**名称：**

数据通信软件设计文档

**摘要：**

本文档为通用软件开发平台数据通信模块的详细设计文档，用于指导程序设计。其功能包括：驱动和应用程序管理、共享内存数据维护、消息管理及转发。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 当前版本 | | 1.0 | | 文件状态 | | | [√]草稿 []正式 | | | | 取代版本 | | - |
| 完成时间 | | 2021-6-22 | | | 被取代文档完成时间 | | | | | | | - | |
| 作者 | | 矫贺岩 | | | | | | 审批 | | - | | | |
| 批准 | | - | | | | | | 存档编号 | | - | | | |
| 版本历史： | | | | | | | | | | | | | |
| 版本状态 | 作者 | | 参与者 | | | 完成日期 | | | 备注 | | | | |
| - | - | | - | | | - | | | - | | | | |

Copyright (c) 2009,天津华宁电子有限公司研发体系优化项目组

All rights reserved.

**目录**

[目录 2](#_Toc19191)

[1 文档介绍 3](#_Toc22788)

[1.1 文档目的 3](#_Toc20939)

[1.2 范围 3](#_Toc31388)

[1.3 参考资料 3](#_Toc4843)

[1.4 术语和缩写词 3](#_Toc27948)

[2 开发环境 3](#_Toc8844)

[3 软件架构 3](#_Toc27896)

[3.1 功能概述 3](#_Toc26940)

[3.2 驱动层 4](#_Toc20549)

[3.2.1 基本设计思路 4](#_Toc31835)

[3.2.2 发送处理 4](#_Toc13946)

[3.3 传输层 5](#_Toc26409)

[3.3.1 基本设计思路 5](#_Toc10597)

[3.4 协议层 6](#_Toc20874)

[3.4.1 基本设计思路 6](#_Toc2429)

[3.5 应用层 8](#_Toc6579)

[3.5.1 基本设计思路 8](#_Toc9556)

[3.5.2 周期性处理 9](#_Toc24658)

[3.6 对外接口及缓存 9](#_Toc7589)

[3.6.1 发送接口 9](#_Toc17143)

[3.6.2 对外缓存 10](#_Toc657)

# 文档介绍

## 文档目的

在网络版支架控制器中，设备间can协议栈负责管理外围can设备的通信，需要兼容V2的基础上，方便扩展V4新协议，用于研发设备进行can通信，同时具备“扩展透传协议”，实现架间can的链路层透传功能。

本文档描述其详细设计，用于指导程序设计，为程序测试和维护提供依据。

## 范围

本文档只适用于ZDYZ网络版电液控项目，描述的为设备间can协议实现方案。

## 参考资料

《网络版支架控制器产品总体方案设计文档》

## 术语和缩写词

无。

# 开发环境

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 软件名称 | 开发平台 | 操作系统 | 编程语言 | 编程工具 |
| 设备间通信协议栈 | PC | Windows 7 | C | Keil uVision4 |

# 软件架构

## 功能概述

为了更好的与控制器外围通信设备通信，层次划分如下图所示。



* 驱动层：

包括SPI-Can、原生Can的设备驱动、读写、IO控制、中断处理。

* Bus层：

包括Can数据的收发，数据和Can驱动的队列、链表等缓存的管理（增删改查）。

* 协议层：

包括V2外围Can通信设备（角度/高度、声光/人员、红外）协议的解析、V4外围can通信设备协议的解析，同时还有数据透传，以及这三部分的管理。

* 应用层：

包括can数据接收和发送任务，以及周期性调用任务，另外应用提供对外调用直接发送API接口，进行数据的发送。

## 驱动层

### 基本设计思路

驱动层负责多个can的设备底层驱动（原生、SPI），对外提供统一的调用接口。

如：open、close、init、write、read、ioctrl等。

### 发送处理

在数据发送时，开启两个定时器保证数据发送准确快速的发出，一个为1ms用于can帧发送失败后的重发，另外一个为10ms用于can发送超时取消发送。

采用中断首发和定时器重发方式进行发送，当can帧存入发送队列前判断当前缓存是否为空，如果为空插入缓存后，若两块发送缓存都为空，直接将数据发出并开启超时定时器，如果不为空，存入对应的缓存中。

当发送完成后进入发送中断后，判断器发送是否失败，如果失败开启重发定时器，等待1ms在重发定时器中断里重新取出重新发送，并关闭重发定时器，开启超时定时器，当重发超过一定次数或发送成功，删除此帧，取出下一个帧发送，重置超时定时器。

当10ms内未进入发送中断，则判断发送超时，删除此帧，取出下一帧发送，重置超时定时器。



## 传输层

### 基本设计思路

传输层封装驱动层的对外接口，创建管理数据收发缓存。

开辟四组Can缓存，分别为接收缓存、高优先级发送缓存、普通发送缓存、已发送缓存。存在当高优先级发送缓存中有数发送时，低优先级缓存中数据无法及时发送。



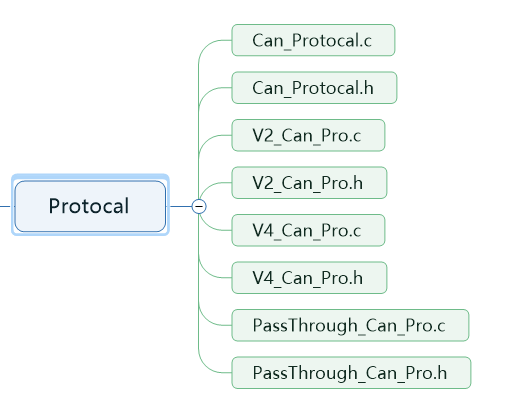




## 协议层

### 基本设计思路

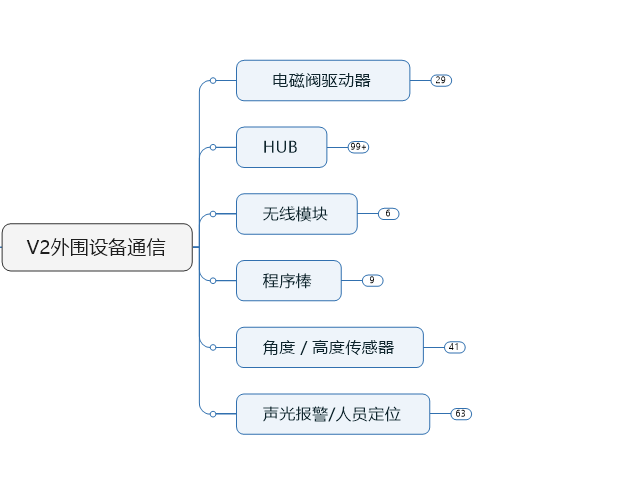
协议层负责协议栈通用功能管理和各协议的解析，其中包V2控制器外围can设备的间协议，扩展V4新外围设备协议，以及扩展透传协议三个协议栈，三个协议栈以单独的文件形式存在。文件格式如图所示。



三部分的通用协议栈的相关参数、相关函数指针的结构体如图所示。



目前V2控制器外围设备（can通信）如图所示。拿角度/高度传感器来说，将角度上报、角度查询、设置工作参数做为V2\_角度/高度协议中一部分，当有发送或接收数据到来，直接通过对应功能的回调函数处理相关数据。



## 应用层

### 基本设计思路

应用层负责开can数据发送和接收的任务，以及周期性处理任务。

can数据接收流程如所示，当can底层接收到数据后，直接将数据存入接收缓存中，并释放信号量，应用层任务接收到信号量后，根据数据在协议层匹配对应结构体，并完成回调。



### 周期性处理

1. 数据重发机制

采用定时查询的方式（任务、定时器）进行数据重发。对于有需要应答的数据帧，当can数据重发间隔时间内未收到主控应答，将其发发送次数减一，从重发缓存中删除此帧，若重发次数不为零，再将此帧放入发送队列，重发时间间隔收到主控应答，则从重发队列删除。

1. 数据周期性发送机制

对于需要某些can设备数据需要一直发送（电磁阀驱动器、声光报警器）设备，在协议层回调函数中采用定时发送，上层通过协议控制发送的开始与结束。

## 对外接口及缓存

### 发送接口

对外的发送接口如下。

1. **协议栈初始化接口：负责选择对应的协议栈（V2、V4、透传）将其初始化。**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：can\_pro\_init

\* 函数功能：can协议层初始化

\* 函数参数：proStack协议栈（v2、V4、透传）

\* 函数输出：初始化成功、失败

\* 注意事项：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unit8\_t can\_pro\_init(ProStack \*proStack);

1. **外围设备初始化接口：新设备接入，开辟对应内存，存储应用、状态数据。**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：can\_device\_init

\* 函数功能：can设备初始化

\* 函数参数：Device设备

\* 函数输出：初始化成功、失败

\* 注意事项：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unit8\_t can\_device\_init(Device device);

1. **Can数据发送接口：负责上层直接调用接口直接发送can数据。**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：can\_pro\_send

\* 函数功能：can发送接口

\* 函数参数：Driver驱动、Device设备、ChildDev子设备、Point点、Data数据

\* 函数输出：发送成功、失败

\* 注意事项：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unit8\_t can\_pro\_send(Driver driver, Device dvice, ChildDev,childDev, Point point, Data data);

1. **外围设备应用数据读取接口：负责上层调用，获取传感器应用数据。**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：can\_device\_app\_msg\_read

\* 函数功能：can外围设备应用数据读取

\* 函数参数：Device设备、ChildDev子设备、Point点、len读取长度、Data数据

\* 函数输出：读取成功、失败

\* 注意事项：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unit8\_t can\_device\_app\_msg\_send(Device dvice, ChildDev,childDev, Point point,uint8\_t len, Data data);

1. **外围设备状态数据读取接口：负责上层调用，获取传感器状态数据。**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名称：can\_device\_status\_msg\_read

\* 函数功能：can外围设备状态数据读取

\* 函数参数：Device设备、ChildDev子设备、Point点、len读取长度、Data数据

\* 函数输出：读取成功、失败

\* 注意事项：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unit8\_t can\_device\_status\_msg\_read(Device dvice, ChildDev,childDev, Point point,uint8\_t len, Data data);

### 对外缓存

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 外围设备应用数据缓存设计 | | | | | | | | |
| 驱动号 | 设备ID | 子设备 | 点 | 设备 | 子设备 | 大小 | 说明 | 目前最多 |
| SPICAN2 | 1 | 1 | 1 | 电磁阀 | 电磁阀 | 2byte | 阀检测（短路、断路、正常） | 2 |
| SPICAN2 | 1 | 1 | 1 | 备用 | 电磁阀 |  |  |  |
| SPICAN1 | 5 | 1 | 1 | 角度 | 角度 | 2byte | X轴2byte | 5 |
| SPICAN1 | 5 | 1 | 2 |  |  | 2byte | Y轴2byte |  |
| SPICAN1 | 5 | 1 | 1 | 角度 | 角度 | 2byte | X轴2byte | 5 |
| SPICAN1 | 5 | 1 | 2 |  |  | 2byte | Y轴2byte |  |
| SPICAN1 | 5 | 2 | 1 | 角度 | 高度 | 2byte | 高度2byte | 1 |
| SPICAN1 | 7 | 1 |  | 声光 | 声光 | null |  |  |
| SPICAN1 | 7 | 2 | 1 |  | 人员 | 2byte | 范围人数1、状态权限1 | 11 |
| SPICAN1 | 7 | 2 | 2 |  |  | 2byte | 编号2byte |  |
| SPICAN1 | 7 | 2 | 3 |  |  | 2byte | 距离2byte |  |
| SPICAN1 | 2 | 1 | 1 | 红外 | 红外 | 2byte | 规定时间次数 | 1 |
| SPICAN1 |  |  |  | 无线 | 不涉及 |  |  |  |
| 原生CAN |  |  |  | HUB | 不涉及 |  |  |  |
| SPICAN2 |  |  |  | 程序棒 | 不涉及 |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 外围设备状态数据缓存设计 | | | | |
| 设备 | 版本号 | 使能 | 在线 | 工作参数（待定） |
| 电磁阀 | 3byte | 1bit | 1bit |  |
| 角度 | 3byte | 1bit | 1bit | 工作方式1、接收设备类型1、上报时间间隔2、角度变化量1、接收方设备类型1 |
| 高度 | 3byte |  |  | 工作方式1、接收设备类型1、上报时间间隔2、高度变化量1、接收方设备类型1 |
| 声光 | 3byte |  |  | 蜂鸣器1、灯1、时间2 |
| 人员 | 3byte |  |  | 支架编号2、工作参数1（bit0:工作模式、bit1:计算距离方式、  bit2-3:变化距离、bit4-7:发射功率）、上报间隔2、检测范围2、支架高度2 |
| 红外 | 3byte |  |  |  |
| 无线 | 不涉及 |  |  |  |
| HUB | 不涉及 |  |  |  |
| 程序棒 | 不涉及 |  |  |  |