ZLG 致远电子

微文摘

ZLG MICRO DIGEST

2024/5 第5期

月刊

新一代高性能小巧型
ZigBee终端采集器

工业级性能

3KM超远距离传输

mesh自组网

网络自愈功能

数据加密

15级路由跳转

项目		描述
ZigBee	协议	ZLG Mesh
	频段	2400~2483.5MHz
	发射功率	-30 ~ 20dBm
	天线	吸盘天线 / 棒状天线 (2.4GHz)
RS-485	波特率	2400~115200bps
DI/DO	DI 输入电平	低电平 0~2.0V，高电平 2.5~24V
	DO 负载 (Max)	最大负载电压 30V，最大负载电流 5A
ADC	采集精度	12 位 ADC，最大可识别电压 36V
按键	DEF	恢复出厂设置
	JOIN	无线组网功能
指示灯	状态指示灯	电源指示灯，RS-485 通信功能灯，ZigBee 通信功能灯
温度	工作温度	-40℃ ~85℃
	存储温度	-40℃ ~85℃
湿度	工作湿度	10%~95%RH(无凝霜)
	存储湿度	10%~95%RH(无凝霜)
供电	供电电压	DC 9~36V
	工作电流 (Max)	9V/200mA
体积	尺寸大小	84.0×63.0×30.0(mm)

致远电子官方网站

致远电子官方微信

CONTENTS
目录

技术平台

EsDA 平台

【技术分享】AWTK 开源串口屏开发 (18) ——用 C 语言自定义命令 …… 04
【AWTK 使用经验】如何自定义 combo_box 下拉框样式 …… 06

ZWS 云平台

【产品应用】EM 储能网关 &ZWS 智慧储能云应用 (3) ——收益接入介绍 …… 08
【产品应用】智慧 CAN 云应用 (1)- 平台功能 …… 10
【产品应用】 智慧 CAN 云应用 (2)-CAN 设备接入 …… 11

边缘计算

工控板 / 工控机

【产品应用】功率电感在 M6442 工控主板中的应用 …… 13
【技术分享】探秘 EPCM3568A-LI：国产高性能工控机的 CAN 通信之旅 …… 15
【技术分享】EPCM3568A-LI 屏幕分辨率和开机 logo 设置 …… 19

行业控制器

【200 个电机驱动】如何快速搭建柔性自动生产线？ …… 22
【200 个电机驱动】如何同步运行？ ——基于 EtherCAT 的柔性电机驱动系统 …… 25
【200 个电机驱动】如何实现 EtherCAT 分布式供电？ …… 27
【技术分享】使用 PCIe EtherCAT 通讯卡控制 IO 从站 step by step（一） …… 29
【产品应用】使用 PCIe EtherCAT 通讯卡控制 IO 从站 step by step（二） …… 32

互联互通

CAN-bus 总线

【技术分享】CAN-bus 应用笔记：节点篇 …… 36
【CAN 总线知识】为什么主机厂愈来愈重视 CAN 一致性测试？ …… 38

无线通讯

【产品应用】致远新一代 LoRa 终端有哪些功能特色？ …… 40
【产品应用】国产蓝牙模组 | BLE5.2 为蓝牙带来了哪些变化？ …… 42

感知控制

电源与隔离

E_UHBCS-6W 系列小体积宽压输入电源模块 …… 44
宽电压输入稳压电源模块 E48_UHFCS-3W 系列 …… 45
E_UHBDD-20W 系列小体积电源模块 …… 46

【技术分享】AWTK 开源串口屏开发(18)——用 C 语言自定义命令

ZLG 致远电子 2024-05-10 11:52:06

如果 AWTK-HMI 内置模型无法满足需求，可以使用 C 语言来扩展默认模型。本文通过一个简单的例子，介绍一下用 C 语言扩展默认模型的方法。

AWTK-HMI 内置了不少模型，利用这些模型开发应用程序，不需要编写代码即可实现常见的应用。但是，有时候我们需要自定义一些命令，以实现一些特殊的功能。

本文档介绍如何使用 C 语言自定义命令。

1. 实现 hmi_model_cmd_t 接口

1.1 exec 函数

本函数用于执行命令。函数原型如下：

```
typedef ret_t (*hmi_model_cmd_exec_t)(hmi_model_cmd_t* cmd, tk_object_t* obj, const char* args);
```

参数：
cmd: 命令对象；
obj: 默认模型对象；
args: 参数。

返回：
RET_OBJECT_CHANGED 表示模型对象发生了变化，界面自动更新。
RET_OK 表示命令执行成功，但模型对象没有发生变化。
其他值表示命令执行失败。

1.2 can_exec 函数

本函数用于判断命令是否可以执行。函数原型如下：

```
typedef bool_t (*hmi_model_cmd_can_exec_t)(hmi_model_cmd_t* cmd, tk_object_t* obj, const char* args);
```

参数：
cmd: 命令对象；
obj: 默认模型对象；
args: 参数。

返回：
TRUE 表示命令可以执行；
FALSE 表示命令不能执行。

1.3 声明命令对象

命令对象一般定义为全局变量。
示例

```
static const hmi_model_cmd_t s_inc_temp_cmd = {
  .name = "inc_temp",
  .exec = inc_temp_exec,
  .can_exec = inc_temp_can_exec,
};
```

2.注册命令

调用函数 hmi_model_add_cmd 注册命令。

```
ret_t custom_cmds_init(void) {
 tk_object_t* model = hmi_service_get_default_model();
 hmi_model_add_cmd(model, &s_inc_temp_cmd);

 return RET_OK;
}
```

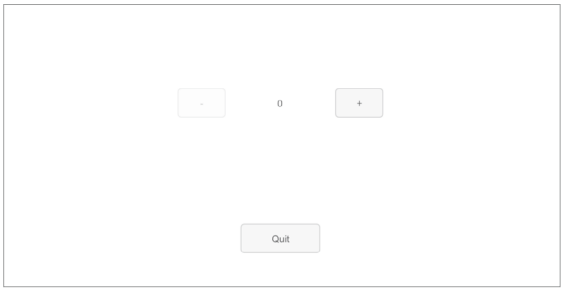
3.完整示例

下面的代码实现了一个命令 inc_temp，用于增加温度属性的值。温度的值小于 100 时，命令可以执行。

```
#define PROP_TEMP " 温度 "

static ret_t inc_temp_exec(hmi_model_cmd_t* cmd, tk_object_t* obj, const char* args) {
 int temp = tk_object_get_prop_int(obj, PROP_TEMP, 0);
 tk_object_set_prop_int(obj, PROP_TEMP, temp + 1);

 return RET_OBJECT_CHANGED;
}

static bool_t inc_temp_can_exec(hmi_model_cmd_t* cmd, tk_object_t* obj, const char* args) {
 int temp = tk_object_get_prop_int(obj, PROP_TEMP, 0);
 return temp < 100;
}
```

```
static const hmi_model_cmd_t s_inc_temp_cmd = {
    .name = "inc_temp",
    .exec = inc_temp_exec,
    .can_exec = inc_temp_can_exec,
};

ret_t custom_cmds_init(void) {
  tk_object_t* model = hmi_service_get_default_model();
  hmi_model_add_cmd(model, &s_inc_temp_cmd);

  return RET_OK;
}
```

完整示例请参考：demo_custom_cmd
https://gitee.com/zlgopen/awtk-hmi/tree/master/hmi/demo_custom_cmd



【AWTK使用经验】如何自定义combo_box下拉框样式

ZLG 致远电子 2024-05-22 11:33:19

AWTK 是基于 C 语言开发的跨平台 GUI 框架。《AWTK 使用经验》系列文章将介绍开发 AWTK 过程中一些常见问题与解决方案，例如：如何加载外部资源？如何设计自定义进度条？这些都会在系列文章进行解答。

假设需要在 ZTP800 示教器 实现一个用于日期选择的下拉框，并且还要求对下拉框做一些美化，此时就需要用户自定义下拉框样式。下面将结合该需求介绍两种修改 combo_box 下拉框样式的方法。

图 1 修改样式后得下拉框效果图

利用theme_of_popup属性修改样式

第一种方式是通过 combo_box 的 theme_of_popup 属性修改下拉框样式，该属性需要指定一个样式文件名称，combo_box 控件会应用该文件内的下拉框样式。

图 2 使用theme_of_popup属性修改下拉框样式

在项目的 design/default/styles 目录下创建一个 xml 文件作为下拉框的样式文件，并配置以下两个样式：

1. 弹出窗口本身的样式，style 名称为“combobox_popup”。

```
<popup>
  <style name="combobox_popup" bg_color="#071F4B" border_color="#00000000">
    <normal/>
    <disable/>
    <focused/>
  </style>
</popup>
```

2. 列表项的样式，style 名称为“default”。

```
<combo_box_item>
  <style name="default" bg_color="#071F4B" font_size="24" icon_at="left" text_color="#FFFFFF">
    <normal/>
    <disable/>
    <disable_of_checked/>
    <focused bg_color="#2E74B5"/>
    <focused_of_checked bg_color="#2E74B5"/>
    <normal_of_checked/>
    <over bg_color="#5B9BD6"/>
    <over_of_checked/>
    <pressed bg_color="#2E74B5"/>
    <pressed_of_checked/>
  </style>
</combo_box_item>
```

最后可以在 AWTK Designer 中设置 combo_box 控件的 theme_of_popup 属性并打包资源文件，另外也可以使用 C 代码方式动态设置样式：

```
combo_box_set_theme_of_popup(combo_box, "my_combo_box");
```

利用open_window属性修改样式

第二种方式是通过修改 combo_box 控件的 open_window 属性修改下拉框样式，该属性接收一个 popup 窗体的名称，用户在 popup 实现自定义下拉框。

若同时设置了options属性与open_window属性，会优先打开open_window属性的popup窗体。

图 3 使用open_window属性修改下拉框样式

1. combo_box控件open_window属性用法

在 AWTK Designer 创建一个 popup 类型窗体并进行布局，使用 combo_box_item 控件作为下拉框选项。

完成 popup 窗体布局后，可以在 AWTK Designer 中设置 combo_box 控件的 open_window 属性，也可以使用 C 代码方式动态设置样式：

```
combo_box_set_open_window(combo_box, "combo_box_menu");
```

2. 获取与设置combo_box当前选择项索引值

默认的 combo_box 控件可以通过控件对象的 selected_index 属性直接读取当前选择项的索引值，也可通过控件函数 combo_box_set_selected_index 设置当前选择项索引值。

需要注意的是设置 open_window 属性后还要设置**相同的 options 属性才可以正常获取与设置当前选项索引值。**例如在 open_window 打开的 popup 三个选项分别为“星期一”、“星期二”和“星期三”，则需要设置 combo_box 控件 options 属性为“0: 星期一 ;1: 星期二 ;2: 星期三”，这样才能通过控件对象属性和控件函数获取与设置当前选项的索引值。

【产品应用】EM储能网关&ZWS智慧储能云应用(3) ——收益接入介绍

ZLG 致远电子 2024-05-16 11:40:32

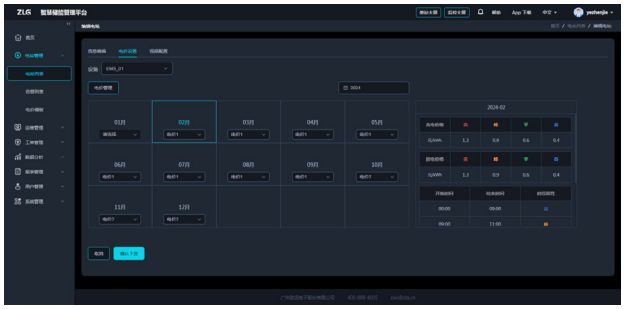
储能系统中，业主单位较为关注的是储能系统带来的收益情况。ZWS智慧储能云平台，支持储能系统根据场景与业务需求，对于储能收益进行接入。

储能系统中，业主单位较为关注的是储能系统带来的收益情况。ZWS智慧储能云平台是为储能系统提供云端数字化运维与管控的行业系统平台，其中可根据地区差异，季度差异，配置未来 12 个月份的电价，根据储能系统充放电数据进行收益计算，并生成不同时间维度收益相关报表。针对不同的储能系统场景与业务需求，云端支持两种接入方式：

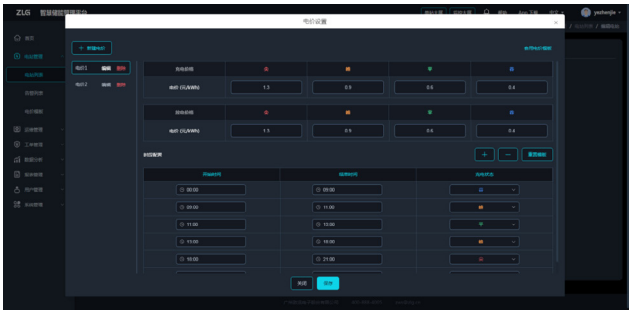
1. 云端统计，便捷式接入，根据充放电量数据，云端实时计算收益；
2. 设备端统计，针对储能系统本地需展示收益数据，根据数据源唯一，云端对设备上报收益数据进行截取统计与显示。

云端统计

1、以电站为维度，由用户配置相应月份的电价。云端只需设备端上报的充电量、放电量数据，到达相应的月份，会取相应的电价配置，统计收益。

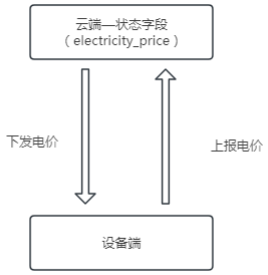


2、电价管理，由用户配置充放电价格与时段配置，形成相应的电价策略。



设备端统计

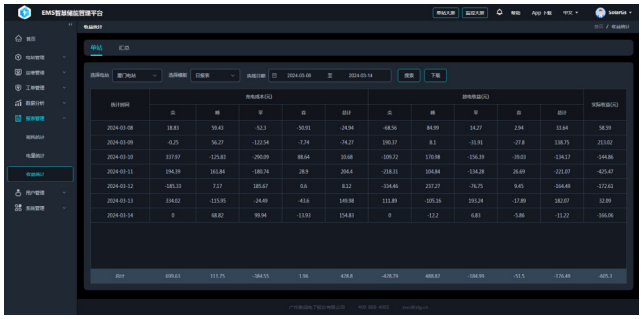
1、设备端统计接入，需对接电价的配置功能。云端以一个状态值的形式存储，设备端以一个配置的方式来存储电价。设备端本地的电价配置发生修改后，需要将修改后的电价配置上报云端；云端下发电价配置后，设备端需要把下发的配置存储应用到设备，并上报一次电价配置。



2、设备端将累计收益，尖峰平谷充电成本、放电收益上报至云平台，云平台作相应的截取统计与显示。

收益显示

两种接入方式，云端都可以作相应支持，其中终端业主，在云端都可以查看相应的收益报表数据。



关联产品

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

【产品应用】智慧CAN云应用(1)-平台功能

ZLG 致远电子 2024-05-24 11:43:42

CANDTU 云平台是一个专业 CAN 报文存储与分析平台， CANDTU 云平台解决方案使汽车路测行业、工业自动化行业能够更有效地采集、管理和分析 CAN 报文，下面将详细介绍 CANDTU 云平台的功能。

CANDTU 云平台的功能有设备状态监控、设备配置管理、数据采集和分析、固件升级与设备管理等基础功能，以及触发管理、UDS 诊断、GPS 轨迹跟踪等高阶功能。本文将重点介绍 CANDTU 云平台的功能。

1. 设备状态监控

CANDTU 云平台能够监控 CANDTU 设备的数据，包括固件版本、通道使能情况等关键信息。用户可以实时查看设备状态，远程下发命令，如“开始 / 停止记录、同步时间、清空记录“指令，实现对设备状态的监控。

2. 设备配置管理

CANDTU 云平台支持对 CANDTU-200 和 CANDTU-400 系列设备进行远程配置。用户可以方便地对 CAN 通道配置使能，对设备记录配置进行修改，以及导入 / 导出配置文件，实现设备的个性化设置。

3. 数据采集与分析

CANDTU 设备上传的实时数据和历史数据在云平台上得到了有效的管理。用户可以实时监控 DBC 数据，并能够自定义配置通道与数据项实现新增曲线，实时查看基于 DBC 数据的可视化曲线。此外，用户还可以通过数据回放功能，对 CAN 数据进行重复观看。

4. 固件升级

CANDTU 云平台支持对 CANDTU 设备的远程固件升级，包括标准固件和自定义固件。用户可以通过云平台方便地进行固件的上传和升级操作，同时支持差分升级，确保固件升级的顺利进行。

除了上述功能，CANDTU 云平台还提供了触发管理、UDS 诊断、GPS 轨迹跟踪等丰富的高阶功能，这些功能进一步扩展了平台的能力，为客户提供了更多种管理诊断、轨迹可视化等功能。通过这些高阶功能，能够帮助客户提高工作效率。

【产品应用】
智慧CAN云应用(2)-CAN设备接入

ZLG 致远电子 2024-05-29 11:39:18

随汽车路试行业的发展，对 CAN 报文采集和分析的需求增加。CANDTU 设备能进行数据高效采集，并通过云平台进行分析。本文将介绍如何将 CANDTU 设备连接到云平台。

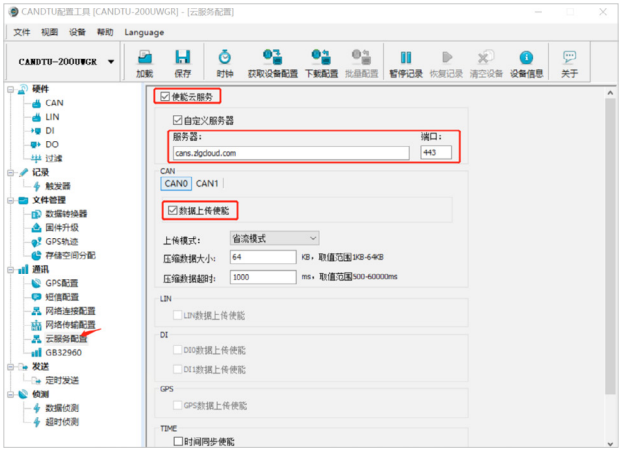
将 CANDTU 设备连接到云平台分两个步骤：

1、使用“CANDTU 配置工具”对设备进行配置。

2、将设备添加到云平台。

使用“CANDTU配置工具”进行配置

1. 给设备插入 4G 的 SIM 卡；使用“CANDTU 配置工具”，进入“云服务配置”配置，对“使能云服务”、“配置云服务器地址和端口”、“数据上传使能”选项进行勾选；对设备完成配置更新 。



2. 复制设备信息中的“云 SN”用于云平台添加设备使用。

将该设备添加到云平台

1. 登录 CAN 云平台，访问网址：cans.zlgcloud.com 并使用账号登录。

2. 点击“添加设备”按钮，选择设备类型 candtu-200，输入设备名称和云 SN，点击“保存“完成设备添加。

3. 点击设备列表中的“查看”按钮，进入设备详情页面。在设备详情中选择“设备数据”-“实时数据”查看实时数据。

选择“设备数据”-“历史数据”可以查看历史数据，并可检索特定时间段的数据。

技术平台▼

【产品应用】
功率电感在M6442工控主板中的应用

ZLG 致远电子 2024-05-27 11:39:10

在现代工业控制系统中，功率电感是不可或缺的组件之一。它们不仅保证了电路的稳定运行，还提高了整个系统的效率。本文将探讨功率电感在 M6442 工控主板中的应用以展示其重要性。

功率电感分类

功率电感一般可分为无屏蔽、带磁屏蔽罩、Coating Resin、以及一体成型四种类型，如表 1 所示。

表 1 功率电感的分类

电感	无屏蔽	带磁屏蔽罩	Coating Resin	一体成型
外形				
优点	便宜	屏蔽效果较好	成本比带磁屏蔽罩的便宜，屏蔽效果好	• 同体积下，电流参数大 • 温度曲线好，屏蔽效果好
缺点	• 屏蔽效果差 • 线圈匝数多，阻抗大	成本稍高，体积大	结构稳定性稍差，因树脂与铁氧体的线膨胀系数不同而容易导致开裂	贵

功率电感主要参数

功率电感的主要参数有电感值、直流电阻、饱和电流、温升电流、以及自谐振频率，如表 2 所示。

表 2 功率电感的主要参数

参数	定义	常见参考值	一般特性
Inductance (μH) 电感值	电感器的标称电感量，一般在100KHz频率下所测得	0.47 ~ 47	由于导磁率和分布电容的存在，电感量将随频率而变化
D.C.R. (mΩ) 直流电阻	电感器的内部线圈的总直流电阻	10 ~ 1000	封装越小或者感量越大，DCR越大
Saturation current (A) 饱和电流	电感值下降10% ~ 35%时的电流值	0.1 ~ 15	感量越小，饱和电流越大
Temperature rise current (A) 温升电流	电感器温升增加40°C时的电流值	0.1 ~ 10	一般比饱和电流小，感量越小，温升电流越大
Self-Resonant Frequency (MHz) 自谐振频率	电感器与其分布电容达到谐振时的频率	1 ~ 1000	感量越小，自谐振频率越大

功率电感基本作用

功率电感主要用于滤波、能量存储和转换、限制电流以及抑制电磁干扰。在工控主板中，这些功能尤为重要，因为它们直接影响到系统的稳定性和效率。

1. 滤波功能

在常见的 BUCK 电源电路中，功率电感可通过下式来实现滤波功能：

$$L(f) = \frac{V_{in} - V_{out}}{2\pi f \Delta I}$$

其中，L(f) 是电感值，Vin 是输入电压，Vout 是输出电压，f 是开关频率，ΔI 是纹波电流。通过调整电感值可有效地滤除噪声，保证输出电压的稳定。

2. 能量存储与转换

在开关电源中，功率电感可存储能量并在需要时释放，其能量转换过程可通过下式描述：

$$E = \frac{1}{2}LI^2$$

其中，E 是电感储存的能量，L 是电感值，I 是任意时刻流过电感的电流。这一过程对于实现高效能量转换至关重要。

3. 限流与保护

功率电感还能限制电流的突变，防止电流过大毁坏负载。电感的限流作用可用下式表示：

$$I(t) = \frac{V}{L} \cdot t$$

其中，I(t) 是某一时刻 t 流过电感的电流，V 是电感两端的电压，L 是电感值。通过将电感连接到电路中，特别是在电流变化较大的情况下，可有效地限制电流的快速变化，从而起到类似保险丝的作用。

工控主板作为工业自动化设备中的关键硬件，承载着电源和数据两大流量，由于处理器在工控主板运行中起着至关重要的作用，因此主板供电部分设计的优劣，直接影响到整个工业控制系统工作的稳定性和安全性。

因此，ZLG 致远电子精心推出 M6442 工控主板，该主板搭配 M64xx 核心板来使用，并提供 1 个 M.2 接口以实现 5G 功能、1 个 SIM 卡座、3 路千兆以太网接口、1 路调试接口、1 路 CAN 接口、1 路 RS232 接口、1 路 RS485 接口、1 路 GPMC 接口、1 路扩展 IO 接口（包含 4 路 PWM、8 路 ADC、4 路 SPI、3 路 UART、3 路 I2C、8 路 GPIO）、1 个 TF 卡座、1 个 JTAG 座。M6442 工控主板既可满足 M64xx 核心板的接口评估，也可直接用于 PLC、电机驱动器、工业网关、远程监控、工厂自动化等应用场合。

边缘计算▼

品质好的功率电感一般采用高导磁率、不易饱和的磁芯，因此不需很多的绕线圈数即可得到足够的磁通量。M6442 工控主板的一级 5V 电源供电和二级 CPU 各路电源供电，由 DCDC 以及多个陶瓷电容和高品质电感组成，能为处理器及其外设提供 10 ～ 20W 的电能，如图 1 所示。

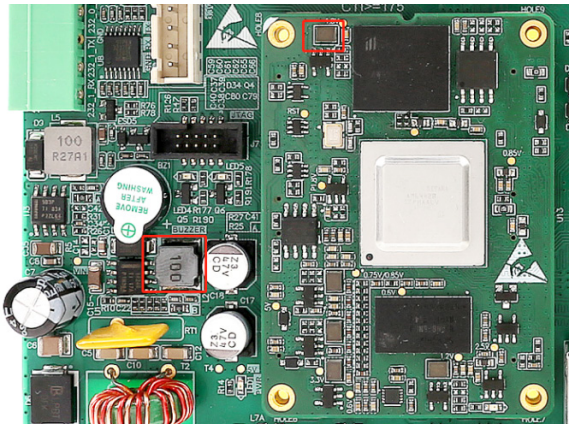


图 1 M6442工控主板中功率电感的位置示意图

后续的文章我们将针对 DCDC 电源电路中的功率电感如何选型以及相关设计经验进行讨论，敬请留意。

【技术分享】探秘EPCM3568A-LI：国产高性能工控机的CAN通信之旅

ZLG 致远电子 2024-05-14 11:39:23

在工业物联网时代，CAN 是一种非常普遍的通信技术。本文将给大家介绍如何使用 EPCM3568A-LI 边缘计算控制器与上位机实现 CAN 通信范例。

EPCM3568A-LI产品简介

EPCM3568A-LI 采用 RK3568 四核处理器，主频高达 2.0GHz，内置 1TOPs NPU，作为边缘计算网关，EPCM3568A-LI 以其强大的计算能力，轻松高效处理数据，为用户带来更加便捷、智能的能源管理体验。

前期准备

1. 软件方面

- ZCANPRO.exe，下载地址：https://manual.zlg.cn/web/#/59/2490
- 下载 EPCM3568A-LI 开发环境，下载地址：https://manual.zlg.cn/web/#/269/10177
- 安装好虚拟机的电脑；
- MobaXterm 虚拟终端。

2. 硬件方面

EPCM3568A-LI 小型边缘计算网关；
网线 1 根 / 交换机 1 个；
USBCAN-8E-U1 个。

CAN概述

CAN(Controller Area Network，控制器局域网) 是一种高可靠性的串行通信协议，专为汽车和工业控制等关键领域设计。它以高可靠性、实时性和出色的抗干扰能力为特点，非常适合在需要精确数据传输的控制系统中使用。

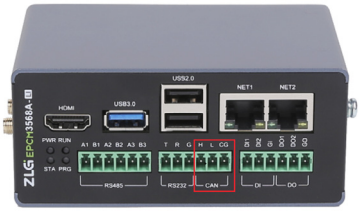


图1 EPCM3568A-LI的CAN接口

USBCAN-8E-U介绍

USBCAN-8E-U 是致远电子开发的一款基于 Linux 操作系统的高性能 CAN 接口卡，其兼容 USB2.0 全速总线规范，集成 1~4 路 CAN-bus 接口，CAN 通道集成独立的电气隔离保护电路。接口卡使 PC 通过 USB 端口连接至 CAN 网络，构成一个 CAN-bus 控制节点。

图2 USBCAN-8E-U

USBCAN-8E-U 高性能 CAN 接口卡是 CAN-bus 产品开发、CAN-bus 数据分析的强大工具；同时、即插即用等特点，也是便携式系统用户的最佳选择。USBCAN-8E-U 接口卡上自带电气隔离模块，使接口卡避免由于地环流的损坏，增强系统在恶劣环境中使用的可靠性。

USBCAN-8E-U 高性能 CAN 接口卡支持 Win2000/XP/7/8/10 等操作系统，也支持 Linux 的操作系统。

在 ZLG 产品用户手册上有 USBCAN-8E-U 的具体介绍和用法说明，用户可以自行查阅，手册地址：https://manual.zlg.cn/web/#/59/2481

EPCM3568A-LI与上位机的CAN通讯实验

本节，我们将利用 EPCM3568A-LI 的 CAN 接口（图 1）和 USBCAN-8E-U（图 2），配合上位机模拟 CAN 通信。操作流程如下：

首先为 EPCM3568A-LI 接通电源，并通过网线将其与笔记本连接至同一交换机。待系统启动后，使用 MobaXterm 的 SSH 登录，IP：192.168.1.136，用户名：zlgmcu，密码：zlgmcu，如图 3。

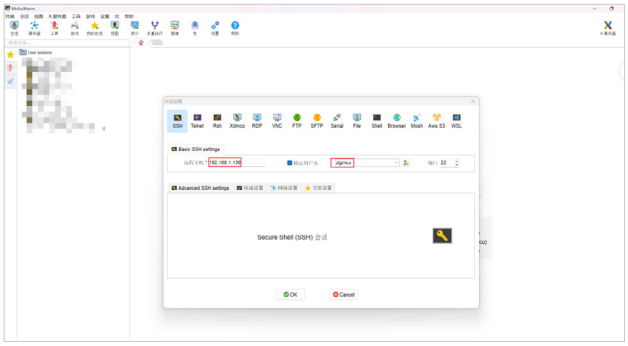


图3 SSH登录EPCM3568A-LI

1. 初始化EPCM3568A-LI的CAN接口

在使用 Socket CAN 之前，需要先设置 CAN 的波特率，波特率为 1000kbps，并激活 CAN 网络接口。执行如下指令：

```
sudo ifconfig can0 down
sudo ip link set can0 type can bitrate 1000000
sudo ifconfig can0 up
```

完成初始化后 , 输入如下指令，可以看到如图 4 所示

```
sudo ifconfig can0
```

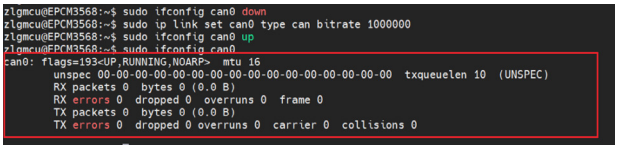


图4 can0设备节点

2. 安装CAN 通信测试工具 can-utils

在确保 EPCM3568A-LI 能够上网的情况下，执行下面指令，安装 CAN 通信测试工具 canutils：

```
sudo apt install can-utils
```

安装完后输入 sudo can 后按 TAB 键发现多了几个工具，如图 5 所示：

图5 can-utils命令行可执行工具

canutils 工具包内含 5 个独立的程序：canconfig、candump、canecho、cansend、cansequence。这几个程序的功能简述如下：

- canbusload：测量 CAN 总线的负载情况，帮助用户评估 CAN 总线的性能和稳定性。
- canfdtest：测试 CAN Flexible Data Rate（CAN FD）总线的工具，用于发送和接收 CAN FD 数据帧。
- canplayer：回放 CAN 总线数据，模拟将保存的 CAN 数据文件发送到 CAN 总线。
- cansend：往指定的 CAN 总线接口发送指定的数据。
- candump：捕获 CAN 总线上的数据帧，并将其显示在终端上。
- cangen：生成 CAN 总线数据帧，模拟发送特定的 CAN 数据。

3. EPCM3568A-LI和USBCAN-8E-U接线

接下来看 EPCM3568A-LI 的 CAN 接口，如下所示：

标识	信号	说明
H	H	CAN_H信号线
L	L	CAN_L信号线
G	G	CAN接地线

图6 EPCM3568A-LI的CAN接口

USBCAN-8E-U 接口卡集成了 1~4 路 CAN 通道，可以连接一个 CAN-bus 网络或者 CAN-bus 接口的设备，其 CAN-bus 通道采用标准公头 DB9 座子引出。DB9 的引脚定义如图 7 所示。

引脚	信号	描述
1	N.C.	未用
2	CAN_L	CAN_L 信号线
3	CAN_GND	参考地
4	N.C.	未用
5	CAN_SHIELD	屏蔽线
6	CAN_GND	参考地
7	CAN_H	CAN_H 信号线
8	N.C.	未用
9	N.C.	未用

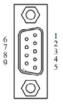


图7 DB9 插座的管脚信号定义

用户可以通过选配的 DB9OPEN5 转换器，将 DB9 插座的 CAN-bus 信号转换至易于连接的 5 引脚 OPEN5 连接器，接口说明见图 8。

引脚	信号	描述
1	V-	未用
2	CAN_L	CAN_L信号线
3	SHIELD	屏蔽线 (FG)
4	CAN_H	CAN_H信号线
5	V+	未用

图8 DB9OPEN5转换器信号定义

CAN-Bus 通讯连接如图 9 所示。

图9 CAN-Bus通讯连接

所以我们将 EPCM3568A-LI 和 USBCAN-8E-U 的 H、L 两两对接，如图 10，然后开始实验。

图10 EPCM3568A-LI和USBCAN-8E-U接线

4. 收发实验

4.1 设置上位机 ZCANPRO

如图 11，启动通道 0，并且将波特率设置为 1000kbps。

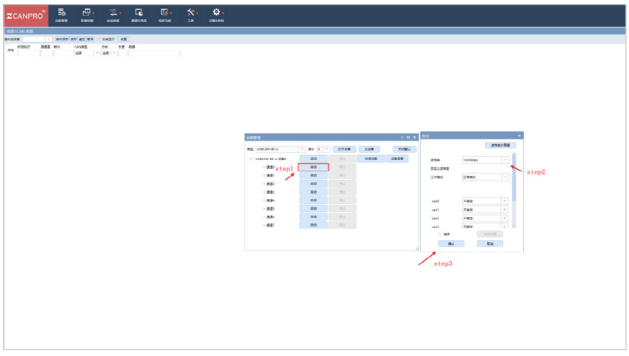


图11 设置上位机ZCANPRO

4.2 EPCM3568A-LI 发送 CAN 数据帧

首先在 EPCM3568A-LI 执行下面指令查看 CAN0 的负载情况，如图 12 所示

```
sudo canbusload can0@1000000 -r -t -b -c
```

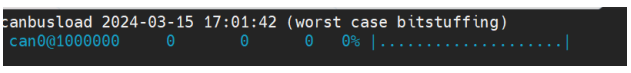


图12 使用canbusload查看can0负载

可以看到当前 CAN0 上没有负载。

接下来，EPCM3568A-LI 发送 CAN 数据帧，上位机接收，在 ZCANPRO 上显示收到的数据。EPCM3568A-LI 发送 11223344，帧 ID 为 123，输入下面指令：

```
cansend can0 123#11223344
```

上位机收到数据 11223344，帧 ID 来自 123，如图 13 所示：

图13 上位机收到EPCM3568A-LI发送的CAN数据帧

4.3 EPCM3568A-LI 接收 CAN 数据帧

接下来，上位机发送 CAN 数据帧，EPCM3568A-LI 接收并显示在终端上。EPCM3568A-LI 设置成接收数据帧模式，输入下面指令：

```
candump -ta can0 &
```

在上位机使用 ZCANPRO 发送数据 00 11 22 33 44 55 66 77，如图 14 所示：

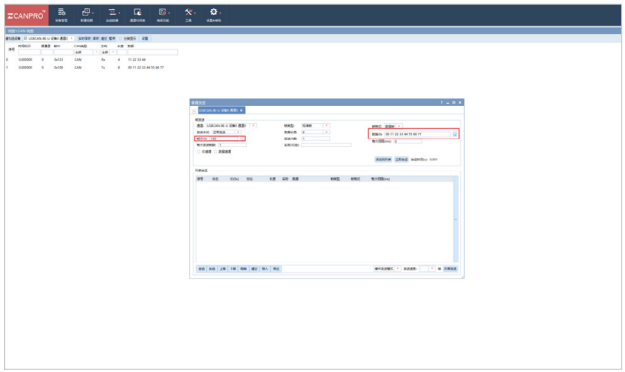


图14 使用ZCANPRO给EPCM3568A-LI发送CAN数据帧

如图，EPCM3568A-LI 接收到 00 11 22 33 44 55 66 77，来自帧 ID：100，如图 15 所示：

图15 EPCM3568A-LI接收到ZCANPRO发送的数据帧

4.4 使用 cangen 循环发送 CAN 数据帧

我们也可以用 cangen，这将在 CAN0 接口上生成 8 字节数据长度的 CAN 帧，并且每 1000 毫秒生成一帧。此外，您也可以根据需要调整其他选项来满足您的测试需求。

```
cangen -g 1000 -D r can0
```

这是 EPCM3568A-LI 循环发送的数据，如图 16 所示：

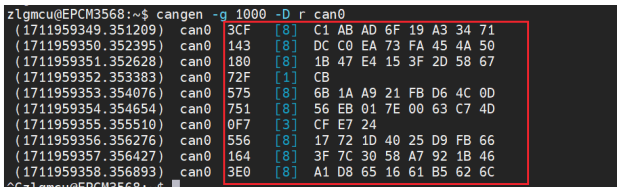


图16 在EPCM3568A-LI使用cangen循环发送CAN数据帧

这是上位机接收到的数据，如图 17 所示：

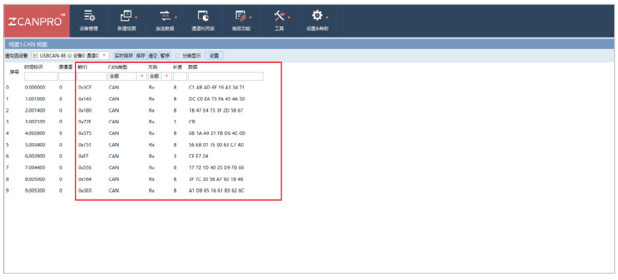


图17 上位机收到EPCM3568A-LI使用cangen循环发送的CAN数据帧

从图 16 和图 17 中可以看到，EPCM3568A-LI 发出的 CAN 数据帧与上位机收到的数据帧是一样的，本实验实现了 EPCM3568A-LI 与上位机的 CAN 通信。

【技术分享】EPCM3568A-LI屏幕分辨率和开机logo设置

ZLG 致远电子 2024-05-21 11:37:51

本文将给大家介绍如何使用 EPCM3568A-LI 边缘计算控制器通过 HDMI 口连接显示器，并实现自定义屏幕显示分辨率和开机 logo 功能。

EPCM3568A-LI产品简介

EPCM3568A-LI 采用 RK3568 四核处理器，主频高达 2.0GHz，内置 1TOPs NPU，作为边缘计算网关，EPCM3568A-LI 以其强大的计算能力，轻松高效处理数据，为用户带来更加便捷、智能的能源管理体验。

前期准备

1. 软件方面

- 下载 EPCM3568A-LI 开发环境，地址：https://manual.zlg.cn/web/#/269/10177
- 安装好虚拟机的电脑；
- MobaXterm 虚拟终端。

2. 硬件方面

- EPCM3568A-LI 小型边缘计算网关；
- 网线 1 根 / 交换机 1 个；
- HDMI 显示屏一个。

SSH登录EPCM3568A-LI

当 EPCM3568A-LI 接通电源后，使用网线将笔记本电脑与该设备连接至同一交换机。系统启动完成后，使用 MobaXterm 的 SSH 登录 IP：192.168.1.136，用户名：zlgmcu，密码：zlgmcu，如图 1。然后，进入系统后，使用指令查看是否生成网络接口，如图 2。

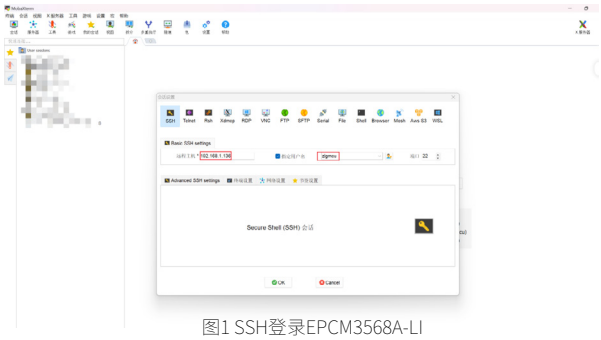


图1 SSH登录EPCM3568A-LI

```
zlgmcu@EPCM3568:~$ sudo ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.137.30  netmask 255.255.255.0  broadcast 192.168.137.255
        ether 6a:23:de:8c:59:35  txqueuelen 1000  (Ethernet)
        RX packets 100  bytes 10650 (10.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 894 (894.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 42

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.136  netmask 255.255.255.0  broadcast 192.168.1.255
        ether 6e:23:de:8c:59:35  txqueuelen 1000  (Ethernet)
        RX packets 1809  bytes 692349 (676.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 829  bytes 97453 (95.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 44

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 34:6f:24:5f:f4:a5  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 197  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

图2 网络配置信息

EPCM3568A-LI切换不同分辨率的液晶屏

EPCM3568A-LI 工控机提供 1 路 HDMI 接口，兼容 HDMI 1.4 和 2.0 规格，支持 1080p@120Hz 或 4096x2304@60Hz 输出。

1. 查看当前显示器的分辨率和屏幕设置

我们将会用到命令行工具：xrandr

xrandr 是一款官方的 RandR (Resize and Rotate)Wikipedia:X Window System 扩展配置工具。它可以设置 屏幕显示的大小、方向、镜像等。

在 MobaXterm 终端执行如下命令，既可查看当前显示器支持的分辨率，如图 3：

```
su zlgmcu -c "DISPLAY=:0 xrandr"
```

```
zlgmcu@EPCM3568:~$ su zlgmcu -c "DISPLAY=:0 xrandr"
密码:
Screen 0: minimum 320 x 200, current 1920 x 1080, maximum 8192 x 8192
HDMI-1 connected primary 1920x1080+0+0 (normal left inverted right x axis y axis) 0mm x 0mm
   1920x1080     60.00*+  74.97    60.00    50.00    59.94
   1920x1080i    60.00    60.00    50.00    59.94
   1600x900      60.00
   1280x1024     75.02    60.02
   1152x864      75.00
   1280x720      60.00    60.00    50.00    59.94
   1024x768      75.03    60.00
   800x600       75.00    60.32
   720x576       50.00    50.00
   720x480       60.00    60.00    59.94    59.94    59.94
   640x480       75.00    60.00    59.94    59.94
   720x400       70.08
DSI-1 connected (normal left inverted right x axis y axis)
   1280x800      60.83 +
```

图3 当前显示器所支持的分辨率

2. 切换不同的分辨率

从图 3 可知，我们使用的显示屏可以支持多种分辨率，目前的分辨率为 1920x1080。

图4 1920x1080的分辨率

接下来，我们尝试切换不同的分辨率，输入以下命令分辨率切换为640x480，如图 5。

```
su zlgmcu -c "DISPLAY=:0 xrandr --output HDMI-1 --mode 640x480"
```

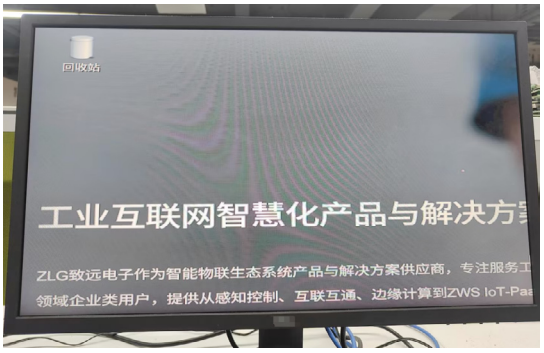


图5 640x480的分辨率

除了调节分辨率，我们还可以在指令的分辨率后加入“--rotate <left|right|inverted|normal> 旋转屏幕：

```
su zlgmcu -c "DISPLAY=:0 xrandr --output HDMI-1 --mode 640x480 --rotate right"
```

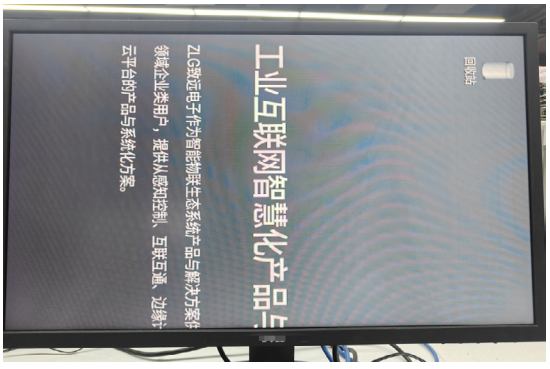


图6 顺时针旋转90度

如果需要恢复屏幕为最原始的状态，只需要在指令中分辨率改为1920x1080，后面加上 --rotate normal：

```
su zlgmcu -c "DISPLAY=:0 xrandr --output HDMI-1 --mode 1920x1080 --rotate normal"
```

此时，屏幕变回图 4 的状态。

修改开机启动LOGO

1. 说明

出厂固件预留 4MB 的 logo 分区，以便用户可以自定义其开机启动 LOGO。

```
zlgmcu@EPCM3568:~$ sudo fdisk -l
Disk /dev/ram0: 4 MiB, 4194304 bytes, 8192 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes


Disk /dev/mmcblk0: 7.3 GiB, 7818182656 bytes, 15269888 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 7F690000-0000-4D36-8000-314F000042B3

Device          Start      End  Sectors  Size Type
/dev/mmcblk0p1  16384    24575     8192    4M unknown
/dev/mmcblk0p2  24576    32767     8192    4M unknown
/dev/mmcblk0p3  32768    98303    65536   32M unknown
/dev/mmcblk0p4  98304   163839    65536   32M unknown
/dev/mmcblk0p5 163840   172031     8192    4M unknown
/dev/mmcblk0p6 172032 15269823 15097792  7.2G unknown
```

图7 系统logo分区

启动LOGO的显示策略

启动过程中优先从 logo 分区加载 LOGO 图片并显示，如果 logo 分区不存在，或者 logo 分区没有对应的 LOGO 图片数据，将显示默认的 LOGO 图片。

2. 自定义启动LOGO示例

2.1 选择一张常见格式的图片（以下测试使用 jpg 格式图片），在 ubuntu 虚拟机安装 ffmpeg 工具。

```
sudo apt install ffmpeg
```

2.2 将 jpg 图片转换成适合屏幕分辨率大小（1920x1080）的 8 位 bmp 图片。

```
ffmpeg -i test.jpg -s 1920x1080 -pix_fmt pal8 mylogo.bmp
```

2.3 制作 logo 镜像。

```
cat mylogo.bmp > logo.img && truncate -s %512 logo.img && cat mylogo.bmp >> logo.img
```

2.4 检测 logo 镜像大小（不能超过 4M）。

```
ll -h logo.img
```

```
zlg@arm-multicross:~$ ll -h logo.img
-rw-rw-r-- 1 zlg zlg 4.0M 4月  22 11:04 logo.img
zlg@arm-multicross:~$
```

图8 查看logo.img的大小

2.5 将 logo.img 拷贝到板子上，使用 dd 命令写进 logo 分区。

```
sudo dd if=logo.img of=/dev/mmcblk0 seek=163840
```

```
zlgmcu@EPCM3568:~$ sudo dd if=logo.img of=/dev/mmcblk0 seek=163840
记录了8105+1 的读入
记录了8105+1 的写出
4149814 bytes (4.1 MB, 4.0 MiB) copied, 0.355553 s, 11.7 MB/s
```

图9 将logo.img写进logo分区

重启开机，此时 LOGO 已更换，如图 10。

图10 EPCM3568A-LI新的开机logo

总结

针对许多客户都有显示屏切换不同分辨率，设置新的开机 logo 需求。本节使用 EPCM3568A-LI 自带的 HDMI 口进行显示屏不同分辨率的切换，并重新设置开机 logo。

【200个电机驱动】如何快速搭建柔性自动生产线?

ZLG 致远电子 2024-05-13 11:34:27

基于 EtherCAT 总线的电机驱动器，可为几百个电机的自动生产线应用提供高速、精确、灵活和可靠的通信和控制解决方案，有助于提高生产效率、降低成本并增强系统的性能和可靠性。

在工业自动化生产线中，电机是必不可少的执行部件，有时一个自动化生产线可能需要控制几十甚至几百个电机。系统中需要控制如此多的电机，此时既要保证系统的实时性，还要方便系统的扩展、维护和工程师进行系统的设计和开发，选用 EtherCAT 总线通讯的驱动器控制电机目前来说是最好的选择。EtherCAT（以太网控制自动化技术）可以控制步进电机实现高精度的定位和运动，EtherCAT 与步进电机的结合可实现高精度、高效率的运动控制，适用于各种工业自动化和机械控制应用。

EtherCAT在电机驱动的应用

EtherCAT 在电机应用中具有以下优势：

1. 支持高速的数据传输，具有极低的延迟，能够实现快速的控制循环，从而提高了电机控制的实时性和精度。

2. 提供了精确的同步机制，使得多个电机能够以高精度的方式进行同步运动，适用于需要精确协同工作的应用。

3. 支持多种拓扑结构，如线型、星型、树形等，可以根据实际需求灵活配置，降低了布线的复杂性。

4. 具有良好的可扩展性，可以轻松地添加或删除电机节点，适应不同规模的系统。

5. 提供了丰富的诊断功能，可以实时监测电机的状态和参数，快速定位故障，提高了系统的可靠性和维护效率。

6. 采用了时分复用技术，保证了数据传输的实时性和确定性，适用于对时间要求严格的电机控制应用。

7.EtherCAT 是一种开放的工业以太网标准，**支持不同厂商的设备之间的互操作性，**例如可以快速接入 DI、DO、模拟量采集等模块，更加方便系统集成。

8. 由于 EtherCAT 使用以太网作为物理底层，利用以太网的优势，**可以减少线缆数量，降低系统成本。**

实际应用中可能还有以下需求：

1. 小体积，结构紧凑，在最小的体积下集成最多的从站。
2. 易于安装和维护。
3. 更灵活的扩展，可快速增加和减少电机数量。
4. 安装更加简洁，降低布线的复杂性。

为满足个性化的电机驱动需求，广州致远电子推出插板式电机驱动系统，该系统采用 EtherCAT 总线通讯，支持高速的数据传输，具有极低的延迟，能够实现快速的控制循环，从而提高了电机控制的实时性和精度。从站电机驱动模组为 59mm*50mm 的小尺寸模块，采用 40 脚 2.54mm 标准排针接口。用户可按需自行设计制作分线底板，驱动器插在底板上通过 EtherCAT 网络级联，总线最多可扩展 65535 个从站设备。

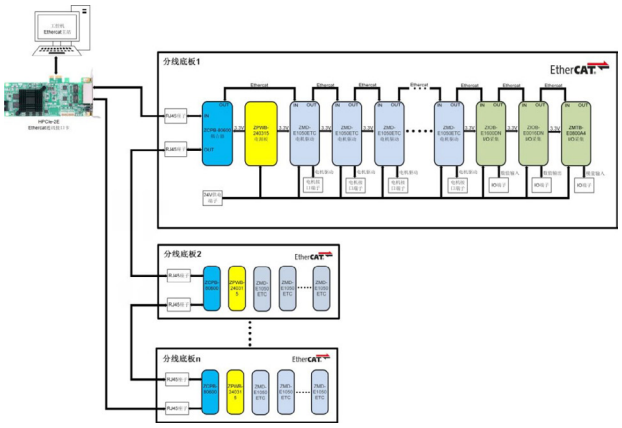


图1 致远电子插板式电机驱动系统方案框图

表1 致远电子EtherCAT电机驱动器

型号	总线类型	电机类型	电源电压	电机路数	电流
ZMD-E1050ECT	EtherCAT	两相步进	24V	1	1.5A
ZMTB-EE1050	EtherCAT	两相步进	24V	1	1.5A
ZMTB-EE2050	EtherCAT	两相步进	24V	2	1.5A*2
ZMTB-EF1200	EtherCAT	两相步进	48V	1	3A

驱动器其他参数：

1.1/2/4/8/16/32/64/128/256 细分选择；

2.CiA402 运动控制协议，支持 CSP/PP/HM/PV 模式；

3.3 个数字输入，1 组 ABZ 正交编码输入；

4. 开环或编码器闭环控制可选。

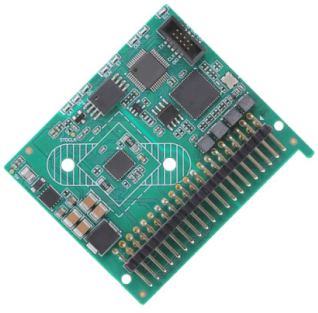


图2 ZMD-E1050ECT电机驱动器

快速实现200个电机驱动应用设计

1. 选择一个拓扑结构

EtherCAT 可以有多种拓扑结构，可以使用线型、环型、树型等拓扑结构，下面介绍使用致远电子 ZMC900E 主站控制器，线型拓扑实施 200 个电机驱动器系统（若要较高的实时性，可使用多个 ZMC900E 主站控制器。若要做冗余设计，可选用致远电子的 HPCIe-2E 主站卡），系统框图如图 3 所示。

图3 200个EtherCAT电机驱动器系统框图

ZMC900E EtherCAT 主站控制器 是广州致远电子股份有限公司开发的最新一代智能总线型控制器，是面向工厂智能化时代的机器控制器。其采用工业领域内先进的嵌入式 ARM 方案，集实时操作系统、智能算法于一身，以 EtherCAT 工业以太网协议为向导，可以快速、有效、便捷的构建数控智能化设备，以适应工厂智能化、信息化产业的变革。

图4 ZMC900E EtherCAT主站控制器

致远电子 **PCIe EtherCAT 通讯卡** 是一款基于 PCIe 的 EtherCAT 总线通讯接口卡。其采用工业领域内先进的 FPGA 控制方案，通讯速度极高，实时性很强。支持 MiniPCIE、PCIe 半卡、PCIe 全卡设计，可兼容任何类型的 3.3 V/DC MiniPCIE 和 PCIe 插槽。

图5 PCIe EtherCAT通讯卡

2. 底板设计

ZMD-E1050ECT 驱动器的典型应用框图如图 6 所示，底板设计只需选择合适自己的接线端子，将线引出即可使用。由于 EtherCAT 网络是通过 PCB 连接，省去了网线，结构可以设计得非常紧凑。

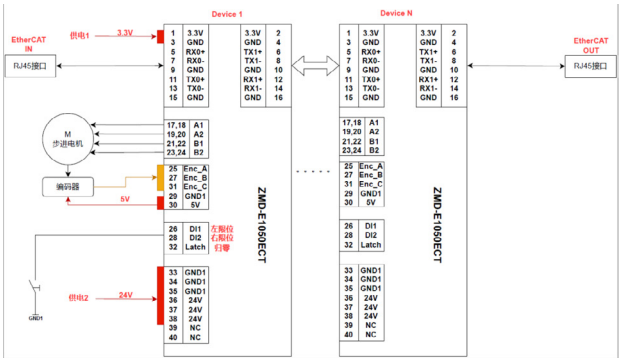


图6 ZMD-E1050ECT典型应用框图

EtherCAT 网口电路参考电路如图 7 所示，在干扰较大的环境下，可使用集成隔离变压器的 RJ45 接口。

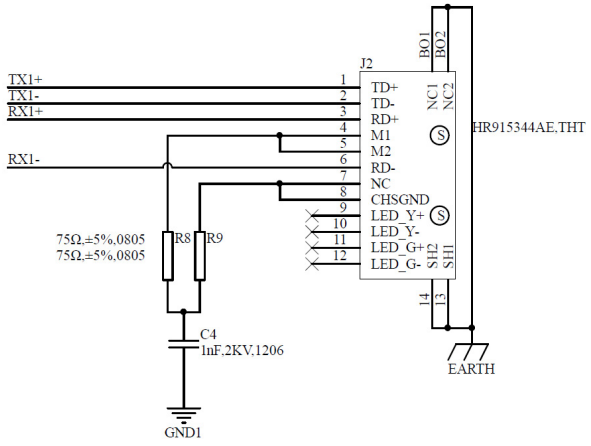


图7 网口设计加隔离变压器

通常情况下数字输入接口通过无源开关接 GND2 即可，若外部需要接 PLC、NPN 型的接近开关等有源信号，可使用普通的光耦进行电平转换，参考电路如图 8 所示：

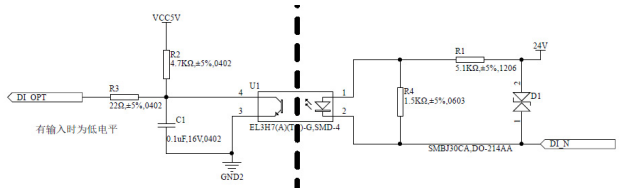


图8 数字输入电路

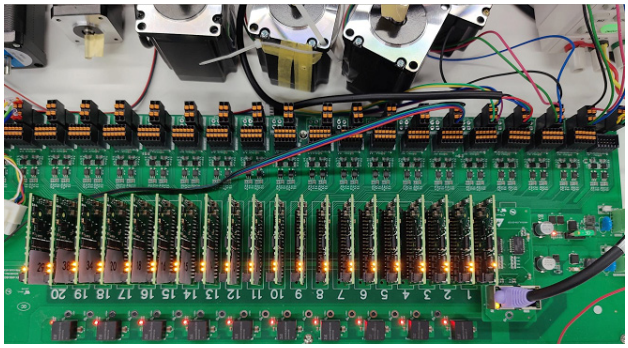


图9 电机控制系统

使用 ZMD-E1050ECT 电机驱动器，只需简单搭建外围电路即可实现多电机应用，分线底板根据使用场景自行设计，使用非常灵活，在较小空间内可布置 200 个电机驱动，致远电子的 20 电机控制系统评估底板如图 9 所示：

【200个电机驱动】如何同步运行？——基于EtherCAT的柔性电机驱动系统

ZLG 致远电子 2024-05-15 11:36:55

在自动化生产线中，EtherCAT 网络轻松实现了过去难以同步控制的成百上千个电机，如流水线等。本文将探讨如何通过柔性电机驱动系统，在 EtherCAT 网络环境中实现对多电机的精确控制。

脉冲信号

传统的伺服电机采用 I/O 脉冲信号控制，主站发一个脉冲，电机走一步。这种驱动器，引线太多、连接不可靠，而且主站也没办法扩展出成百上千组信号出来！

图1

CAN总线

既然 I/O 脉冲控制接线多、又不能控制太多电机，那能否使用总线统一控制？ 答案是可以！首先流行起来是 CAN 总线。CAN 总线的时间同步是通过在总线上传输时间戳来实现，能保证总线上的全部从节点统一时间处理数据！但 CAN 总线理论上最多只支持 110 个从节点，在实际应用中，由于电气特性的限制，通常不能多于 100 个节点。而且数据传输速度也不高，常用的 CAN2.0A 在超高速模式下才 1Mbps。这些也导致满足不了数百个电机驱动器的同步要求。但在 CAN 总线的使用中，CANopen 协议应运而生。CANopen 中最重要的通讯协议栈是 DS301，而 DS402 是在 DS301 基础之上拓展出的伺服类控制协议。DS402 把一个伺服控制系统应该具有的功能都定义好了，厂家和使用者按照协议即可开发和使用符合标准的设备。正是有了 DS402 协议标准，为后面的 EtherCAT 总线铺平了伺服控制的道路。

图2

EtherCAT总线

EtherCAT 是一个开发的架构，基于太网为基础的现场总线系统。具有以下优点：

1. 高速传输，使用双绞线或光线，可以在 30uS 内处理 1000 个分布式 I/O 信号，或在 100uS 内处理 100 个电机轴。

2. 精确同步，具有纳秒级别的同步性，协议栈处理延迟仅需几纳秒。

3. 灵活拓扑，EtherCAT 网络最多可支持 65535 个从设备，且对拓扑的结构没有限制，线性、树形、星型等拓扑都支持。

对于通讯协议，CANopen 被直接移植到 EtherCAT 总线上使用，称为 COE（CANopen over EtherCAT）。对于电机控制，CANopen 下的 DS402 协议被顺理成章地使用。在 CAN 总线上数据传输速度不高、节点不能多、同步性一般的问题都被迎刃而解。

柔性电机驱动系统

致远电子推出柔性电机驱动系统，能满足个性化的接口需求。该系统采用 EtherCAT 为网络，从站模块面积 59mm*50mm，采用 40 脚标准排针接口。电机驱动、I/O、模拟采集、电源、耦合器等多种模块选择。用户按需制作分线底板即可，从站板插在底板上通过 EtherCAT 网络级联，最多支持 255 个。

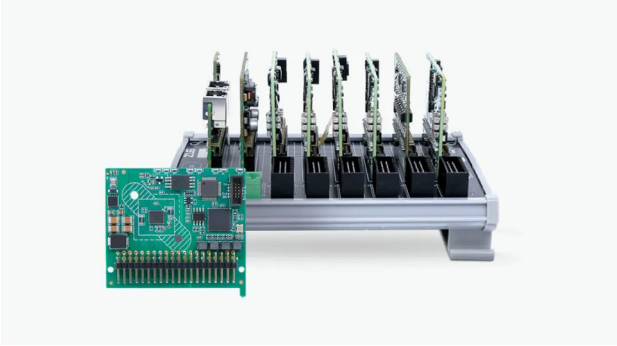


图3

ZMD-E1050ETC 步进电机驱动模块

- 24V~48V/1.5A 的两相步进电机；
- 1/2/4/8/16/32/64/128/256 细分选择；
- CiA402 运动控制协议，支持 CSP/PP/HM/PV 模式；
- 3 个数字输入，1 组 ABZ 正交编码输入；
- 开环或编码器闭环控制可选。

理论上只要底板做得足够大，就可以串接数百个步进电机驱动模块。但是为了提高可靠性和易用性，每个底板通常只安装 10~50 个电机模块或其他功能的模块；并通过 ZPWB-240302 电源模块给 EtherCAT 部分独立供电；然后通过 ZCPB-80600 EtherCAT 耦合器，用网线接通上级和下级的底板；最后形成数百个的电机驱动系统。

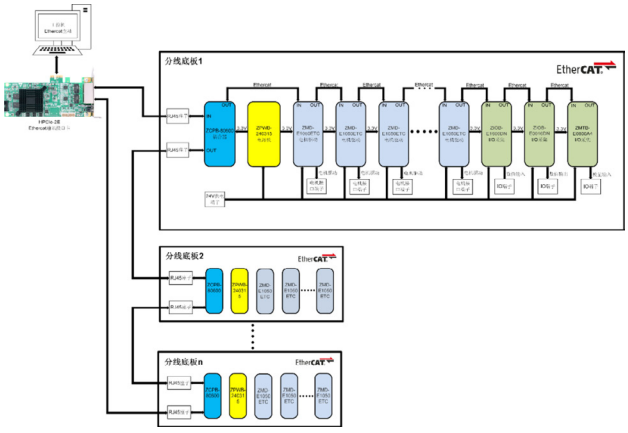


图4

后面的专题，我们将剖析 DS402 协议中的两种位置控制模式 CSP 和 PP，如何在柔性电机驱动系统中发挥作用。敬请留意。

表1 电机模组选型表

型号	总线类型	电机类型	电源电压	电机数	电流
ZMD-E1050ETC	EtherCAT	两相步进	24V	1	1.5A

表2 I/O模组选型表

型号	总线类型	模组类型	通道数量	接口特征
ZIOB-E1600DN	EtherCAT	数字输入	16	NPN，0~7V
ZIOB-E1600DP	EtherCAT	数字输入	16	PNP，11~30V
ZIOB-E0016DN	EtherCAT	数字输出	16	NPN，单路0.5A
ZIOB-E0016DP	EtherCAT	数字输出	16	PNP，单路0.5A
ZIOB-E0800AI	EtherCAT	模拟输入	8	电流型，4~20mA
ZIOB-E0800AU	EtherCAT	模拟输入	8	电压型，0~10V
ZIOB-E0800AU1	EtherCAT	模拟输入	8	电压型，2x(0~10V)+6x(-10~10V)
ZIOB-E0008AU	EtherCAT	模拟输出	8	电压型，0~10V
ZCPB-80600	EtherCAT	耦合器	-	IN/OUT 双网口
ZPWB-240302	-	通讯电源	-	EtherCAT 模组电源，3.3V/2.5A

另外我们还提供配套的各种模块，详细资料请看：
https://manual.zlg.cn/web/#/333/13047

【200个电机驱动】如何实现EtherCAT分布式供电?

ZLG 致远电子 2024-05-17 11:43:44

致远电子推出的插板式模组系统，通过分线底板轻松实现 EtherCAT 网络级联，最多支持 255 个！那如此多的模组该如何供电？本文来教你如何使用电源模组实现分布式供电！

插板式模组

为满足个性化大型系统的控制需求，致远电子推出了插板式电机驱动模组。该系统采用 EtherCAT 总线，从站模组尺寸 59mm*50mm，采用 40 脚标准排针接口。用户按需制作分线底板，从站板插在底板上通过 EtherCAT 网络级联，最大支持 255 个节点。

- 以不变应万变，只需最少的工作量，按需制作分线底板，适应千变万化的布线要求。
- 小体积大系统，在最小的体积下集成最多的从站，实现大型系统的控制。
- 高精度快布局，基于 EtherCAT，实现高精度分布控制，以及即插即用快速布局。

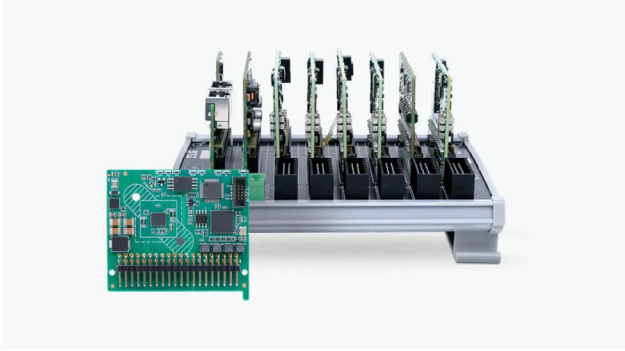


图1 插板式电机驱动模组

各个模组的主要参数如下：

步进电机驱动模组

- 24V~48V/1.5A~3A 的两相步进电机；
- 1/2/4/8/16/32/64/128/256 细分选择；
- CiA402 运动控制协议，支持 CSP/PP/HM/PV 模式；
- 3 个数字输入，1 组 ABZ 正交编码输入；
- 开环或编码器闭环控制可选。

I/O采集模组

- 数值输入，24V 电平，3ms 消抖，16 通道，高低电平可选；
- 数值输出，24V 电平，单路 0.5A，16 通道，高低电平可选；
- 模拟输入，12bit，8 通道，电流型 / 电压型可选；
- 模拟输出，12bit，8 通道，电压型。

电源模组

- 24VDC 输入，3.3VDC/2.5A 输出。

电源模组的使用

插板式电机驱动模组可以插接多个从站模组，轻松实现数百个电机驱动的同步控制。但如此多的模组对 EtherCAT 通讯用的 3.3V 电源要求很高，一方面功率要足够大（250mA 左右 / 模组），另一方面纹波等干扰要尽量低。插板式模组配套的电源模组 ZPWB-240302，是专门给 EtherCAT 通讯提供高效稳定的电源。

步进电机驱动模组、I/O 采集模组、电源模组是分布式的插接在分线底板上的。其中电源模组输入 24VDC，输出 3.3VDC/2.5A，该电源模块主要为步进电机驱动模组和 I/O 采集模组的 EtherCAT 电路部分提供 3.3V 供电，整个系统的分布框图如下图所示，整个系统的 EtherCAT 电路电源由 ZPWB-240302 电源模组提供，单个模组的 EtherCAT 电路最大需要 250mA 电流供电，即 ZPWB-240302 电源模组最多支持 10 个模组供电，如图中的 ZCPB-80600 耦合器后面接了一块 ZPWB-240302 电源模组，那该电源模组后面最多插接包括耦合器在内的 10 个步进电机驱动模组或 I/O 采集模组，插满 10 个模组后需要再插接一块 ZPWB-240302 供电，给后续其他模组提供电流。分线底板是按需制作的，如果每个底板多于 10 个模组，需要插入更多的电压模组供电。

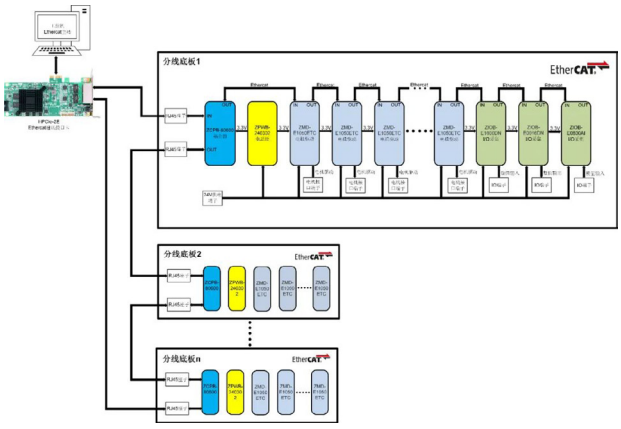


图2 分线底板

电源模组的接口说明

信号排针由 40 脚间距 2.54mm 的标准排针组成，如表 1 所示。一般需要根据此信号排针制作底板。

表1 引脚定义说明

引脚		信号		说明
1	2	3.3VETC	3.3VETC	控制电源 3.3V 正极
3	4	GNDETC	GNDETC	控制电源 3.3V 负极
5	6	NC	NC	一般底板要将 5 和 6、7 和 8 分别短路，将耦合器 OUT 发送和从站 IN 接收的差分信号连通
7	8	NC	NC	
9	10	GNDETC	GNDETC	控制电源 3.3V 负极
11	12	NC	NC	一般底板要将 11 和 12、13 和 14 分别短路，将耦合器 IN 接收和从站 OUT 发送的差分信号连通
13	14	NC	NC	
15	16	GNDETC	GNDETC	控制电源 3.3V 负极
17	18	NC	NC	
19	20	NC	NC	
21	22	NC	NC	
23	24	NC	NC	
25	26	NC	NC	
27	28	NC	NC	
29	30	NC	NC	
31	32	NC	NC	
33	34	GNDUp	GNDUp	驱动电源
35	36	GNDUp	24VUp	24VUp：24V 正极
37	38	24VUp	24VUp	GNDUp:24V 负极
39	40	SGND	SGND	SGND:一般悬空，或连接金属外壳

电源模组的经典接线图

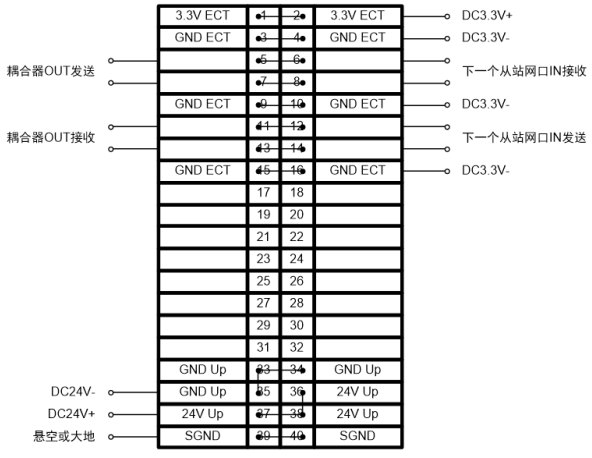


图3 电源模组的经典接线图

上图显示了电源模组的引脚定义及连接方法。下面是各种模组的选型表。

表2 电机模组选型表

型号	总线类型	电机类型	电源电压	电机数	电流
ZMD-E1050ETC	EtherCAT	两相步进	24V	1	1.5A

表3 I/O模组选型表

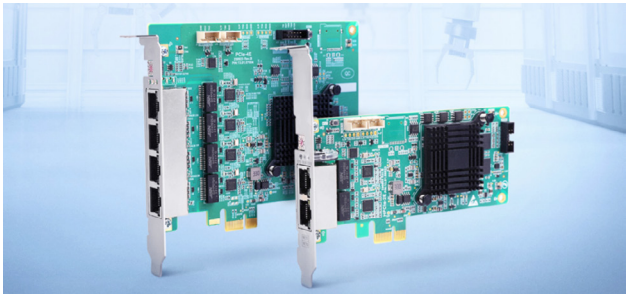
型号	总线类型	模组类型	通道数量	接口特征
ZIOB-E1600DN	EtherCAT	数字输入	16	NPN，0~7V
ZIOB-E1600DP	EtherCAT	数字输入	16	PNP，11~30V
ZIOB-E0016DN	EtherCAT	数字输出	16	NPN，单路0.5A
ZIOB-E0016DP	EtherCAT	数字输出	16	PNP，单路0.5A
ZIOB-E0800AI	EtherCAT	模拟输入	8	电流型，4~20mA
ZIOB-E0800AU	EtherCAT	模拟输入	8	电压型，0~10V
ZIOB-E0800AU1	EtherCAT	模拟输入	8	电压型，2x(0~10V)+6x(-10~10V)
ZIOB-E0008AU	EtherCAT	模拟输出	8	电压型，0~10V
ZCPB-80600	EtherCAT	耦合器	-	IN/OUT 双网口
ZPWB-240302	-	通讯电源	-	EtherCAT 模组电源，3.3V/2.5A

插板式模组的详细信息请关注：
https://manual.zlg.cn/web/#/333/13047

【技术分享】使用PCIe EtherCAT通讯卡控制IO从站step by step（一）

ZLG 致远电子 2024-05-20 11:38:33

EtherCAT 是一种高效且常用的工业通讯协议。本系列文章将带领您使用 ZLG 致远电子的 PCIe EtherCAT 通讯卡，一步步实现从搭建编译环境到程序运行。



ZLG 致远电子 **PCIe EtherCAT 通讯卡** 是一款高性能的总线控制卡，专为满足高实时的工业控制需求而设计。基于 PCI Express 技术，该卡通过使用实时内核 + 商业授权 EtherCAT 协议栈的方式，实现了高达 125us 的控制周期。

为了方便用户的使用，该卡配套提供了全面的上位机配置软件 AWStudio，用户通过使用 AWStudio 软件，可快速扫描从站设备，轻松地得到 ENI 的配置文件。

此外，该卡还支持 Windows、Linux、Vxworks 等多平台，并提供了简洁易用的 SDK 接口。无论是在复杂的工业控制中，还是在追求高性能的场景中，ZLG 致远电子的 PCIe EtherCAT 通讯卡都是您的理想选择。

运行框架：

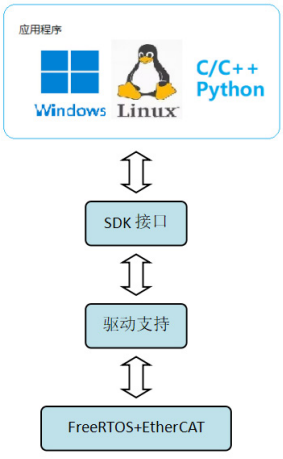


表1 PCIe EtherCAT通讯卡型号

型号	结构	EtherCAT	线路冗余	热插拔	CAN/CAN FD	DI/DO	编码器	PWM
HPCIe-2E	PCIe 半高	1 路主站，2 网口	支持	支持	--	--	--	--
PCIe-2E	PCIe 全高	1 路主站，2 网口	支持	支持	--	--	--	--
PCIe-4E	PCIe 全高	2 路主站，4 网口	支持	支持	--	--	--	--
MiniPCIe-2E	MiniPCIe	1 路主站，2 网口	支持	--	--	--	--	--

搭建步骤

为了优化客户使用 PCIe EtherCAT 通讯卡的体验，我们将详细介绍在 Ubuntu 环境下开发项目搭建环境的步骤，以及如何快速上手使用 PCIE 卡和 IO 从站进行通讯。这将使您能够快速而轻松地将 ZLG 致远电子的 PCIe EtherCAT 通讯卡接入到您的工程项目中。

1. 前期准备

1.1 在目标机器上安装驱动；

```
robot@robot-zlg:~$ ls /dev | grep zpcidev
zpcidev
zpcidev0
```

1.2 获得 SDK 包；

1.3 使用 AWStudio 导出 EtherCAT 网络信息文件（ENI）。

2. 项目创建

2.1 构建工程目录

新建 Project 文件夹。

```
vmuser@ubuntu:~$ mkdir -p Project/src
```

将 sdk 移动到工程目录中。

```
vmuser@ubuntu:~$ cp -r sdk/ Project
```

移动完后，工程目录 Project 将会是下图的样子。

```
vmuser@ubuntu:~$ tree Project/
Project/
├── sdk
│   ├── include
│   │   ├── pci_dbg.h
│   │   ├── pci_errno.h
│   │   └── pci_zecm.h
│   ├── shared
│   │   └── libpci_zecm.so
│   └── staticlib
│       └── libpci_zecm.a
└── src
```

2.2 CMakeLists.txt 编写

在 Project 目录下，创建 CMakeLists.txt 文件。

```
vmuser@ubuntu:~$ cd Project
vmuser@ubuntu:~/Project$ toucd CMakeLists.txt
```

在 CMakeLists.txt 文件中输入。

```
M CMakeLists.txt
1   cmake_minimum_required(VERSION 3.10)
2
3   project(Demo)
4
5   set(SRC ./src/ecat_api_io_test.cpp) #将需要编译的源码保存到SRC变量中
6   set(TARGET_NAME demo) #设置生成的app名字为demo
7   set(ZPCIE_SDK_PATH ./sdk) #设置sdk路径
8
9   #将库的相对路径转换为绝对路径
10  get_filename_component(ZPCIE_SDK_PATH ${ZPCIE_SDK_PATH} ABSOLUTE)
11
12  add_executable(${TARGET_NAME} ${SRC})
13  #设置包含路径
14  target_include_directories(${TARGET_NAME} PUBLIC ${ZPCIE_SDK_PATH}/include)
15  target_link_libraries(${TARGET_NAME} ${ZPCIE_SDK_PATH}/shared/libpci_zecm.so)
```

2.3 创建测试程序

在 src 目录中创建主程序 ecat_api_io_test.cpp, 该文件名需要和 CMakeLists.txt 中的 SRC 变量相同。

```
vmuser@ubuntu:~/Project$ cd src
vmuser@ubuntu:~/Project/src$ touch ecat_api_io_test.cpp
```

打开 ecat_api_io_test.cpp 文件。输入内容：

```
vmuser@ubuntu:~$ cd Project
```

```
#include <iostream>
#include <chrono> // 用于 sleep
#include <thread> // 用于 sleep
#include "pci_errno.h"
#include "pci_zecm.h"
#include "pci_dbg.h"

int32_t testDemo(int alias, int channel, const char* fileName){
int result = 0;
char buff[256];
    ECAT_HANDLE hHandle;
    // 初始化 hHandle 句柄
     EXIT_IF_FAIL(EcatOpen(&hHandle, BOARD_ALIAS(buff, alias), channel));
// 启动主站
    EXIT_IF_FAIL(EcatBusRun(hHandle, fileName));
// 将状态切换为 8(Operational)
      EXIT_IF_FAIL(EcatRequestMasterState(hHandle, EcatStateO));
EXIT_IF_FAIL(EcatClose(hHandle));
    return result;
}

int main(int argc, char* argv[]){
  ECAT_HANDLE hHandle;
  char buff[256];
  uint32_t channel = 0, alias = 0;
  std::string eniFile;
  if (argc != 4){
      std::cout << "usage: " << argv[0] << " encoder_id channel eni.xml" << std::endl;
      return 1;
  }
  alias = atoi(argv[1]);
  channel = atoi(argv[2]);
  eniFile = argv[3];
  if (channel > 1){
    channel = 1;
  }
  testDemo(alias, channel, eniFile.c_str());
  return 0;
}
```

上面的示例代码实现了主站的启动以及关闭，具体的过程数据修改将在下一章节做介绍。

2.4 测试编译环境

构建项目，先去到项目的根目录。

```
vmuser@ubuntu:~/Project$ mkdir build
```

创建构建的文件夹 build。

```
vmuser@ubuntu:~/Project$ cd build
```

进入到文件夹 build。

```
vmuser@ubuntu:~/Project/build$ cmake ..
```

执行 cmake 构建项目。

```
vmuser@ubuntu:~/Project/build$ cmake --build .
```

编译 (编译完成后，生成的执行程序将会在 build 目录下，可执行程序的名字为 CMakeLists.txt 中 TARGET_NAME 变量指定的名字)。

【产品应用】使用PCIe EtherCAT通讯卡控制IO从站step by step（二）

ZLG 致远电子 2024-05-23 11:40:46

ZLG 致远电子的 PCIe EtherCAT 通讯卡该如何使用？ PDO 过程数据该如何操作？具体编程又该如何实现？续接上一章节，本文将为您详细讲解。

EtherCAT 工业总线技术在工业自动化领域展现出了广泛的应用价值，特别是在运动控制、机器人技术和测量技术等方面。ZLG 致远电子 **PCIe EtherCAT 通讯卡** 基于自主知识产权的系统之上开发，实现了软硬件间的无缝连接，极大地提升了系统的稳定性、可靠性以及安全性。同时，该通讯卡还支持线路冗余以及热插拔功能，可轻松实现多轴同步控制和数据的高速传输。

此外，ZLG 致远电子 PCIe EtherCAT 通讯卡还为用户提供了便捷的二次开发库，支持 VC、C#、Linux、Python 等各类主流开发环境，满足客户不同层次的开发需求。

值得一提的是，ZLG 致远电子 PCIe EtherCAT 通讯卡通过将商业级 EtherCAT 主站协议和实时内核相结合的方式，有效释放主机资源，完美解决传统 EtherCAT 主站在非实时操作系统下运行所带来的各类问题，为用户带来了更加高效、稳定的解决方案。

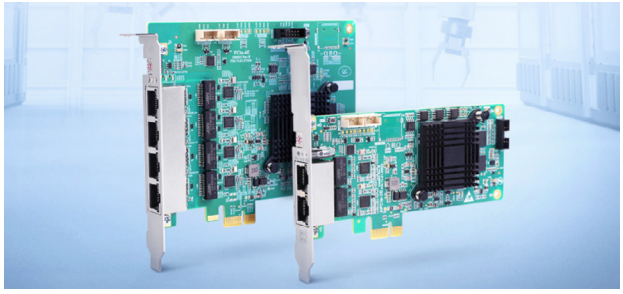


图1 PCIe EtherCAT通讯卡

表1 PCIe EtherCAT通讯卡型号

型号	结构	EtherCAT	线路冗余	热插拔
HPCIe-2E	PCIe 半高	1 路主站，2 网口	支持	支持
PCIe-2E	PCIe 全高	1 路主站，2 网口	支持	支持
PCIe-4E	PCIe 全高	2 路主站，4 网口	支持	支持
MiniPCIe-2E	MiniPCIe	1 路主站，2 网口	支持	--

基于上一章**《使用 PCIe EtherCAT 通讯卡控制 IO 从站 step by step（一）》**中所讲诉的内容，我们已经完成了开发环境的搭建，以及主函数的建立，接下来，我们将会进一步完善主函数中的代码，实现对从站的 PDO 数据读写。

1. 代码编写

打开 ecat_api_io_test.cpp 文件。根据 AWStudio 软件导出的 eni 文件定义过程数据的结构体，打开 eni 文件，移动光标到文件尾部，找到注释的节点 ENI_PDO_LIST。

```
<!-- @@##ENI_PDO_LIST##@@
{
    "slaves" : [
        {
            "name" : "Slave 1 (EL6695-0002)",
            "vendor_id" : "2",
            "vendor_name" : "Beckhoff Automation GmbH & Co. KG",
            "type" : "EL6695 EtherCAT Bridge terminal (Secondary)",
            "product_code" : "438775890",
            "revision_no" : "393218",
            "product_revision" : "EL6695-0002-0006",
            "phys_addr" : "1001",
            "auto_inc_addr" : "0",
            "outputs" : [
                {
                    "channel" : "IO Inputs",
                    "name" : "DI_1",
                    "index" : "28672",
                    "subindex" : "1",
                    "type" : "ARRAY [0..0] OF BYTE",
                    "bit_size" : "8",
                    "bit_offset" : "0"
                },
                {
                    "channel" : "IO Inputs",
                    "name" : "AI_1",
                    "index" : "28672",
                    "subindex" : "2",
                    "type" : "USINT",
                    "bit_size" : "8",
                    "bit_offset" : "8"
                },
                {
                    "channel" : "IO Inputs",
                    "name" : "AI_2",
                    "index" : "28672",
                    "subindex" : "3",
                    "type" : "INT",
                    "bit_size" : "16",
                    "bit_offset" : "16"
                }
            ],
            "inputs" : [
                {
```

图2 ENI文件(1)

```
            "inputs" : [
                {
                    "channel" : "IO Outputs",
                    "name" : "DO_1",
                    "index" : "24576",
                    "subindex" : "1",
                    "type" : "ARRAY [0..0] OF BYTE",
                    "bit_size" : "8",
                    "bit_offset" : "0"
                },
                {
                    "channel" : "IO Outputs",
                    "name" : "AO_1",
                    "index" : "24576",
                    "subindex" : "2",
                    "type" : "USINT",
                    "bit_size" : "8",
                    "bit_offset" : "8"
                },
                {
                    "channel" : "IO Outputs",
                    "name" : "AO-2",
                    "index" : "24576",
                    "subindex" : "3",
                    "type" : "INT",
                    "bit_size" : "16",
                    "bit_offset" : "16"
                }
            ]
        }
    ]
} -->
```

图3 ENI文件(2)

节点中的 inputs 为从站返回的 PDO 过程数据，outputs 为主站发送到从站的 PDO 过程数据，根据每个变量的位宽 bit_size，我们可以定义对应的类型。

根据图 2 的信息，我们可以看出输出有三个变量，DI_1,AI_1,AI_2, 长度分别为 8 位，8 位，16 位，输入有三个变量 DO_1,AO_1,AO_2, 长度分别位 8 位，8 位，16 位，定义结构体：

```
// 过程数据，写入从站的数据格式
typedef struct{
  uint8_t DI_1;
  uint8_t AI_1;
  uint16_t AI_2;
}PDO_OUTPUTS_T;

// 过程数据，从站返回的数据格式
typedef struct{
  uint8_t DO_1;
  uint8_t AO_1;
  uint16_t AO_2;
}PDO_INPUTS_T;
```

定义完过程数据的结构体后。

第一步，输入需要控制的 pcie 卡别名以及通道号，获取 Ecat 控制句柄。

```
EXIT_IF_FAIL(EcatOpen(&hHandle, BOARD_ALIAS(buff, alias), channel));
```

第二步，输入上位机程序导出的 eni 文件，启动 Ecat 主站。

```
EXIT_IF_FAIL(EcatBusRun(hHandle, fileName));
```

第三步，将 Ecat 主站状态切换为 8（Operational）。

```
EXIT_IF_FAIL(EcatRequestMasterState(hHandle, EcatStateO));
```

等待主站切换状态。

```
uint8_t query = EcatStateNotSet;
do{
  EXIT_IF_FAIL(EcatGetMasterState(hHandle, &query));
  // 输出当前状态
   _DBG_("request_state=%d, query_state=%d", EcatStateO, query);
  if (query == EcatStateO){
    break;;
  }
  std::this_thread::sleep_for(std::chrono::seconds(1));
}while(1);
```

第四步，定义 PDO 过程数据的指针并指向本地缓存空间，这一步将会让我们更加方便快捷地读写 PDO 数据。

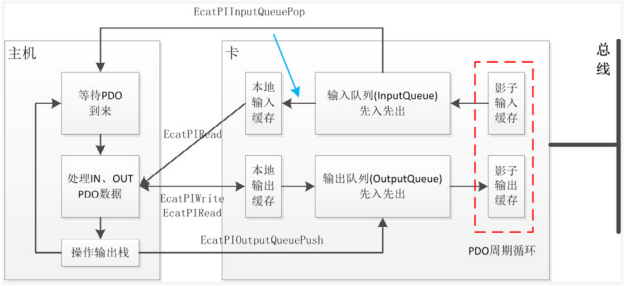


图4 PDO数据的收发原理

执行 EcatPINMap 函数，将会修改第 3 个参数 inputBuff,outputBuff 的指向，让其直接指向本地输入输出缓存区，固不需要再额外申请空间。

```
PDO_OUTPUTS_T *outputBuff;
PDO_INPUTS_T *inputBuff;
// 将指针 inputBuff,outputBuff 分别指向本地缓存的空间
EXIT_IF_FAIL(EcatPINMap(hHandle, PI_AREA_LOCAL_INPUT, (void**)&inputBuff));
EXIT_IF_FAIL(EcatPINMap(hHandle, PI_AREA_LOCAL_OUTPUT, (void**)&outputBuff));
```

第五步，向发送队列中添加空数据，添加空数据的数量取决于 PC 系统抖动的程度，抖动越小，添加的空数据越少，控制指令的滞后性越小。

```
for(auto i = 0; i < 2; i++){
  EcatPIOutputQueuePush(hHandle, false, 100);
}
```

第六步，使能过程数据 PDO 通信。

```
EXIT_IF_FAIL(EcatPIEnable(hHandle));
```

第七步，通过 EcatPIInputQueuePop 接口，等待从站数据返回，然后读写 PDO 过程数据，最后调用 EcatPIOutputQueuePush 接口将数据添加到发送队列。当前例子中，在收到从站返回的数据后，主站会将收到的输入数据写到输出的变量。

```
    bool loopFlag = true;
    while(loopFlag){
      // 阻塞式等待 PDO 数据
      if (!EcatPIInputQueuePop(hHandle, false, 100)){
        /*************************************************************/
        // 修改过程数据
          printf("0x%02x, 0x%02x, 0x%04x\r\n", inputBuff->DO_1,
inputBuff->AO_1, inputBuff->AO_2);
        outputBuff->DI_1 = inputBuff->DO_1;
        outputBuff->AI_1 = inputBuff->AO_1;
        outputBuff->AI_2 = inputBuff->AO_2;
        /*************************************************************/
        // 将数据添加到 PDO 的发送队列中
        if (EcatPIOutputQueuePush(hHandle, false, 100)){
          _ERR_("PI push error.");
          break;
        }
      }
    }
```

第八步，释放句柄。

```
    EXIT_IF_FAIL(EcatClose(hHandle));
```

完整的 ecat_api_io_test.cpp 文件。

```
    #include <iostream>
    #include <chrono> // 用于 sleep
    #include <thread> // 用于 sleep
    #include "pci_errno.h"
    #include "pci_zecm.h"
    #include "pci_dbg.h"

    // 过程数据，写入从站的数据格式
    typedef struct{
      uint8_t DI_1;
      uint8_t AI_1;
      uint16_t AI_2;
    }PDO_OUTPUTS_T;

    // 过程数据，从站返回的数据格式
    typedef struct{
      uint8_t DO_1;
      uint8_t AO_1;
      uint16_t AO_2
    }PDO_INPUTS_T;
```

```
    int32_t testDemo(int alias, int channel, const char* fileName){
      int32_t result = 0;
      char buff[256];
      ECAT_HANDLE hHandle;
      // 初始化 hHandle 句柄
      EXIT_IF_FAIL(EcatOpen(&hHandle, BOARD_ALIAS(buff, alias),
channel));
      // 启动主站
      EXIT_IF_FAIL(EcatBusRun(hHandle, fileName));
      // 将状态切换为 8(Operational)
        EXIT_IF_FAIL(EcatRequestMasterState(hHandle,
EcatStateO));

      // 等待主站切换状态
      uint8_t query = EcatStateNotSet;
      do{
        EXIT_IF_FAIL(EcatGetMasterState(hHandle, &query));
        // 输出当前状态
         _DBG_("request_state=%d, query_state=%d", EcatStateO,
query);
        if (query == EcatStateO){
          break;;
        }
        std::this_thread::sleep_for(std::chrono::seconds(1));
      }while(1);

      PDO_OUTPUTS_T *outputBuff;
      PDO_INPUTS_T *inputBuff;
      // 将指针 inputBuff,outputBuff 分别指向本地缓存的空间
       EXIT_IF_FAIL(EcatPINMap(hHandle, PI_AREA_LOCAL_INPUT,
(void**)&inputBuff));
        EXIT_IF_FAIL(EcatPINMap(hHandle, PI_AREA_LOCAL_
OUTPUT, (void**)&outputBuff));

      // 向发送队列中添加空数据，添加空数据的数量取决于 PC 系
统抖动的程度，抖动越小，添加的空数据越少，控制指令的滞后性越
小
      for(auto i = 0; i < 2; i++){
        EcatPIOutputQueuePush(hHandle, false, 100);
      }
      // 使能过程数据 PDO 通信
      EXIT_IF_FAIL(EcatPIEnable(hHandle));

      bool loopFlag = true;
      while(loopFlag){
        // 阻塞式等待 PDO 数据
        if (!EcatPIInputQueuePop(hHandle, false, 100)){
          /*************************************************************
*/
          // 修改过程数据
```

```
                printf("0x%02x, 0x%02x, 0x%04x\r\n", inputBuff->DO_1,inputBuff->AO_1, inputBuff->AO_2);
                outputBuff->DI_1 = inputBuff->DO_1;
                outputBuff->AI_1 = inputBuff->AO_1;
                outputBuff->AI_2 = inputBuff->AO_2;
                /*************************************************************/
                // 将数据添加到 PDO 的发送队列中
                if (EcatPIOutputQueuePush(hHandle, false, 100)){
                  _ERR_("PI push error.");
                  break;
                }
              }
            }

            // 释放句柄
            EXIT_IF_FAIL(EcatClose(hHandle));
            return result;
        }

        int main(int argc, char* argv[]){
            ECAT_HANDLE hHandle;
            char buff[256];
            uint32_t channel = 0, alias = 0;
            std::string eniFile;
            if (argc != 4){
                std::cout << "usage: " << argv[0] << " encoder_id channel eni.xml" << std::endl;
                return 1;
            }
            alias = atoi(argv[1]);
            channel = atoi(argv[2]);
            eniFile = argv[3];
            if (channel > 1){
                channel = 1;
            }
            testDemo(alias, channel, eniFile.c_str());
            return 0;
        }
```

编译

```
vmuser@ubuntu:~/Project/build$ cmake --build .
```

运行测试

```
vmuser@ubuntu:~/Project/build$ ./demo 0 0 eni.xml
```

输出，程序将持续打印从站的输入状态

```
0x00, 0x00, 0x0001
0x00, 0x00, 0x0001
0x00, 0x00, 0x0001
0x00, 0x00, 0x0001
0x00, 0x00, 0x0002
0x00, 0x00, 0x0002
0x00, 0x00, 0x0002
0x00, 0x00, 0x0002
0x00, 0x00, 0x0002
0x00, 0x00, 0x0002
0x00, 0x00, 0x0002
```

但真正开发的时候，建议将打印信息等耗时的操作注释后再编译，否则，程序将可能会因为打印动作耗时过长而导致主机无法快速填充 pdo 数据，最终将产生控制抖动等问题。

互联互通▼

【技术分享】CAN-bus应用笔记：节点篇

ZLG 致远电子 2024-05-11 11:40:00

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CAN总线节点数的因素

影响总线节点数的因素有多种，本文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 1 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。

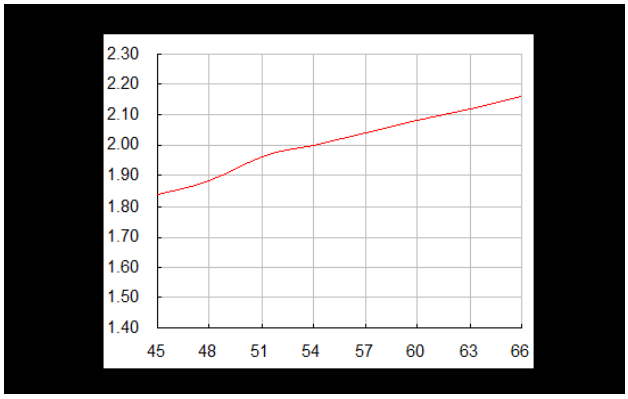


图1 不同负载下的差分电压

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 1 测试的范围内波动。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，如图 2 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，可以看到电源电压升高 0.5V，差分电压幅值会升高约 0.3V。

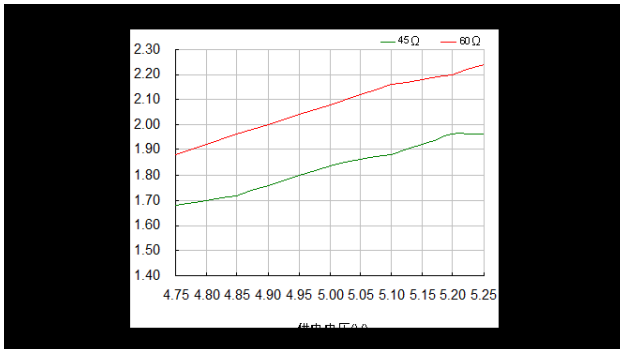


图2 不同供电电压下的差分电压

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 接口典型参数如表 1 所示。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。

表1 CAN接口典型参数

类别	最小值	典型值	最大值
输出显性电平	1.5V	--	3V
输入显性电平	0.9V	--	8.0V
输入隐性电平	-3V	--	0.5V

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 3 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。画出其等效电路如图 4 所示。

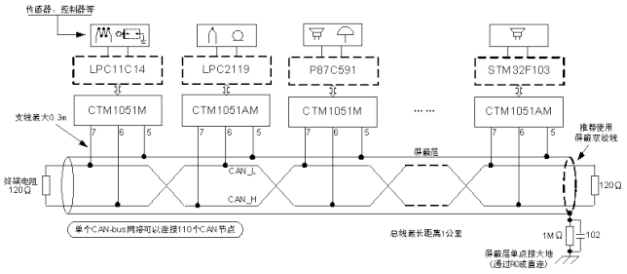


图3 CTM1051M推荐组网

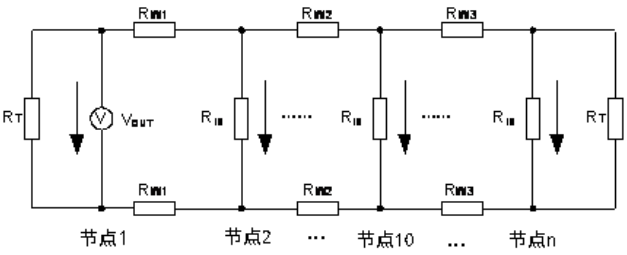


图4 CTM1051M组网等效电路

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 4 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 4 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。

总结

无论总线网络长短，网络两端都需要加终端电阻；

通讯距离长时，适当增加终端电阻值，减少总线电阻对信号的衰减，如 150Ω~300Ω；

有强烈干扰的场合使用屏蔽双绞线，屏蔽层单点接大地；

收发器 CAN 接口输出的差分电平会随着供电电压的变化而变化，应确保供电电压在手册规定范围内。

高可靠性CAN-bus隔离收发器模块

在户外或其他复杂的工业现场环境，除了要使用 CAN 总线隔离收发器进行信号隔离传输，还需要考虑如何避免雷击、浪涌、过压等有害信号对信号传输系统的影响。针对这类恶劣的应用环境，致远电子推出新一代高防护等级隔离 CAN 收发器 CTM1051HP，在普通的隔离收发器基础上集合了总线保护器的功能，可有效避免雷击、浪涌、过压对 CAN 总线产生的影响，大大加强总线的可靠性。且 CTM1051HP 与常规 CAN 隔离收发器相比体积不变，可广泛应用于各类高体积要求或紧凑型产品中，亦可作为现有 CAN 隔离收发电路的优化方案。

互联互通▼

【CAN总线知识】为什么主机厂愈来愈重视CAN一致性测试?

ZLG 致远电子 2024-05-11 11:40:00

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。

CAN 一致性测试，就是要求整车 CAN 网络中的节点都满足 CAN 总线节点规范要求，缩小 CAN 网络中节点差异，保证 CAN 网络的环境稳定，有效提高 CAN 网络的抗干扰能力。

那主机厂为什么愈来愈重视 CAN 一致性测试呢?

整车CAN网络架构

以往的传统车的 CAN 总线网络节点较少，如仪表、发动机 ECU 等。但随着新能源汽车行业发展，整车 CAN 网络中的节点演变得极为复杂，现在新能源汽车内部 CAN 节点已经高达 60 个，细分为多个 CAN 网络系统，如车身部含有空调、车门、导航等节点，安全系统又含有气囊、引爆管等节点。

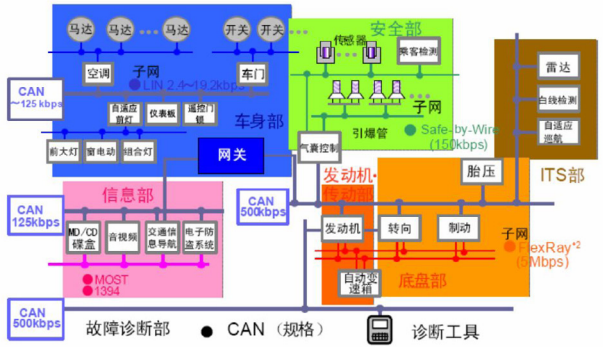


图1 整车复杂的CAN网络

CAN总线不一致的危害

复杂的 CAN 网络，各个节点质量良莠不齐会对 CAN 总线网络存在较大的安全隐患，通常会因为其中某一个节点的错误进而影响整体总线正常运行，乃至导致整体总线的瘫痪。

1. 总线瘫痪

比如一个 CAN 网络包含节点 A、B、C，节点 A 差分电压是 1.2V，而节点 B 的差分电压是 2.0V，节点 C 差分电压是 1.8V。当整车 CAN 网络工作在强电磁干扰的环境下，环境的共模干扰串扰到 CAN 总线中会使节点 A 的差分电压影响到 0.9V 以下，导致节点从显性电平翻转成为隐性电平，进而导致了节点 A 工作故障，频繁发出错误帧。在 CAN 总线中，错误帧虽然不被接收，但是依然占用总线传输时间，所以导致其他正常节点发送延迟或者无法发送，影响整车 CAN 总线正常运行环境。

解决方案: 主机厂必须要求节点 A、B、C 的工作电压必须要工作在 1.8V，乃至 2.0V，这个问题便得以解决。

图2 错误帧占用总线

2. 波特率不一致导致CAN网络系统死机

位时间（位宽）和波特率是 CAN 总线通讯的最基本要素。位时间 =1/波特率，比如波特率是 500k，那位时间是 2us。在相同的 CAN 总线采样频率下，当某一个节点的位时间发生抖动时，即位时间为 1.8us 或者 2.2us，将导致采样点的逻辑判断出现异常，出现总线错误，导致 CAN 网络系统死机。

解决方案: 在 CAN 网络准入阶段，如果对接入网络的节点进行规范化，每个节点的位时间必须满足 t=2us±3%，那么 CAN 网络的位时间将高度一致，则可以从 CAN 总线物理层进行规避该问题。

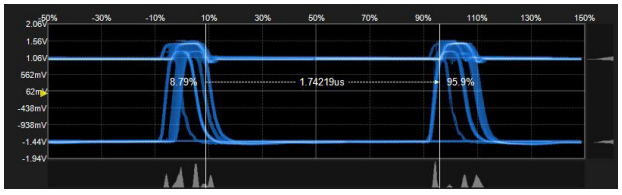


图3 波特率抖动导致位时间变化

3. 显性阈值电平错误判断导致整车网络故障

通常而言，CAN 总线判断显隐性的机制如下：在差分电平大于 0.9V 时，为显性电平；而在小于 0.5V 时，为隐性电平，其中在 05V 至 0.9V 之间为不确定区域。但在实际网络中，CAN 总线网络中某一节点在差分电平为 0.9V 时，依然判断为隐性，则出现位逻辑判断错误，进而导致节点发出错误帧，使总线陷入网络故障状态。

解决方案: 如在 CAN 网络节点准入阶段，对每个节点进行显性阈值测试，利用电压源将差分电压升高至 0.9V，，保证所有节点在此差分电压都能判断为显性，并且停止发送报文，将减少该总线故障问题出现，并且减轻 CAN 总线网络调试的工作量。

差分电平幅值	识别成的逻辑值
>0.9V	显性电平（0）
0.5~0.9V	不确定区域
<0.5V	隐形电平（1）

图4

CAN一致性测试内容及解决方案

1. CAN一致性测试内容

在国内，大部分的主机厂都有 CAN 总线网络测试规范，主要内容包括物理层、链路层以及应用层。

- 物理层：通信介质的物理特性，如幅值、边沿时间等，是最重要的部分。
- 链路层：规定在介质上传输的排列和组织，如帧结构；
- 应用层：在用户、软件、网络终端之间进行信息交换，客户自定义内容较多。

物理层在介质的物理特性方面的规定，主要源自于标准 ISO 11898，该部分标准高度一致。而链路层和应用层方面，因为主机厂的整车网络设计不同，其应用层测试各不一致。

序号	物理层测试	序号	网络管理层测试
1	隐性输出电压	14	CAN 协议一致性
2	显性输出电压	15	扩展报文帧(CAN 2.0B Passive)的兼容性
3	位上升/下降时间	16	100%总线负载下的报文接收
4	位时间精度	17	短时突增报文的接收
5	信号对称性	序号	应用层测试
6	过压情况下的报文发送持续时间	18	接收报文数据长度
7	内部延迟	19	发送报文数据类型
8	地偏移	20	发送报文 ID
9	通信电压范围	21	发送报文 DLC
10	欠压情况下的报文发送持续时间	22	未使用字节填充规则
11	欠压恢复情况下的报文发送启动时间	23	信号缺省值
12	过压恢复情况下的报文发送启动时间	24	发送周期容差
13	ESD 保护	25	事件型报文的发送延迟时间

图5 CAN一致性测试内容（节选）

2. CAN一致性测试方案

CAN 一致性测试工具包含 CAN 卡、示波器、电源等设备，当前国内 CAN 总线工程师只能采用手动搭建测试平台并进行逐项测试，主要通过 CAN 卡采集报文数据，以及通过示波器进行测试波形，进而达到测试位时间、幅值、位宽等目的，但是测试方案效率较低，一般完成整体 CAN 一致性测试项目需要 10 小时乃至 1 天时间。

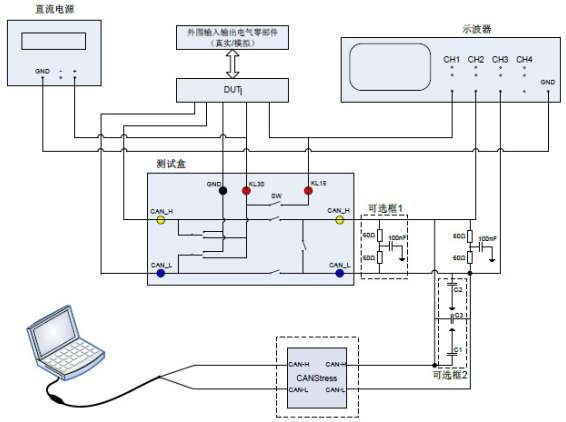


图6 手动测试架构搭建

ZLG 致远电子发布的 CANDT 一致性测试系统基于 CANScope 底层分析能力，集成示波器、电源等必要设备，可覆盖主机厂 CAN 一致性测试标准，全自动化实现 CAN 总线一致性测试，为主机厂及零部件企业建立 CAN 总线测试及保障体系。另外，CANDT 还可以根据测试结果进行输出测试报告，作为主机厂准入依据，大大降低主机厂在网络调试环节的工作量，并保证整车 CAN 网络环境的稳定。

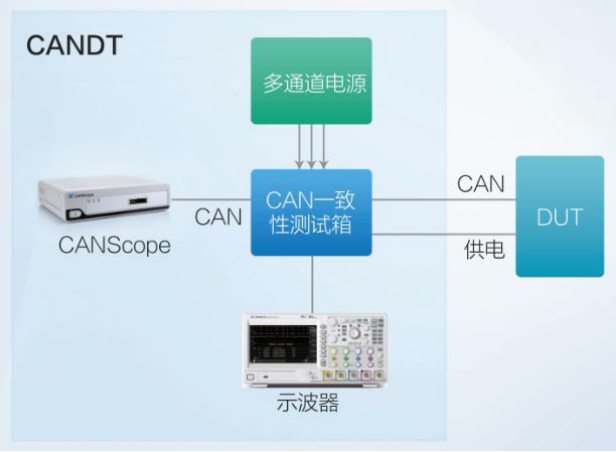


图7 CANDT一致性测试系统架构

互联互通▼

【产品应用】致远新一代LoRa终端有哪些功能特色？

ZLG 致远电子 2024-05-09 11:35:55

GLCOM-NODE-100 是广州致远电子股份有限公司 2024 年推出的一款智能数据采集终端新品。本期文章咱们一起来看看这款新产品有哪些功能特色吧。

GLCOM-NODE-100 是新一代的高性能 LoRa 终端采集器。其拥有工业级性能，支持数据采集、周期上报、自组网和 5km 超远距离通信，适用于多种行业应用场景。

图1 GLCOM-NODE-100

小巧轻便

GLCOM-NODE-100 是一款小巧型数据采集终端，其体积仅有 84×63×30mm（长 × 宽 × 高）。这个尺寸是一个什么概念呢？这个尺寸的长度约为手机的一半，宽度比手机略小一点。而且这款终端的重量仅为 151.6 克，约等于 3 个鸡蛋。这样小巧轻便的体积和重量，让它能够轻易地被安装在各种狭小的场景里，或者像一个小荷包一样被揣在兜里。

图2 比手机更小的终端

图3 整机重量

功能齐全

俗话说，麻雀虽小五脏俱全。别看体积小巧，GLCOM-NODE-100 内置了高性能的 LoRa 智能组网芯片，能够实现 485 串口通信、DI 数字信号采集、DO 数字信号控制以及 ADC 模拟数据采集功能。同时可随时装配 LoRa 天线，可以满足数据透传和组网的需求。在开启组网功能后，只需要按下 JOIN 按键，就可以完成组网，不需要再手动配置任何参数。除此之外，3 个 LED 指示灯能够帮助您轻松地检查终端的工作状态。如果想要重置设置的参数，也可以按住 DEFAULT 键并重新上电，恢复出厂设置。

图4 强大的LoRa无线通信功能

简易安装

得益于 LoRa 的通信距离和抗干扰能力，GLCOM-NODE-100 终端采集器可以广泛应用于室内外环境，通过在背面安装挂耳的方式，可以把它配置在几乎任何地方。比如可以在施工现场的水泥墙壁上固定两个水泥钉把它挂起来，然后接上电源和 470MHz 天线，使用 ADC 接口连接温度传感器，这样就可以采集室外的实时气温并且定期地上报至 LoRa 网关来监控施工现场的温度。又比如，在室内也可以安装在大型机箱中，连接上吸盘天线，并把吸盘天线吸附于机箱的铁质外壳表面，就可以节省许多空间了。如图 5 所示，用两颗螺丝和螺母即可挂在在室内的金属壁上，十分方便。

图5 室内挂壁

超远通信距离

GLCOM-NODE-100 内置高性能 LoRa 智能组网芯片，具备超强的射频性能，最高可以支持 62.5kbps 的数据速率，空旷环境下极限通信距离可达 5km，完全可以满足工业控制、智慧楼宇、城市基建、安防监控等众多场景的使用需求。

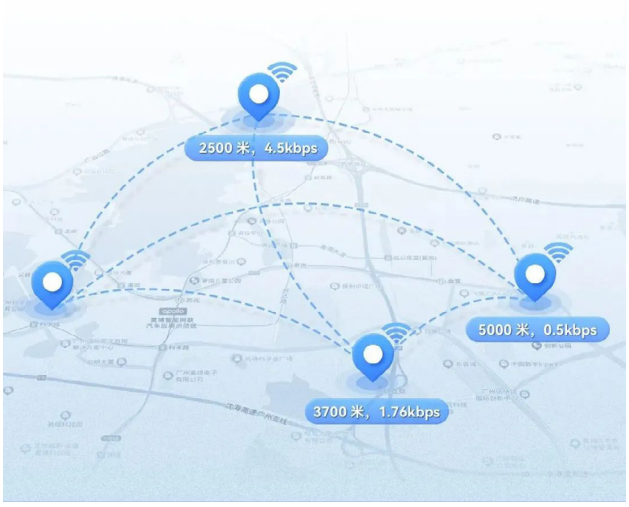


图6 LoRa超远距离传输

【产品应用】国产蓝牙模组 | BLE5.2为蓝牙带来了哪些变化?

ZLG 致远电子 2024-05-30 11:38:04

经过多年的发展，蓝牙已经从最初的 1.0 版本演变到了最新的 5.3 版本，目前最常用的是 BLE5.2 版本。在历代的版本更迭中，蓝牙技术有了非常大的进步。本期文章将带大家一起了解 BLE5.2 的主要特点。

BLE5.2简介

图1 BLE5.2

2020 年国际消费电子展上，蓝牙 SIG 推出了最新版本的 BLE5.2，增加了多项功能，这三项功能是：增强属性协议（EATT）、功率控制（LEPC）、同步通道（ISOC）。这些更新提高了蓝牙设备的可靠性、能效和用户体验。

ZM8258P 是广州致远电子股份有限公司设计的一款国产 BLE5.2 主从一体蓝牙模组，支持 8 路数据传输通道，最大支持 4 主 4 从。同时支持数据透传和 OTA 升级等功能，具有低成本、低功耗、远距离、小尺寸等优点。

增强型ATT协议

蓝牙 5.2 中对 ATT 协议进行了加强，简称为 EATT。EATT 修改了顺序事务模型，允许堆栈处理并发事务，并且新增的流量控制提升了 EATT 的稳定性。EATT 协议允许并发事务可以在不同的 L2CAP 通道上执行。这归功于 EATT 协议中的 ATT MTU 和 L2CAP MTU 是独立配置的，并且可以在连接期间重新配置。

在蓝牙 5.1 协议及之前的传输协议版本中事务的处理是顺序的，不支持并发，事务必须在一个完整的 PDU/SUD 之后才能执行；MTU 是一一对应且固定的，MTU 一旦建立连接便不可更改。而对于 LE5.2 的 EATT，MTU 在 ATT 和 L2CAP 之间不再一一对应，可以互相独立配置。

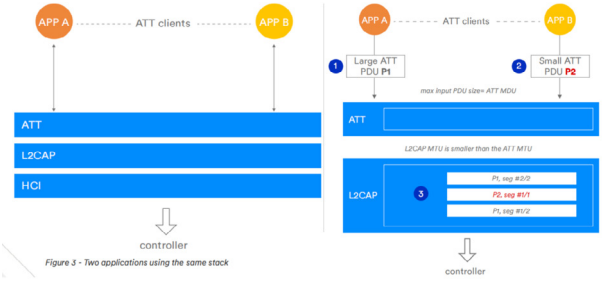


图2 EATT 事务处理模型（右）

LE功耗控制

BLE5.2 规定了对于发射功率的动态管理。通过对接受信号强度 RSSI 的监控，来通知发射方增加或减少发射功率。这对于在使用时设备之间的距离经常处于变化中的应用来说比较节省功耗，从而达到刚好满足应用的功耗。设备会根据不同距离时的 RSSI 值，控制发射功率使接收灵敏度保持在一个最佳的范围内，实现更好的控制功耗。

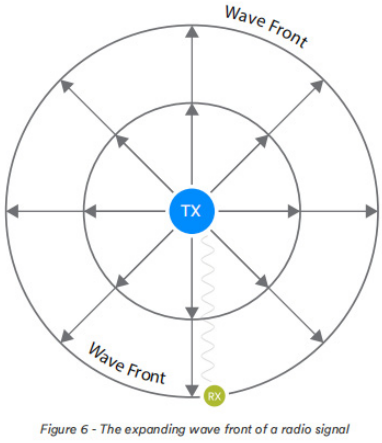


图3 无线电波路径损耗

LE同步信道

在 BLE5.1 及之前的版本中，仅支持面向连接的异步通信链路及非连接模式的广播链路，然而 BLE5.2 为了基于低功耗实现下一代 Bluetooth Audio 而定义了 LE 同步信道，包括连接模式下的同步音频流传输信道以及广播模式下的同步音频流传输信道。

LE 同步通道定义了一个有时间依赖的数据的传输通道和传输策略。首先是一个对于多接收方同步获取数据的机制；其次是定义了发送方在允许的时间外丢弃数据，从而保证接收方收取的数据满足时效要求。此同步通

道支持连接和非连接模式，内容包括：连接模式下同步音频流传输信道 / 多声道音频流，广播模式下同步音频流传输信道 / 广播模式共享音频流。

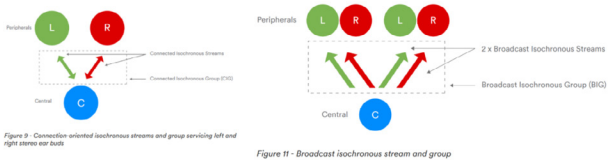


图4 连接同步信道及广播同步信道

总结

增强版 ATT(Enhanced ATT)：用于快速读取属性值，这一新增功能将提高基于 ATT 协议的信息沟通效率，实现快速服务发现等功能。

LE 功率控制 (LE Power Control)：BLE5.2 定义了低功耗蓝牙的双向功率控制协议 (LE Power Control)，可用于实现多种应用场景，有助于在保持连接的情况下进一步降低功耗并提高设备连接的稳定性和可靠性。

LE 同步信道 (LE Isochronous Channels)：LE 同步信道为实现下一代蓝牙音频的多声道音频流和基于广播音频流的共享音频应用打下了基础。根据 5.2 版本核心规范一个同步组可以包括最多 31 个不同的同步音频流，在广播同步模式下可以实现通讯范围内无限多个音频接收端同时收听分享的音频流。

感知控制 ▼

E_UHBCS-6W系列小体积宽压输入电源模块

ZLG 致远电子 2024-05-06 11:48:47

在高集成度的控制系统上，电源模块体积越做越小，但是小体积难以做到大功率。为满足需求，致远电子推出一款小体积、大功率宽压输入电源模块，拥有比 1W/3W 产品更高的功率，比普通 6W/10W 产品更小的体积。

E_UHBCS-6W 系列产品为单路输出（5V、12V、24V），拥有 9~36V 宽电压输入范围。在拥有与 1W/3W 产品相似的小体积（22.00×9.50×12.00）的情况下，输出功率高达 6W。满足 -40°C ~+105°C工作温度范围，EMC 性能优越，具有输入欠压保护，过流保护。并带有软启动功能，能够有效抑制起机电压尖峰。

产品应用

产品广泛用于工业控制、轨道交通、储能、电力电气、仪表仪器、医疗等领域。

图1 应用场景

产品特点

- 小体积：22.00×9.50×12.00mm；
- 大功率：输出功率高达 6W；
- 宽输入电压范围：9~36VDC；
- 工作温度范围：-40°C ~+105°C；
- 隔离耐压高达 1600VDC；
- 输入欠压保护、过载保护、输出短路保护等。

EMC性能

E_UHBCS-6W 系列产品 EMC 性能优越，仅需简单的外围电路就能保证优秀的 EMC 性能。以 E2412UHBCS-6W 为例，增加外围电路（参考数据手册电路参数）。测试辐射结果如图 2、图 3 所示。辐射测试余量非常充足，可见 E_UHBCS-6W 系列产品 EMC 性能优越，可放心选用。

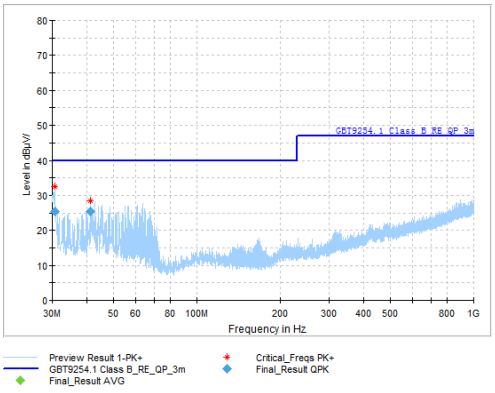


图2 产品加外围电路垂直方向

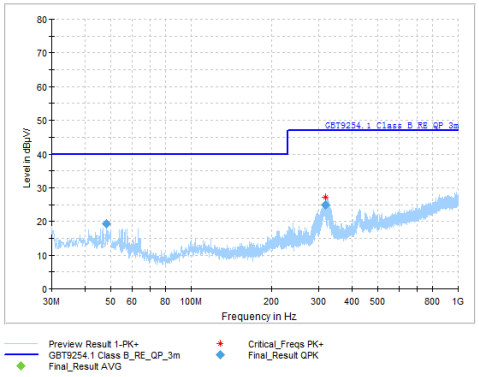


图3 产品加外围电路水平方向

E_UHBCS-6W 系列产品是一款高性价比的优秀的电源模块，欢迎大家测试与选用。

宽电压输入稳压电源模块 E48_UHFCS-3W系列

ZLG 致远电子 2024-05-07 11:36:51

在高集成度的控制系统上，电源模块体积越做越小，为满足需求，致远电子推出一款小体积、宽压输入电源模块。E48_UHFCS-3W 系列 18~72VDC 宽输入电压范围，小体积产品。

E48_UHFCS-3W 系列产品拥有 18~72VDC 宽输入电压范围，单路输出电压 5V、12V、24V，可满足 -40°C ~+85°C的工作温度范围，隔离耐压 4000VDC，EMC 性能优越，具有输入欠压保护，输出短路保护。

产品特点

- 4:1 超宽输入电压范围：18~72VDC；
- 隔离耐压高达 4000VDC；
- 工作温度范围：-40°C ~+85°C；
- 尺寸为 22.00 x 9.50 x 12.00mm；
- 输入欠压保护、输出短路保护等；
- 国际标准引脚方式。

EMI性能

E48_UHFCS-3W 系列产品有优越的 EMC 性能指标，以产品 E4805UHFCS-3W 为例，增加外围电路（详情见规格书），测试的传导测试传导性能如图 1 所示，测试辐射结果如图 2、3 所示。

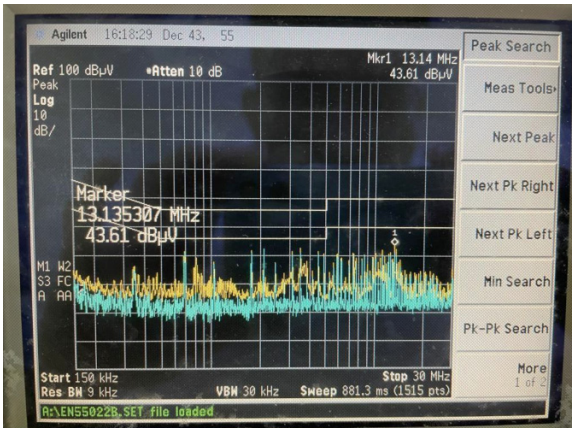


图1 E4805UHFCS-3W加外围传导测试数据
（QP&AVG值）

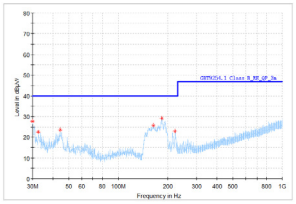


图2 产品加外围电路水平方向测试数据

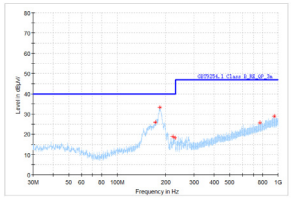


图3 产品加外围电路垂直方向测试数据

由以上可知，E4805UHFCS-3W 产品具有传导和辐射性能非常好，余量也充足，特别是辐射性能。

宽压 3W 系列产品有优秀的 EMC 特性，只需要简单的外围电路，就不用担心产品影响产品的 EMC 性能，是一款高性价比的小功率电源模块。欢迎大家测试与选用。

公司网址：www.zlg.cn。

产品应用

可广泛用于工控、电力、仪器仪表、通信等领域。

E_UHBDD-20W系列小体积电源模块

ZLG 致远电子 2024-05-08 11:43:10

在高集成度的系统应用中，电源模块的体积要求越来越小，功率密度要求越来越高，致远电子新推出的 E_UHBDD-20W 即为你解决空间问题又解决功率问题。

图1 E_UHBDD-20W系列小体积电源模块

E_UHBDD-20W 系列产品是一款宽电压输入，稳压单路输出电源模块。输入电压涵盖了 9-75V，输出电压有 3.3V、5V、12V、15V、24V，输出功率 20W。模块具有超宽工作温度范围 -40°C ~105°C，EMC 性能优越，拥有多种保护功能，输入欠压保护，输出过流保护，输出短路保护。

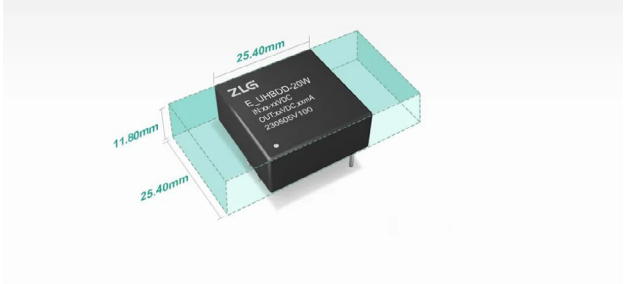


图2 产品尺寸展示图

产品体积小，仅 25.40×25.40×11.80mm，输出功率高达 20W，适用于既要空间又要功率的应用场合。

产品特点

- 小体积：25.40×25.40×11.80mm；
- 大功率：输出功率高达 20W；
- 转换效率：高达 91%；
- 隔离耐压：1500VDC；
- 静电抗电强度：6kV；
- 保护功能：输入欠压、输出过压、过流、短路保护功能。

性能亮点

输出低纹波噪声，产品的纹波噪声典型值能到 50mv 以下，如图 3 所示，测试条件及测试方法参照手册。

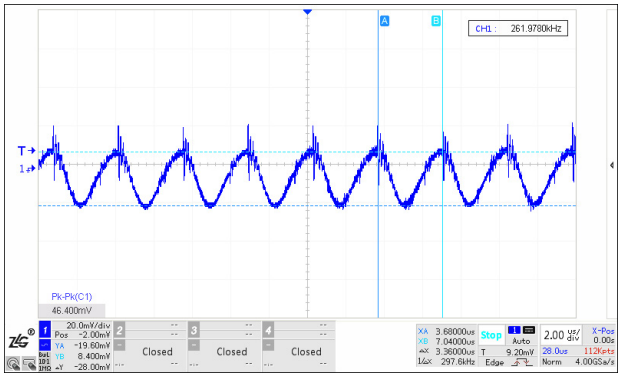


图3 标称输入满载输出纹波噪声图

EMC 性能优越，仅需简单的外围电路就能拥有较优秀的 EMC 性能，如图 4、5、6 所示，测试条件及测试方法参照手册。

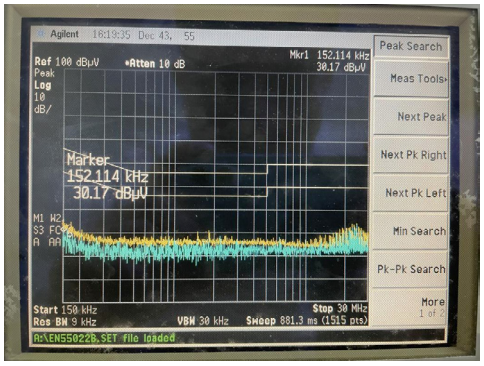


图4 传导骚扰测试结果图

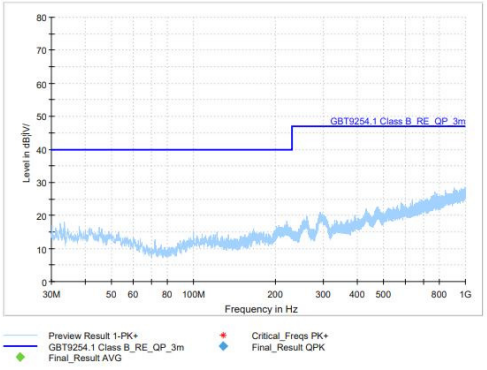


图5 水平方向辐射骚扰测试结果图

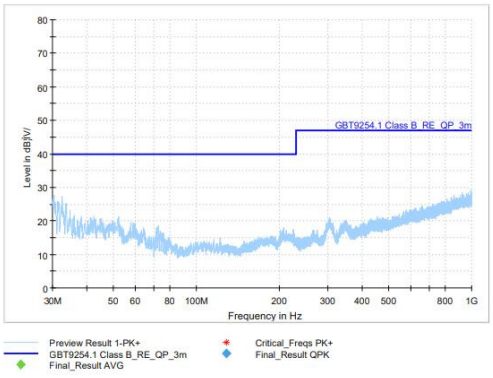


图6 垂直方向辐射骚扰测试结果图

产品应用场景

产品适用于工业控制、轨道交通，电力电器监控，楼宇自动化，仪器仪表，石油化工等领域。

图7 应用场景图

E_UHBDD-20W 系列产品小的躯体集众多优点，可靠性高，欢迎申样测试。

2024/5 第5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