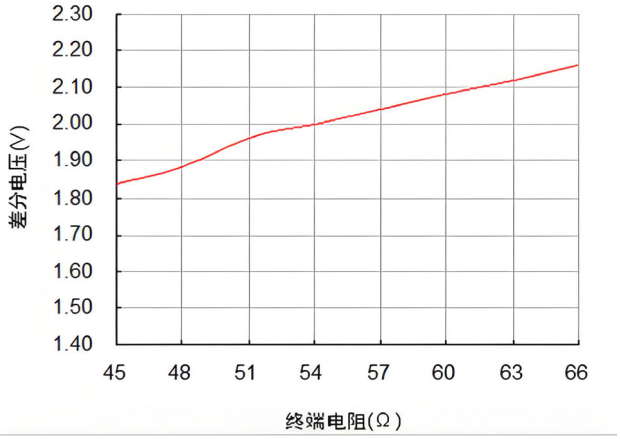


图3 不同负载下的差分电压

在负载电阻由 45Ω 不断增大到 66Ω 时，节点的输出差分电压也随着由 1.84V 增大到 2.16V，两者近似线性关系。为了使发送节点的输出差分电压不至于过低，实际组网时负载电阻应在图 3 的范围内波动。我们分析 RL 的组成有 3 个：终端电阻、总线节点的差分输入电阻、总线本身的有效电阻。

终端电阻：

总线两端均需要增加终端电阻，当总线距离长时，总线有效电阻大，损耗大，可以适当增加终端电阻值以减小总线有效电阻的损耗，如 150Ω~300Ω。

差分输入电阻：

ISO 11898 中规定的收发器差分输入电阻范围为 10kΩ~100kΩ 之间，CTM1051M 系列收发器的差分输入电阻为 19kΩ~52kΩ，其典型值为 30 kΩ，如果我们以最多节点组网，按典型值考虑，则整个总线的差分输入电阻会达到 30 kΩ/110=273Ω，与终端电阻并联时会显著增加节点的负载。

总线有效电阻：

使用较小截面积的双绞线，其有效电阻达到几十欧姆，长距离通信，总线对差分信号的影响会很大，如常用的 RVS 非屏蔽双绞线的电阻从 8.0Ω/km 到 39.0Ω/km 不等。严重时会使接收点的电平达不到识别范围。

差分电压除负载电阻的影响外，还会受到供电电压的影响，图 4 是我们测试了 CTM1051M 模块在不同电压，不同负载下的差分电压幅值得到的曲线。可以看到电源电压升高 0.5V，差分电压幅值会升高约 0.3V。

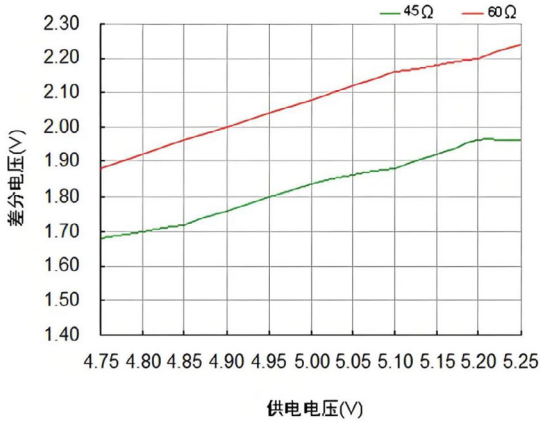


图4 不同供电电压下的差分电压

2. 接收节点的识别电平

接收节点有一定的电平识别范围，CTM1051M 的 CAN 接口典型参数如图 5 所示。

输出特性							
参数		符号	条件	最小值	典型值	最大值	单位
显性电平（逻辑 0）	CANH	$V_{O(D)CANH}$	R_L=60Ω	2.75	3.5	4.5	VDC
	CANL	$V_{O(D)CANL}$	R_L=60Ω	0.5	1.5	2.25	
隐性电平（逻辑 1）	CANH	$V_{O(R)CANH}$	no load	2	2.5	3	
	CANL	$V_{O(R)CANL}$	no load	2	2.5	3	
差分电平	显性（逻辑 0）	$V_{dif(d)}$	R_L=60Ω	1.5	2	3	
	隐性（逻辑 1）	$V_{dif(r)}$	no load	-0.05	0	0.05	
总线引脚最大耐压		V_X	CANH、CANL	-58	--	+58	
总线瞬时电压		V_{tt}	CANH、CANL	-150	--	+100	
总线引脚漏电流		I_L	V_{CC}=0V，$V_{CANH,L}$=5V	-5	--	+5	μA
CAN 总线接口		符合 ISO 11898-2 标准，双绞线输出					

图5 CAN接口典型参数

节点输入显性电平应大于 0.9V。ISO 11898 中，总线上的任意点的最小电平应大于 1.2V，组网时我们应使差分电压大于此值。

3. 实际组网分析

目前收发器的最大组网节点数为 110 个，组网时我们考虑以上的电阻参数，确保总线上的差分电压在合理的范围内即可。图 6 所示为 CTM1051M 推荐的组网拓扑，我们要考虑总线电阻，终端电阻，发送点，接收点电压参数。

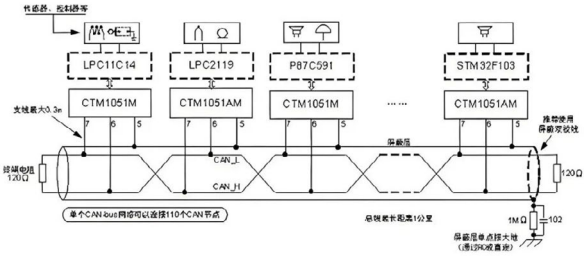


图6 CTM1051M推荐的组网拓扑

其等效电路如图 7 所示。

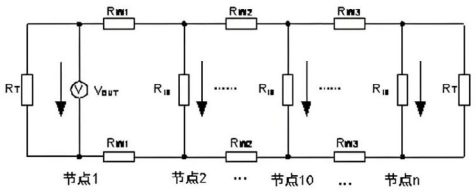


图7 CTM1051M组网等效电路

根据等效电路，我们可以调整的参数有终端电阻 RT、发送节点电压 VOUT、总线有效电阻 RW。图 7 中，各节点的 RW、RIN 难以准确确定，组网时以公式计算较为繁琐，简便的方法便是测量总线两端的节点电压。如网络的总线电阻过大时，节点 1 到节点 n 总线对信号的损耗会很大，当节点 n 接收的差分电压低于 1.2V 时，需要增大终端电阻。

在使用浪涌抑制器的场合，比如在图 7 的节点 1 和节点 2 之间增加 SP00S12 信号浪涌抑制器，其直流等效电阻为 9.5Ω，可以将其等效为总线的有效电阻，当节点 1 收到的电压过低时可通过减小总线有效电阻，提高节点 1 处的终端电阻来弥补浪涌抑制器带来的损耗。

边缘计算 ▼

【新品发布】RK3568核心板降成新方案 - M3562核心板新上市！

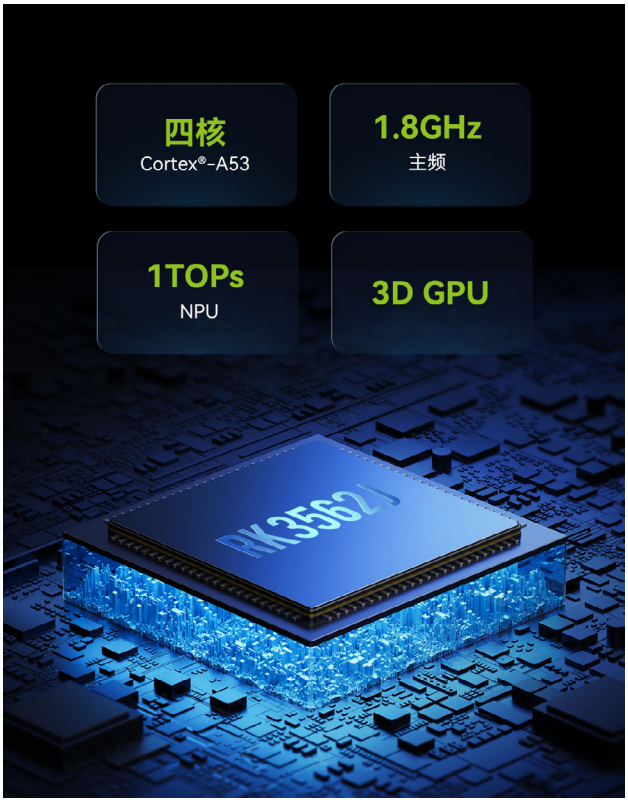
ZLG 致远电子 2025-01-06 11:34:38

选型表

型号	M3562-2GF16GI-B	M3562-4GF32GI-B
处理器	RK3562J	RK3562J
内核	四核 Cortex-A53	四核 Cortex-A53
主频	1.8GHz	1.8GHz
操作系统	buildroot，Debian	buildroot，Debian
内存	2GB	4GB
存储	16GB	32GB
GPU 3D	Mail-G52 核， 支持 OpenGL ES 3.2	Mail-G52 核， 支持 OpenGL ES 3.2
GPU 2D	支持 ARGB/RGB888 等多种格式，裁剪、旋转等操作	支持 ARGB/RGB888 等多种格式，裁剪、旋转等操作
NPU	1TOPs	1TOPs
视频解码	H.265 4K@30fps	H.265 4K@30fps
视频编码	H.264 1080p@60fps	H.264 1080p@60fps
最高显示分辨率	2048 x 1080@60Hz	2048 x 1080@60Hz
LCD 接口（RGB）	可提供方案支持	可提供方案支持
LVDS/MIPI DSI	1 路 4 lanes	1 路 4 lanes
MIPI CSI	2 路 4 Lanes	2 路 4 Lanes
电阻触摸屏	可提供方案支持	可提供方案支持
电容触摸屏	可提供方案支持	可提供方案支持
USB3.0 OTG	1 路	1 路
USB2.0 HOST	1 路	1 路
以太网	1 路千兆，1 路百兆	1 路千兆，1 路百兆
SDIO	2 路	2 路
CAN BUS	2 路	2 路
UART	8 路（最高 10 路）	8 路（最高 10 路）
IIC	3 路（最高 6 路）	3 路（最高 6 路）
IIS	1 路（最高 3 路）	1 路（最高 3 路）
PCIE2.0	最高 1 路	最高 1 路
SPDIF	最高 1 路	最高 1 路
PDM	最高 1 路	最高 1 路
SPI	最高 3 路	最高 3 路
PWM	最高 16 路	最高 16 路
ADC	最高 16 路	最高 16 路
GPIO	最高 108 路	最高 108 路
RTC 时钟	提供接口驱动支持	提供接口驱动支持
独立硬件看门狗	支持	支持
供电电压 (PM362 电源模块)	3.3V	3.3V
机械尺寸	35mm x 40mm	35mm x 40mm
环境测试	-40℃ ~85℃	-40℃ ~85℃
评估底板	M3562-EV-Board(搭载 M3562-4GF32GI-B 核心板)	
核心板形态	BGA 封装	BGA 封装

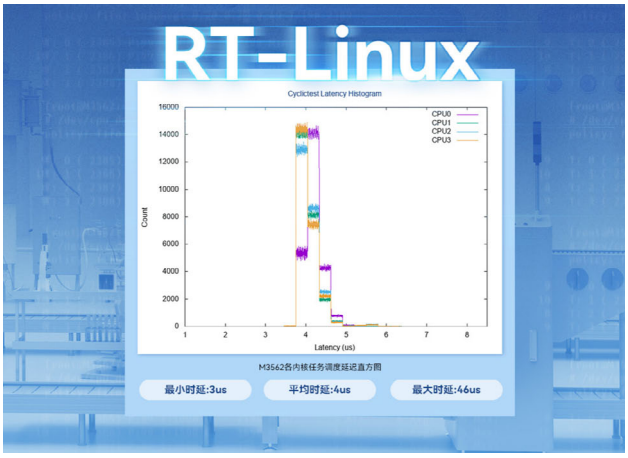
四核Cortex®-A53，主频1.8GHz

M3562 核心板，搭载瑞芯微 RK3562J 处理器，四核 Cortex®-A53 内核架构，最高主频 1.8GHz，同时集成 Mali-G52 GPU 以及 1TOPs 的 NPU。

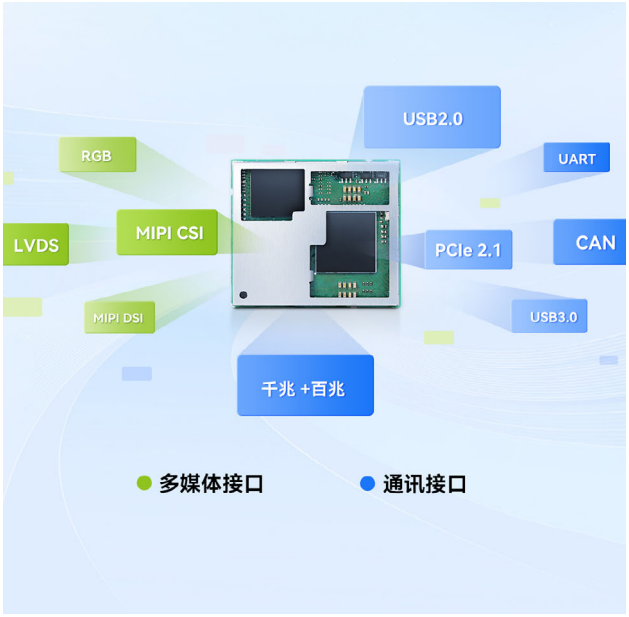


精准实时控制

M3562 核心板搭载 RT-Linux 系统，专注于精准实时控制，满足高性能计算与实时性要求，为工业自动化、机器人等领域提供强大的硬件与软件支持。



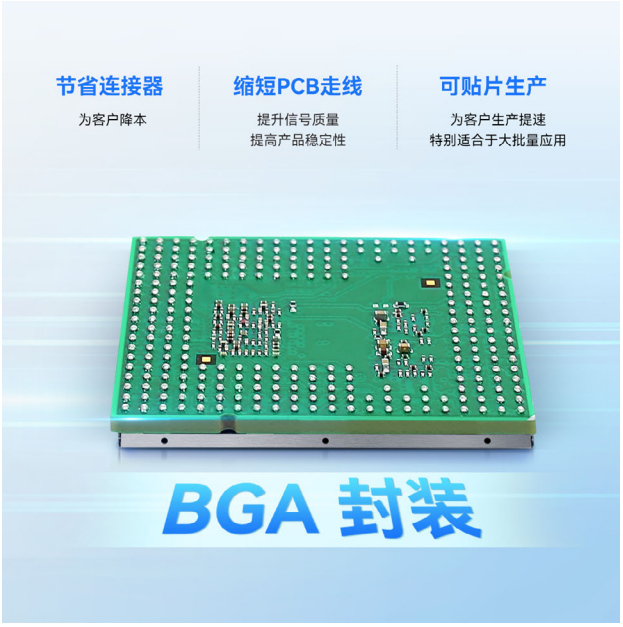
接口丰富，覆盖更多应用场景



VPU 高清编解码，三种显示接口

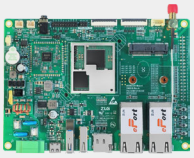
边缘计算 ▼

BGA封装，工艺革新



M3562-4GF16GI-B

点击购买



M3562-4GF16GI-B

点击购买

行业应用

M3562-2GF16GI-B

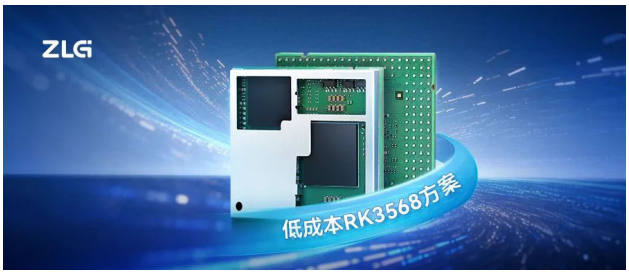
点击购买

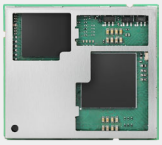
【技术分享】为什么要选择BGA核心板？

ZLG 致远电子 2025-01-07 11:33:45

M3562 核心板不仅在性能上表现卓越，还采用了先进的 BGA 封装技术。那么，BGA 封装核心板究竟有哪些独特的优势呢？本文将带您深入探讨。

继 MX2000 和 CPMG2UL BGA 核心板之后，ZLG 致远电子又重磅推出了一款全新的 BGA 核心板——M3562。





M3562-4GF16GI-B

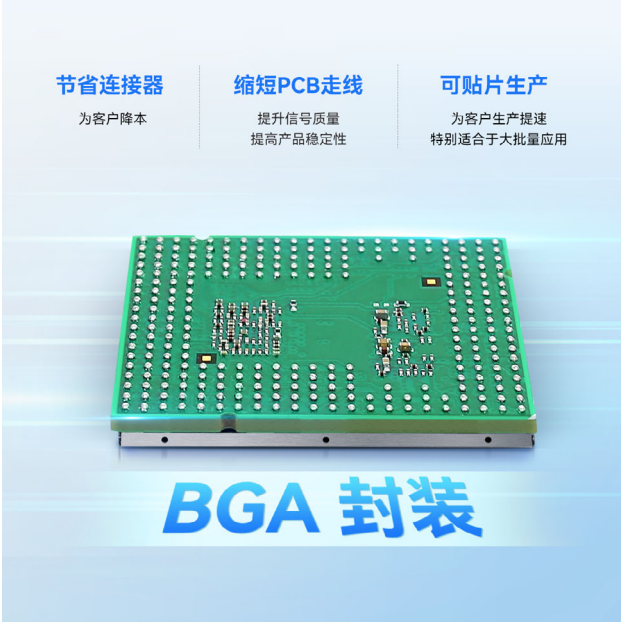
点击购买

作为一款高性能、低功耗、高性价比的嵌入式核心板，M3562 具备四核 Cortex-A53 架构内核，最高主频可达 1.8GHz，同时集成 Mali-G52 GPU、NPU，以及 1080p@60fps 视频编码器和 4K@30fps 视频解码器。它拥有 35mm×40mm 的超小尺寸，支持 -40~85°C的工业级工作温度，提供 2GB~4GB 的内存容量和 16GB~32GB 的存储容量可供选择。除了性能卓越之外，M3562 核心板还有一个特色就是采用了 BGA 封装，那么 BGA 封装核心板有什么优点呢？本文继续深入探讨。

为什么开发BGA核心板

最常见的核心板形态是采用板对板连接器对外连接，这种连接方式工艺简单、易于安装。但随着电子产品对性能和尺寸要求的提升，越来越多的应用场景已无法满足常规板对板连接器核心板的尺寸需求。于是，邮票孔和 LGA 封装的核心板应运而生，它们使核心板能够做得更小、更薄，但也引入了一些问题。邮票孔封装由于只有最外侧一排能够布置引脚，引脚密度较低，难以满足尺寸要求，而 LGA 封装则存在翘曲和空洞风险。

相比之下，BGA 核心板完美规避了这些问题，在几乎不增加成本的情况下，实现了比邮票孔和 LGA 封装核心板更小的尺寸、更高的引脚密度、更优的电气性能、更低的底板贴片工艺要求和更高的贴片质量。



解决翘曲导致的接触不良

由于 PCB 印制电路板的工艺限制，不可避免地会有翘曲现象。虽然目前 PCB 翘曲率已能控制在 0.75% 以内，但回流焊印刷的锡膏厚度通常只有 0.12~0.15mm，对于边长 10~20mm 的小型功能模块尚可接受，但对于尺寸达 30~40mm 以上的 LGA 核心板，翘起程度可能导致核心板接触不到锡膏或接触不足，引发焊接不良。为提高良率，LGA 核心板不得不将焊盘做得较大，从而降低引脚密度，无法充分利用空间。即便如此，处于 LGA 核心板边缘的引脚仍面临上锡不足的风险。

要确保 LGA 核心板有足够的可靠性，需要将翘曲度控制在 0.2%~0.3%，这会增加 PCB 板厂的费用。设计 PCB 时还需考虑残铜率、对称性、导热率、拼板强度等诸多因素，以配合实现 0.2%~0.3% 的翘曲度，这进一步限制了 PCB 尺寸，无法最大限度利用空间。

BGA 封装核心板在焊接到底板时，半熔融的锡球不会完全塌陷，而是略微下沉。这种下沉恰好弥补了 PCB 翘曲带来的间隙，焊盘无需做得过大，使 BGA 封装核心板在引脚密度更高的情况下，焊接不良的概率反而更低。

解决焊接气泡/空洞问题

气泡和空洞问题，对 LGA 核心板也是个大问题。气泡 / 空洞是指在回流焊过程中，熔融的锡膏、助焊剂、PCB 表面氧化层等物质中逸散的气体无法及时排出，聚集形成气泡 / 空洞，导致焊接不良，这是 PCB 贴片中常见的不良现象之一。LGA 封装由于核心板与底板紧贴，且焊盘较大，排气空间有限，加剧了空洞的形成。

边缘计算 ▼

焊点中的空洞可能导致散热变差、焊点强度降低、延展性、蠕变和疲劳寿命等机械性能下降、焊锡溢出焊盘等问题，严重时甚至引发短路或虚焊。传统降低 LGA 封装空洞率的方法包括调整温度曲线、调整锡膏、优化钢网设计、预上锡、N2 回流焊或真空回流焊等，但这些方法在实际应用中受到限制，增加工艺难度和生产成本，效果往往不尽如人意。

BGA 封装核心板由于核心板与底板不再像 LGA 一样紧贴，底板上的焊盘更小，锡膏更少，这些因素使得气体更容易排出。在相同工艺条件下，BGA 封装核心板的空洞率远低于 LGA 封装。因此，BGA 封装核心板的贴装使用一般工艺，就能达到较理想的空洞率，无需再担心空洞过大导致的各种问题。

提升信号质量

在信号质量方面，BGA 封装核心板也优于其他形式的核心板。板对板连接器核心板由于连接器引脚长度较长，会引入较大的寄生电感，连接器引脚资源珍贵，设计时不得不避免分配太多地信号，导致阻抗跨分割、信号回流路径变长，最终影响信号质量。

LGA 核心板和邮票孔核心板则因较大的方形或异型焊盘，传输线经过此处形成严重的阻抗突变，影响信号质量。直角的焊盘边缘还容易产生 EMI 和 EMS 问题。此外，LGA 和邮票孔核心板由于紧贴底板，核心板底层走线和底板顶层走线之间存在串扰风险。

由此可见，BGA 封装的优点众多！快来看看你的项目是否适合采用这种先进工艺的核心板吧！

ZLG嵌入式笔记(连载17)| 面对这样的“放电”不要心动，要杜绝！

ZLG 致远电子 2025-01-09 11:39:31

随着电子设备集成度的不断提高，静电放电（ESD）对电路的干扰和破坏愈发严重，成为影响设备可靠性的关键因素。本文探讨了 ESD 的成因、危害及防护方法。

现在各类电子产品的功能越来越强大，而电路板却越来越小，集成度越来越高，或多或少都装有部分接口用于人机交互，这样就存在着人体静电放电的 ESD 问题。一般电子产品中需要进行 ESD 防护的有：USB 接口、HDMI 接口、电源接口、以太网接口、天线接口、VGA 接口、DVI 接口、MIPI 接口、按键、SIM 卡、耳机、4G/5G 接口、 触摸接口及其他各类数据传输接口。现代半导体器件的规模越来越大，工作电压越来越低，ESD 对于电路引起的干扰、对元器件以及接口电路造成的破坏等问题务必引起高度重视。

下面从静电的特点、危害和防护等方面进行一些介绍，并给出电源、USB 3.0 电路的 ESD 防护示例，最后对 ESD 防护器件选型进行一些分享。

什么是静电？有什么危害？

所谓静电，就是一种处于静止状态的电荷或者说不流动的电荷（流动的电荷就形成了电流）。当电荷聚集在某个物体上或表面时就形成了静电，而电荷分为正电荷和负电荷两种，也就是说静电现象也分为两种即正静电和负静电。当正电荷聚集在某个物体上时就形成了正静电，当负电荷聚集在某个物体上时就形成了负静电，但无论是正静电还是负静电，当带静电物体接触零电位物体（接地物体）或与其有电位差的物体时都会发生电荷转移，就是我们日常见到的火花放电现象。例如北方冬天天气干燥，人体容易带上静电，当接触他人或金属导电体时就会出现放电现象。人会有触电的针刺感，夜间能看到火花，这是化纤衣物与人体摩擦，人体带上正静电的原因。

静电并不是静止的电，是宏观上暂时停留在某处的电。人在地毯或沙发上立起时，人体电压也可高达 1 万多伏，而橡胶和塑料薄膜表面的静电更是可高达 10 多万伏。[摘自百度百科“静电”词条]

1. 特点

- ESD 可形成高电位、强电场、瞬时大电流，大多数情况下 ESD 过程往往会产生的瞬时脉冲电流强度可达到几十安培甚至上百安倍。
- 在 ESD 过程会产生强烈的电磁脉冲辐射，这种电磁脉冲上升时间极快、持续时间极短。

2. 危害

- ESD 对电子设备的危害主要有两种机理，其一是 ESD 电流直接流过电路造成破坏，另一种是 ESD 电流产生的电磁场通过近场的电容耦合、电感耦合或远场的空间辐射耦合等途径对电路造成干扰。
- ESD 可能引起易燃易爆物的起火或爆炸，也可能导致电子元件被击穿或失效。随着微电子技术的快速发展，静电危害已成为微电子器件的重要破坏源。静电防护问题越来越被关注，而 ESD 防护领域也日渐广泛。

静电防护

静电防护，一方面要在原理图设计时考虑静电释放，另一方面，在 PCB 设计的时候也需要考虑静电因素。

1. 原理图设计

- 阻止静电的产生和积累，消除 ESD 源；
- 隔离导体，阻止放电；
- 为 ESD 电流提供替换通道，使其旁路；
- 屏蔽电路，阻止 ESD 产生的电磁干扰耦合到电路或设备；
- 通过选择抗静电级别较高的器件，设计合理的工艺和电路来增强系统的抗干扰能力。

2. PCB设计要点

- PCB 板边与其他布线之间的距离应大于 0.3mm；
- PCB 的板边最好用 GND 网络包围，并间隔 100~300mil 打一个缝合孔；
- 重要的信号线如 Reset、Clock 等与其他布线之间的距离最好要满足 3W 规则，有空间条件的话包地更好；
- 大功率的线与其他布线之间的距离保持在 0.2mm 以上；
- ESD、TVS 等保护器件布局应该尽量靠近接口，并且在 GND 管脚附近多打过孔；
- 铺地时应尽量避免尖角，有尖角的地方应尽量使其平滑。

常见电路应用案列

1. 电源电路的ESD设计

致远电子工控板为满足在工业环境下的防护等级要求，在输入端做了一系列的保护，参考电路如图 1 所示。

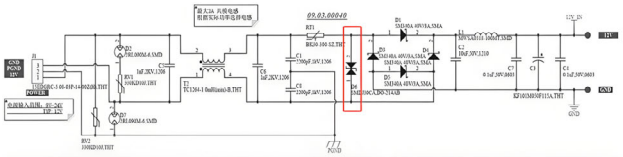


图1 输入电源电路

该电路的 RT1 自恢复保险丝提供过流、短路保护，D1、D3、D4、D5 组成防反接电路，RV1 压敏电阻和 D2 气体放电管组成一级浪涌防护，D3 的 TVS 瞬态抑制二极管为二级浪涌保护和静电泄放，T2 共模电感作为一级浪涌和二级浪涌的退耦器件，同时抑制共模干扰，大容量电容 C3 可避免浪涌回落时一级电源欠压保护或负压损坏一级电源芯片，C2、C7、L1 组成的 π 型滤波电路可抑制差模干扰。客户可针对性增加或者删减，以适应于不同的应用场景。

2. USB的ESD参考电路设计

USB3.0 参考电路如图 2 所示。

边缘计算 ▼

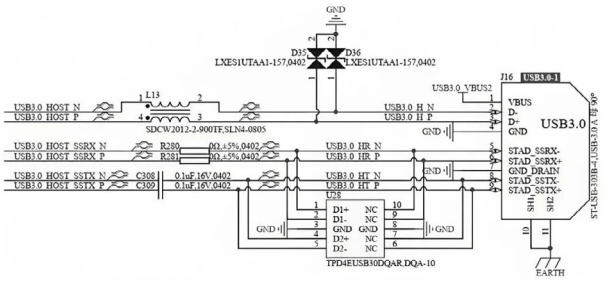


图2 USB3.0参考电路

USB3.0 接口的 ESD 电路设计主要是在接口处放置 ESD 保护器件 LXES1UTAA1-157,0402 和 TPD4EUSB30DQAR,DQA-10。两个 ESD 保护器件的关键参数分别如图 3 和图 4 所示。

Parameter	Reverse Working Voltage	Channel Leakage Current	Break down voltage	ESD per IEC 61000-4-2 (air)	ESD per IEC 61000-4-2 (contact)	ESD per IEC 61000-4-5 (Surge) (8/20μs)	Capacitance
Symbol	V_{RWM}	I_{leak}	V_{br}	V_{esd}	V_{esd}	I_{pp}	C
Unit	V	uA	V	kV	kV	A	pF
Condition		VP_{in1}=5V, VP_{in2}=0V	I_{br}=1mA, P_{in1} to P_{in2}	Ta=25℃	Ta=25℃		$VP_{in1,2}$=0V, f = 1MHz, Between Channel pins
LXES1UTAA1-157 LXES1UTBB1-157	+/-8.0	1.0 (max)	7 (min)	+/- 15	+/- 8	1.5	0.5

图3 LXES1UTAA1-157 ESD参数

ELECTRICAL CHARACTERISTICS

over operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{RWM}	Reverse stand-off voltage, TPD2EUSB30, TPD4EUSB30	D+,D– pins to ground				5.5	V
	Reverse stand-off voltage, TPD2EUSB30A	D+,D– pins to ground				3.6	V
V_{clamp}	Clamp voltage	D+,D– pins to ground,	I_{IO} = 1 A			8	V
I_{IO}	Current from IO port to supply pins	V_{IO} = 2.5 V,	I_O = 8 mA		0.01	0.1	μA
V_D	Diode forward voltage	D+,D– pins, lower clamp diode,	V_{IO} = 2.5 V, I_D = 8 mA	0.6	0.8	0.95	V
R_{dyn}	Dynamic resistance	D+,D– pins	I = 1 A		0.6		Ω
C_{IO-IO}	Capacitance IO to IO	D+,D– pins	V_{IO} = 2.5 V		0.05		pF
C_{IO-GND}	Capacitance IO to GND	D+,D– pins (DRT)	V_{IO} = 2.5 V		0.7		pF
		D1+, D1-, D2+, D2- (DQA)			0.8		
V_{BR}	Break-down voltage, TPD2EUSB30, TPD4EUSB30	I_{IO} = 1 mA		7			V
	Break-down voltage, TPD2EUSB30A	I_{IO} = 1 mA		4.5			V

图4 TPD4EUSB30DQAR ESD参数

ESD静电二极管选型指南

- ESD 二极管的截止电压要大于电路中最高工作电压；
- 脉冲峰值电流 IPP 和最大钳位电压 VC 的选择，要根据线路上可能出现的最大浪涌电流来选择合适 IPP 的型号，需要注意的是，此时的 VC 应小于被保护 IC 管脚所能耐受的最大峰值电压；
- 用于信号传输电路保护时，一定要注意所传输信号的频率和传输速率，当信号频率或传输速率较高时，应选用低电容系列的 ESD 二极管；
- 根据电路设计布局及被保护线路数选择合适的封装。

ESD 涉及的知识广泛，这里仅仅是抛砖引玉。

ZLG嵌入式笔记(连载18)｜电磁兼容的这些问题，你都考虑到了吗？(上)

ZLG 致远电子 2025-01-14 11:36:47

随着电子设备的广泛使用，电磁兼容性问题愈发关键。本文将深入解析开发生产电子设备时需关注的电磁兼容性要点，并结合案例分析，助你全面了解如何确保电子设备在复杂电磁环境中的稳定运行。

在电子设备日益普及的今天，电磁兼容性问题越来越受到人们的关注。电磁兼容性是指电子设备在电磁环境中能正常工作，且不对该环境中其他设备产生干扰。在开发和生产电子设备时，需考虑哪些电磁兼容性问题呢？下面为您解析。

设备是否符合电磁兼容性标准？

在开发和生产电子设备时，需遵循相关的电磁兼容性标准，以确保设备在特定的电磁环境中能正常工作，且不会对其他设备造成干扰。这包括对设备的电源线路、信号线路、屏蔽材料等进行分析和优化，并进行一系列的电磁兼容测试，如传导发射测试、辐射发射测试、传导抗扰度测试和辐射抗扰度测试等，以确保其符合相关的电磁兼容标准。

电磁兼容标准规范是指为了确保电子设备在电磁环境中能正常工作，而制定的一系列电磁兼容性标准和规范。这些标准和规范通常由国际电工委员会（IEC）等权威机构制定，以确保不同国家之间的兼容性。最新的电磁兼容标准规范是 GB/T17799.1-2017《电磁兼容通用标准第 1 部分：总则》，该标准已于 2018 年 5 月 1 日实施，替代了 GB/T17799-2003。

电磁兼容标准规范主要涉及到以下几个方面的要求：

传导发射：

指电子设备在正常工作过程中，通过电源线、信号线等传导途径向外发射的电磁干扰。为了控制传导发射，需采取一系列的措施，如选择合适的滤波器、优化线路设计、使用屏蔽材料等。

辐射发射：

指电子设备在正常工作过程中，向外辐射的电磁干扰。为了控制辐射发射，需采取一系列的措施，如优化设备的结构和设计、使用屏蔽材料等。

传导抗扰度：

指电子设备在受到外部电源线、信号线等传导途径传入的电磁干扰时，能正常工作的能力。为了提高传导抗扰度，需采取一系列的措施，如优化电路设计、使用滤波器和保护器件等。

辐射抗扰度：

指电子设备在受到外部电磁辐射干扰时，能正常工作的能力。为了提高辐射抗扰度，需采取一系列的措施，如优化设备的结构和设计、使用屏蔽材料等。

案例分析：

某款手机在上市后发现存在严重的电磁干扰问题，导致无法与其他设备正常通信。为了解决这一问题对手机的设计进行分析，发现是电源线路设计不合理导致了电磁干扰。通过重新设计电源线路，并增加滤波器等措施，最终解决了电磁干扰问题，并符合相关的电磁兼容性标准。

设备是否会产生电磁干扰？

电子设备在运行过程中会产生电磁干扰，这些干扰可能会影响其他设备的正常运行。产生电磁干扰的原因有很多种，包括电源线、信号线、地线等引起的传导干扰，以及电磁辐射引起的辐射干扰。为了减少电磁干扰的产生，需在设计阶段采取一系列的措施，如选择合适的滤波器、优化布线、使用屏蔽材料等。此外，还需对设备的电源和信号进行滤波处理，以减少干扰的传播。

设备是否能承受电磁干扰？

除了产生电磁干扰外，电子设备还需能承受电磁干扰，以确保其正常运行。在开发和生产电子设备时，需采取相应的措施来提高设备的电磁抗干扰能力。这包括使用具有抗干扰能力的元件、优化电路设计、使用滤波器和屏蔽材料等。此外，还需对设备进行一系列的电磁抗扰度测试，包括传导和辐射抗扰度测试等，以确保其能在各种电磁环境中正常工作。

边缘计算 ▼

案例分析：

某航空公司的航班在着陆时突然出现了一次意外事故，据调查是由于客舱门控制系统的电磁干扰导致的。为了解决这一问题，对客舱门控制系统和周围的电磁环境进行分析和测试。结果发现是客舱门控制系统的电磁抗干扰能力不足，无法承受电磁干扰的影响。通过增加滤波器、优化电路设计等措施，最终提高了客舱门控制系统的电磁抗干扰能力。

关注我们，下期继续讲解如何测试和解决电磁兼容性。

ZLG嵌入式笔记(连载19)｜电磁兼容的这些问题，你都考虑到了吗？(下)

ZLG 致远电子 2025-01-16 11:36:34

在电子设备日益普及的今天，电磁兼容性问题越来越受到人们的关注。上期我们探讨了电子设备开发生产中电磁兼容性的要点及案例，本期我们将继续探讨如何测试和解决电磁兼容性问题。

如何测试设备的电磁兼容性？

为了确保电子设备的电磁兼容性符合要求，需进行相关的测试。这些测试包括传导和辐射发射测试、传导和辐射抗扰度测试等。这些测试需在专业的电磁兼容实验室中进行，以确保测试结果的准确性和可靠性。

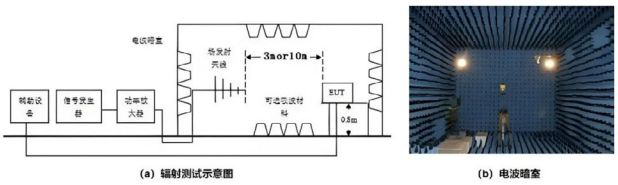


图1 辐射测试和电波暗室

案例分析：

某电力公司的电力监控系统在运行过程中出现了数据传输错误的问题，经过排查发现是电磁干扰导致的。为了解决这一问题，对电力监控系统进行了详细的电磁兼容性测试。测试内容包括传导和辐射发射测试、传导和辐射抗扰度测试等，以确定电磁干扰的具体来源和影响程度。通过测试发现，电力监控系统的数据传输线路存在电磁干扰问题，导致数据传输错误。通过增加滤波器和优化数据传输线路等措施，最终解决了电磁干扰问题。

如何解决电磁兼容性问题？

在开发和生产电子设备时，需考虑如何解决电磁兼容性问题。这包括选择合适的滤波器、屏蔽材料等。此外，还需对设备的电源和信号进行滤波处理，以减少干扰的传播。同时，也需在设备的结构和设计上进行优化，以提高其电磁抗干扰能力。例如，可以使用金属屏蔽盒来保护设备的内部电路，以减少电磁干扰的产生和传播。还可以使用具有抗干扰能力的元件、优化电路设计等措施来提高设备的电磁抗干扰能力。

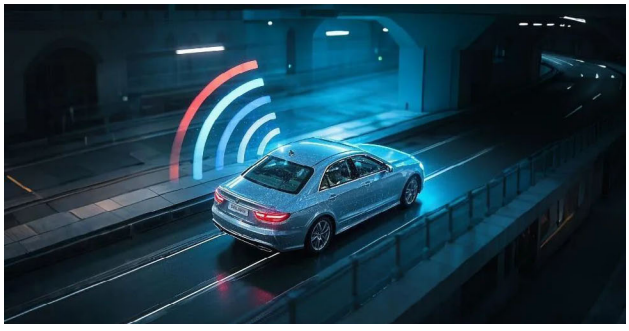


图2 汽车无线信号中断

案例分析：

某汽车制造公司的汽车在行驶过程中出现了无线通信中断的问题，经过排查发现是汽车内部的电子元件产生了电磁干扰。为了解决这一问题，采取了多种措施，包括优化汽车内部线路设计、使用具有抗干扰能力的元件、增加滤波器等。通过这些整改措施的实施，汽车的无线通信问题得到了有效解决。

边缘计算 ▼

ZLG嵌入式笔记(连载20)｜“三防”防什么？什么时候需要“三防”？

ZLG 致远电子 2025-01-21 11:44:50

在电子产品设计与应用中，三防处理是保障设备可靠性的重要环节。本文通过案例分析，介绍了三防处理的必要性、应用场景、三防漆的性能及使用工艺。

三防处理一般都在产品阶段来做，根据产品应用场合以及销售区域而定。ZLG 核心板作为通用二次开发产品，没有做三防处理，这一点务必注意。曾有一个案例，客户的产品销往南方海边城市，结果在雨季时候产品出了故障，根据天气情况，那段时期城市湿度维持在 80% 以上，并有凝露现象。后来将故障产品拿回分析，基本定位在没做三防上，当然还有与结构相关的其它问题。下面对三防做一些简单介绍。

“三防”防什么？

三防指的是防潮、防盐雾、防霉。

就电子产品而言，潮湿、盐雾和霉菌是导致其损坏变质的主要因素，它们会腐蚀和破坏材料，导致产品的电气性能下降，机械强度降低，严重时甚至会导致产品功能失效。潮湿、盐雾和霉菌对电子设备的影响不是孤立的，而是相互影响，共同破坏电子设备。潮湿为霉菌的生长提供了有利条件，而霉菌是潮湿的，当其跨过绝缘表面而繁殖时，可能引起短路，对电子产品金属材料、电子元件等造成腐蚀破坏。盐雾最显著的特点是能从相对干燥的大气中吸附水分，当电子产品表面附着这些含盐水分时，就会长期保持潮湿状态，除自身的腐蚀作用外，还加剧了潮湿的破坏作用。

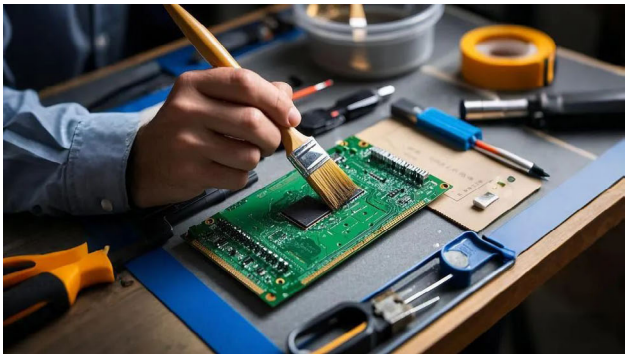
什么时候需要“三防”？

当电子产品应用于以下情形时需做三防处理。

- 沿海地区，由于空气比较潮湿，需要做好防潮、防盐雾处理。
- 多雨地区，由于雨水较多，需要做好防水、防霉处理。
- 高寒地区，由于气温较低，需要做好防冻处理。
- 化工行业，由于会接触到各种化学物质，需要做好防腐蚀处理。
- 农业领域，在干旱、多涝、大风等环境下，需要做好防水、防涝、防风处理。

三防漆是一种配方非常特殊的涂料，它不仅可以防潮、防盐雾、防霉，还可以对整个外界环境都起到良好的防护作用，并且还可以起到防漏电、防尘、防静电、防老化、防腐蚀等性能，同时还具有良好的耐高温性。

当三防漆被涂抹在元器件表面以后，就会形成一道透明的保护膜，这层保护膜就可以很好的保护元器件，避免被周边环境所侵蚀或者是损坏。由于三防漆可防止漏电，因此允许更高的功率和更近的 PCB 板间距。



三防漆使用工艺有刷涂、喷涂、自动浸涂和选择性涂覆着膜。在涂三防漆时，一般连接器、开关、蜂鸣器、电池座、接地螺丝孔、散热器、散热区域、插板区域等是不允许有涂覆材料的，涂覆时这些区域要进行遮蔽。

ZLG 致远电子所有核心板都不带三防功能，需要用户根据产品的实际需求进行三防处理。

ZLG嵌入式笔记(连载21)｜使用了致远电子MPU核心板后的产品热设计，你考虑周全了么？

ZLG 致远电子 2025-01-23 11:33:51

在嵌入式系统设计中，散热是影响处理器性能与稳定性的关键问题。本文聚焦于高端嵌入式处理器的散热设计，探讨核心板的热设计与系统级热设计方法，以及导热材料和布局的建议，为解决高温问题提供参考。

用高端嵌入式处理器设计系统，散热是一个避不开的问题。就算标称工业级处理器，在进行环境温度实测的时候，都需要加散热片才能通过。在不加散热器的情况下，处理器的结温和环温，一般相差可在 20°C ~50°C范围内，加上散热片后，温差可大幅度缩小。如果一个处理器标称结温为 105°C，如果不加散热片，大概率是过不了 85°C的高温测试的。在封闭的高温测试环境下，结温和环温的温差通常会超过 30°C。

选用 ZLG 的核心板，如果处理器主频超过 1GHz，多核处理器，通常都需要考虑散热需求，特别是带 GPU 和 NPU 的芯片，在运行 GPU 和 NPU 的时候发热量急剧上升，更是需要散热片，否则可能会出现一些异常：

1. 当环境温度升高，处理器运行速度下降，界面卡顿或者通信延迟增加；
2. 高温下系统意外重启，或者出现界面黑屏。

如果产品本身还是封闭式设计，那更需要在散热上下功夫。

使用核心板的热设计主要涉及两个大方面：核心板本身的热设计、包含核心板的系统级热设计。

核心板本身的热设计

核心板在设计之初已经融入了热设计理念，使用的时候只需关注发热量大的器件（一般为 MPU，有时也包括 DDR 和 eMMC），对其散热处理即可。散热方式主要有自然散热、强迫风冷、强迫液冷，其中致远电子的核心板一般选择自然散热和强迫风冷，最终选择何种散热方式需结合工作温度、器件结温、器件功耗等因素。通过调试指令读出 MPU 的结温可以知道 MPU 散热的实际效果，如果裸机能满足 MPU 结温条件甚至都可以不用额外加散热处理。

1. 自然散热

自然散热是指物体通过热传导、对流和辐射等方式将热量传递给周围环境的过程，实现方法是发热器件通过散热器把热量传递到环境中。散热路径为发热器件→导热界面材料→散热器→环境。核心板的发热器件一般为 MPU，导热界面材料主要有导热硅脂、导热硅胶片、导热相变片、导热凝胶等，常见的散热器材质主要是铝合金和铜合金，它们的导热效能和经济性综合表现较好。

对于核心板来说，导热界面材料推荐选择导热硅胶片，其优点是导热性能稳定，耐高温，电绝缘，很好的柔软度可以弥合一定程度的结构件高度差。在选型导热硅胶片时一项很重要的参数是导热系数，导热系数在市面上以 3W/(m.K) 居多，也有 5W/(m.K)、8W/(m.K)。

曾经在同一核心板使用同一个散热器对比测试过 3W/(m.K) 和 5W/(m.K) 的导热硅胶片，80°C环境温度下，使用 5W/(m.K) 导热硅胶片读取的 MPU 结温比使用 3W/(m.K) 导热硅胶片时低 5°C。从对比测试结果看导热硅胶片的导热系数对散热效果影响是比较大，应综合测试和成本选择合适的导热硅胶片。

热量传导有三种基本方式：热传导、对流、辐射，这可作为散热器设计的根据。导热系数和导热截面积是热传导中影响传热效率的关键参数。铜合金比铝合金导热系数高，但成本相对较高，需综合考量。加大齿厚、基板厚度、导热截面积改善散热性能需综合考量系统级空间，在有金属外壳的产品中，把金属外壳当作散热器不失为很好的散热方案。换热面积和对流换热系数是对流的重要参数，细密齿间距散热器可以加大散热面积但降低了换流系数，稀疏齿间距散热器有更高的换流系数但散热面积减少，需综合它们的乘积获得最优值。散热器表面进行氧化发黑处理，可以增强辐射换热效果，在自然对流情况下，辐射换热作用突出，可以提高 25% 的散热量。所以，除非是器件附近有高热源，否则散热器表面都应涂覆或氧化发黑处理以提高辐射性能。

图 1 是某款核心板采用安装散热器的自然散热方式示例。

图1 核心板安装散热片

2. 强迫风冷

强迫风冷的实现方式是用风扇加剧空气流动，提高对流换热系数来强化换热能力。在用自然散热方式实现不了散热需求时可以采用风扇的散热方案。常用散热风扇有轴流风扇和离心风扇。轴流风扇进风口与出风口平行，特点是风量大、风压小、噪声小，适合风阻低但风量需求大的场合。离心风扇的进风口与出风口垂直，风量小，风压高，适合风阻大风量需求小的场合。

对于核心板一般采用轴流风扇，可分抽风设计和吹风设计：

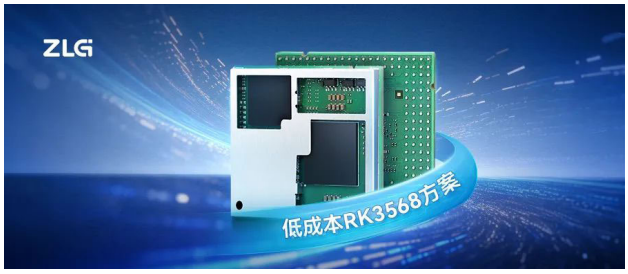
1. 抽风设计中系统内流场比较均匀，适合热源比较分散的场景；
2. 吹风设计在出风口空气流动状态通常是湍流，更适合热量集中的产品。

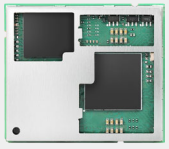
抽风设计时器件散发的热量经过风扇时高于环境温度，吹风设计时流过风扇的空气是新鲜的未经系统内元器件加热，风扇工作在常温或低温，所以吹风设计的风扇寿命相对更长。

包含核心板的系统级热设计

系统级热设计主要考虑核心板在整个系统中的位置布局，核心板应与发热量大的器件或模块拉开距离，在冷却气流路径上不应把核心板放在发热量大的器件或模块的下游。

边缘计算 ▼





M3562 Cortex®-A53核心板

点击购买

ZLG嵌入式笔记(连载22) | 为什么你的串口总是丢一些特殊字符?

ZLG 致远电子 2025-02-06 13:24:23

在嵌入式开发和物联网应用中，Linux 系统的串口通信至关重要。本文将简单介绍 Linux 下串口设备的命名规则和 termios 结构体，并解析终端的三种工作模式，帮助开发者更好地掌握 Linux 串口编程。

Linux 下串口设备名称一般为 "/dev/ttySn"，n 是数字，若串口是 USB 扩展的，则串口设备文件命名多为 /dev/ttyUSBn。

串口是一个终端设备，在 Linux 中用 termios 结构体来描述：

```
struct termios {
  tcflag_t c_cflag;        /* 控制标志 */
    tcflag_t c_iflag;        /* 输入标志 */
    tcflag_t c_oflag;        /* 输出标志 */
    tcflag_t c_lflag;        /* 本地标志 */
    tcflag_t c_cc[NCCS];     /* 控制字符 */
};
```

termios 是在 POSIX 规范中定义的标准接口，表示终端设备（包括虚拟终端，串口等）。终端有 3 种工作模式，分别为规范模式（canonical mode）、非规范模式（non-canonical mode）和原始模式（raw mode）。

c_lflag 中设置 ICANNON 标志来定义终端的三种模式：规范模式、非规范模式和原始模式。

1. 规范模式

规范模式下所有的输入都是基于行进行处理的。在用户输入一个行结束符（回车符、EOF 等）之前，系统调用 read() 函数读不到用户输入的任何字符。除了 EOF 之外的行结束符（回车符等）和普通字符一样都会被 read() 函数读取到缓冲区之中。在规范模式中，行编辑是可行的，而且一次调用 read() 函数最多只能读取一行数据。

2. 非规范模式

非规范模式所有的输入是即时有效的，不需要用户另外输入行结束符，而且不可进行行编辑。在非规范模式下，对参数 MIN(c_cc[VMIN]) 和 TIME(c_cc[VTIME]) 的设置决定 read() 函数的调用方式。设置可以有 4 种不同的情况。

- VMIN=0 和 VTIME=0：read() 函数立即返回。若有可读数据，则读取数据并返回被读取的字节数，否则读取失败并返回 0。
- VMIN > 0 和 VTIME = 0：read() 函数会被阻塞直到 VMIN 个字节数据可被读取。
- VMIN = 0 和 VTIME > 0： 只要有数据可读或者经过 VTIME 个 1/10 秒的时间，read() 函数则立即返回， 返回值为被读取的字节数。如果超时并且未读到数据， 则 read() 函数返回 0。
- VMIN > 0 和 VTIME > 0： 当有 VMIN 个字节可读或者两个输入字符之间的时间间隔超过 VTIME 个 1/10 秒时，read() 函数才返回。因为在输入第一个字符之后系统才会启动定时器，所以在这种情况下，read() 函数至少读取一个字节之后才返回。

3. 原始模式

原始模式是一种特殊的非规范模式。在原始模式下，所有的输入数据以字节为单位被处理。在这个模式下，终端是不可回显的，而且所有特定的终端输入 / 输出控制处理不可用。通过调用 cfmakeraw() 函数可以将终端设置为原始模式。

边缘计算 ▼

ZLG嵌入式笔记(连载23) | 串口数据错乱？从标志位设置入手

ZLG 致远电子 2025-02-11 11:34:42

在串口通信开发中，数据错乱是常见问题。本文将快速介绍串口标志位的作用及配置方法，帮助解决数据传输错误。

有用户反馈“串口应用层数据错乱、与发送端发送的字节数不匹配”，简单来说就是接收和发送的数据不对，而且还存在帧数据错乱的情况。

经排查，是程序没用设置 ISIG 标志位设置引起的。本节以此为切入点进行展开，对串口设置的一些标志位进行简单说明。

1. c_cflag：可以设置波特率、字符大小、数据位、停止位、奇偶校验位和硬软流控等

1.1 波特率

B0	0 波特率（放弃 DTR）。
…	
…	
B115200	115200 波特率。

1.2 其它

EXTA	外部时钟率。
EXTB	外部时钟率。
CSIZE	数据位的位掩码。
CS5	5 个数据位。
CS6	6 个数据位。
CS7	7 个数据位。
CS8	8 个数据位。
CSTOPB	2 个停止位（不设则是 1 个停止位）。
CREAD	接收使能，允许输入被接收。
PARENB	校验位使能。
PARODD	使用奇校验而不使用偶校验。
HUPCL	最后关闭时挂线（放弃 DTR）。
CLOCAL	本地连接，不检测载波信号。
CRTSCTS	硬件流控。

初始化 c_flag 时，不能直接对 c_cflag 成员进行赋值， 而是需要通过“与”、“或” 操作使用其中的某些选项，例如：

设置硬件流控制

```
c_cflag |= CNEW_RTSCTS
```

无硬件流控制

```
c_cflag &= ~CNEW_RTSCTS
```

INPCK	奇偶校验使能。
IGNPAR	忽略奇偶校验错误。
PARMRK	奇偶校验错误掩码。
ISTRIP	裁减掉第 8 位比特。
IXON	启动输出软件流控。
IXOFF	启动输入软件流控。
IXANY	输入任意字符可以重新启动输出（默认为输入起始字符才重启输出）。
IGNBRK	忽略输入终止条件。
BRKINT	当检测到输入终止条件时发送 SIGINT 信号。
INLCR	将接收到的 NL（换行符）转换为 CR（回车符）。
IGNCR	忽略接收到的 CR（回车符）。
ICRNL	将接收到的 CR（回车符）转换为 NL 换行符）。
IUCLC	将接收到的大写字符映射为小写字符。
IMAXBEL	当输入队列满时响铃。

禁用软件流控制是禁止上面的选项。

```
c_iflag &= ~(IXON | IXOFF | IXANY);
```

3. c_oflag：用于控制终端端口发送出去的字符处理

OPOST	启用输出处理功能， 如果不设置该标志， 则其他标志都被忽略。
OLCUC	输出中的大写字符转换成小写字符。
ONLCR	将输出中的换行符（‘\n’） 转换成回车符（‘\r’）。
ONOCR	如果当前列号为 0， 则不输出回车符。
OCRNL	将输出中的回车符（‘\r’） 转换成换行符（‘\n’）。
ONLRET	不输出回车符。
OFILL	发送填充字符以提供延时。
OFDEL	如果设置该标志， 则表示填充字符为 DEL 字符， 否则为 NUL 字符。
NLDLY	换行延时掩码。
CRDLY	回车延时掩码。
TABDLY	制表符延时掩码。
BSDLY	水平退格符延时掩码。
VTDLY	垂直退格符延时掩码。
FFLDY	换页符延时掩码。

因为现在终端的速度比以前快得多， 所以大部分延时掩码几乎没什么用途。

启用输出处理需要在 c_oflag 成员中启用 OPOST 选项。

```
c_oflag |= OPOST;
```

使用原始输出，就是禁用输出处理，使数据能不经过处理过滤的完整地输出到串口。当 OPOST 被禁止，c_oflag 其它选项也被忽略。

```
c_oflag &= ~OPOST;
```

4. c_lflag：用于控制控制终端的本地数据处理和工作模式

ISIG	若收到信号字符（INTR、 QUIT 等） ， 则会产生相应的信号。
ICANON	启用规范模式。
ECHO	启用本地回显功能。
ECHOE	若设置 ICANON，则允许退格操作。
ECHOK	若设置 ICANON，则 KILL 字符会删除当前行。
ECHONL	若设置 ICANON，则允许回显换行符。
ECHOCTL	若设置 ECHO，则控制字符（制表符、换行符等）会显示成“^X”，其中 X 的 ASCII 码等于给相应控制字符的 ASCII 码加上 0x40。例如：退格字符（0x08）会显示为“^H”（‘H’的 ASCII 码为 0x48）。
ECHOPRT	若设置 ICANON 和 IECHO，则删除字符（退格符等）和被删除的字符都会被显示。
ECHOKE	设置 ICANON，则允许回显在 ECHOE 和 ECHOPRT 中设定的 KILL 字符。
NOFLSH	在通常情况下，当接收到 INTR、QUIT 和 SUSP 控制字符时，会清空输入和输出队列。如果设置该标志，则所有的队列不会被清空。
TOSTOP	若一个后台进程试图向它的控制终端进行写操作，则系统向该后台进程的进程组发送 SIGTTOU 信号。该信号通常终止进程的执行。
IEXTEN	启用输入处理功能。

4.1 选择规范模式

```
c_lflag |= (ICANON | ECHO | ECHOE);
```

4.2 选择原始模式

```
c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);
```

5. c_cc：定义特殊控制特性

VINTR	中断控制字符，对应键为 CTRL+C。
VQUIT	退出操作符，对应键为 CRTL+Z。
VERASE	删除操作符，对应键为 Backspace（BS）。
VKILL	删除行符，对应键为 CTRL+U。
VEOF	文件结尾符，对应键为 CTRL+D。
VEOL	附加行结尾符，对应键为 Carriage return（CR）。
VEOL2	第二行结尾符，对应键为 Line feed（LF）。
VMIN	指定最少读取的字符数。
VTIME	指定读取的每个字符之间的超时时间。

边缘计算 ▼

ZLG嵌入式笔记(连载24) | “串口阻塞”你真的会用吗?

ZLG 致远电子 2025-02-13 11:39:21

在串口通信开发中，数据错乱是常见问题。本文将快速介绍串口标志位的作用及配置方法，帮助解决数据传输错误。

这是一个真实案例，用户反馈“串口向另外的设备发送数据，发现运行一段时间后，发送的消息会阻塞很久才会发出来，一下子出来很多数据”。经过帮客户检查应用程序源码，发现应用程序在串口阻塞方面没有做正确的处理，修改后解决。

非阻塞打开串口

```
open("/dev/ttyS1", O_RDWR | O_NOCTTY | O_NONBLOCK);
```

- O_NOCTTY：如果打开的是一个终端设备，这个程序不会成为对应这个端口的控制终端，如果没有该标志，任何一个输入，例如键盘中止信号等，都将影响进程。
- O_NONBLOCK：该标志与早期使用的 O_NDELAY 标志作用差不多。程序不关心 DCD 信号线的状态，也就是不关心端口另一端是否已经连接。如果不指定该标志，进程将一直在休眠状态，直到 DCD 信号线为 0。简单点就是以非阻塞方式打开串口。

设置串口成阻塞方式

可用 fcntl 设置串口的阻塞 / 非阻塞。

1. 阻塞：fcntl(fd, F_SETFL, 0)

fcntl 中的 F_SETFL 只可以更改标志 O_APPEND，O_NONBLOCK，O_SYNC 和 O_ASYNC；而 0 则表示清空这几个标志，其中 O_NONBLOCK 也没了，所以就变成了阻塞。

2. 非阻塞：fcntl(fd, F_SETFL, O_NONBLOCK)

检测打开的文件描述符是否连接到一个终端设备，进一步确认串口是否正确打开。

获取和设置termios

1. 获取termios结构体(串口属性)

- int tcgetattr(int fd, struct termios *termptr);
- termptr：接收返回的 termios，成功：0，失败：-1。

2. 保存先前的串口配置

int tcsetattr(int fd, int opt, const struct termios *termptr);

3. 设置串口属性

3.1 opt：在串口驱动程序里有输入缓冲区和输出缓冲区。在改变串口属性时，缓冲区可能有数据存在，如何处理缓冲区中的数据，可通过 opt 参数实现。

- TCSANOW：更改立即发生；
- TCSADRAIN：发送了所有输出后更改才发生，若更改输出参数则应用此选项；
- TCSAFLUSH：发送了所有输出后更改才发生，在更改发生时未读的所有输入数据被删除（Flush）。

3.2 成功：0。

3.2 失败：-1。

设置波特率

1. 设置输入波特率

int cfsetispeed(struct termios *termptr, speed_t speed);

2. 设置输出波特率

int cfsetospeed(struct termios *termptr, speed_t speed);

设置数据位(也称设置字符大小)

通过 c_cflag 设置。

```
CSIZE    // 数据位屏蔽
CS5      //5 个数据位
CS6      //6 个数据位
CS7      //7 个数据位
CS8      //8 个数据位
```

例如，设置串口的数据位为 8 位：

```
c_cflag &= ~CSIZE;    // 清除 CSIZE
c_cflag |= CS8;       // 设置 CS8
```

设置奇偶校验位

设置串口的奇偶校验是在 c_cflag 设置。

- PARENB　进行奇偶校验。
- PARODD　奇校验，否则为偶校验。

1. 无校验

```
c_cflag &= ~PARENB;
```

2. 偶校验

```
c_cflag |= PARENB;
c_cflag &= ~PARODD;
```

3. 奇校验

```
c_cflag |= PARENB;
c_cflag |= ~PARODD;
```

设置停止位

设置串口停止位是在 c_cflag 设置。

1. 设置 1 位停止位

```
c_cflag &= ~CSTOPB;     // 清除 CSTOPB 标志位
```

2. 设置 2 位停止位

```
c_cflag |= CSTOPB;      // 设置 CSTOPB 标志位
```

设置最少字符和等待时间

c_cc[VTIME] 和 c_cc[VMIN] 设置最少字符和等待时间，针对 read 而言。

如果设置为 0 的话，则在任何情况下 read() 函数立即返回：

```
c_cc[VTIME] = 0;
c_cc[VMIN] = 0;
```

清除串口缓冲

由于串口在重新设置之后，需要对当前的串口设备进行适当的处理，通常使用 tcflush 实现。

```
int tcdrain(int fd);      // 使程序阻塞，直到输出缓冲区的数据全部发送完毕。
int tcflow(int fd, int action);      // 用于暂停或重新开始输出。
int tcflush(int fd, int queue_selector); // 用于清空输入 / 输出缓冲区。
```

使用 tcflush() 函数，对于在缓冲区中的尚未传输的数据，或者收到的，但是尚未读取的数据进行处理。

queue_selector 设置：

- TCIFLUSH：对接收到而未被读取的数据进行清空处理。
- TCOFLUSH：对尚未传送成功的输出数据进行清空处理。
- TCIOFLUSH：即对尚未处理的输入输出数据进行清空处理。

激活选项

CLOCAL 和 CREAD 分别用于本地连接和接收使能。

激活这两个选项：

```
c_cflag |= CLOCAL | CREAD;
```

激活串口配置(属性)

在完成全部串口配置之后，要激活刚才的配置并使配置生效。使用属性设置函数 tcsetattr()，前面已有其说明。

向串口写数据

直接调用 wtrite() 即可。

从串口读数据

调用 read() 读取串口数据，但在非规范模式 / 原始模式下需要设置 VMIN 和 VTIME。

- VMIN：非规范模式下读取的最小字符数。
- VTIME：非规范模式下读数据时的延时，VTIME 个 1/10 秒。

VMIN 和 VTIME 组合有四种情况：

- VMIN=0,VTIME=0：读取的最少字符数为 0，延时时间为 0，read 立即返回。
- VMIN>0,VTIME=0：read 阻塞到读取 VMIN 个字符才返回。
- VMIN=0,VTIME>0：有数据就返回，无数据等待 VTIME 个 1/10 秒返回。
- VMIN>0,VTIME>0：读取 VMIN 个字符或前后两个字符的输入间隔超过 VTIME 个 1/10 秒后返回，因为在输入第一个字符之后系统才会启动定时器，所以在这种情况下，read 至少读取一个字符。

串口操作顺序

1. 保存原有串口属性（可选）；
2. 设置波特率；
3. 设置激活选项，如 c_cflag |= CLOCAL | CREAD；
4. 设置数据位大小；
5. 设置奇偶校验位；
6. 设置停止位；
7. 设置输出（可选），如 c_oflag=0；0 是清空标志；c_oflag&=~OPOST；
8. 设置输入（可选）；
9. 设置 c_lflag，如原始模式 cfmakeraw(&termios)；
10. 设置读取特性，c_cc[VTIME] 和 c_cc[VMIN]；
11. 刷新缓冲区，tcflush()；
12. 设置串口属性，tcsetattr()。

边缘计算 ▼

ZLG嵌入式笔记(连载25)｜CAN丢帧很常见，你知道有哪些主要原因吗？（上）

ZLG 致远电子 2025-02-18 11:36:41

在工程应用中，CAN 通信的稳定性至关重要，但丢帧和错误帧现象却时有发生。本文将简要分析导致这些问题的常见原因，并给出针对性的解决方案。

一般来说，使用 CAN 通信的场合，对通信的稳定性都有很高的要求。但在工程应用现场，经常遇到各种原因引起的丢帧或者出现错误帧的现象，下面对各种可能的原因进行简要说明。

CAN终端匹配电阻

当涉及 CAN 总线上的终端匹配电阻时，确保其正确性至关重要。终端匹配电阻的阻值不正确可能导致信号反射和干扰，进而导致数据丢失。

在 CAN 总线上，终端匹配电阻的作用是消除信号在总线上的反射。当信号到达总线的终点时，终端匹配电阻会吸收信号的能量，防止信号反射回总线上。如果终端匹配电阻的阻值不正确，可能会导致信号在总线上的反射，造成信号干扰和失真。这种干扰和失真可能会导致接收端无法正确解析数据，从而导致数据丢失。

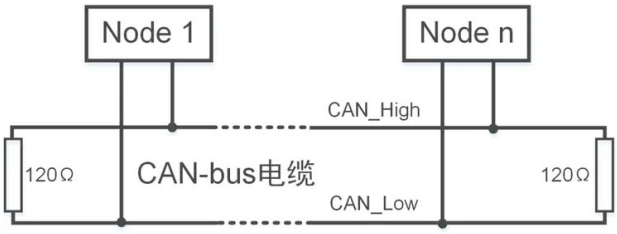


图1 CAN总线终端电阻

要解决这个问题，首先需要确保终端匹配电阻的数值符合 CAN 总线标准和硬件规范。通常情况下，CAN 总线上的终端匹配电阻数值应该等于总线特性阻抗，通常为 120Ω。确保终端匹配电阻的数值正确是防止信号反射和干扰的关键。

此外，还需要确保终端匹配电阻的连接正确，以及在总线的两端正确地安装终端匹配电阻。如果终端匹配电阻的连接不正确或者缺失，也可能导致信号反射和干扰，进而导致数据丢失。

在设计和部署 CAN 总线系统时，需要特别注意终端匹配电阻的设置和连接，以确保信号的稳定传输，避免干扰和数据丢失的问题。

CAN采样点设置

在 CAN 总线通信中，采样点是指在一个位时间内进行采样的时间点。CAN 总线的通信速率是由位时间确定的，因此采样点的准确性对于正确解析数据至关重要。

如果采样点设置不准确，可能导致在信号传输过程中采样时机不准确，造成数据采样错误或丢失。

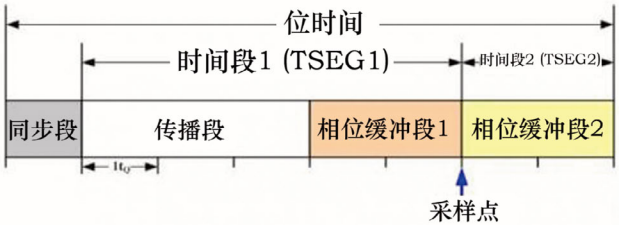


图2 CAN总线采样点

对于 CAN 总线的采样点设置，需要考虑到总线的通信速率、时钟源的稳定性以及传输线的信号衰减等因素。如果采样点设置过早或过晚，可能导致对数据位的采样出现偏差，进而影响数据的正确解析。此外，在高速传输或长距离传输的情况下，信号的衰减可能会导致采样点的漂移，进一步影响数据的准确采样。

要解决 CAN 总线的采样点设置不正确可能导致的数据丢失问题，需要对 CAN 控制器的配置进行仔细评估和调整。确保采样点设置符合总线通信速率和硬件规范，以及考虑到信号传输的稳定性和可靠性。通常情况下，可以通过调整 CAN 控制器的定时器和同步段长度来调整采样点设置，以确保在不同情况下都能够准确采样数据。

综上所述，正确设置 CAN 总线的采样点对于确保数据的准确采样和解析至关重要。通过仔细评估和调整 CAN 控制器的配置，可以有效地避免由于采样点设置不正确而导致的数据丢失问题。

底层驱动

底层驱动程序是整个 CAN 通信系统的关键组成部分，它负责与硬件进行交互并提供数据传输的基本功能。如果底层驱动程序存在 bug 或者不稳定，可能会导致数据丢失和通信故障。

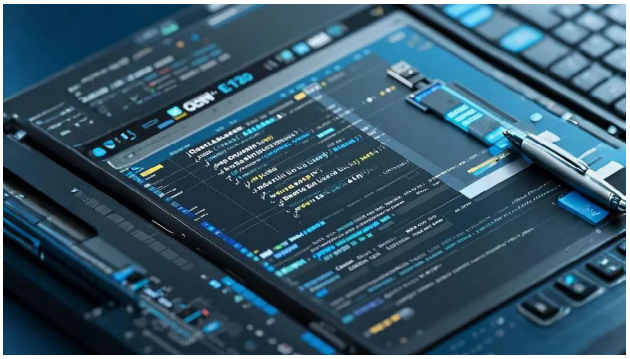


图3 底层驱动程序

缓冲区溢出：

底层驱动程序通常会使用缓冲区来存储接收到的数据，以便应用程序对其进行处理。如果缓冲区大小不合适或者底层驱动程序没有有效地管理缓冲区，可能会导致缓冲区溢出。当数据量超出缓冲区容量时，会导致数据丢失。为解决这个问题，需要对缓冲区的大小进行合理的设计，并确保底层驱动程序能够有效地处理缓冲区溢出的情况。

错误处理不当：

另一个可能导致数据丢失的问题是底层驱动程序没有正确处理错误情况。例如，当出现通信错误、总线故障或其他异常情况时，底层驱动程序应该能够及时识别并进行适当的错误处理，而不是简单地丢弃数据或者忽略错误。如果底层驱动程序没有正确处理错误情况，可能会导致数据丢失或者错误解析。因此，正确的错误处理机制对于确保数据的可靠传输至关重要。

为解决底层驱动程序可能存在的 bug 或者不稳定性问题，需要进行严格的测试和验证。确保底层驱动程序能够稳定地与硬件进行交互，并能够有效地处理各种异常情况。此外，对于缓冲区的管理和错误处理机制也需要进行仔细的设计和实现，以确保数据在传输过程中不会丢失，并能够得到正确的处理。

您还想到哪些原因？下期我们接着分析。

边缘计算 ▼

ZLG嵌入式笔记(连载26) | CAN丢帧很常见，你知道有哪些主要原因吗？（中）

ZLG 致远电子 2025-02-20 11:41:07

在工程应用中，CAN 通信的稳定性至关重要，但丢帧和错误帧现象却时有发生。本文将简要分析导致这些问题的常见原因，并给出针对性的解决方案。

一般来说，使用 CAN 通信的场合，对通信的稳定性都有很高的要求。但在工程应用现场，经常遇到各种原因引起的丢帧或者出现错误帧的现象，下面对各种可能的原因进行简要说明。

总线冲突

CAN（Controller Area Network）总线是一种常用于汽车和工业控制系统中的通信协议和总线系统。在 CAN 总线中，当多个节点同时尝试发送消息时，可能会发生总线冲突。这种冲突可能会导致消息丢失，从而引发丢帧和通信错误。

在 CAN 总线中，每个消息都具有一个唯一的标识符（ID），用于确定消息的优先级。当多个节点尝试发送消息时，CAN 总线使用一种基于非争用的访问机制，即辨识出发消息的节点优先级，并将较低优先级的消息挂起，以确保高优先级消息能够顺利发送。这种机制使得 CAN 总线在大多数情况下能够避免碰撞和冲突。

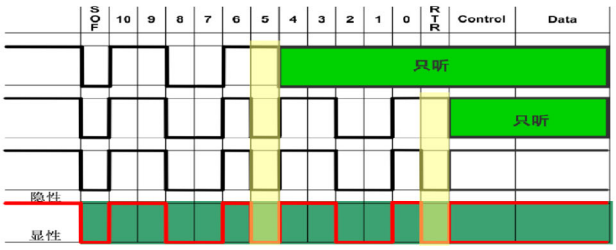


图1 总线冲突

然而，当多个节点具有相同优先级并且同时尝试发送消息时，仍然有可能发生总线冲突。为了减少这种冲突的发生，CAN 总线使用了非常快速的算法来检测碰撞，并且具有重发机制，以确保消息能够成功传输。

此外，CAN 总线还使用了一种错误检测和纠正机制，以确保即使发生了冲突或丢帧，接收节点仍然能够检测到错误并进行纠正，从而提高了通信的可靠性。

总线噪声和抖动

CAN 总线上的噪声和抖动可能导致信号失真，使得节点无法正确接收到消息，这种信号失真可能会导致丢帧。

图2 总线噪声

首先，噪声和抖动可能来自于 CAN 总线上的电磁干扰或者节点之间的电气连接问题。为了减少这种影响，可以采取以下措施：

- 使用合适的屏蔽和滤波器：在 CAN 总线的连接线路上使用屏蔽电缆和滤波器可以有效地减少电磁干扰对信号的影响。
- 确保良好的接地和电气连接：良好的接地和电气连接可以减少抖动和噪声的产生，确保信号的稳定传输。
- 优化节点的布局和电气设计：合理安排 CAN 节点的布局和电气设计可以减少节点之间的干扰，减少噪声和抖动的产生。

其次，针对已经产生的噪声和抖动，可以采取以下措施来减少其对信号的影响：

- 使用抗干扰的芯片和设备：选择具有良好抗干扰性能的芯片和设备可以有效地减少噪声和抖动对信号的影响。
- 优化信号调节和处理算法：通过优化信号调节和处理算法，可以减少噪声和抖动对信号的影响，提高节点对消息的正确接收率。

总之，针对 CAN 总线上的噪声和抖动问题，可以通过合理设计和布局、使用抗干扰的设备和优化信号处理算法等措施来减少其对信号的影响，确保节点能够正确接收消息，避免丢帧等问题的发生。

CAN总线负载过重

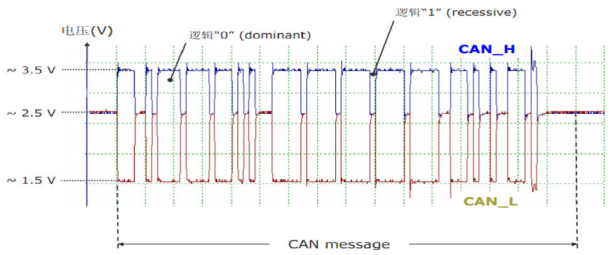


图3 总线负载过重

导致 CAN 总线负载过重的原因可能包括以下几个方面：

消息频率过高：

CAN 总线上的消息频率过高会导致总线负载增加，从而可能导致丢帧的问题。特别是在网络中存在大量节点同时发送消息或者某些节点频繁发送消息的情况下，会导致总线负载过重。

网络中节点数量增加：

当 CAN 网络中的节点数量增加时，总线负载也会增加，因为更多的节点需要共享总线带宽，可能导致消息冲突和丢失。

消息长度过长：

过长的消息长度会占用更多的总线带宽，导致总线负载过重。

针对上述可能的导致 CAN 总线负载过重的原因，给出的解决方法如下：

优化消息设计：

对于频率过高的消息，可以考虑优化消息设计，减少不必要的消息发送，或者将多个消息合并为一个更大的消息以降低总线负载。

考虑节点数量：

在设计 CAN 网络时，需要考虑网络中所需的节点数量，并合理规划总线带宽和消息发送频率，以避免总线负载过重。

使用CAN FD协议：

如果可能的话，可以考虑使用 CAN FD 协议。CAN FD 提供了更高的数据传输速率和更大的数据帧长度，可以减轻总线负载过重的问题。

消息过滤和优先级设置：

通过合理设置消息过滤和优先级，可以确保重要消息优先传输，避免总线负载过重导致丢帧。

性能评估和调整：

对 CAN 网络的性能进行评估，包括总线负载、消息冲突等情况，根据评估结果对网络进行调整和优化。

通过以上解决方法，可以有效地解决 CAN 总线负载过重可能导致丢帧的问题，确保 CAN 网络的稳定和可靠性。

您还想到哪些原因？下期我们接着分析。

边缘计算 ▼

ZLG嵌入式笔记(连载27) | CAN丢帧很常见，你知道有哪些主要原因吗？（下）

ZLG 致远电子 2025-02-25 11:30:54

CAN 总线在汽车和工业领域广泛应用，但通信问题可能影响其稳定性。本文探讨总线速率错误、电源不稳定和线程处理不当三大常见问题，分析原因并提供解决方法，助力优化通信可靠性。

总线速率设置错误

当总线速率设置错误时，可能会导致节点在接收消息时无法正确同步和解析，从而造成丢帧问题。

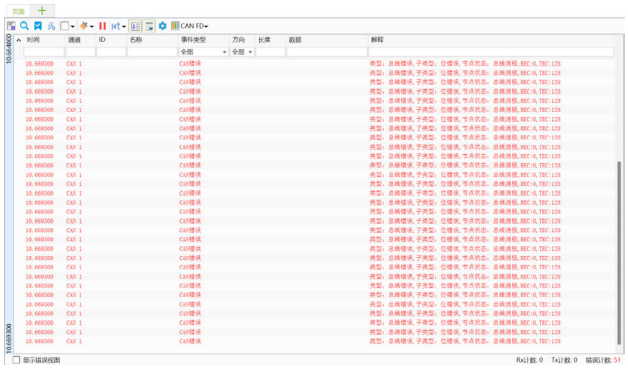


图1 总线速率设置错误

CAN 总线速率是指在 CAN 总线上传输数据的速率，它是 CAN 通信中的一个重要参数。总线速率的设置直接影响着节点之间消息的传输效率和可靠性。

如果总线速率设置错误，可能导致节点无法正确接收消息。这是因为节点在接收消息时需要根据正确的总线速率进行同步，如果速率设置错误，节点将无法正确同步消息的位时钟，导致消息无法正确解析，从而造成丢帧问题。

正确的总线速率设置是确保 CAN 通信稳定和可靠的关键因素之一。因此，对于总线速率设置错误可能导致的丢帧问题，需要对总线速率进行仔细的规划和设置。在设置总线速率时，需要考虑总线物理层特性、总线长度、总线负载等因素，确保速率设置与实际应用环境匹配。同时，对于不同的 CAN 控制器和总线标准（如 CAN 2.0、CANFD 等），也需要根据其规范进行正确的速率设置。

电源稳定性问题

电源稳定性对于 CAN 总线系统的稳定运行至关重要。电源不稳定可能导致节点的工作频率波动，电压波动等问题，这可能会影响节点的正常工作，包括消息发送和接收。不稳定的电源可能导致节点无法稳定地发送或接收消息，从而引发丢帧问题。

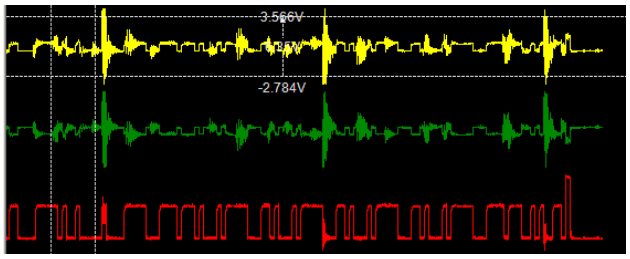


图2 电源稳定性问题

电源噪声可能会通过电源线传播到节点，影响节点的正常工作。这种噪声可能导致消息发送时的干扰，或者影响节点对消息的正确接收和解析，进而导致丢帧问题。

为了减少电源问题带来的丢帧，可以采取一些措施来确保良好的电源稳定性：

- 使用稳定的电源：选择高品质、稳定的电源设备，确保节点供电稳定。
- 添加电源滤波器：在节点的电源线上添加电源滤波器，可以有效地减少电源噪声对节点的影响。
- 确保良好的接地：良好的接地设计可以帮助减少电源问题对节点的影响，确保电源稳定性。
- 电源线路隔离：对于对电源稳定性要求较高的节点，可以考虑使用电源隔离设备，将其与其他电源线路隔离开，减少电源波动对节点的影响。

线程处理失当

这是一个真实案例。使用 CAN0 收数据，在压力大的情况下，接收会缺帧，但是用 ifconfig can0 看， err 和 drop 数量都为零。

最初程序 CAN 收发都在一个线程里，后来将收和发分别放在不同线程中，问题消失。

线程处理失当可能导致数据丢失，这可能是由于线程优先级设置不当，或者线程竞争条件导致的。当数据处理线程的优先级设置不当时，可能会导致其他高优先级的线程抢占资源，从而导致数据处理线程无法及时响应，造成数据丢失。这种情况下，需要仔细评估每个线程的优先级，并确保数据处理线程具有足够高的优先级，以确保数据的及时处理。

另一方面，线程竞争条件也可能导致数据丢失。例如，多个线程同时访问共享资源而没有正确的同步机制，可能导致数据处理不一致或丢失。这可能需要通过使用互斥锁、信号量或其他同步机制来确保对共享资源的访问是线程安全的，从而避免数据丢失的问题。ZLG 致远电子 CAN 设备的二次开发函数库 dll 均是线程安全的，有互斥锁。

ZLG嵌入式笔记(连载28) | CAN总线故障排查：从问题到解决的实战案例

ZLG 致远电子 2025-02-28 11:35:53

在工业现场的煤安监控网络中，CAN 总线通信常因复杂环境出现数据丢失问题。本文以一起煤安监控网络中 CAN 总线数据丢失的故障排查案例，简述了排查过程和解决方法，为工业现场 CAN 通信故障提供了参考。

现场通讯故障描述

用户现场为一煤安监控网络，节点包括一个主站设备、两个分站设备，主站设备对分站设备进行轮询式数据查询。故障出现时发现分站上传的数据出现缺失现象。致远电子工程师将专业工具——CANScope 分析仪接入故障网络捕获数据，然后进行分析。

图1 CANScope总线综合分析仪

对故障通讯网络快速体检

监控系统上电工作后，记录一段时间数据，通过【报文帧统计功能】分析，此次测试样本为 135 个帧，成功报文 119 个，正确率为 88.15%。错误帧类型主要包括 CRC 定界符错误、帧结束错误、应答定界符错误、数据场填充错误。使用 CANScope 分析仪捕获到网络中的错误数据，如图 2 所示。

图2 报文帧统计结果

数据丢失分析

用户程序采取轮询的方式查询分站数据，存在固有的数据周期，通过【流量分析功能】发现，未丢数据的周期中，包含 7 条有效报文，如图 3 所示。

图3 未丢数据周期

丢数据的周期中出现了红色的错误帧，如图 4 所示。有效报文数量 = 周期报文总数 - 错误帧数量。正常模式下 CAN 总线中出现错误帧后底层会实现自动重发，保证报文不会因为错误帧而丢失，然而在本网络中这一机制未能实现。通过检查软件，发现软件工程师在 CAN 控制器初始化代码中禁用了重发功能，导致错误帧不能重发。

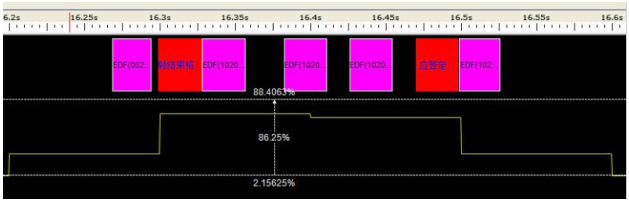


图4 丢数据周期

错误帧分析

通过修改代码解决了数据重发的问题，但是网络中偶尔冒出的错误帧仍然是工程师的一块心病，错误重发机制的使能仅仅是治标不治本，那么究竟是什么原因导致 CAN 网络中出现错误帧呢？需要对型号质量进行分析，这里用到了 CANScope 的信号质量分析功能。通过观察错误帧的波形，发现 CAN 信号上存在很严重的共模干扰，使得 CAN_H 和 CAN_L 上的单线波形畸变严重，如图 5 所示。

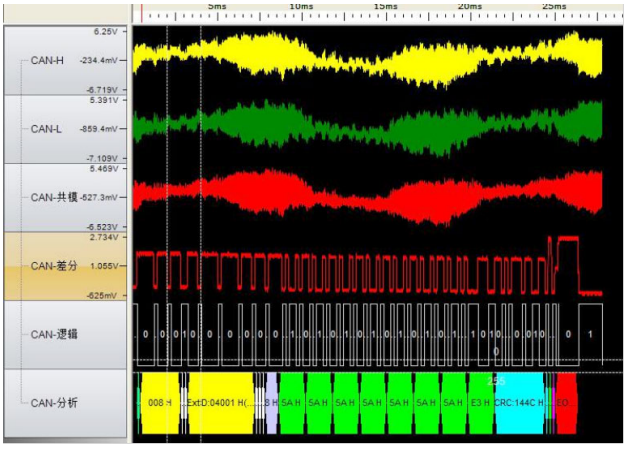


图5 波形分析

边缘计算 ▼

选取一条错误帧使用【FFT 分析功能】进行分析可以看到该错误帧信号上的频域特性，其中在 27KHz 频点上存在很强的能量，幅值甚至达到了 1.38V，如图 6 所示。CAN 总线的显隐性电平的压差很小，1V 左右的干扰很容易导致电平识别错误，从而出现错误帧。

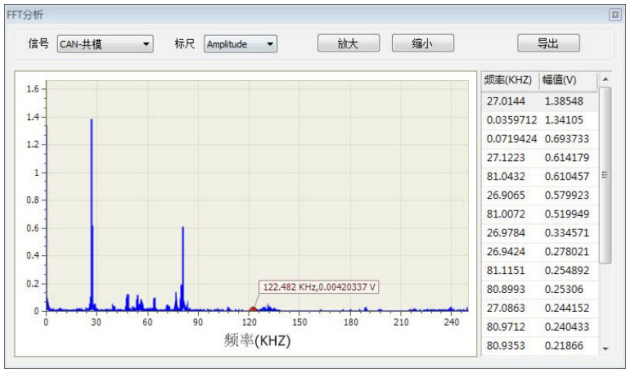


图6 FFT分析

这一干扰是否具有统计特性呢？使用【干扰统计功能】针对所有样本数据做干扰频点强度排序，发现干扰最强的频点集中在 27KHz 附近，如图 7 所示。因此判断在 CAN 网络附近应该存在这样一个干扰源。后经过仔细排查，发现这一干扰频率与开关电源的开关频率最吻合，由于所有 CAN 节点未做隔离导致电源串扰，引发错误帧。更换电源后故障消失，问题解决。

序号	幅度(V)	频率(HZ)	帧ID
109	1.777	27.028K	08AA0008 H
1	1.674	27.012K	错误帧
120	1.665	27.028K	08AA0008 H
98	1.596	72	08AA0008 H
64	1.575	27.028K	08AA0008 H
73	1.553	27.013K	81021100 H
91	1.518	38	01020008 H
134	1.505	38	01020008 H
97	1.497	27.021K	81020100 H
86	1.462	72	08AA0008 H
128	1.459	36	08AA0108 H
79	1.456	38	01020008 H
78	1.455	36	08AA0108 H
71	1.442	27.019K	错误帧
67	1.434	36	08AA0108 H
95	1.427	27.019K	错误帧
83	1.427	27.013K	81021100 H
117	1.400	27.013K	81021100 H

图7 干扰统计

RISC-V生态崛起：政策落地与高性能芯片的崛起

ZLG 致远电子 2025-03-06 11:33:35

近期，RISC-V 生态取得重要进展，包括高性能核心板的推出和在工业自动化、物联网等领域的广泛应用。其中，致远电子 MR6450 系列 RISC-V 核心板，凭借高性能和丰富的通信接口，成为工业控制和物联网领域的理想选择。

RISC-V生态崛起

2025 年，RISC-V 生态迎来重大发展机遇。据路透社报道，中国计划首次发布政策指导，鼓励在全国范围内使用开源 RISC-V 芯片，以加速减少对西方技术的依赖。

在技术与产品发布方面，阿里巴巴达摩院于 2025 年 2 月 28 日举办的“玄铁 RISC-V 生态大会”上，发布了玄铁 RISC-V 系列芯片及“无剑 600”开发平台，并宣布其首款服务器级处理器 C930 有望于 3 月正式交付。C930 的通用算力性能达到 SPECint2006 基准测试 15/GHz，具备 512 bits RVV1.0 和 8 TOPS Matrix 双引擎，将通用高性能算力与 AI 算力原生结合。

截至 2022 年年底，全球 RISC-V 芯片累计出货量已超 100 亿颗，中国贡献了其中的 50%，显示出其在全球市场的强劲增长。中国科学院计算技术研究所副所长包云岗预计，到 2030 年，全球 RISC-V 芯片市场规模将达到 927 亿美元（约合 6752.08 亿元人民币），年均复合增长率为 47.4%。

RISC-V 的应用场景不断拓展，尤其在汽车领域，其灵活性和可定制性使其在高级驾驶辅助系统（ADAS）等应用中展现出潜力。此外，雄安新区发布了涵盖智慧城市、交通管理等领域的 RISC-V 应用场景，推动其在城市级数字化中的应用。

目前，中国已汇聚超 300 家生态企业，形成从 IP 核到芯片量产的完整产业链。RISC-V 正通过 AI、智能汽车等高增长场景挑战 ARM 的垄断地位。

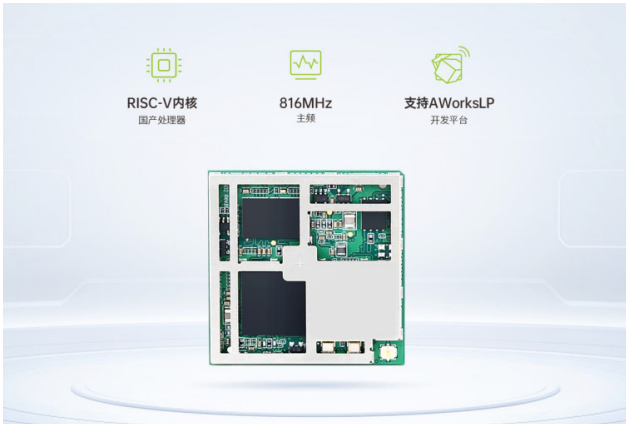
RISC-V架构的优势

RISC-V 是一种基于“精简指令集（RISC）”原则的开源指令集架构，诞生于 2010 年。其最大特性在于开源和精简，避免了专利和授权问题，为物联网等对成本敏感的领域提供了新选择。与 x86 和 ARM 指令集相比，RISC-V 融入了现代设计理念，具有以下优势：

- 更低的成本：开源特性无需支付授权费，降低了开发门槛。
- 更高的灵活性：模块化设计允许针对不同应用领域进行定制和优化。
- 更高的编程效率：基本指令数目仅 40 多条，简洁且易于扩展，提升了开发效率。

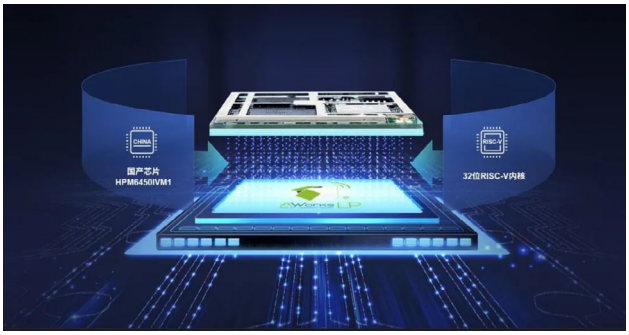
MR6450系列RISC-V核心板

去年，ZLG 致远电子推出了首款 RISC-V 核心板 -MR6450，凭借高性能和丰富接口，成为物联网和工业控制领域的理想选择。



核心特点：

- 816MHz 主频，高于 9000 CoreMark™和 4500 以上的 DMIPS 性能，支持双精度浮点运算、DSP 扩展及硬件图形加速；
- 板载 8MB/32MB SDRAM，8MB QSPI Flash；
- 双千兆以太网，iperf 测试带宽大于 900Mbps；
- 支持 IEEE 1588 PTP 规范，时间同步精度可达 17ns；
- 四路 CANFD 接口，最高速率可达 8Mbps；
- 多种通信接口：15 路 UART，4 路 SPI，4 路 I2C；
- 双路高速 USB2.0，内置 PHY；
- 4 个 PWM 定时器，调制精度可达 2.5ns，3 个 12 位 ADC，1 个 16 位 ADC。



MR6450 系列核心板基于国产芯片 HPM6450IVM1 开发，主频高，支持高速数据处理，适合工业控制、仪器仪表、电机控制等应用场合。

边缘计算 ▼

ZLG嵌入式笔记(连载29) | Linux固件烧写中的陷阱：文件系统异步写入引发的问题

ZLG 致远电子 2025-03-07 11:32:08

在工业生产中，固件烧写是确保产品正常运行的关键环节。本文通过一个实际案例，揭示了 Linux 系统下因文件写入异步性导致的固件烧写不完全问题。

客户案例

客户产线上批量生产时，用 SD 卡进行固件烧写，烧写完成后蜂鸣器提示，产线工人听到蜂鸣器鸣叫后就直接断电重启，进入测试环节，结果在测试中发现有部分产品启动就出现异常。

客户用解压方式烧写系统固件，解压命令执行完后，再运行一个二进制可执行程序，然后鸣叫蜂鸣器。乍一看逻辑非常正确，没有任何问题。可问题却较高概率出现，问题出在哪里呢?

后来经过反复排查，发现客户解压脚本里的可执行程序里面还有二次解压操作，而且用 system() 调用了 Linux 的 Shell 命令。System() 调用 Shell 命令会生成一个新的子进程，这样蜂鸣器鸣叫和二次解压分别在不同进程中，两者没有同步，解压完成和蜂鸣器鸣叫没有必然的先后顺序。按照蜂鸣器鸣叫就断电重启，这样就不可避免的出现了解压尚未完成就被断电的情况，从而引起文件烧写不完全，系统启动异常。

下面围绕该问题，对 Linux 文件系统写文件操作进行一些说明。

原理说明

由于 Linux 系统存在页高速缓存，对写入的内容起到了缓存作用，用户的写操作实际上会被延迟。当页高速缓存中的数据比后台存储的数据新的时候，这些数据就被称作脏数据。发生以下三种情况时，脏页才会被写回磁盘:

1. 当空闲内存低于一个特定的阈值时，内核必须将脏页写回磁盘以释放内存。
2. 当脏页在内存中驻留时间超过一个特定的阈值时，内核必须将超时的脏页写回磁盘，以确保脏页不会无限期地驻留在内存中。
3. 当用户进程调用 sync() 和 fsync() 系统调用时，内核会按照要求执行回写操作。

应对方案

如果更新脚本在解压命令后没有 sync 指令，或者程序更新代码在执行解压或者写操作后没有调用 fsync() 函数执行写操作，Linux 系统就会按照默认机制来实现写操作；如果没有达到如上前 2 个条件，发生了断电操作，则会导致写入数据不完整。

在程序更新脚本里，解压后必须执行 sync 指令完成写入同步，或者应用程序代码在写操作后调用 fsync() 函数完成写同步操作，确保数据写入存储器。

ZLG嵌入式笔记(连载30)｜看门狗，你真的会用吗?

ZLG 致远电子 2025-03-12 11:34:28

在嵌入式系统开发中，看门狗是一种重要的机制，用于在程序异常时自动重启系统，保障其稳定性。本文将深入探讨看门狗及其在程序稳定性中的作用，帮助您更好地理解和应用看门狗技术。

看门狗

首先需要明确一点：看门狗并不能直接提升程序本身的稳定性和可靠性。它仅能在程序运行出现异常时，通过重启应用或系统来恢复功能。程序的稳定性和可靠性，从根本上来说，取决于编程过程中的规范性和严谨程度。

看门狗主要分为硬件看门狗和软件看门狗，二者在实现方式和使用场景上存在显著差异。

1. 硬件看门狗

硬件看门狗的核心是一个定时电路，由被监控的 CPU 提供周期性的“清狗”信号，即“喂狗”，以清零定时器。在 CPU 正常工作时，能够定时“喂狗”，因此看门狗内的定时器不会溢出。一旦 CPU 出现故障，无法继续提供“喂狗”信号，看门狗内的定时器就会溢出，触发复位信号，导致 CPU 重启，俗称“被狗咬”。

硬件看门狗又可分为两种类型：

1.1 外部独立硬件看门狗

这种看门狗独立于 MCU/MPU 之外，无需驱动程序支持，只需在规定时间内为喂狗引脚提供一个电平变化即可，通常通过 GPIO 实现电平切换。其喂狗时间不可设置，一旦硬件连接完成，就无法关闭，除非断开硬件连接。这种看门狗具有极高的可靠性，适用于对产品可靠性要求极高的场合。不过，由于它是独立的硬件设备，因此会增加一定的硬件成本。

外部独立硬件看门狗，主要用于解决恶劣外部环境导致系统死机且无法人工干预的问题，也能应对硬件故障引发的系统死机重启需求。

1.2 内置硬件看门狗

内置硬件看门狗是利用处理器内部的定时器实现定时操作。它需要在系统中编写驱动程序，初始化定时器并实现喂狗操作，甚至可以调整定时器的参数。这种看门狗的喂狗时间可调，还可以通过特殊指令关闭，成本较低。然而，在处理器死机的情况下，它可能会失效，因此适用于对硬件可靠性要求不是特别严格的场合。

内置硬件看门狗，系统驱动 + 应用程序喂狗，主要用于解决应用程序问题引起的复位重启，也能应对部分外部环境导致的复位重启。

2. 软件看门狗

软件看门狗的原理与硬件看门狗类似，只是将硬件电路上的定时器替换为处理器的内部定时器。这种方式可以简化硬件电路设计，但在可靠性方面不如硬件定时器。例如，如果系统内部定时器自身发生故障，就无法检测到问题。虽然可以通过双定时器相互监视来提高可靠性，但这不仅会增加系统开销，也无法解决所有问题，比如中断系统故障导致定时器中断失效。

软件看门狗，系统驱动 + 应用程序喂狗，主要用于监控应用程序死机问题。

看门狗的正确使用

看门狗本身并不是用来解决系统出现的问题。在调试过程中发现的故障，应该通过排查设计本身的错误来解决。加入看门狗的目的是应对程序潜在错误和恶劣环境干扰等因素导致的系统死机，在无人干预的情况下自动恢复系统正常工作状态。然而，看门狗也不能完全避免故障造成的损失。毕竟从发现故障到系统复位恢复正常这段时间内，系统会处于停滞状态。此外，一些系统还需要在复位前保护现场数据，并在重启后恢复现场数据，这可能需要额外的软硬件开销。

可靠性排序与喂狗操作

综合来看，看门狗的可靠性排序为：外部独立看门狗 > 内置硬件看门狗 > 软件看门狗。

对于喂狗操作，外部独立看门狗由系统自动完成，应用程序无需关心。而应用程序需要喂的狗要么是内置硬件看门狗，要么是软件看门狗，这取决于平台本身提供的资源。应用程序必须在规定时间内进行喂狗操作，以监控程序本身的运行健康状况。如果程序编写不当导致异常不喂狗，就会引发系统复位重启。这就需要程序开发人员进行问题定位和排查，解决可能影响系统运行的各种潜在问题。

边缘计算 ▼

ZLG嵌入式笔记(连载31) | 如何正确选择嵌入式文件系统?

ZLG 致远电子 2025-03-17 11:31:53

在 Linux 嵌入式系统中，文件系统和缓存机制常导致数据存储稳定性问题。本文通过案例分析原因，对比不同文件系统特性，为开发者提供优化建议，助力提升数据稳定性和系统可靠性。

前言

基于 Linux 的嵌入式操作系统，由于有文件系统以及缓存的存在，在数据存储方面的使用注意事项比单片机直接写存储器的应用会有更多的要求才能达到数据的稳定可靠。如下都是比较常见的异常：

1. 系统没有正常关机，导致磁盘出现文件系统错误或者变为只读；
2. 频繁读写数据，文件系统中文件被损坏或者变为只读；
3. 在读写过程中突然断电，导致文件系统变为只读。

案例回顾

先看几个真实案例：

案例1

有客户反馈在使用 M3354-512LI-F1GT 的核心板当中。发现会出现文件系统变成只读的现象，此时主机就不能正常使用。

案例2

EPC-287C-L ARM9 系列核心板，当网关程序和配置信息放在 /opt 文件夹下面执行一段时间，偶尔出现 /opt 文件夹下的信息都被清空或者里面权限变为只读。

案例3

M280 工控板在运行一段时间后，/opt 目录下的文件不能编辑，编辑的时候会出现只读的情况，但是文件的属性又不是只读的。

案例分析

- 案例 1 的问题可能是本身 ubifs 的机制问题，解决方式是通过更新 ubifs 数据分区的格式为 yaffs 格式。
- 偶发性的 opt 目录只读问题，暂时没有好的办法去解决。一般情况下，建议在对 opt 分区或者文件系统进行写操作时，使用 sync 命令或函数同步数据。由于 SLC NandFlash 的标准读写次数上限为 10 万次，因此，不建议太频繁的进行擦除写入。
- 硬件上，为了保证产品稳定，建议添加备份电源或者大电容，保证 10s 左右的续航时间，让系统有足够时间去做文件系统同步。

要点提示

当然，不单只有 ubifs 文件系统才会出现 opt 目录只读问题，在 ext4 和 yaffs2 的文件系统也有出现。比如没有正确按照文档的步骤操作打包或者烧写文件系统的镜像，也会直接造成文件系统镜像损坏。当文件系统出现损坏或错误时，操作系统会将其自动挂载为只读模式，以防止进一步的数据损坏。这是为了保护数据的完整性和安全性。有时，操作系统在引导过程中会自动检测文件系统错误。为了避免这些错误进一步扩散，操作系统会将文件系统以只读模式挂载，以便用户可以修复错误。

还有一种是人为主动将其挂载为只读模式，将文件系统以只读模式挂载可以增加系统的安全性，防止未经授权的用户修改或删除关键文件。特别是在网络环境中，只读文件系统可以起到一定的保护作用。可以在发生损坏、错误、硬件故障时保护数据的完整性和安全性。

根据应用场景选择合适的文件系统，能有效减少 opt 分区变为只读的概率，提高产品的稳定性。

图 1 和图 2 摘自 TOSHIBA 的公开文档，为我们在做文件系统的选择时提供了很好的参考。图 1 列出了电子产品对文件系统的参数要求，如快速启动、IO 性能、内存消耗、使用寿命和掉电忍受能力。

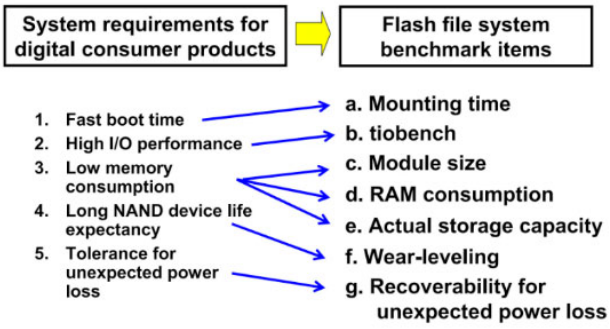


图1 选择文件系统考虑的因素

不同的文件系统因为设计理念的不同，在上述因素的实现以及应用场合也是不一样的。在图 2 所列的 4 种文件系统特性对比中，可以很清楚看到，YAFFS2 对内存消耗比较小，但不太适合于经常写数据的应用，比较适合于存储静态数据的应用场景。如果要进行动态数据存储，建议用户进行数据写平衡加强处理。而 UBIFS 文件系统则相反，对内存消耗较大，能适用于频繁写数据的场合，但因为文件系统缓存的原因，在抗掉电能力稍弱，需要在硬件和系统方面做一些补救。

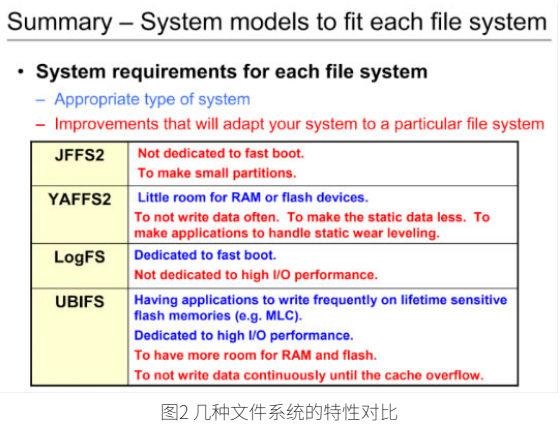


图2 几种文件系统的特性对比

简单归纳一下，在产品设计中一定要根据应用场景来选择合适的文件系统才能保证产品稳定和数据可靠。不经常写数据可优先选择 YAFFS2，经常写数据则应当考虑 UBIFS；经常异常掉电，优先选择 YAFFS2，有掉电保护则可选择 UBIFS。

边缘计算 ▼

ZLG嵌入式笔记(连载32) | 拯救NANDeMMC：延长闪存寿命

ZLG 致远电子 2025-03-25 11:41:16

随着电子设备的广泛应用，NAND 闪存和 eMMC 作为主流存储介质，其使用寿命受到广泛关注。本文将探讨其损坏的软件原因，并提供延长使用寿命的实用方法。

前言

长时间运行后出现 NAND 或者 eMMC 损坏，可能的原因有物理损坏如雷击损坏，也有可能因为频繁擦写操作引起寿命到期损坏。下面就应用软件方面的可能性进行探讨，寻求延长 NAND/eMMC 使用寿命的方法。

闪存的寿命和计算公式

在 NAND 闪存中，P/E Cycle 也称为擦除次数，是判断 NAND 闪存寿命的关键参数。随着 P/E Cycle 的增加，浮栅与沟道之间的氧化层被磨损的越来越严重，导致浮栅中电子的控制越来越艰难，最终结果就是：NAND 的寿命走到了尽头。每颗 NAND 闪存，在出厂的那一刻，寿命就固定了。

NAND 闪存根据存储颗粒密度和结构差异，可分为 SLC、MLC、TLC 和 QLC。存储密度越来越高，容量越来越大，但单位可擦写次数却越来越少。SLC 的密度最低，擦写次数最多，使用寿命最长，但现在很少能买到了，特别是大容量闪存，基本都是 MLC、TLC 甚至 QLC 了。

一般都说 MLC 的擦写寿命是 3000~10000 次，但实际上去咨询半导体原厂，得到的答案通常都是 3000 次，哪怕三星、海力士以及 Skyhigh 这样的国际品牌，都是这个答案。eMMC 实际上是 NAND 闪存加了控制器，底层存储还是 NAND 闪存，所以可在此一并讨论。

如何延长闪存的使用寿命，确保存储的数据可靠，是工业产品一直在试图解决的问题。一个 NAND 闪存能使用多久，我们先给出一个预测公式：

$$\text{预期使用寿命(天)} = \frac{\text{分区容量大小(字节)} * \text{可擦写次数} * (1 - \text{文件系统开销})}{\text{写入放大} * \text{每天写入的次数} * \text{每次写入的数据量（字节）}}$$

对于特定的 Linux 系统，文件系统开销可看成一个常数，暂时不考虑。根据公式不难看出，预期使用寿命与分区容量大小、可擦写次数正相关，与写入放大、每天写入的次数以及每次写入的数据量成负相关。在产品设计方案选型阶段，容量、可擦写次数是一个正相关变量，但对于特定的一个产品，闪存一旦选定，可擦写次数也就定了，如果有条件扩大分区容量，也是能改善使用寿命的，但改善非常有限。要想比较有效的提高闪存寿命，必须从分母的写入放大、每天写入的次数以及每次写入的数据量上面来优化。

写入放大

我们知道，闪存在写入数据前必须先进行擦除，而擦除操作的粒度与写入操作相比低得多，执行这些操作就会多次移动（或改写）用户数据和元数据。因此，要改写数据，就需要读取闪存某些已使用的部分，更新它们，并写入到新的位置，如果新位置在之前已被使用过，还需连同先擦除；由于闪存的这种工作方式，必须擦除改写的闪存部分比新数据实际需要的大得多，这就是写入放大，此倍增效应会增加请求写入的次数。

写入放大，简单的计算公式如下：

$$\text{写入放大 WA} = \frac{\text{闪存写入的数据量}}{\text{主控写入的数据量}}$$

影响写入放大的因素：

- 垃圾回收，启用垃圾回收，WA 会减小。
- 预留空间，增大预留空间，能减小 WA。
- 顺序写入，理论上顺序写入，WA 为 1，当然其他因素会影响到 WA。
- 随机写入，写入到非连续的 LBA 对写入放大的影响最大。零散写入会带来极大的 WA 影响。例如写入一个字节，实际上闪存最小写入单位是页，擦除单位是块，这样会影响到在这个块内的所有数据的搬移和写入，数据量会非常大。
- 数据压缩，数据压缩后再写入，能减少数据量的写入。
- 删除重复数据，这样能减少磁盘占用，能减小 WA。

从公式来看，减小 WA 能增加闪存寿命，具体方法有启用垃圾回收、增大预留空间、尽量顺序写入、进行数据压缩以及删除重复数据等。

如何计算数据量?

写入 NAND 数据量的计算，并不是按照应用程序数据来计算的，这与 NAND 闪存的结构和擦写方式紧密相关。

一般 NAND 闪存内部结构分为多个块 (Block)，每个块包含多个页面 (page)，每个页面又是由有效个数据区和 spare area 区 (即 OOB 区) 组成。NAND 闪存以块为单位进行擦除，以页为单位进行读写。

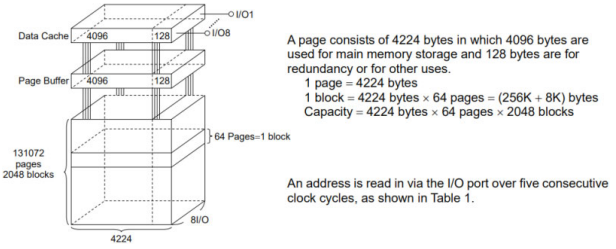


图2 几种文件系统的特性对比

图 1 所示的 NAND 闪存，每个页面为 4096 字节，一个块为 64 页，整个器件为 2048 块，总容量为 4096*64*2048=512MB。

如果产品使用的 NAND 是这样的结构，哪怕写入 1 字节数据，在计算数据量的时候，也得按 4096 向上取整，即 4096 字节，而不是 1 字节。当然，写入 4095 字节也是按 4096 向上取整，4096 字节。同理，写入 4097 字节数据，则向上取整为 8192 字节。

如果不清楚 NAND 闪存的这些具体信息，可以查看数据手册或者内核启动信息：

```
nand: device found, Manufacturer ID: 0x01, Chip ID: 0xda
nand: AMD/Spansion S34ML02G2
nand: 256 MiB, SLC, erase size: 128 KiB, page size: 2048, OOB size: 128
```

改善措施

综合上述信息，要延长 NAND/eMMC 的读写寿命，就要尽量减少对闪存的擦写次数，特别是零散数据写入。归纳一下，可以采取以下措施：

1. 合理分区，动静分离

- 动静数据分离，将数据按修改频率分组。一般可将系统分区与数据分区分开，确保系统分区不受数据写的影响。
- 有效地使用 RAM 文件系统。系统 log 信息，以及应用程序的 log 信息，不要直接写入闪存，尽量写在 RAM 文件系统中，仅对异常 log 定期写入闪存，减少闪存写入次数。

2. 减少数据写入次数

- 先将数据写在内存里，可以在系统中生成一个 Ramdisk 并挂载到某个目录，将需要写入到闪存的数据先暂存在这个目录，到一定时间再写到闪存里面去。也可以使用系统的 ramfs 目录存放暂存数据。
- 将频繁修改的数据存储在一块连续的区域中，并定期将数据迁移到新的区域，以避免在同一区域反复擦写。
- 进行数据压缩，减小写入的数据量，特别是对于海量数据的应用，数据压缩尤为重要。

3. 避免零碎散数据写入

确保在写入数据时，数据块的大小是擦除块大小的整数倍，以避免无效的擦除操作。

4. 维持合理的分区占用率

- 不要写满分区，维持较低的磁盘占用率，能有效地提高闪存使用寿命。磁盘占用率管理分两种情况，采用系统的磁盘配额管理或者应用程序自行管理。
- 对于 eMMC 闪存，在使用 Ext3/4 文件系统的时候，启用磁盘配额管理，确保磁盘使用率在一个合理的范围之内。注意，启用磁盘配额管理，需要普通用户，建议产品应用程序都运行在普通用户模式，而不是 root 用户来运行。
- 对于 NAND 闪存，一般都会使用 Yaffs/Yaffs2 文件系统，支持不了磁盘配额。这种情况要想维持合理的磁盘占用率的话，可以写一个磁盘占用率监控程序，当磁盘占用率超过某个阈值，就进行磁盘清理工作，删除不重要或者陈旧的文件，以保持闪存处于比较健康的状态。
- 无论是 NAND 还是 eMMC，都要定期进行数据处理，对于过期的数据要及时删除，减小无用数据在闪存使用中的搬运和写入次数。

5. 闪存健康管理

对于 eMMC，可以在系统中通过 mmc_erase_info 文件查看 eMMC 的擦写次数，以判断 eMMC 的健康状况。一旦 eMMC 的擦写次数已经接近厂商理论值，就要特别留意此闪存上的数据，并做好预案处理。

```
/ #
/ # dmesg | grep mmc_earse_info
mmc_earse_info:data_size=5006234M
mmc_earse_info:earse_count_sum=4328963 max_earse_count=64562 max_earse_count_num=547
mmc_earse_info:blk_sum=1024 bad_block_sum=3
/ #
```

对于 NAND 闪存，不能像 eMMC 这样方便的查看 NAND 闪存的擦写次数，则可以自行统计 NAND 擦写次数，并结合寿命预测公式对 NAND 闪存的健康状况进行大致判断。

6. 坏块检查和替换

定期进行坏块检测和替换，以防止使用坏块导致的数据丢失和写失败。一旦发现 NAND 坏块数量超过一定数值，就要对 NAND 整体健康做重新评估，并启动数据安全性预案处理。

AWTK-WEB 快速入门(3) - C 语言 Http 应用程序

ZLG 致远电子 2025-01-15 11:35:29

XMLHttpRequest 改变了 Web 应用与服务器交换数据的方式，fetch 是 XMLHttpRequest 继任者，具有更简洁的语法。本文介绍一下如何使用 C 语言开发 AWTK-WEB 应用程序，并用 fetch 访问远程数据。

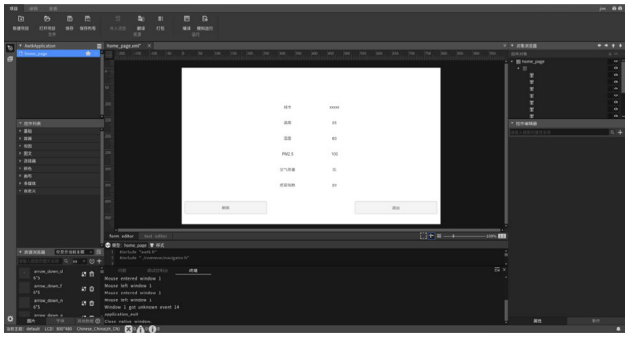
用 AWTK Designer 新建一个应用程序

先安装 AWTK Designer：

https://awtk.zlg.cn/web/index.html

1. 新建应用程序

这里假设应用程序的名称为 AwtkApplicationCHttp，后面会用到，如果使用其它名称，后面要做相应修改。



2. 编写代码

2.1 用 VSCode 打开目录 AwtkApplicationCHttp，并打开文件 src/pages/home_page.c。

使用其它文本编辑器或 IDE 也可以，推荐使用 VSCode，并开启 Copilot，将大幅提升开发效率。

2.2 添加事件处理函数。可以参考下面的代码：

```
// C 函数，用于处理接收到的数据
void home_page_on_fetch_data(void* ctx, const char* data, uint32_t data_size) {
  widget_t* win = WIDGET(ctx);

  conf_doc_t* doc = conf_doc_load_json(data, data_size);
  if (doc != NULL) {
    const char* city = conf_doc_get_str(doc, "cityInfo.city", "unknown");
    const char* shidu = conf_doc_get_str(doc, "data.shidu", "unknown");
    const char* wendu = conf_doc_get_str(doc, "data.wendu", "unknown");
    const char* quality = conf_doc_get_str(doc, "data.quality", "unknown");
    const char* ganmao = conf_doc_get_str(doc, "data.ganmao", "unknown");
    double pm25 = conf_doc_get_float(doc, "data.pm25", 0);

    widget_set_child_text_utf8(win, "city", city);
    widget_set_child_text_utf8(win, "shidu", shidu);
    widget_set_child_text_utf8(win, "wendu", wendu);
    widget_set_child_text_utf8(win, "quality", quality);
    widget_set_child_text_utf8(win, "ganmao", ganmao);
    widget_set_child_text_with_double(win, "pm25", "%.0f", pm25);

    conf_doc_destroy(doc);
  }

  printf("Received data (%d bytes): %s\n", data_size, data);
}

// C 函数，用于处理错误
void home_page_on_fetch_error(void* ctx, const char* error_message) {
  printf("Error: %s\n", error_message);
}

static ret_t on_update(void* ctx, event_t* e) {
  widget_t* win = WIDGET(ctx);
  const char* url = "http://localhost:8080/AwtkApplicationCHttp/res/assets/default/raw/data/weather.json";
  return_value_if_fail(win != NULL, RET_BAD_PARAMS);

  fetch_data(win, "GET", url, home_page_on_fetch_data, home_page_on_fetch_error, NULL);

  return RET_OK;
}
/**
 * 初始化窗口
 */
ret_t home_page_init(widget_t* win, void* ctx) {
  (void)ctx;
  return_value_if_fail(win != NULL, RET_BAD_PARAMS);

  widget_child_on(win, "update", EVT_CLICK, on_update, win);
```

```
    return RET_OK;
}
```

注意：控件的名称一定要和 home_page.xml 保持一致。

3. 在 AWTK Designer 中，执行“打包” “编译” “模拟运行”

正常情况下可以看到如下界面：

点击“关闭”按钮，退出应用程序。

编写配置文件

具体格式请参考，特殊平台编译配置：
https://github.com/zlgopen/awtk/blob/master/docs/build_config.md
这里给出一个例子，可以在此基础上进行修改，该文件位于：
examples/AwtkApplicationCHttp/build.json

```
{
  "name": "AwtkApplicationCHttp",
  "version": "1.0",
  "assets": "res/assets",
  "vendor": "zlgopen",
  "app_name": "org.zlgopen.AwtkApplicationCHttp",
  "copyright": "Guangzhou ZHIYUAN Electronics Co.,Ltd.",
  "themes":["default"],
  "sources": [
    "src/*.c",
    "src/common/*.c",
    "src/pages/*.c",
    "src/*.h",
    "src/common/*.h",
  "web": {
    "app_type": "c",
    "assets": "design",
    "config": {
      "fontScale": "0.8",
      "defaultFont": "sans"
    }
  }
```

编译 WEB 应用程序

进入 awtk-web 目录，不同平台使用不同的脚本编译：
Windows 平台

./build_win32.sh examples/AwtkApplicationCHttp/build.json release
Linux 平台

./build_linux.sh examples/AwtkApplicationCHttp/build.json release
MacOS 平台

./build_mac.sh examples/AwtkApplicationCHttp/build.json release
请根据应用程序所在目录，修改配置文件的路径。

运行

1. 正常启动

./start_web.sh
2. 调试启动

start_web_debug.sh
3. 用浏览器打开 URL：http://localhost:8080/AwtkApplicationCHttp

点击 “更新” 按钮，可以看到数据更新。

云平台 ▼

AWTK DEMOUI
ZLG开源GUI引擎AWTK: https://github.com/zlgopen/awtk

AWTK-WEB 快速入门(4) - JS Http 应用程序

ZLG 致远电子 2025-01-22 11:29:09

XMLHttpRequest 改变了 Web 应用程序与服务器交换数据的方式，fetch 是其继任者。本文介绍一下如何使用 JS 语言开发 AWTK-WEB 应用程序，并用 fetch 访问远程数据。

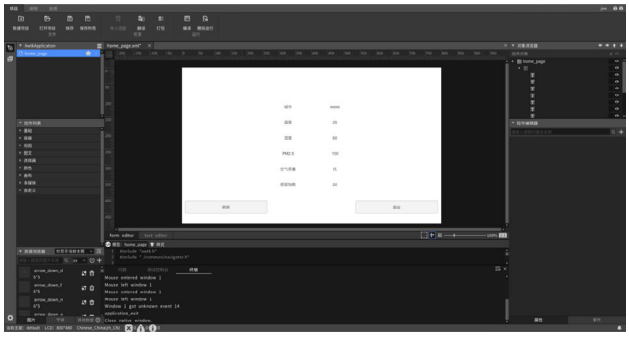
用 AWTK Designer 新建一个应用程序

先安装 AWTK Designer：

https://awtk.zlg.cn/web/index.html

1. 新建应用程序

这里假设应用程序的名称为 AwtkApplicationCHttp，后面会用到，如果使用其它名称，后面要做相应修改。



2. 编写代码

2.1 删除 src 目录下全部文件（留着也可以，只是看起来比较乱），在 src 目录创建 js 目录。

2.2 在 src/js 下创建 application.js，内容如下：

```
function applicationInit() {
  home_page_open();
}

applicationInit()
```

2.2 添加事件处理函数。可以参考下面的代码：

```
async function on_update_clicked(evt) {
  var e = TPointerEvent.cast(evt);
  var widget = TButton.cast(e.target);

 const win = widget.getWindow();
  const url = "http://localhost:8080/AwtkApplicationJSHttp/res/assets/default/raw/data/weather.json";
 try {
   const response = await fetch(url);
   if (!response.ok) {
    throw new Error("Network response was not ok " + response.statusText);
   }
   const json = await response.json();
   win.setChildText("city", json.cityInfo.city);
   win.setChildText("wendu", json.data.wendu);
   win.setChildText("ganmao", json.data.ganmao);
   win.setChildText("quality", json.data.quality);
   win.setChildText("shidu", json.data.shidu);
    win.setChildTextWithDouble("pm25", "%.0f", json.data.pm25);
  } catch (error) {
    console.error("There was a problem with the fetch operation:", error);
  }
 }

 function home_page_open() {
  var win = TWindow.open("home_page");
  var update = win.lookup("update", true);

  update.on(TEventType.CLICK, on_update_clicked);

  win.layout();
 }
```

注意：控件的名称一定要和 home_page.xml 保持一致。

3. 在 AWTK Designer 中，执行“打包” “编译” “模拟运行”

正常情况下可以看到如下界面：

点击“关闭”按钮，退出应用程序。

编写配置文件

具体格式请参考，特殊平台编译配置：
https://github.com/zlgopen/awtk/blob/master/docs/build_config.md

这里给出一个例子，可以在此基础上进行修改，该文件位于：
examples/AwtkApplicationJSHttp/build.json

```
{
  "name": "AwtkApplicationJSHttp",
  "version": "1.0",
  "app_type":"js",
  "author": "xianjimli@hotmail.com",
  "copyright": "Guangzhou ZHIYUAN Electronics Co.,Ltd.",
  "themes":["default"],
  "sources": [
    "src/js/*.js"
  ]
}
```

编译 WEB 应用程序

进入 awtk-web 目录，不同平台使用不同的脚本编译：
Windows 平台

./build_win32.sh examples/AwtkApplicationJSHttp/build.json release
Linux 平台

./build_linux.sh examples/AwtkApplicationJSHttp/build.json release
MacOS 平台

./build_mac.sh examples/AwtkApplicationJSHttp/build.json release
请根据应用程序所在目录，修改配置文件的路径。

运行

1. 正常启动

./start_web.sh

2. 调试启动

start_web_debug.sh
3. 用浏览器打开 URL：http://localhost:8080/AwtkApplicationJSHttp

点击 “更新” 按钮，可以看到数据更新。

AWTK-WEB 快速入门(5) - C 语言 WebSocket 应用程序

ZLG 致远电子 2025-02-19 11:32:49

WebSocket 可以实现双向通信，适合实时通信场景。本文介绍一下使用 C 语言开发 AWTK-WEB 应用程序，并用 WebSocket 与服务器通讯。

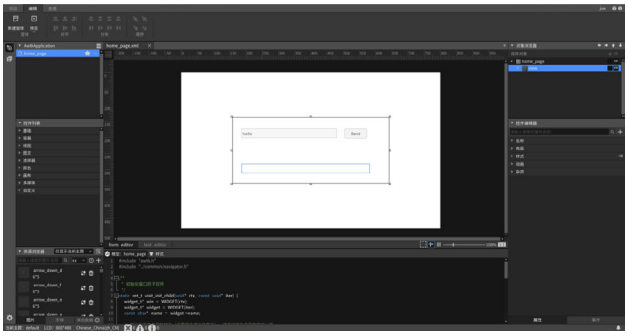
用 AWTK Designer 新建一个应用程序

先安装 AWTK Designer：

https://awtk.zlg.cn/web/index.html

1. 新建应用程序

这里假设应用程序的名称为 AwtkApplicationCHttp，后面会用到，如果使用其它名称，后面要做相应修改。



2. 编写代码

2.1 用 VSCode 打开目录 AwtkApplicationCWebSocket，并打开文件 src/pages/home_page.c。

使用其它文本编辑器或 IDE 也可以，推荐使用 VSCode，并开启 Copilot，将大幅提升开发效率。

VSCode 下载地址：https://code.visualstudio.com

2.2 添加事件处理函数。可以参考下面的代码：

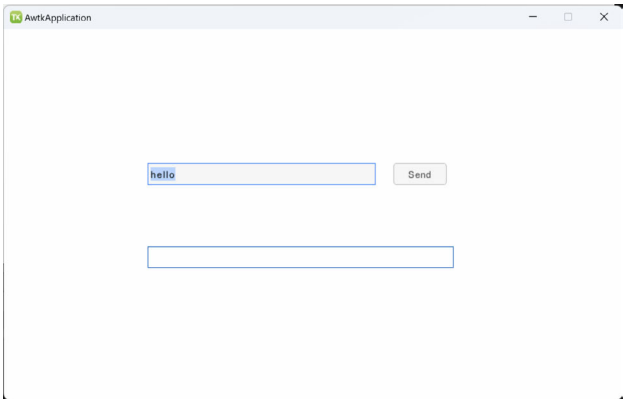
```
typedef struct _home_page_t {
  widget_t* send;
  widget_t* send_text;
  widget_t* recv_text;
  EMSCRIPTEN_WEBSOCKET_T sock;
} home_page_t;

static home_page_t* home_page_create(widget_t* win) {
  home_page_t* home_page = TKMEM_ZALLOC(home_page_t);

assert(win != NULL);
 assert(home_page != NULL);
  if (home_page != NULL) {
    home_page->send = widget_lookup(win, "send", TRUE);
     assert(home_page->send != NULL);
      home_page->send_text = widget_lookup(win, "send_text", TRUE);
     assert(home_page->send_text != NULL);
      home_page->recv_text = widget_lookup(win, "recv_text", TRUE);
     assert(home_page->recv_text != NULL);
    }

    return home_page;
  }

  static ret_t home_page_destroy(home_page_t* home_page) {
    if (home_page != NULL) {
      emscripten_websocket_close(home_page->sock, 0, 0);
      TKMEM_FREE(home_page);
    }

    return RET_OK;
  }

  bool WebSocketOpen(int eventType, const EmscriptenWebSocketOpenEvent* e, void* userData) {
    home_page_t* home_page = (home_page_t*)userData;
    return_value_if_fail(home_page != NULL, FALSE);

    widget_set_text_utf8(home_page->recv_text, "opened");

    /*
     emscripten_websocket_send_utf8_text(e->socket, "hello on the other side");
     char data[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
      emscripten_websocket_send_binary(e->socket, data, sizeof(data));
     */

    return 0;
  }

  bool WebSocketClose(int eventType, const EmscriptenWebSocketCloseEvent* e, void* userData) {
    home_page_t* home_page = (home_page_t*)userData;
    return_value_if_fail(home_page != NULL, FALSE);
```

```
    emscripten_websocket_delete(e->socket);
    widget_set_text_utf8(home_page->recv_text, "closed");
    home_page->sock = 0;

    return 0;
  }

  bool WebSocketError(int eventType, const EmscriptenWebSocketErrorEvent* e, void* userData) {
    home_page_t* home_page = (home_page_t*)userData;
    return_value_if_fail(home_page != NULL, FALSE);

    widget_set_text_utf8(home_page->recv_text, "error");

    return 0;
  }

  bool WebSocketMessage(int eventType, const EmscriptenWebSocketMessageEvent* e, void* userData) {
    home_page_t* home_page = (home_page_t*)userData;
    return_value_if_fail(home_page != NULL, FALSE);

    if (e->isText) {
      widget_set_text_utf8(home_page->recv_text, (const char*)(e->data));
    }

    return 0;
  }

  static ret_t on_send(void* ctx, event_t* e) {
    home_page_t* home_page = (home_page_t*)ctx;
    return_value_if_fail(home_page != NULL, RET_BAD_PARAMS);

    if (home_page->sock > 0) {
      char text[1024];
      widget_get_text_utf8(home_page->send_text, text, sizeof(text) - 1);
      if (text[0] != '\0') {
        emscripten_websocket_send_utf8_text(home_page->sock, text);
      }
    }

    return RET_OK;
  }

  EMSCRIPTEN_WEBSOCKET_T create_socket(void* user_data, const char* url) {
    EmscriptenWebSocketCreateAttributes attr;
    emscripten_websocket_init_create_attributes(&attr);

    attr.url = url;
    attr.protocols = "binary,base64";
    EMSCRIPTEN_WEBSOCKET_T socket = emscripten_websocket_new(&attr);

    emscripten_websocket_set_onopen_callback(socket, user_data, WebSocketOpen);
    emscripten_websocket_set_onclose_callback(socket, user_data, WebSocketClose);
    emscripten_websocket_set_onerror_callback(socket, user_data, WebSocketError);
    emscripten_websocket_set_onmessage_callback(socket, user_data, WebSocketMessage);

    return socket;
  }

  /**
   * 初始化窗口
   */
  ret_t home_page_init(widget_t* win, void* ctx) {
    (void)ctx;
    const char* url = "ws://localhost:8090/";
    home_page_t* home_page = home_page_create(win);
    return_value_if_fail(win != NULL && home_page != NULL, RET_BAD_PARAMS);

    assert(emscripten_websocket_is_supported());

    home_page->sock = create_socket(home_page, url);
    widget_on(home_page->send, EVT_CLICK, on_send, home_page);

    return RET_OK;
  }
```

注意：控件的名称一定要和 home_page.xml 保持一致。

3. 在 AWTK Designer 中，执行“打包” “编译” “模拟运行”

正常情况下可以看到如下界面：



编写配置文件

具体格式请参考，特殊平台编译配置：

https://github.com/zlgopen/awtk/blob/master/docs/build_config.md

这里给出一个例子，可以在此基础上进行修改，该文件位于：

examples/AwtkApplicationCWebSocket/build.json

```
{
  "name": "AwtkApplicationCWebSocket",
  "version": "1.0",
  "assets": "res/assets",
  "vendor": "zlgopen",
  "app_name": "org.zlgopen.AwtkApplicationCWebSocket",
  "author": "xianjimli@hotmail.com",
  "copyright": "Guangzhou ZHIYUAN Electronics Co.,Ltd.",
  "themes":["default"],
  "sources": [
    "src/*.c",
    "src/common/*.c",
    "src/pages/*.c",
    "src/*.h",
    "src/common/*.h",
    "src/pages/*.h"
  ],
  "web": {
    "app_type": "c",
    "assets": "design",
    "config": {
      "fontScale": "0.8",
      "defaultFont": "sans"
    }
  }
```

编译 WEB 应用程序

进入 awtk-web 目录，不同平台使用不同的脚本编译：

Windows 平台

./build_win32.sh examples/AwtkApplicationCWebSocket/build.json release

Linux 平台

./build_linux.sh examples/AwtkApplicationCWebSocket/build.json release

MacOS 平台

./build_mac.sh examples/AwtkApplicationCWebSocket/build.json release

请根据应用程序所在目录，修改配置文件的路径。

运行

1. 正常启动

./start_web.sh

2. 调试启动

start_web_debug.sh

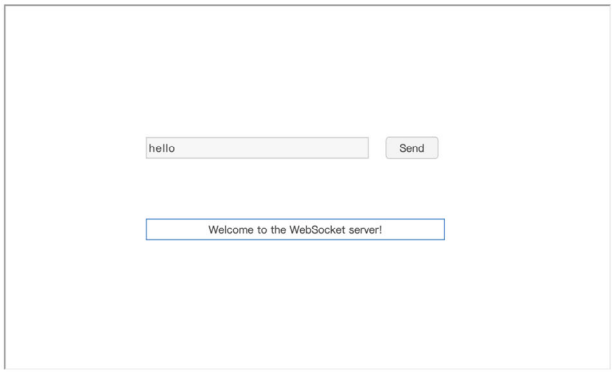
3. 启动 websocket 服务器（先安装 nodejs）

进入 awtk-web 目录下的 tools/websocket，执行：

node websocket_echo_server.js

4. 用浏览器打开 URL：

http://localhost:8080/AwtkApplicationCWebSocket



AWTK-WEB 快速入门(6) - JS WebSocket 应用程序

ZLG 致远电子 2025-02-26 11:38:50

WebSocket 可以实现双向通信，适合实时通信场景。本文介绍一下使用 Javacript 语言开发 AWTK-WEB 应用程序，并用 WebSocket 与服务器通讯。

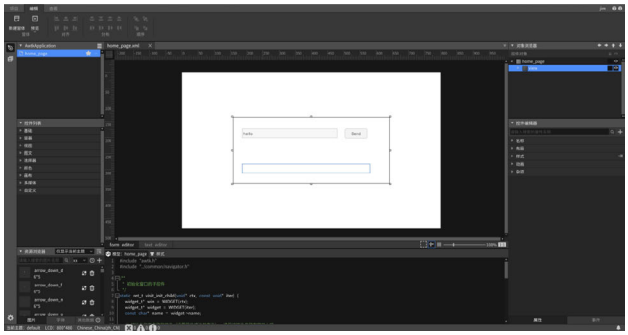
用 AWTK Designer 新建一个应用程序

先安装 AWTK Designer：

https://awtk.zlg.cn/web/index.html

1. 新建应用程序

这里假设应用程序的名称为 AwtkApplicationCHttp，后面会用到，如果使用其它名称，后面要做相应修改。



2. 编写代码

2.1 删除 src 目录下全部文件（留着也可以，只是看起来比较乱），在 src 目录创建 js 目录。

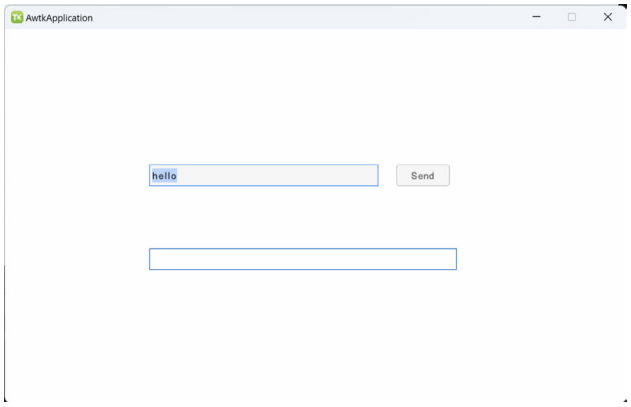
2.2 在 src/js 下创建 application.js，内容如下：

```
function home_page_open() {
  var win = TWindow.open("home_page");
  var send = win.lookup("send", true);
  var send_text = win.lookup("send_text", true);
  var recv_text = win.lookup("recv_text", true);
  const ws = new WebSocket("ws://localhost:8090");

  send.on(TEventType.CLICK, function (evt) {
    const message = send_text.getText();

    ws.send(message);
    return TRet.OK;
  });

  ws.onopen = () => {
recv_text.setText("Connected to the server");
  };

  ws.onmessage = (event) => {
    recv_text.setText(event.data);
  };

  ws.onclose = () => {
    recv_text.setText("close");
  };

  win.layout();
}
```

注意：控件的名称一定要和 home_page.xml 保持一致。

3. 在 AWTK Designer 中，执行“打包” “编译” “模拟运行”

正常情况下可以看到如下界面：



点击“关闭”按钮，退出应用程序。

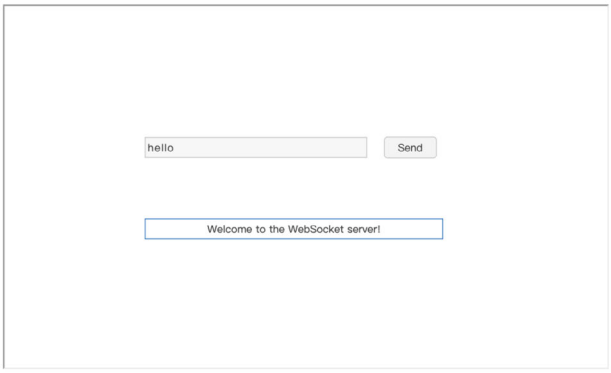
编写配置文件

具体格式请参考，特殊平台编译配置：

https://github.com/zlgopen/awtk/blob/master/docs/build_config.md

这里给出一个例子，可以在此基础上进行修改，该文件位于：examples/AwtkApplicationJSWebSocket/build.json

```
{
 "name": "AwtkApplicationJSWebSocketWebSocket",
 "version": "1.0",
 "app_type":"js",
 "author": "xianjimli@hotmail.com",
 "copyright": "Guangzhou ZHIYUAN Electronics Co.,Ltd.",
 "themes":["default"],
 "sources": [
   "src/js/*.js"
 ]
}
```



AWTK DEMOUI
ZLG开源GUI引擎AWTK: https://github.com/zlgopen/awtk

编译 WEB 应用程序

进入 awtk-web 目录，不同平台使用不同的脚本编译：

Windows 平台

./build_win32.sh examples/AwtkApplicationJSWebSocket/build.json release

Linux 平台

./build_linux.sh examples/AwtkApplicationJSWebSocket/build.json release

MacOS 平台

./build_mac.sh examples/AwtkApplicationJSWebSocket/build.json release

请根据应用程序所在目录，修改配置文件的路径。

运行

1. 正常启动

./start_web.sh

2. 调试启动

start_web_debug.sh

3. 启动 websocket 服务器（先安装 nodejs）

进入 awtk-web 目录下的 tools/websocket，执行：

node websocket_echo_server.js

4. 用浏览器打开 URL：

http://localhost:8080/AwtkApplicationJSWebSocket

【产品应用】1分钟，实现传感器通过串口服务器接入ZWS云

ZLG 致远电子 2025-01-10 11:35:59

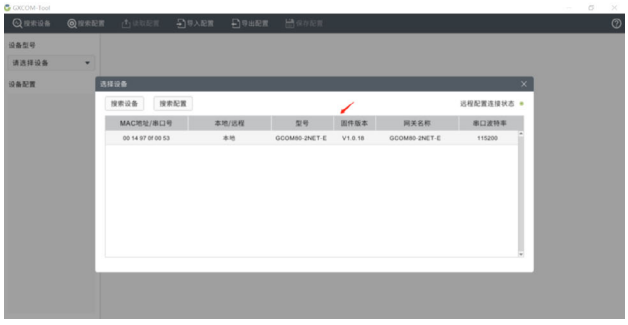
本文介绍如何在 1 分钟内，将传感器通过串口服务器 GCOM80 接入 ZWS 云平台，实现数据上云。

硬件准备

1. 光照传感器；
2. 串口服务器 GCOM80；
3. 网线和串口线。

步骤一：配置串口和网络参数

1. 串口服务器 GCOM80 上电，将 GCOM80 与光照传感器的串口相连，并连接网线。打开串口服务器配置软件“GXCOM-Tool”，点击“搜索设备”，选择串口服务器设备。



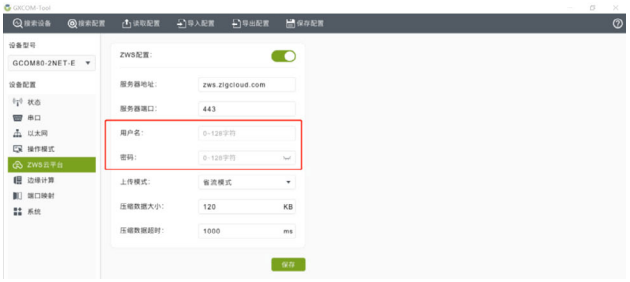
2. 进入“设备配置 > 串口”，配置串口服务器串口参数（如波特率、数据位、停止位）与光照传感器参数一致。

3. 配置网络，将网络的 IP 地址更新到配置工具”配置工具 > 以太网”的 IP 地址和网关地址。

步骤二：配置串口服务器上云参数

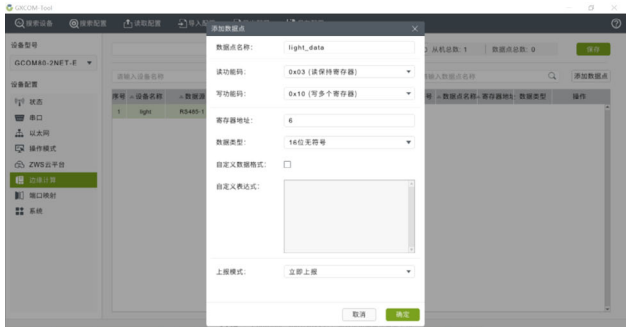
1. 进入“设备配置 > 操作模式”，将 RS485 操作模式配置成“MODBUS RTU 采集 ZWS 上报”。

2. 进入“设备配置 >ZWS 云平台”，配置 ZWS 物联网云平台用户名和密码。



3. 进入“设备配置 > 边缘计算”，配置从机和从机对应数据点。点击“添加从机”，然后配置“设备名”、“从机地址”、“轮询间隔”等参数。

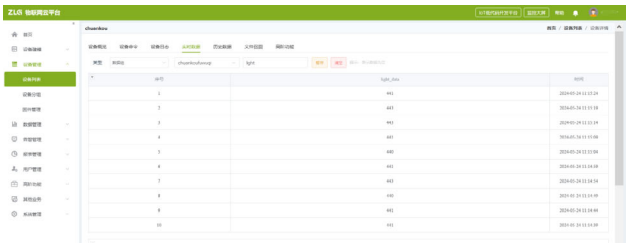
4. 添加从机对应数据点，点击“添加数据点”按钮，配置“数据点名称”、“寄存器地址”、“数据类型”等参数。



步骤三：云端查看光照传感器数据

1. 登录 ZWS 物联网云平台，点击“设备列表”，可以看到串口服务器已经连接上云平台。

2. 点击设备详情，查看实时数据，可以看到光照传感器数据。



3. 点击“数据大盘 > 数据看板”，可添加查看数据曲线图表。

【产品应用】EM储能网关&ZWS智慧储能云应用(6) — 账号体系

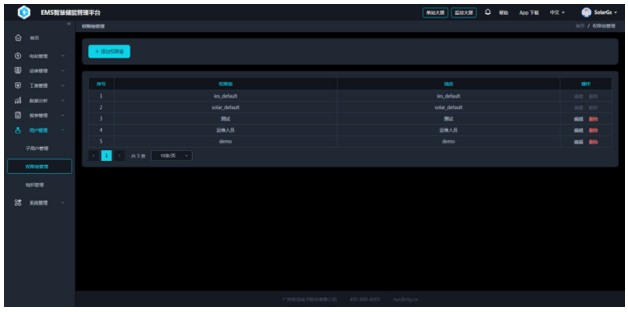
ZLG 致远电子 2025-01-17 11:35:14

ZWS 智慧储能云的账号体系，根据各营销体系或企业体系，各司其职，可为不同的账户赋予不同的角色权限，实现不同企业，不同场景 ，精细化完成账号、电站与功能管理。

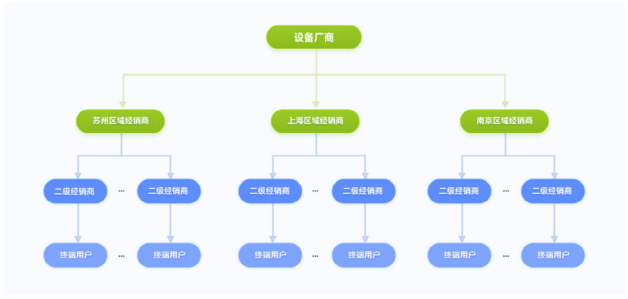
储能系统从研发、销售、安装、应用中，有不同的角色参与，其中有厂家、分销商、运维人员、能源投资商、终端业主等。基于他们的角色不同，对储能系统的分配与应用（运营、运维、调度等）各有差异。储能云作为储能系统的监控与运维系统，在云平台上，不同角色也应各有体现。每个人员账号，拥有着不同的功能权限，能对储能系统的信息与控制权限，各有差异。

树结构的账号体系

1. ZWS 储能云中，拥有着角色的权限组，其中可以管控着不同角色的功能权限与页面权限。



2. ZWS 储能云的账号体系为树结构的账号体系，其中以树结构的形式展开，上级用户的可以管控下级用户的账号管理、功能、电站权限等。



3. 在相应子用户登录到储能云系统时，只能查看到上级分配的电站与功能、页面。并在数据统计时，只会统计权限下的电站能耗、电量、收益等数据。

EM系列储能边缘智能网关

EM 系列储能边缘智能网关是 ZLG 致远电子专为新能源储能系统设计的一款高性能、多接口通讯管理设备，可在储能系统应用中作为边缘 EMS(能源管理系统) 总控、通讯管理机、规约转换器或 BAU(电池管理总控) 使用。该系列产品集成丰富的外设接口，支持各类 BMS、PCS、空调、电表、屏显等设备的通讯传输，且软件上支持 RT-Linux、Ubuntu 等操作系统，支持 IEC-61850/IEC-104/EtherCAT 等专用协议，可广泛满足各类储能系统的本地能源管理应用需求。

【产品应用】EM储能网关&ZWS智慧储能云应用(7) — 数据修正

ZLG 致远电子 2025-02-08 11:36:58

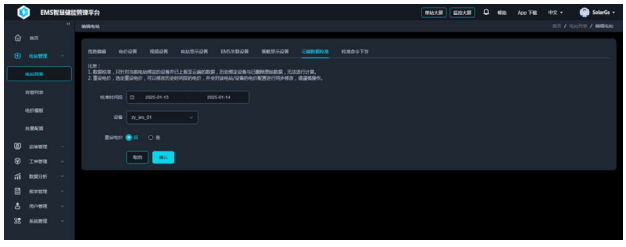
ZWS 智慧储能云，会统计电量、能耗等数据，这些统计数据与客户收益情况息息相关。在日常运营中，常有各种异常因素，导致统计有误。ZWS 储能云，针对异常情况是如何数据修正的？

ZWS 储能云中，有着对储能系统的能耗、电量、收益等统计。其中在线下应用中，这些统计数据，与用户的运营调整、真实收益，有着密切的关系，其中统计数据的真实性与时效性，极为重要。

储能系统通常是以 4G 网络连接到云平台，将原始数据上报至云平台。设备可能会因为网络问题、堵塞问题，bug 问题，而导致数据延迟或有误。也有可能是终端业主，配错了电价或忘配电价，多种因素导致统计有误等。ZWS 储能云，可对此类情况进行相应的处理。

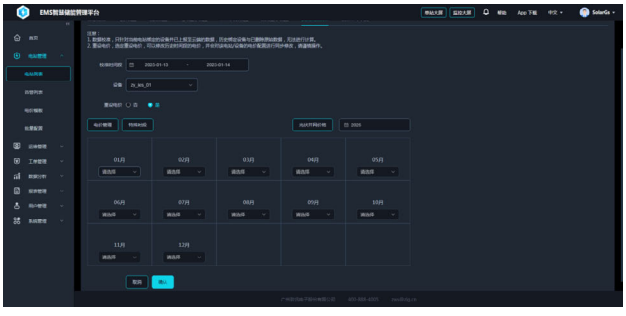
1. 数据延迟

云端的统计机制是定时统计，EMS 网关对于数据堵塞或因信号问题导致离线，从而上报的源数据延迟比较大，需要设备端补充后，再次重新校准统计历史数据。云端可针对历史统计数据，再次重跑数据。



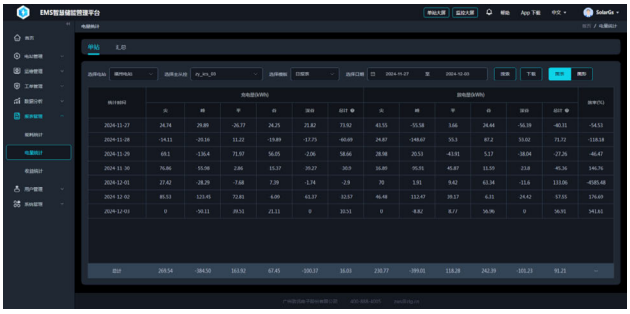
2. 配错电价

对于特殊节假日，运营用户经常忘记配置特殊电价。云端中，可以针对配错电价的情况，而再次重新配置。针对重新配置后的电价，再次计算尖、峰、平、谷、深谷的电量和收益等。



3. 源数据有误

设备在有异常 BUG 的情况下，会给云平台上报错误的源数据，导致云端的统计数据有误。此种情况下，需要电站的管理人员，将云端异常的源数据进行删除。再根据情况，是否需要设备端补充完善数据，再进行数据重跑重新修正统计数据。



EM系列储能边缘智能网关

EM 系列储能边缘智能网关是 ZLG 致远电子专为新能源储能系统设计的一款高性能、多接口通讯管理设备，可在储能系统应用中作为边缘 EMS(能源管理系统) 总控、通讯管理机、规约转换器或 BAU(电池管理总控) 使用。该系列产品集成丰富的外设接口，支持各类 BMS、PCS、空调、电表、屏显等设备的通讯传输，且软件上支持 RT-Linux、Ubuntu 等操作系统，支持 IEC-61850/IEC-104/EtherCAT 等专用协议，可广泛满足各类储能系统的本地能源管理应用需求。

云平台 ▼

EM储能网关&ZWS智慧储能云应用(8) — 电站差异化支持

ZLG 致远电子 2025-03-14 11:36:01

面对不同项目、种类繁多的储能产品，如何在储能云平台上进行电站差异化支持尤为关键，ZWS 智慧储能云从多方面支持储能电站差异化。

简介

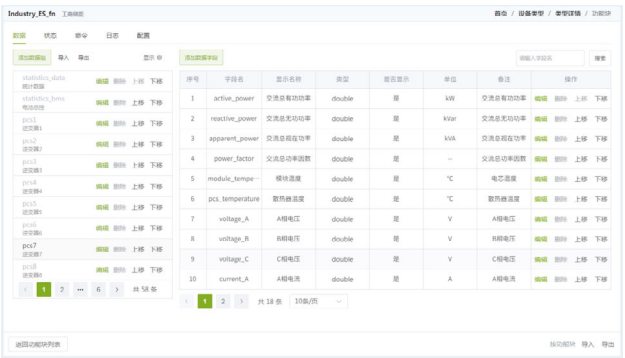
随着行业发展，市场“内卷”之下，各大储能企业推陈出新的速度加快。面对不同项目、种类繁多的储能产品，如何在储能云平台上进行电站差异化支持尤为关键。

ZLG 致远电子储能网关 EM-1000 搭配 ZWS 智慧储能云，可实现储能系统远程集控，为储能企业提供云端数字化运维的工商储能、户用储能的管控方案，助力推动储能系统安全运维，拓展收益。

不同储能项目的特性不同，储能电站也会有差异，比如每个电站的储能单元 PCS 数量、BMS 数量、光伏、动环监控等展示元素可能要求不一样。面对储能电站的多样性，ZWS 智慧储能云支持对电站差异化显示和处理，主要通过以下几个方面。

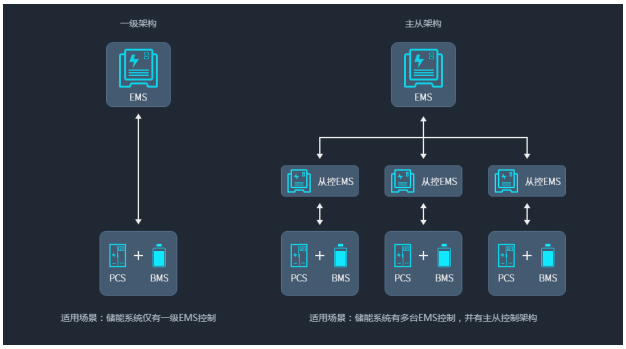
1. 物模型差异拓展

云平台拥有标准的储能物模型，包含了（PCS、BMS、BCU、电表、动环）等常见设备。用户只需按云端的协议，按点位进行上报，即可在储能平台中查看到相应数据，快速上云。如果标准物模型无法满足项目应用需求，用户可以自行拓展设备和数据点。



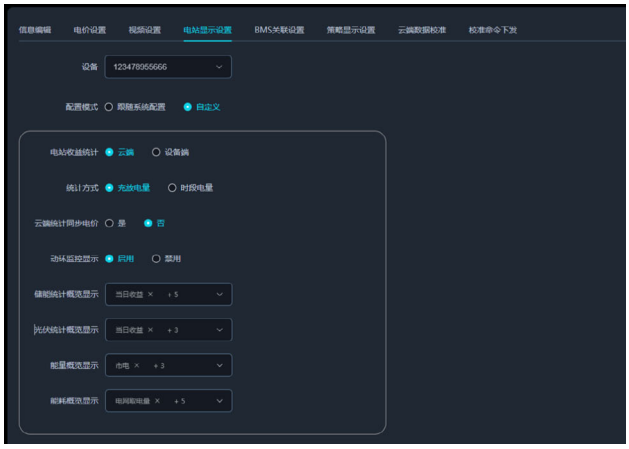
2. 电站部署架构适配

储能云平台，电站云部署架构，支持“一级架构”与“主从架构”，实现多种储能系统按功能需求上云。

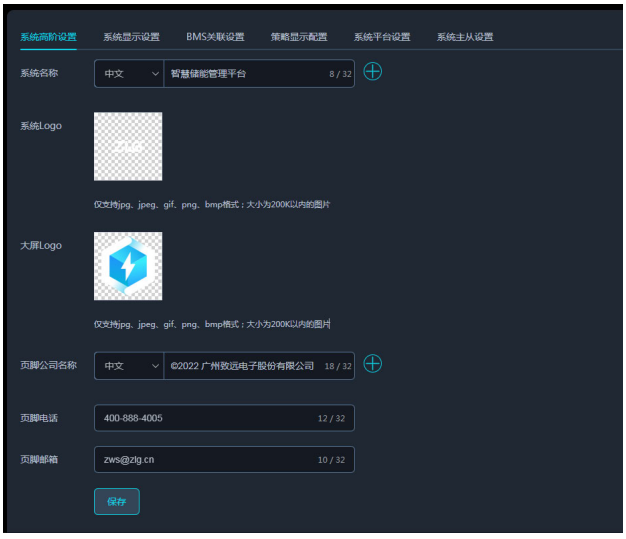


3. 电站级别差异化配置

云平台可根据设备读取的数据与电站级别的配置项，对该电站展示的元素，如 PCS、BMS 数量、此电站是否拥有光伏等，进行差异化展示。使每个电站查看到的元素，与实际储能系统一致。



此外，还支持储能平台名称 logo 定制化，云平台的 logo 与系统名称可由用户自定义配置，其中域名也可进行单独处理，让企业拥有专属平台属性。



EM系列储能边缘智能网关

EM 系列储能边缘智能网关是 ZLG 致远电子专为新能源储能系统设计的一款高性能、多接口通讯管理设备，可在储能系统应用中作为边缘 EMS(能源管理系统) 总控、通讯管理机、规约转换器或 BAU(电池管理总控) 使用。该系列产品集成丰富的外设接口，支持各类 BMS、PCS、空调、电表、屏显等设备的通讯传输，且软件上支持 RT-Linux、Ubuntu 等操作系统，支持 IEC-61850/IEC-104/EtherCAT 等专用协议，可广泛满足各类储能系统的本地能源管理应用需求。

云平台 ▼

EM储能网关&ZWS智慧储能云应用(9) — 远程OTA升级

ZLG 致远电子 2025-03-20 11:33:29

ZWS 智慧储能云平台支持远程 OTA 固件升级，可以针对具体的储能设备进行升级，升级储能网关、EMS 主控软件、PCS、BMS 等。

简介

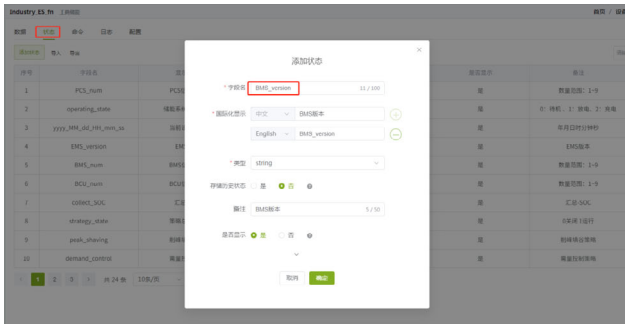
储能系统通常高度集成化，一体化设计，将 EMS、BMS（电池管理系统）、PCS（电力转换系统）、空调以及消防系统等集成在一起，大大提高了系统的便捷性和灵活性。

随着储能产品的迭代更新，储能系统也需要长期且持续的运维服务，便捷的运维解决方案在储能系统中非常重要，比如，远程控制、远程固件升级的功能可以大大减少现场售后服务的时间成本和服务本身的费用。

致远电子的储能网关 EM-1000 搭配 ZWS 智慧储能云平台，可实现储能系统远程集控，远程 OTA 固件升级，可以针对具体的设备进行升级，升级储能网关、EMS 主控软件、PCS、BMS 等，助力储能系统安全运维。

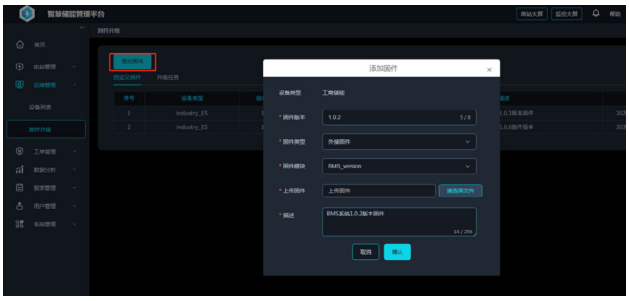
1. 配置设备版本字段

云端内置了储能网关本身、EMS 的版本字段，升级这两个设备则不需要创建新字段。若要升级其他设备，像 BMS、PCS，则需要自定义拓展对应的版本字段。如下图，创建一个 BMS_version 的版本字段，用于云端存储和显示对应的版本状态。



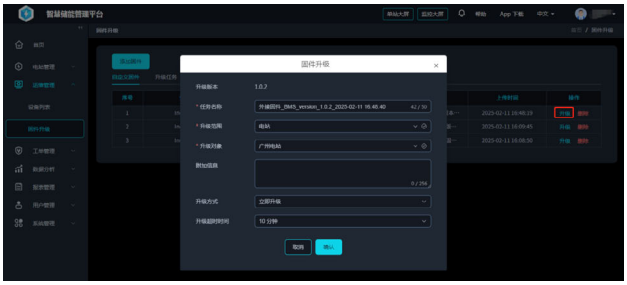
2. 上传固件

在“运维管理 - 固件升级”页面，点击【添加固件】，输入版本号，选择固件类型：自身固件（默认是储能网关的固件）、外接固件（即外接设备的固件）。若升级外接设备的固件，比如，BMS 固件，则需选择对应的固件模块字段（即上一步创建的版本字段）。上传固件，点击“确认”。

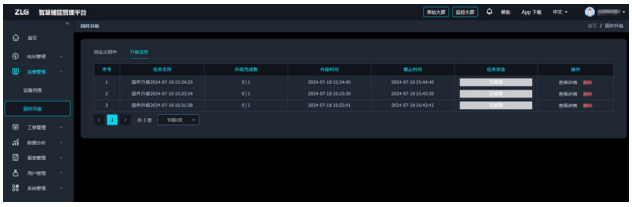


3. 升级固件

固件列表里，在对应的版本固件右侧，点击【升级】，选择升级范围、升级对象、升级方式，点击“确认”后，云端自动创建升级任务，完成升级。



升级进度和升级完成情况，可以在“升级任务”中查看。



EM系列储能边缘智能网关

EM 系列储能边缘智能网关是 ZLG 致远电子专为新能源储能系统设计的一款高性能、多接口通讯管理设备，可在储能系统应用中作为边缘 EMS(能源管理系统) 总控、通讯管理机、规约转换器或 BAU(电池管理总控) 使用。该系列产品集成丰富的外设接口，支持各类 BMS、PCS、空调、电表、屏显等设备的通讯传输，且软件上支持 RT-Linux、Ubuntu 等操作系统，支持 IEC-61850/IEC-104/EtherCAT 等专用协议，可广泛满足各类储能系统的本地能源管理应用需求。

EM储能网关&ZWS智慧储能云应用(10) — 智能化电站管理

ZLG 致远电子 2025-03-28 11:34:00

ZWS 智慧储能云平台针对储能电站监控管理等问题，通过整合设备监控、收益统计与策略下发等核心功能，为用户提供一站式的智能化电站管理解决方案，助力实现电站高效运维与收益优化。

简介

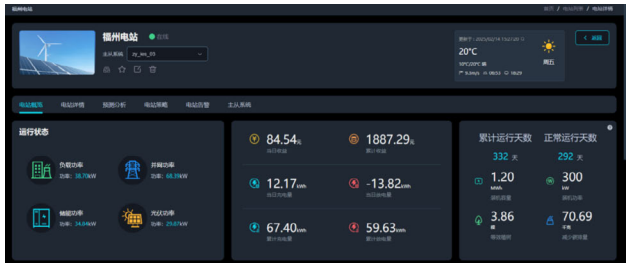
近年来，储能技术不断进步，使得储能电站的建设和运营具备了更强的经济性和可行性。在此基础上，ZWS 智慧储能云平台通过智能化电站管理，能够充分发挥储能技术的优势，进一步提升储能电站的性能和效益。

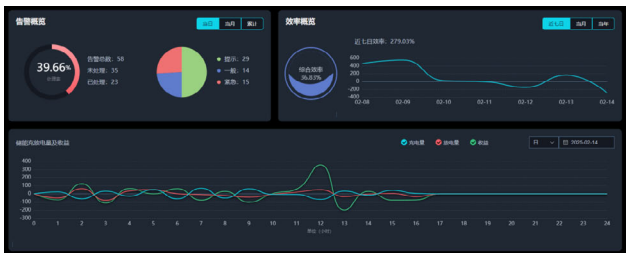
1. 多电站集中监控

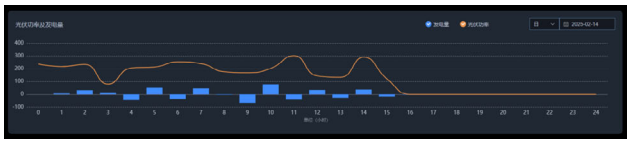
在电站列表下对多电站集中监控，支持查看由用户创建或上级分配的电站列表，并实时监控电站的在线状态、运行数据（如当日充电量、当日放电量）及系统架构。

2. 电站架构支持

在对具体电站管理的支撑上，云上支持用户基于一级架构和主从架构对电站进行数据监控、统计，同时可以随时查看当前电站的天气情况、运行情况以及发电收益情况等。

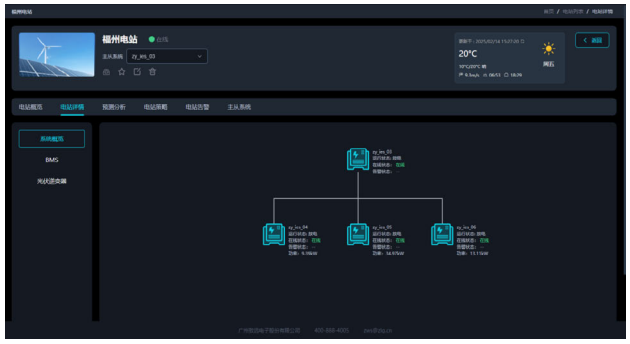






3. 拓扑图和实时数据

在电站详情下还可以监控主从架构的设备数据以及状态，看 EMS 下的 PCS、BMS 以及电池簇等拓扑图和实时数据。

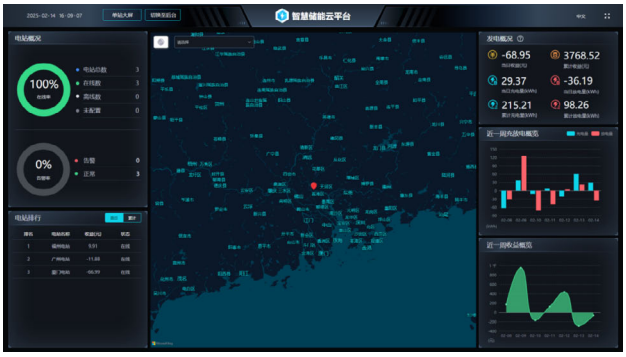


4. 日志记录

记录设备绑定 / 解绑、策略下发等操作日志，标注执行账号与时间戳，满足安全审计需求。

5. 数据监控大屏

云上支持监控大屏和单站大屏两个监控维度的数据大屏，提供给用户多站以及单站维度的全面监控，提供电站健康度全景视图。



EM系列储能边缘智能网关

EM 系列储能边缘智能网关是 ZLG 致远电子专为新能源储能系统设计的一款高性能、多接口通讯管理设备，可在储能系统应用中作为边缘 EMS(能源管理系统) 总控、通讯管理机、规约转换器或 BAU(电池管理总控) 使用。该系列产品集成丰富的外设接口，支持各类 BMS、PCS、空调、电表、屏显等设备的通讯传输，且软件上支持 RT-Linux、Ubuntu 等操作系统，支持 IEC-61850/IEC-104/EtherCAT 等专用协议，可广泛满足各类储能系统的本地能源管理应用需求。

如需了解更多产品详情，可填写申请表单，
我们会有专人与您联系。

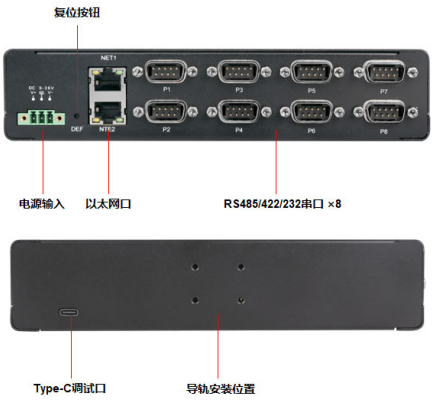
点击申请

【新品发布】致远电子三合一8路串口服务器，附带新年第一份心意！

ZLG 致远电子 2025-01-13 11:33:13

ZLG 致远电子全新推出国产化透传型三合一 8 路串口服务器，让用户轻松实现串口信号与以太网的无缝数据交互。性能强悍，等你来评，还有机会获得新年第一份心意！

GCOM88-2NET-P硬件速览

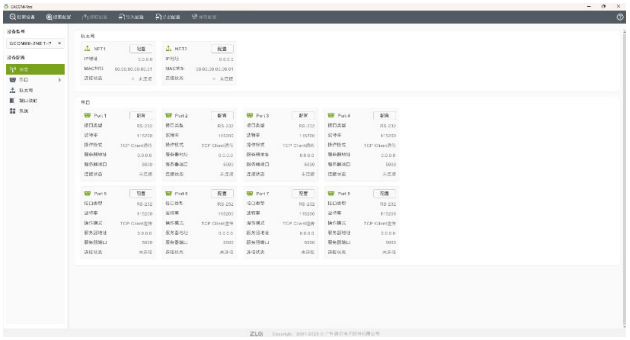


- 国产高性能 816MHz 处理器；
- 2 路 10/100M 自适应以太网端口；
- 8 路独立 RS485/RS232/RS422 三合一 DB9 端口；
- 最高支持 2Mbps 通信速率（@RS485）；
- 额定供电电压范围 9V~36V 直流，1500V 耐压隔离；
- 工作温度：-40℃～ 85℃，湿度：5% – 95% RH，无凝露；
- 坚固的金属外壳，专为工业环境设计；
- 长 x 宽 x 高：199mm x 124.4mm x 43.8mm，支持导轨及底部安装。

GCOM88-2NET-P功能速览

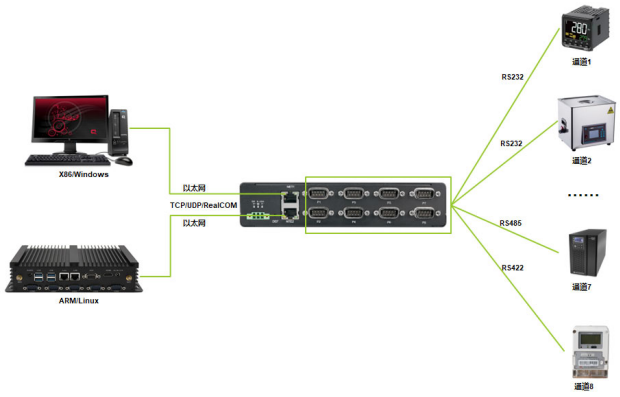
产品开发平台：

1. 软件平台：ZLG 自研开发平台 EsDA，自主可控；
2. 操作系统：ZLG 自研 AWorksLP 实时操作系统；
3. 配置工具：ZLG 自研 GUI 引擎 AWTK 设计的上位机 GXCOM-TOOL。



- 工作模式：TCP/UDP/RealCOM 3 种通讯模式；
- Microsoft 认证虚拟串口内核驱动；
- 网口功能 1：交换机级联功能，可柔性扩展串口通道数量；
- 网口功能 2：双网口链路冗余功能；
- 支持 NTP 网络对时功能；
- 支持批量导入导出，省事方便；
- 设备加密保护，安全无忧。

GCOM88-2NET-P的典型应用



更多详细内容请查看：
https://www.zlg.cn/wireless/wireless/product/id/343.html

GCOM88系列
高性能串口服务器

点击购买

互联互通▼

智能粉尘监测解决方案 | 守护工业安全，杜绝爆炸隐患

ZLG 致远电子 2025-02-27 11:33:51

在厂房轰鸣的生产线上，一粒微小粉尘的聚集可能成为一场灾难的导火索。如何实现粉尘浓度的精准监控与快速响应？我们为您打造了一套 " 感知 - 预警 - 处置 " 全闭环的智能安全方案！

行业痛点：粉尘管理的生死线

在金属加工、化工、木制品等工业场景中，粉尘浓度超标可能导致：

- 燃爆风险：悬浮粉尘遇明火可引发爆炸（如 2014 年昆山工厂爆炸事故）；
- 职业伤害：长期吸入导致尘肺病等呼吸道疾病；
- 环保处罚：粉尘外泄面临环保法规严惩；

传统人工巡检存在响应滞后、数据缺失等致命缺陷，数字化转型已成必然。

智能监测方案架构

1. "3 分钟部署，24 小时守护 "

通过 GCOM80-2NET 智能网关 + 工业级粉尘传感器 + 智能风机系统构建三重防护网，粉尘传感器阶段性采集粉尘浓度数据 , 集中传输至后台。如图 1 所示 :

图1 示意图

在图 1 中，当左侧粉尘探测器检测到粉尘浓度超过设定阈值时，会通过 GCOM80-2NET 将数据上报至后台。后台系统会立即启动应急响应机制，一方面及时发送短信或电话通知相关负责人，另一方面同步下发排风扇指令，迅速降低粉尘浓度，从而有效避免粉尘爆炸事故的发生。

为什么选择我们?

1. 快速配置与高效数据获取

GCOM80-2NET 支持快速配置，能够迅速投入使用，无需复杂的安装和调试过程。此外，该设备采用 Modbus 协议，用户只需简单操作即可获取所需数据，大大提高了工作效率。

2. 灵活的设备管理

GCOM80-2NET 支持单通道 MOTT 主题配置，设备种类丰富，一个通道可管理一类设备。这种灵活的管理方式不仅方便现场布置，还能简化维护流程，降低企业的运营成本。

3. 健全的物联生态

我们的方案基于健全的物联网生态，硬件设备与云平台可快速接入。通过 GCOM80-2NET 和 ZWS 云平台，用户能够快速搭建个性化需求的控制界面及业务需求。这种高度灵活的系统设计，能够满足不同客户的多样化需求，同时支持未来的可持续扩展。

总结

在工业生产中，粉尘监测是保障安全与健康的关键环节。我们的智能化粉尘监测与管理方案，凭借快速配置、高效数据传输、灵活设备管理等优势，为企业提供了一种可靠的解决方案。当每一粒粉尘都被精准监控，安全生产才能真正落到实处。

如需了解更多详情，欢迎随时联系我们!

工业4.0下，应急电源的智能化升级之路

ZLG 致远电子 2025-03-11 11:31:16

在工业 4.0 和智慧工厂快速发展的背景下，电力稳定性成为保障生产连续性的关键。本文结合客户案例展示 GCOM80-2NET-E 方案如何通过数据防冲突机制和高效传输解决传统问题，助力电力保障智能化升级。

行业介绍

随着工业 4.0 与智慧工厂的推进，生产场景对电力稳定性的需求持续升级。工业自动化渗透率突破 65% 的背景下，应急电源设备（如 UPS、EPS）成为保障生产连续性、规避突发断电风险的核心基础设施。我国双碳政策与新基建战略的落地，进一步推动应急电源向高可靠性、智能化方向迭代，行业规模预计 2025 年突破 1200 亿元。

1. 应急电源

应急电源通常分为两类：UPS 与 EPS。两者的核心工作点与应用场景虽不一致，但核心功能均是在市电断电时提供应急电源。具体差异如表 1 所示：

名称	核心工作点	典型应用场景
UPS	市电停电时负载用电不会中断	数据机房
EPS	市电停电时负载用电会短时间断电后才能切换到电池供电	应用照明

表1 UPS 与 EPS 差异对照表

由于UPS 电源在断电再供电切换速度快的特点，其单个设备的响应时间、电池电量状态监测等数据需要集中化管理，用于能耗分析、设备管控。

2. UPS电源

- 运作方式：动态式、静态式；
- 工作原理：后备式（离线式）、在线式、在线互动式；
- 其他维度：备用时间、结构形态、电池类型等。

客户方案

方案中，电池到交流电源的转换器（逆变器）始终连接到 UPS 的输出端。当输入交流电源正常时，逆变器反向操作可为电池充电；一旦输入电源出现故障，转换开关打开，通过电池向 UPS 输出端供电。框图如图 1 所示：

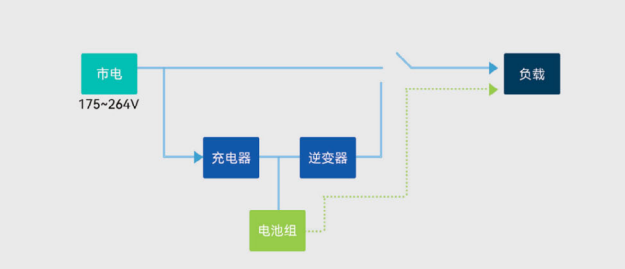


图1 UPS工作原理示意图

在图 1 的工作原理示意图中，UPS 状态一直处于被监测状态。因此，当 UPS 批量化部署时，就需要将 UPS 数据汇总。示意图如图 2 所示：

图2 UPS部署示意图

在图 2 中，GCOM80-2NET-E 南向与 UPS 通过 Modbus 连接，北向通过以太网与交换机连接，数据传输至不同的 PC 上。GCOM80-2NET-E 整体工作模式为 Modbus RTU 转 Modbus TCP，电脑 PC 作为客户端。

然而，在此场景下，用户的业务逻辑可能会出现数据收发冲突问题。例如，当 PC 端同时向网关设备发送命令时，传统模式下，网关设备在接收到 PC 指令后，只会接收当前最新的指令，无法对其他指令进行缓存，从而导致数据丢失或查询失败。而 GCOM80-2NET-E 则可以进行 Modbus 数据缓存，对于普通的单寄存器查询，能够保证每条查询指令不丢失。

方案核心优势

1. 国产化适配

- 主器件全国产化（GCOM80-2NET-P 型号），硬件接口满足工业级需求；
- 可靠性指标：MTBF 无故障时间达 492,295.6 小时。

2. 数据防冲突机制

- 传统模式缺陷：多指令并发时仅响应最新指令，易导致数据丢失；
- GCOM80-2NET-E 优化：支持 Modbus 数据缓存，确保单寄存器查询指令不丢失；提供多模式数据转换及冲突规避逻辑。

3. 实测验证与性能提升

- 多客户端场景：在 PC 端并发查询压力下，设备通过缓存队列和优先级调度实现零丢包；
- 转换效率：Modbus RTU 至 TCP 转换延迟低于 10ms，满足实时监控需求。

工厂老化测试解决方案：GCOM80-2NET-E如何赋能智能制造

ZLG 致远电子 2025-03-18 11:36:54

老化测试是产品质量把控的关键，但传统方式效率低、成本高。GCOM80-2NET-E 通过智能协议解析、多设备兼容和数据轻量化，大幅提升测试效率，缩短周期，助力企业实现智能化升级与降本增效。

行业洞察：电源测试的关键挑战

在电源制造领域，产品性能与质量直接决定其市场竞争力。作为核心检测环节，电源老化测试通过模拟长期运行场景，对电池、适配器等产品的稳定性、兼容性及可靠性进行严苛验证。尤其在全球化竞争下，满足国际标准、降低故障率、提升用户信任度已成为企业立足市场的关键。

老化测试

老化测试是电源生产中极为重要的环节，其主要目的是模拟电源在长期使用过程中的性能变化，提前发现潜在的故障和问题。

1. 故障预警与风险控制

模拟电源全生命周期运行状态，提前暴露潜在缺陷，降低实际使用中的故障风险。

2. 设计优化与效能提升

基于老化数据逆向改进产品设计，延长使用寿命并提升能效表现。

3. 标准合规与市场准入

确保产品通过国内外严苛环境测试标准，为全球化布局奠定基础。

4. 品牌信任与用户黏性

以可靠品质建立用户口碑，强化品牌技术背书。

客户场景与方案架构

1. 设备需求

实时监控：动态追踪不良率，按需生成数据报表。

多单元协同：集成马达、导杆、工控机等模块，支持大规模并行测试。

灵活部署：兼容不同被测设备的协议与地址配置，降低集成复杂度。

图1 设备示意图

2. GCOM80-2NET-E的核心角色

作为数据交互中枢，该设备实现：

协议转换：通过 Modbus 连接待测终端，以 TCP/MQTT 协议对接中控工控机。

智能寻址：支持从机 ID 小板或设备自主地址配置，适配多品牌终端。

高效解析：将原始报文转换为标准化 JSON 数据流，简化中控系统处理负担。

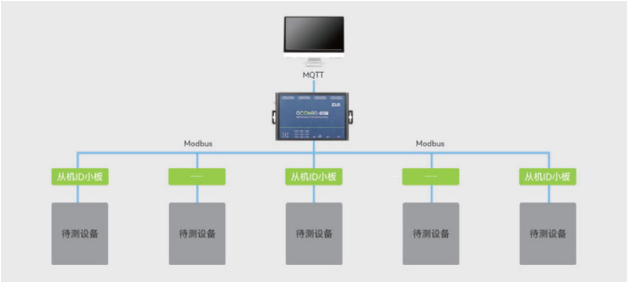


图2 应用框图

方案优势：直击行业痛点

用户痛点	GCOM80-2NET-E 解决方案	创造价值
多协议解析开发周期长	内置 Modbus-JSON 双向转换引擎	部署效率提升 50%+
定制化需求导致成本高	统一接口标准，支持模式灵活切换	研发成本降低 30%
设备兼容性差	自适应地址分配与协议扩展	支持所有 modbus 主流电源设备
数据追溯难	全链路实时监控与报表自动化	不良品溯源效率提升 70%

1. 核心技术创新

- 双模通道技术：同一端口支持协议解析与透明传输模式秒级切换，满足复杂测试场景。
- 边缘计算能力：本地化数据清洗与压缩，降低工控机负载 20%-40%。
- 开放式架构：可提供界面 API 接口与 SDK 工具包，支持界面二次开发与系统深度集成。

赋能智慧工厂的未来价值

- GCOM80-2NET-E 通过标准化、智能化、柔性化的测试方案重构电源老化检测流程：
- 缩短产品上市周期：从“定制开发”转向“即插即用”，设备调试时间从周级降至小时级。
- 提升产线良品率：通过数据建模预判故障模式，实现从“事后检测”到“前瞻性品控”转型。
- 驱动数字化转型：构建测试大数据平台，为工艺优化提供持续的数据资产支撑。

在智能制造升级浪潮中，该方案正成为电源行业提质增效的核心基础设施，助力企业以更低成本、更高可靠性赢得全球市场先机。

储能行业新机遇：PCS与UPS产线测试解决方案

ZLG 致远电子 2025-03-27 11:43:36

聚焦于储能行业的快速发展背景，探讨 PCS 储能变流器和 UPS 不间断电源作为核心设备的市场需求，展示基于 GCOM80-2NET-P 的测试系统方案及其优势，助力企业提升竞争力。

行业背景

随着全球能源转型和“双碳”目标的推进，储能行业迎来快速发展。2025 年被视为储能行业发展的关键转折点，政策支持、技术创新与市场需求形成共振，推动产业从试点示范迈向规模化普及。

中国政府明确到 2025 年新型储能将步入规模化发展阶段，国际能源署预测 2030 年全球储能市场规模将突破 5000 亿美元，中国有望占据 30% 以上份额。同时，欧盟和美国等也纷纷出台政策支持储能发展，极大地激发了市场主体投资储能项目的热情。在此背景下，PCS 储能变流器和 UPS 不间断电源作为储能系统的核心设备，市场需求快速增长。

PCS和UPS

1.PCS(Power Conversion System) 储能变流器，又称双向储能逆变器，是储能系统中实现电能双向转换的关键设备。它能够将直流电转换为交流电（DC/AC），并将其馈入电网，同时也能将交流电转换为直流电（AC/DC），为储能电池充电。

2.UPS(Uninterruptible Power Supply) 不间断电源，是一种含有储能装置的电源设备，能够在市电中断时，瞬间切换到备用电池供电，确保负载设备的不间断运行。这两种设备在储能系统中扮演着至关重要的角色，广泛应用于数据中心、通信基站、工业自动化、可再生能源发电等领域。

客户方案

1. 具体方案

针对企业对 PCS 和 UPS 产品稳定性、可靠性和兼容性的严格要求，我们提供了基于 GCOM80-2NET-P 的测试系统解决方案。GCOM80-2NET-P 被广泛应用于 PCS、UPS 的老化测试场景。在老化测试阶段，测试系统通过 RS485 总线将状态数据收集到后台。客户 PCS、UPS 产线自测方案的简化示意图分别如图 1、图 2 所示。

图1 客户 PCS、UPS 自测方案示意图

图2 测试平台

而用户的完整系统拓扑结构示意图如图 3 所示，这一拓扑结构图也是目前主流的测试方案。

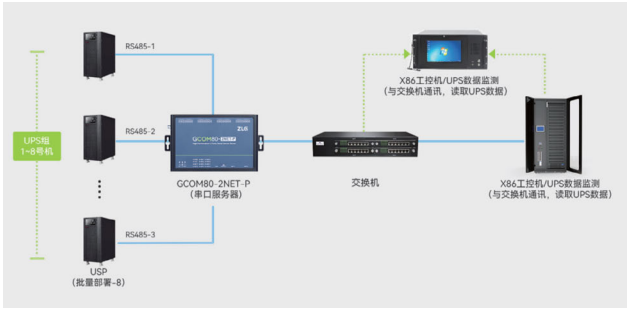


图3 完整拓扑结构示意图

2. 方案优势

2.1 硬件稳定性

高温环境适应性：设备能够在 75° C 的高温环境下长期稳定运行，确保系统不死机、电源不烧毁、数据不丢包。

高波特率通信能力：支持 8 路 2Mbps 全速运行，确保数据传输的高效性和可靠性。

2.2 虚拟串口工具

- 稳定性与性能：在与国内外竞争对手的产品对比中，我们的虚拟串口工具在稳定性、创建端口的速度以及长时间传输数据不丢包等方面表现出色。
- 兼容性：除了兼容 Windows 操作系统外，还支持 .NET 框架指令，能够无缝对接客户基于 .NET 开发的上位机软件，极大地提升了系统的兼容性和易用性。
- 性价比：价格优势显著，功能与稳定性出色，可作为同类型产品理想的国产化替代方案。

- 2.3 系统拓扑结构
- 通用性与扩展性：高达 8 路 RS485 通道，且具有交换机功能、良好的通用性和扩展性，能够满足客户当前及未来的发展需求。

总结

在储能行业快速发展的背景下，企业对 PCS 和 UPS 产品的质量要求越来越高，尤其是在硬件稳定性和通信能力方面。我们的 GCOM80-2NET-P 测试系统方案凭借其卓越的硬件稳定性、高效的通信能力和强大的兼容性，能够全面满足客户的需求。选择我们的方案，不仅能够确保客户产品的高质量出厂，还能为客户的产能扩张提供有力支持，助力客户在全球储能市场中占据更大的份额。

【新品发布】500mA带延时开关功能的低压差线性稳压器

ZLG 致远电子 2025-03-03 11:37:44

带延时控制功能的高性能低压差线性稳压器 ZL6305，支持 2.3V 至 7V 输入，具备 500mA 带载能力 , 特别适合需要长按键开关的产品。

功能描述

ZL6305 是广州致远微电子有限公司设计的一款带延时开关功能的低压差线性稳压器。ZL6305 具有极低的关断电流和静态功耗，特别适用于 2.3V 至 7V 的供电设备。ZL6305 的初始输出电压精度为 ±1.5%。当输出电流 500mA 时典型压差为 240mV。

ZL6305 内置快速放电电路，当输入电压掉电到设定值时，内部快速放电电路开启使输出快速放电。ZL6305 同时具有欠压保护、过流保护、短路保护和过温保护等保护功能。ZL6305 采用 SOT23-5 封装，外围仅需要极少元件，减少了所需电路板的空间和成本。

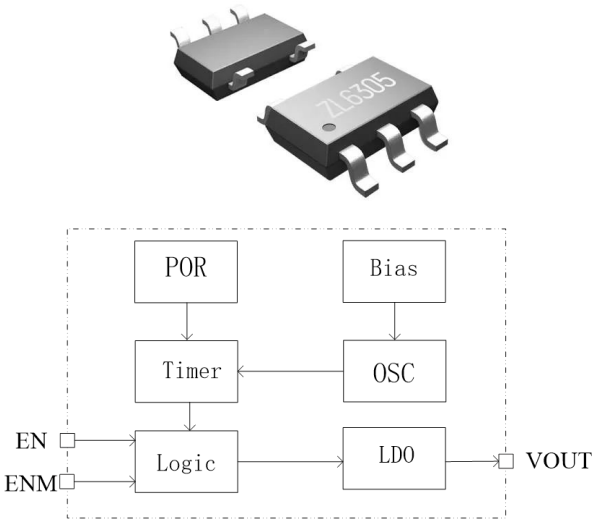


图1 ZL6305内部电路示意图

1. 延时开关功能功能与常规开关功能功能配置

- 常规开关功能与延时开关功能通过功能选择端 ENM 的上拉或者下拉来选择。
- ENM 下拉或悬空时，选定延时开关功能：
- 电源上电后 LDO 无输出，EN 下拉持续超过 3s，LDO 正常输出；当 LDO 正常输出时，EN 再次下拉持续超过 3s，LDO 关闭输出；每次 EN 下拉持续时间小于 3s，LDO 输出状态不作响应，避免了误动作对输出的影响。
- ENM 上拉时，选定常规开关功能：
- 电源上电后 LDO 无输出，EN 上拉 LDO 正常输出，EN 下拉 LDO 关闭输出。

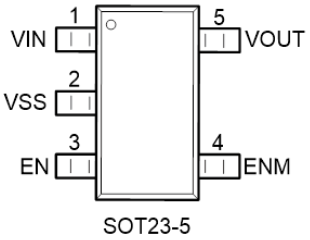


图2 管脚信息

2. ZL6305各管脚的详细功能描述

管脚编号	名称	描述
1	VIN	电源输入端，VIN 引脚与芯片地之间需要靠近芯片接一个不小于 10μF 陶瓷电容（建议 10μF~100μF）。工作输入电压范围为 2.3V 至 7V。
2	VSS	芯片接地端，该引脚必须连接到 PCB 的地。
3	EN	开关功能端，芯片开关功能控制，通过 ENM 选择常规开关功能或延时开关功能。
4	ENM	功能选择端，上拉时 EN 为常规开关功能；下拉或悬空时 EN 为延时开关功能。
5	VOUT	电压输出端，VOUT 引脚和芯片地之间需要接一个 1μF 的陶瓷电容，为了获得更好的瞬态响应，其值可以增加到 10μF，输出电容应尽可能靠近该管脚。

表1 管脚描述

设计实例

如图 3 是 ZL6305 的典型应用电路图。

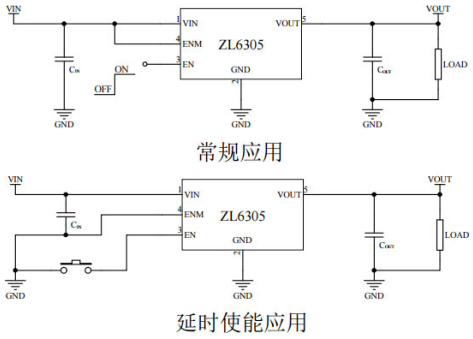


图3 ZL6305典型应用电路

ZL6305 特别适合需要长时间按键开关机的产品。应用中采用 ZL6305 方案可实现不增加任何成本的条件下安全可靠地取代传统延时开关电路控制 LDO 输出。采用图 3 的应用电路即可实现所需的延时开关功能。

致远微电子芯片 ▼

ZL6305和ZL6300的完美结合：解决加载时间长系统的复位难题

ZLG 致远电子 2025-03-19 11:31:33

ZLG 致远微电子新推出带延时开关功能的 LDO-ZL6305，特别适用于低功耗、长按开关机的产品，搭配经典复位监控芯片 ZL6300，可有效解决系统加载时间过长的复位问题，快来了解它们的协同工作方式！

ZL6305：延时开关，轻松控制

ZL6305 是一款带有延时开关功能的 LDO，它的特别之处在于，你可以只用此一颗芯片，就能搭建出长按按键来控制设备的开关机电路，具体来说：

- 常规模式：ENM 脚上拉，此刻 EN 脚与我们常用 LDO 的使能脚一样，上拉使能输出，下拉关闭输出；
- 延时模式：ENM 脚下拉，此刻 EN 脚可外接按键，长按 3 秒即可开机，松开按键后再长按 3 秒关机，非常适合长按开关机的低功耗产品。

图1 ZL6305

此外，ZL6305 还具备快速放电功能，和多种保护机制，下图是 ZL6305 的应用原理图：

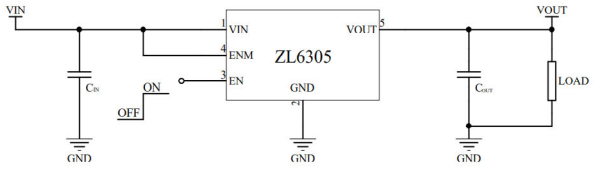


图2 ZL6305的常规应用

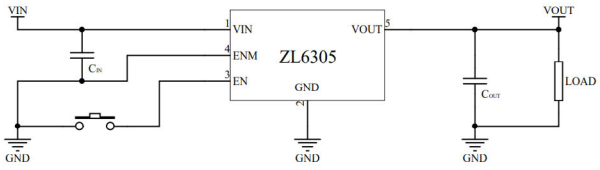


图3 ZL6305的延时应用

ZL6300：复位监控，保障稳定

ZL6300 是我司经典的复位监控芯片，主要用于监控电源电压和程序的“喂狗”情况，如果程序“跑飞”了，ZL6300 会及时产生复位信号，确保系统稳定运行。

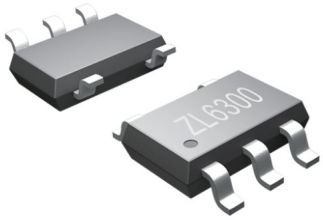


图4 ZL6300

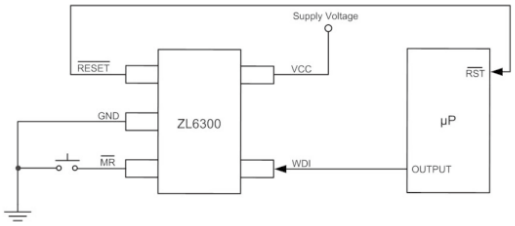


图5 ZL6300常规应用

不过，如果系统加载时间较长（比如超过 3 秒），而 ZL6300 产生复位信号的间隔只有 1.7 秒，系统就会在加载过程中就被复位了，我们之前有篇文章《ZL6300 在加载时间过长系统的巧妙应用》提出了一种解决方案：系统在加载前，先让 ZL6300 自行“喂狗”，等加载完成后再切换成程序“喂狗”，这个方案解决了大部分的问题，但有一种情况没有考虑，那就是假如系统在加载过程中“奔溃”，那系统就永远不会复位了。

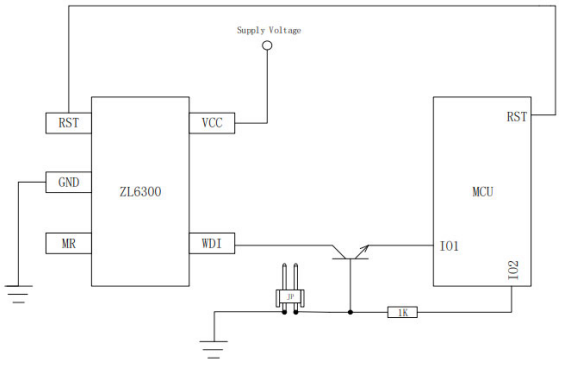


图6 ZL6300的巧妙应用方案

ZL6305与ZL6300的完美结合

为了解决上述问题，我们想到了 ZL6305 的长按 3 秒开机功能，能不能他和 ZL6300 结合起来，延长 ZL6300 的复位信号的时间间隔呢？答案是肯定的！

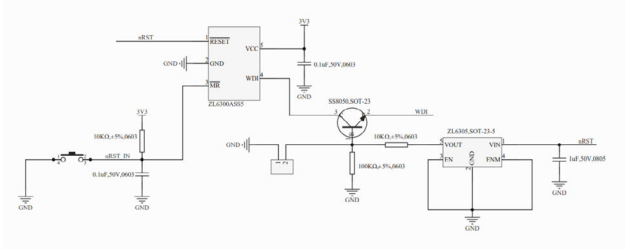


图7 ZL6305与ZL6300相结合的方案

工作原理：

1. 当 ZL6300 上电时，延时 200ms 后 nRST 输出 3.3V 高电平；

2. 由于 ZL6305 的 ENM 下拉，是延时开关功能，开始时 VOUT 没有输出，三极管处于截止状态，ZL6300 的 WDI 脚高阻态，内部自动“喂狗”，所以 nRST 保持高电平；

3. 而 ZL6305 的 EN 脚一直拉低，3 秒后 VOUT 有输出，WDI 不再是高阻态，此时需要“喂狗”，否则 nRST 就会产生复位信号，一旦产生了复位信号，ZL6305 就会重置，所以我们看到，假如不喂狗的话，ZL6300 输出的复位信号间隔就有 3 秒以上了。

注意事项：由于 nRST 是信号线，带载能力不强，请在 ZL6305 的 VIN 处加一个 1uF 电容，而且负载不能过大，建议电阻 5K~10K。

有关 ZL6305 和 ZL6300 的详细资料，请到官网地址下载：
https://www.zlg.cn/microelectronics/down/down/id/296.html

2024/12 第12期

微文摘

ZLG MICRO DIGEST

ZLG致远电子官方微信

广州致远电子股份有限公司
更多详情请访问 www.zlg.cn
欢迎拨打全国服务热线 400-888-4005