仓储搬运设备远程故障诊断系统

智能终端详细设计书

东南大学自动化学院

2015年9月13日

版本历史

版 本 号	更 改 描 述	更 改 日 期	修 订 人

目录

1 引言

1.1 编写目的

本说明书为宁波如意股份有限公司与东南大学自动化学院合作的“基于物联网的仓储搬运设备远程故障诊断系统”的智能终端的详细软、硬件说明文档。本说明为经过认真的调研与系统论证之后确定的最终设计方案，包括智能终端的嵌入式软件设计和硬件设计。嵌入式软件设计包括数据采集、故障诊断、远程通信和参数配置等，硬件设计包括各模块原理图、PCB电路板和硬件选型等。

1.2 背景

仓储搬运设备在现代企业的仓储系统中扮演着非常重要的角色，而叉车作为仓储搬运设备的主要工具，是仓储物料搬运设备中的主力军，被广泛应用于车站、港口、机场、工厂、仓库等国民经济中的各个部门。第二次世界大战期间，叉车得到发展。中国从20世纪50年代初开始制造叉车。特别是随着中国经济的快速发展，大部分企业的物料搬运已经脱离了原始的人工搬运，取而代之的是以叉车为主的机械化搬运。因此，在过去的几年中，中国叉车市场的需求量每年都以两位数的速度增长。

仓储叉车主要是为仓库内货物搬运而设计的叉车，目前应用较多的仓储叉车都是以电动机驱动的，因其车体紧凑、移动灵活、自重轻和环保性能好而在仓储业得到普遍应用。在多班作业时，电机驱动的仓储叉车更是处于长时间使用的状态，而叉车等仓储搬运设备在使用过程中，总是会出现各种类型的故障。当厂商将设备卖到世界各地的用户处，一旦叉车出现故障后，用户拨打厂商的服务电话求助，厂商派出维护工程师前往客户现场进行检查和维修，在这个过程中，有可能出现维护工程师携带的配件不合适，有可能维护工程师在现场不能解决问题，需要更高级的工程师到现场才能解决问题。另一方面，如果故障问题不严重，用户可以自行维护时，而用户对设备不够熟悉，则可能维护不当，会造成搬运设备使用寿命下降，并影响整个仓储系统的运作效率。仓储在现代企业的供应链中起到了重要的作用，随着近些年我国物流业的蓬勃发展，现代企业渐渐将注意力转到了仓储方面，如何提高仓储系统的管理水平和保障仓储系统的高效运转是当前业内专家和企业相关人士所共同研究的方向。

随着物联网、云计算等技术的快速发展，为仓储搬运设备智能维护提供了新的解决方案，越来越多的厂商希望能够通过互联网络对所售出或持有的产品进行远程诊断

和维护，这样可以减少维护工程师到现场的时间和费用，不仅节约了大量的人力和物力的成本，同时也能为客户提供更为快捷的服务，减少客户的损失。因此，远程诊断和远程维护成了厂商迫切需要解决的问题。物联网的传感器技术和嵌入式系统技术的发展为信息的实时采集和实时在线监测提供了可能。仓储搬运设备智能维护作为一个新的应用领域，能够让企业为客户提供更为完善的服务。

智能终端是整个系统的重要组成部分，为了给远程故障诊断系统提供实时的现场采集数据，设计一个可靠的远程故障诊断终端。故障诊断终端通过传感器技术、采样电路和现场总线采集叉车的各种开关量、模拟量和运行信息，获取最基本的数据，结合叉车的电气连接图和运行状态，制定出故障诊断逻辑，实时监测和判断叉车的运行状态，并通过 GPRS 或 WIFI 与互联网连接，将采集到的状态信息和故障诊断信息发送给诊断服务器或局域诊断中心，供维护人员查看。智能终端还具有单机配置和信息查看功能，方便维护人员对智能终端进行参数配置和现场叉车运行状态诊断。

1.3 参考资料

[1] 姚旭东. 国内外物联网技术发展的比较研究[D]. 西南交通大学，2012

[2] 余宁. 中国物联网的发展及前景分析[J]. 科技创新导报. 2011(08)

[3] 黎洪生，何岭松，史铁林，杨叔子. 基于因特网远程故障诊断系统架构[J]. 华中理工大学学报，2000(3)

[4] Luigi Atzori，Antonio Iera，Giacomo Morabito. The Internet of Things：A survey[J]. Computer Networks. 2010 (15)

[5] 刘德勇，朱明富. 基于Internet 的远程协作故障诊断系统技术[J]. 现代电子技术，2001(12)

[6] 唐志航，唐北平，陈世清. 基于Internet的远程故障诊断系统的研究与设计[J]. 机床与液压， 2009(2)

[7] 朱顺华. 基于GPRS的汽车远程故障诊断系统中车载终端的研究设计[D]：[硕士论文]. 长沙:国防科学技术大学，2006

1.4 术语定义及说明

文档中采用的专门术语的定义及缩略词简要如下：

WIFI：Wireless Fidelity， 基于IEEE 802.11b标准的无线局域网

GPRS:General Packet Radio Service，通用分组无线服务技术

2 设计概述

2.1 任务和目标

为了对仓储搬运设备运行状态实时监控,并给远程故障诊断系统提供实时的现场采集数据，设计一个可靠的故障诊断终端设备，使维护人员可以在远程获取搬运设备的实际运行信息，便于故障诊断。任务内容包括远程故障诊断终端的功能需求和硬件电路设计、故障诊断逻辑的分析与设计、远程通信协议的规划和设计和嵌入式系统软件的开发和设计。

任务目标是设计基于物联网的仓储搬运设备远程故障诊断终端，最终通过对搬运设备叉车的故障情况分析，建立严谨的故障判断逻辑，设计和开发一个完善、可靠的远程故障诊断终端。故障诊断终端能够实时采集叉车的运行状态信息，通过互联网与远程服务器进行数据通信实现在线监控，故障诊断逻辑能够及时发现故障问题，并在故障发生时及时向远程服务器发送故障信息，便于售后人员及时发现存在的故障，提高售后服务质量。同时，诊断终端与单机软件通过串口通信，用来配置故障诊断终端的运行参数，也在用来获取叉车的实时状态和故障信息，便于现场故障诊断。

2.1.1 需求概述

故障诊断智能终端：采集数据及状态信息，数据存储及故障判断，将状态信息或故障报警信息发送给服务器，或经局域诊断中心汇总后发送给服务器。

系统要求具有WIFI和GPRS通信功能。

2.1.2 运行环境概述

嵌入式软件运行于ARM架构的MCU上，STM32F107是意法半导体推出全新STM32互连型（Connectivity）系列微控制器中的一款性能较强产品，此芯片集成了各种高性能工业标准接口，且STM32不同型号产品在引脚和软件上具有完美的兼容性，可以轻松适应更多的应用。

新STM32的标准外设包括10个定时器、两个12位1-Msample/s AD(模数转换器)（快速交替模式下2M sample/s)、两个12位DA(数模转换器)、两个I2C接口、五个USART接口和三个SPI端口和高质量数字音频接口IIS，另外STM32F107拥有全速USB（OTG）接口，两路CAN2.0B接口，以及以太网10/100 MAC模块。此芯片可以满足工业、医疗、楼宇自动化、家庭音响和家电市场多种产品需求。

2.1.3 条件与限制

嵌入式软件的运行需要硬件设备的支持，需要保证硬件设备的稳定与可靠，其他条件与限制暂无。

2.1.4 开发工具

嵌入式软件的开发采用windows下的Keil uVision4集成开发环境，开发语言为C，并采用官方提供的stm32标准外设库、启动文件和微控制器软件接口标准等。

硬件采用windows下的Altium Designer Summer 09进行开发，Altium的一体化设计结构将硬件、软件和可编程硬件集合在一个单一的环境中。

3 系统详细需求分析

3.1 功能需求分析

3.1.1 系统配置

1) 设置设备ID信息

2) 设置GPRS相关信息

3) 设置WiFi相关信息

4) 设置其它信息

3.1.2 CAN数据采集

1) 控制器故障码及状态信息采集

故障信息：

01：高踏板故障；02：预充电故障；03：过流；04：控制器过热；05：主回路断电；06：电流信号采集故障；07：编码器故障/电机堵转；09:：电池组欠压；10：电池组过压；11：电机过热；12：I^2C存储故障；13：加速器故障；

18：电机开路；19：输出缺相。

状态信息：

运行方向/高低速模式、转速、低功耗模式、小计里程、直流电压、电机电流、电机温度。

2) 数据预处理

解析CAN总线数据

3.1.3 IO数据采集

1) 叉车开关量数据采集

喇叭：开关1个，喇叭1个，地线1个，共3个开关量

上升：开关1个，接触器线圈两端2个，触点两端2个，共5个开关量

下降：开关1个，电磁阀两端2个，共3个开关量

主接触器：接触器线圈两端2个，触点两端2个，共4个开关量

制动器：制动器两端2个，共2个开关量　线圈通断,

2）叉车模拟量数据采集

起升电机电流：电流范围0-150A，用霍尔电流传感器换检测

电机温度：温度范围100℃以下，用温度传感器检测

3）数据预处理

对模拟量采集的数据进行滤波。

3.1.4 数据处理

1）故障逻辑判断

根据采集的数据，进行故障决策

2）数据缓存

故障诊断模块保存最近2小时内的状态信息

3.1.5 通信功能

1）WiFi与局域诊断中心数据通信

2）WiFi与服务器（Internet）数据通信

3）GPRS数据通信

4）调试接口数据通信

3.2 性能需求分析

3.2.1 嵌入式软件

智能终端软件在搬运设备上电时自动运行，直到设备断电，软件正常运行时不受突然断电的影响，不会丢失重要数据，需要长时间稳定运行，不能因为异常问题发生崩溃。

3.2.2 智能终端硬件

硬件设备要求在较宽的温度范围内正常运行，且具有一定的抗震能力，在搬运设备正常运行产生的震动条件下，硬件设备稳定可靠。

4 总体方案

4.1 嵌入式软件结构

嵌入式软件功能划分如下图1所示：

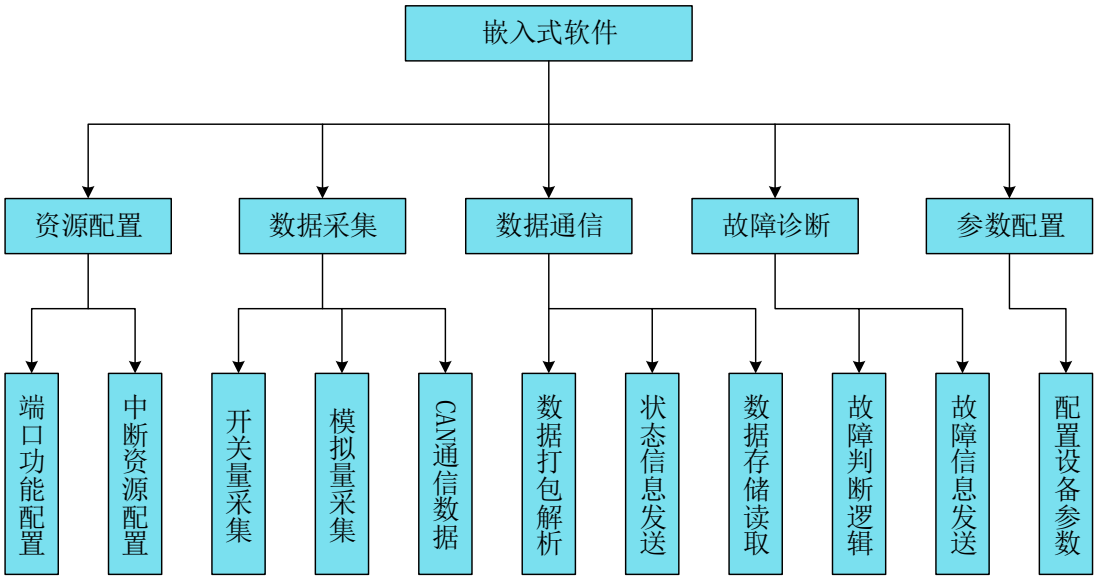

图1 嵌入式软件结构图

嵌入式软件主要分为五个功能模块：资源配置模块、数据采集模块、数据通信模块、故障诊断模块和参数配置模块。每个功能模块由若干子功能组成，子功能是具体的实施函数。

4.2 硬件设计方案

智能终端硬件设计结构如下图2所示。

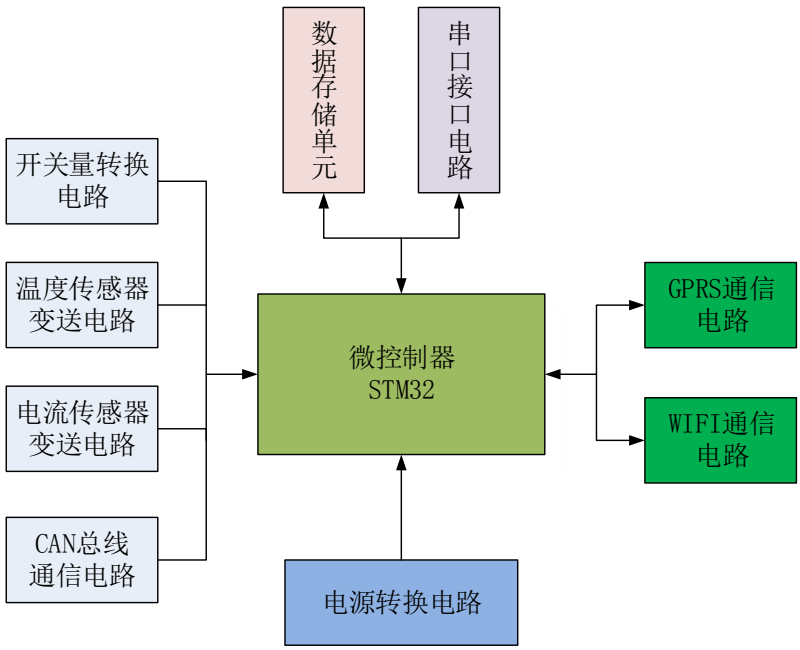

图2 硬件设计结构图

硬件设计按模块划分有：数据采集模块、电源供电模块、GPRS通信模块、WIFI通过模块和处理器系统模块。数据采集模块由开关量转换电路、温度传感器变送电路、电流传感

器变送电路和CAN总线通信电路。处理器系统模块包括数据存储单元和串口接口电路。

4.3 嵌入式软件开发文件结构

软件开发文件结构截图，如图3所示：

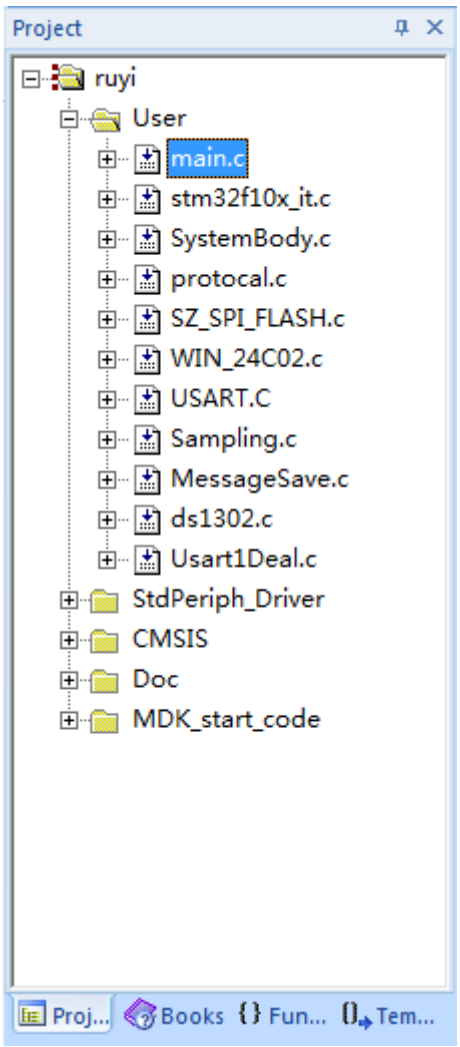

图3 软件开发文件文件结构截图

表1 文件夹说明

文件夹名	功能
User	用户开发代码文件夹，包含所有用户编写的代码文件
StdPeriph_Driver	stm32标准外设库，官方提供
CMSIS	微控制器软件接口标准，官方提供
MDK_start_code	启动文件，官方提供

表2 用户文件说明

文件名	功能
main.c	主功能文件，主函数入口，统一调用所有资源和函数

Stm32f10x_it.c	中断服务程序文件，包含所有中断服务程序
SystemBody.c	系统主体文件，包含状态信息定时发送，故障诊断
Protocol.c	通信处理文件，包含通信数据打包和解析所需函数
SZ_SPI_FLASH.c	Flash芯片驱动，主要包含数据写入和读取
WIN_24C02.c	EEPROM芯片驱动，主要包含数据写入和读取
USART.c	串口功能文件，包含串口数据接收和发送功能
Samping.c	采集文件，包含数据采集函数
MessageSave.c	数据包存储文件，包含数据包循环存储和读取函数
Ds1302.c	DS1302始终芯片驱动，包含时间设置和读取函数
Usart1Deal.c	串口1文件，包含配置功能函数

5 软件模块详细设计

5.1 资源配置（软件入口）

功能：

资源配置主要功能是使所有用到的硬件资源工作在合适的状态，包括配置用到的IO资源到合适的工作模式用于数据采集和数据通信，初始化串口资源用于设备数据通信，初始化ADC和DMA资源用于采集模拟量数据，中断优先级设置保证设备的正常工作。资源配置功能从软件入口处开始。

功能划分：

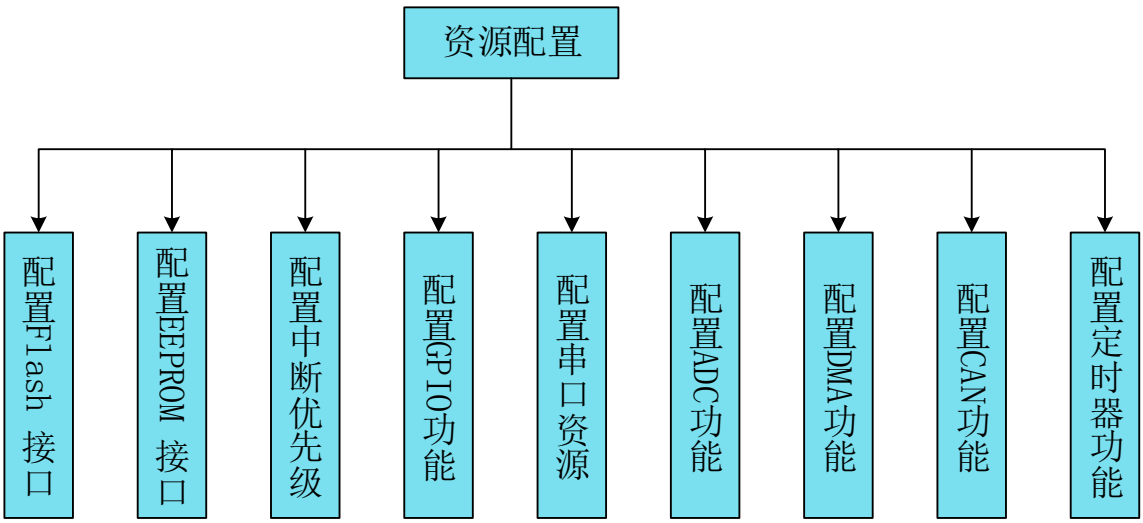

图4 资源配置功能划分

表3 全局变量：

变量名	类型	含义	定义文件
AD_Value[30][2]	u16	两路 AD 采样值缓存	Main.c
canMsg1[8]	uint8_t	接收到第一包 CAN 数据缓存	Main.c
canMsg2[8]	uint8_t	接收到第二包 CAN 数据缓存	Main.c
Usart1Rxcount	uint8_t	串口 1 接收数据个数	USART.c
Usart1RxOkFlag	FlagStatus	串口 1 一帧数据接收完成标志	USART.c
Usart1OverTimeFlagCount	uint8_t	串口 1 接收超时计数	USART.c
Usart1OverTimeFlag	FlagStatus	串口 1 接收超时标志	USART.c
Usart1RxBuffer[RxC]	uint8_t	串口 1 数据接收缓存	USART.c
Usart2Rxcount	uint8_t	串口 2 接收数据个数	USART.c
Usart2RxOkFlag	FlagStatus	串口 2 一帧数据接收完成标志	USART.c
Usart2OverTimeFlagCount	uint8_t	串口 2 接收超时计数	USART.c
Usart2OverTimeFlag	FlagStatus	串口 2 接收超时标志	USART.c
Usart2RxBuffer[RxC]	uint8_t	串口 2 数据接收缓存	USART.c
Usart3Rxcount	uint8_t	串口 3 接收数据个数	USART.c
Usart3RxOkFlag	FlagStatus	串口 3 一帧数据接收完成标志	USART.c
Usart3OverTimeFlagCount	uint8_t	串口 3 接收超时计数	USART.c
Usart3OverTimeFlag	FlagStatus	串口 3 接收超时标志	USART.c
Usart3RxBuffer[RxC]	uint8_t	串口 3 数据接收缓存	USART.c
Uart4Rxcount	uint8_t	串口 4 接收数据个数	USART.c
Uart4RxOkFlag	FlagStatus	串口 4 一帧数据接收完成标志	USART.c
Uart4OverTimeFlagCount	uint8_t	串口 4 接收超时计数	USART.c
Uart4OverTimeFlag	FlagStatus	串口 4 接收超时标志	USART.c
Uart4RxBuffer[RxC]	uint8_t	串口 4 数据接收缓存	USART.c

软件入口：

首先按顺序实现硬件资源的配置和初始化，完成后进入设备功能实现大循环。

```
/*********************************************************************
*FunName:     int main(void)
*
*Discription: 主函数，程序入口
*
*Notes:       配置所有硬件资源，循环调用功能函数以实现设备功能
*
*Define File: Main.c
*
*********************************************************************/
int main(void)
{
      SPI_FLASH_Init();//SPI flash config
      AT24C02_Init();  //eeprom config
      delay_ms(1000);
      TIM4_Init();  //tim4 init  1min  没有用到
      FirstSaveSystemInfo();//设备第一次运行时  数据初始化
      getTimeandDeviceInfo();  //read runningtime and device id from eeprom;
      NVIC_Configuration();//中断配置 nvic config
      GPIO_Config();//GPIO配置 digital config
      USART_Config();//串口配置 usart config
      ADC_Config();//ADC配置 adc config
      DMA_Config();//DMA配置 dma config
      CAN_Config();//CAN配置 can config
      SysTickConfig();//系统滴答定时器   50ms中断
      Delay((uint32_t)0xFFFF);
      TIM5_Init();  //初始化定时器  init 1s for record the time for send data
      TIM3_Int_Init(19, 7199);  //10kHZ的计数频率，计数到20用时2ms。
      ReadFlashpara();  //读取Flash存储数据参数
      ReadResistorCoefficientPara(); //读取温度标定系数
      ReadMessageCacheCnt();   //读取数据缓存个数
      RELAY_OFF;  delay_ms(1);   //继电器复位

      while (1) //主循环体，设备功能实现部分
      {
            Get_DS1302(&realtime);

            if (flag_Running_time == SET)   //是否1分钟 IS 1 MIN
            {
                  recordTime(); // update the running time in eeprom spiflash
                  flag_Running_time = RESET;
            }

            . . . . . .
```

```
        RT = ResistorCoefficient.fdat*anaMsg[0] * 3300 / 4095 / 1000;
        TemperatureMeasureCalibration(RT); //温度系数标定
        Temper = CalculateTemperature(RT);   //计算浮点数温度
        TemperErrorValue = (u16)Temper;        //整数温度

        ScanData.allinfo.colData.anaData[1] = (u8)Temper;
        ReadSwitch();//读拨码开关
        ReadHC165Data();//获取32个开关量信号
        RecordActionNumber();//获取动作次数
        IO_OUT = Read_IO(); //IO输入  没有用到

        if (ArresterControlEn == 1) //制动器控制及故障检测
        {
              . . . . . .
        }

        ComunitationLinkTest();//网络状态检测
        Scan_Current_Info();     //扫描 scan_info 并发送状态信息

        if (Usart2RxOkFlag == SET)  // GPS
        {
              . . . . . .
        }

        if (Usart1RxOkFlag == SET)  // 调试及配置串口
        {
              . . . . . .
        }

        if (Usart3RxOkFlag == SET)  // WIFI
        {
              . . . . . .
        }

        if (Uart4RxOkFlag == SET)  // GPRS
        {
              . . . . . .
        }

        StatusMessageReadFromFlash();//读取Flash中的数据
        SendOnceMessage(); //响应实时刷新数据一次

}
```

}

5.2 数据采集

功能：

数据采集主要功能通过硬件接口，不断采集IO输入信号、模拟量数值、CAN总线数据和芯片数据，一部分为搬运设备运行数据的采集，为故障诊断提供数据，另一部分为设备运行状态数据的采集，为设备工作情况提供判断依据，主要数据如下。

1) 控制器CAN数据，即故障码及状态信息采集；

2) 搬运设备开关量数据采集、模拟量数据采集；

3) 开关动作次数；

4) 实时时间信息读取；

5) 网络通信状态检测

功能划分：

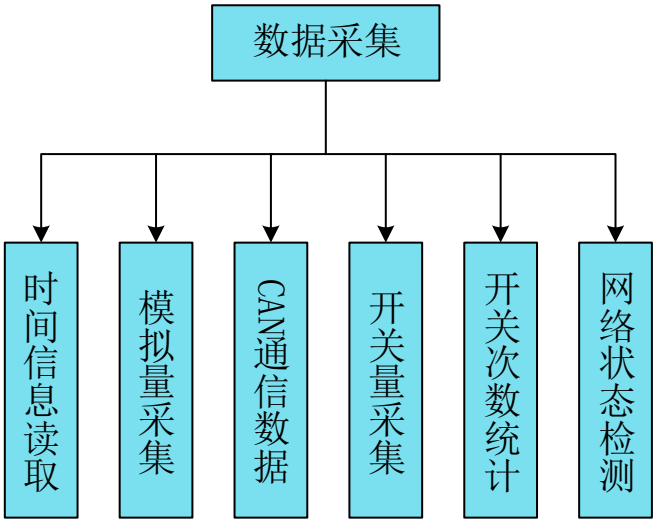

图5 数据采集功能划分

结构体：

ALLKINDSTRUCT 结构体（protocal.h）

```
typedef struct
{
      COMINFO cominfo;      //通讯设置结构体
      DEVICEINFO devinfo;         //ID设置结构体;
      ALLINFO  allinfo;        //数据采集结构体;
```

```
    DATETIME timeinfo;
}ALLKINDSTRUCT;
```

ALLKINDSTRUCT 结构体定义表

成员	含义
COMINFO cominfo	通讯设置结构体
DEVICEINFO devinfo;	ID 设置结构体
ALLINFO allinfo;	叉车数据采集结构体
DATETIME timeinfo;	时间结构体

DEVICEINFO 结构体（protocal.h）

```
typedef struct DevInfo
{
    uint8_t deviceData[10];
}DEVICEINFO;
```

DEVICEINFO 结构体定义表

成员	含义
deviceData[10]	10 位叉车编号

ALLINFO 结构体（protocal.h）

```
typedef struct BagInfo
{
    CANINFO canData;   //can data ;
    COLLECTINFO colData;//collect data;
}ALLINFO;
```

ALLINFO 结构体定义表

成员	含义
CANINFO canData	CAN 通信数据结构
COLLECTINFO colData	叉车采集数据结构体

CANINFO 结构体（protocal.h）

```
typedef struct ControlInfo
{
    uint8_t  canData1[8];   //CAN data 1;
    uint8_t  canData2[8];   //CAN  data 2;
}CANINFO;
```

CANINFO 结构体定义表

成员	含义
canData1[8]	存储第一包 CAN 通信数据
canData2[8]	存储第二包 CAN 通信数据

COLLECTINFO 结构体（protocal.h）

```
typedef struct DataInfo
{
```

```
    uint8_t digData[32];   //17 digital data
    uint16_t  anaData[2]; // 2 anaData
} COLLECTINFO;
```

COLLECTINFO 结构体定义表

成员	含义
digData[32]	采集 32 路开关量状态
anaData[2]	采集两路模拟量数据

DATETIME 结构体（protocal.h）

```
typedef struct TimeInfo
{
    uint8_t year;
    uint8_t month;
    uint8_t day;
    uint8_t hour;//IP add;
    uint8_t minute;
}DATETIME;
```

DATETIME 结构体定义表

成员	含义
year;	记录时间的年
month	记录时间的月
day	记录时间的日
hour	记录时间的时
minute	记录时间的分

TIME 结构体（protocal.h）

```
struct TIME
{
    unsigned char second;
    unsigned char minute;
    unsigned char hour;
    unsigned char day;
    unsigned char month;
    unsigned char year;
};
```

TIME 结构体定义表

成员	含义
second	时钟数据的秒
minute	时钟数据的分
hour	时钟数据的时
day	时钟数据的日

month	时钟数据的月
year	时钟数据的年

全局变量：

变量名	类型	含义	定义文件
ScanData	ALLKINDSTRUCT	记录所有采集叉车数据	Main.c
Temper	float	计算得到的温度值	Main.c
RT	float	计算得到的PT100阻值	Main.c
realtime	TIME	当前时间信息	Ds1302.h
DigDataBuffer[6]	uint8_t	开关状态历史数据	Samping.c
DigDataCurrent[6]	uint8_t	开关状态当前数据	Samping.c
ActionCnt[6]	uint8_t	开关次数记录	Samping.c
ActionStatus[6]	uint8_t	带开关状态的次数记录	Samping.c
RecordFlag	uint8_t	第一次记录开关状态	Samping.c
linkstatu[3]	uint8_t	网络连接状态输入信号	Samping.c
LinkStatuOnlineFlag	uint8_t	网络连接状态标志	Samping.c
MessageCacheCnt	int32TOCHAR	离线数据包缓存个数	Samping.c
LinkTimeCnt	uint32_t	网络恢复计时变量	Samping.c
LinkTimeCntLimit	uint32_t	网络恢复计时上限	Samping.c

数据采集功能函数：

时间信息读取

```
/*****************************************************************************
*FunName:     void Get_DS1302(struct TIME * ptr)
*
*In:          ptr 指向实时时间变量的结构体指针
*
*Discription: 获取实时时间信息
*
*Notes:       读取DS1302时钟芯片获取实时时间信息
*
*Define File: Ds1302.c
*
```

```
*****************************************************************************/
void Get_DS1302(struct TIME * ptr)
{
      unsigned char addr = 0x81;
      ptr->second = DS1302_Read(addr);
      addr += 2;
      ptr->minute = DS1302_Read(addr);
      addr += 2;
      ptr->hour = DS1302_Read(addr);
      addr += 2;
      ptr->day = DS1302_Read(addr);
      addr += 2;
      ptr->month = DS1302_Read(addr);
      addr += 4;
      ptr->year = DS1302_Read(addr);

}
```

模拟量采集

```
/*****************************************************************************
*FunName:     无
*
*Discription: 代码段
*
*Notes:       根据AD值，计算得到电流值
*
*Define File: Main.c
*
*****************************************************************************/
ScanData.allinfo.colData.anaData[0] = abs(250 - 2 - anaMsg[1] * 330 /
4096);//anaMsg[0]*3.3/0x1000;//    电流计算
if(ScanData.allinfo.colData.anaData[0]<3)
ScanData.allinfo.colData.anaData[0] = 0; //    电流最小区间限制

/*****************************************************************************
*FunName:     无
*
*Discription: 代码段
*
*Notes:       根据AD值，计算出PT100阻值，查表算出温度值，最后取整数部分
*
*Define File: Main.c
*
*****************************************************************************/
//温度计算
```

```
RT = ResistorCoefficient.fdat*anaMsg[0] * 3300 / 4095 / 1000;     //电阻值
TemperatureMeasureCalibration(RT); //温度系数标定
Temper = CalculateTemperature(RT);   //计算浮点数温度
TemperErrorValue = (u16)Temper;        //整数温度
```

CAN 数据采集

```
/*******************************************************************************
*FunName:     void CAN1_RX0_IRQHandler(void)
*
*Discription: CAN1中断函数，接收CAN数据
*
*Notes:       根据ExtID的不同，存储CAN数据包一、二的数据
*
*Define File: stm32f10x_it.c
*
*******************************************************************************/
void CAN1_RX0_IRQHandler(void)
{
      uint8_t i = 0;
      CAN_Receive(CAN1, CAN_FIFO0, &RxMessage);
      if ((RxMessage.ExtId == 0x10F8109A) && (RxMessage.IDE == CAN_ID_EXT))
      //判断为第一包CAN数据
      {
            if (canupdateflag1 == 200)
            {
                  canupdateflag1 = 0;
            }
            canupdateflag1++;
            for (i = 0; i<8; i++)
            {
                  canMsg1[i] = RxMessage.Data[i];//存储第一包CAN数据
            }
            CanComOvertimeCnt = 0;   //通信超时计数清零
      }
      if ((RxMessage.ExtId == 0x10F8108D) && (RxMessage.IDE == CAN_ID_EXT))
     //判断为第二包CAN数据
      {
            if (canupdateflag2 == 200)
            {
                  canupdateflag2 = 0;
            }
            canupdateflag2++;
            for (i = 0; i<8; i++)
            {
```

```
                canMsg2[i] = RxMessage.Data[i];      //存储第二包CAN数据
            }
            CanComOvertimeCnt = 0;    //通信超时计数清零
        }

}
```

开关量采集

```
/*****************************************************************************
*FunName:     void ReadHC165Data(void)
*
*Discription: 74HC165芯片数据读取
*
*Notes:       读取所有32路开关量数据
*
*Define File: SystemBody.c
*
*****************************************************************************/
void ReadHC165Data(void)
{
      uint8_t i = 0;
      HC165_CLK_L;//始终拉低
      HC165_SHLD_L;
      HC165_SHLD_H;
      . . . . . .
}
```

5.3 数据通信

功能：

数据通信主要功能是根据制定的通信协议，将采集到的叉车信息打包为一个数据包，按设定的通信方式定时向服务器发送状态信息，同时接受服务器发送的指令并解析，执行对应的操作，主要是数据刷下指令。数据存储功能用于记录历史状态数据包，并在网络异常时，提供数据缓存功能，待通信恢复后，将缓存数据一起发送给服务器。

功能划分：

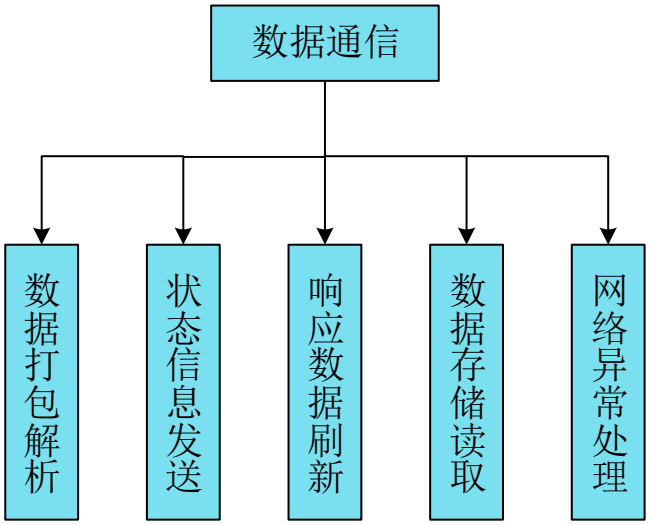

图6 数据通信功能划分

结构体：

HEADandTAIL 结构体（protocal.h）

```
typedef  struct tagCMDHEADandTAIL
{
      uint32_t  Dest;    //destination;
      uint32_t   Source;    // source;
      uint16_t   FC;
      uint16_t DataLenth;
      uint16_t CheckSum;
      uint16_t CheckSumRx;
}HEADandTAIL;
```

表 5 HEADandTAIL 结构体定义

成员	含义
Dest	目标地址
Source	源地址
FC	功能码
DataLenth	数据长度
CheckSum	计算到的校验和
CheckSumRx	解析到的校验和

表6 全局变量

变量名	类型	含义	定义文件
flag_TIMER_RX	FlagStatus	状态信息定时发送标志	Main.c

flag_Sendontime	FlagStatus	状态信息发送完成标志	Main.c
DESTNATION	uint32_t	目标设备地址	Main.c
DEVICEADD	uint32_t	源设备地址	Main.c
timeCounter	uint8_t	开机发送数据控制变量	Main.c
SendTimeCnt	u16	定时发送计时变量	SystemBody.c
diglenth	uint8_t	实际开关量数据长度	protocal.c
analenth	uint8_t	模拟量数据长度	protocal.c
canlenth	uint8_t	CAN 数据包长度	protocal.c
StatusMessageStartAddr	u32	Flash 状态信息存储起始地址	MessageSave.c
StatusMessageEndAddr	u32	Flash 状态信息存储结束地址	MessageSave.c
StatusMessageNumber	u32	Flash 状态信息存储总数	MessageSave.c
StatusReadStartAddr	u32	Flash 状态信息读取起始地址	MessageSave.c
StatusReadEndAddr	u32	Flash 状态信息读取结束地址	MessageSave.c
StatusReadNumber	u32	Flash 状态信息读取个数	MessageSave.c
StatusReadEnFlag	u8	Flash 状态信息读取使能标志	MessageSave.c
StatusReadBuff [DatagramLength]	u8	Flash 状态信息读取一包数据缓存	MessageSave.c
MessageReadFlag[2]	u8	缓存数据读取标志	MessageSave.c
MessageReadNumberBuf[2]	u32	缓存数据读取个数存储	MessageSave.c
MessageReadStatus	u8	缓存数据读取状态	MessageSave.c

表7 通信协议

1 字节	4 字节	4 字节	10 字节	2 字节	2 字节	N 字节	2 字节	1 字节
@	ADR	SRC	ID	FC	LEN	DATA	RC	$
帧头	目的地址	源地址	叉车 ID	功能码	帧长	数据	校验	帧尾

其中数据部分内容如下：

表 8

10 字节	23 字节	2 字节	18 字节	12 字节
运行时间	开关量	模拟量	CAN 数据	信息采集时间
年 月 日 时 分	17 个开关量 6 个开关动作次数	一个电流 一个温度	第一包 CAN 数据 8 个 一个状态变量 第二包 CAN 数据 8 个 一个状态变量	年 月 日 时 分 秒

功能码介绍如下：

表 9

功能码	数值	说明
FC_DAT	0x01	系统状态信息
FC_WNG	0x02	系统报警信息
FC_GSI	0x03	获取状态信息

数据通信功能函数：

定时将采集的数据打包为一帧状态信息，将状态信息发送给服务器，同时存储这一帧状态信息。

```
/*****************************************************************************
*FunName:      void Scan_Current_Info(void)
*
*Discription: 定时1分钟记录数据，发送状态信息
*
*Notes:        定时发送状态信息，存储状态信息
*
*Define File: SystemBody.c
*
```

```
*****************************************************************************/
void Scan_Current_Info(void)
{
      uint8_t i = 0;
      uint8_t *Tx_currentInfo = NULL;
      /*CAN数据读取到结构体中 get CAN      data */
      for (i = 0; i<8; i++)
      {
            ScanData.allinfo.canData.canData1[i] = canMsg1[i];
            ScanData.allinfo.canData.canData2[i] = canMsg2[i];
      }
      ///到时间了发送采集信息
      if (flag_TIMER_RX == SET || timeCounter == 0)//开机发送或定时发送
      {
            if (CanErrorFlag == 1) //can error CAN通信故障
            {
                  canupdateflag1 = 0xff; canupdateflag2 = 0xff;
            }
            . . . . . .
            changeMessageCacheCnt(1);  //缓存数据个数加一
            Send_Instruction(Tx_currentInfo); //发送数据
            StatusMessageSaveToFlash(Tx_currentInfo);//数据保存
            releaseChar_ptr(Tx_currentInfo);  //释放空间
            ActionNumberClear();  //开关次数清零
            timeCounter = 1;//开机发送成功
            flag_Sendontime = SET; //定时发送成功
      }
      /* if  send on time;reset the timer flag*/
      if (flag_Sendontime == SET)  //发送成功   复位标志
      {
            flag_TIMER_RX = RESET;
            flag_Sendontime = RESET;
      }
}
```

接收服务器发送的数据帧，并主要响应数据刷新指令

```
/*****************************************************************************
*FunName:     uint8_t ProtocolReply(uint8_t *Rx_info)
*
*Out:               解析结果
*
*Discription: 响应数据刷新指令
*
*Notes:       主要用于接收解析服务器的刷新指令。   功能码为03
```

```
*
*Define File: protocal.c
*
*********************************************************************/
uint8_t ProtocolReply(uint8_t *Rx_info)
{
      uint16_t recieveFC = 0; //用于存储接收到的功能码
      uint8_t* Tx_info = NULL;
      uint8_t data = 0; //返回值
      ALLKINDSTRUCT unionstruct;     //所有参数
      recieveFC = parseChar_ptr(Rx_info, &unionstruct);//解析功能码
      switch (recieveFC)
      {
            . . . . . .
      }
      releaseChar_ptr(Tx_info);
      return data; //返回结果
}
```

5.4 故障诊断

功能：

故障诊断是设备的主要功能，根据采集的控制器数据、开关量数据和模拟量数据，结合搬运设备的工作原理，分析出在不同工作情况下的搬运设备工作状态，形成核心的故障诊断逻辑，智能终端设备对各状态实时监测，并结合故障诊断逻辑实时判断是否有故障发生，如有故障发生则立即发送对应的故障报警信息，故障持续存在时，则每隔一分钟发送一次故障报警信息，某一时刻某故障恢复则立即发送对应的故障恢复信息。实现一个完整的故障检测、故障报警和故障恢复功能。

功能划分：

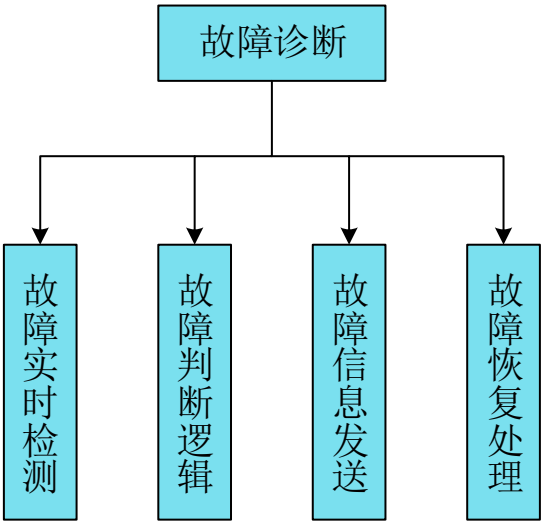

图7 故障诊断功能划分

表10 全局变量

变量名	类型	含义	定义文件
CanComOvertimeCnt	Uint16_t	CAN 通信超时计数	SystemBody.c
ErrorFlag[20]	Uint8_t	故障发生数值	SystemBody.c
CanErrorFlag	Uint8_t	CAN 通信故障标志	SystemBody.c
CanErrorPast[14]	Uint8_t	CAN 数据中历史故障代码	SystemBody.c
WarnTimeCon	Uint16_t	故障报警信息连续发送，时间控制变量	SystemBody.c
CurrentErrorCnt	Uint16_t	电流传感器故障时间计数	SystemBody.c
CurrentHistoryValue	Uint16_t	电流历史数据	SystemBody.c
CurrentErrorFlag	uint8_t	电流传感器故障标志	SystemBody.c
TemperErrorFlag	uint8_t	温度传感器故障标志	SystemBody.c
ArresterErrorFlag	uint8_t	制动器离线或线圈故障标志	SystemBody.c
TemperErrorValue	Uint16_t	温度传感器故障值	SystemBody.c
LiftErrorCnt	uint8_t	提升接触器故障判断次数	SystemBody.c
CurrentTestCnt	uint8_t	电流传感器故障判断次数	SystemBody.c
TemperTestCnt	Uint8_t	温度传感器故障判断	SystemBody.c

		次数	
CurrentBoundaryValue	Uint8_t	电流故障判断阈值	SystemBody.c
TemperBoundaryValue	Uint8_t	温度故障判断阈值	SystemBody.c

表11 故障代码对应表

故障代码	故障名称	故障判断逻辑
01	高踏板故障	由控制器 CAN 数据提供
02	预充电故障	由控制器 CAN 数据提供
03	过流	由控制器 CAN 数据提供
04	控制器过热	由控制器 CAN 数据提供
05	主回路断电	由控制器 CAN 数据提供
06	电流采样电路故障	由控制器 CAN 数据提供
08	BMS 故障	由控制器 CAN 数据提供
09	电池组欠压	由控制器 CAN 数据提供
10	电池组过压	由控制器 CAN 数据提供
11	电机过热	由控制器 CAN 数据提供
13	加速器故障	由控制器 CAN 数据提供
38	电流传感器故障	根据电流传感器断线或离线时的异常值，判断异常值持续存在的时间，如果超过一定时长，则判断为电流传感器故障
39	温度传感器故障	根据温度传感器断线或离线时的异常值，由于异常值较正常值偏差很大，直接判断是否为异常值，是则判断为温度传感器故障
40	提升电机过流	电流传感器正常情况下，判断电流值是否超过设定的阈值，是则判断为提升电机过流
41	提升电机过热	温度传感器正常情况下，判断温度值是否超过设定的阈值，是则判断为提升电机过热
42	CAN 通讯故障	CAN 通信正常情况下每隔 50ms 收到一次数据，如果持续一段时间没有收到数据，则判断 CAN 通信故障
43	喇叭开关正极断路	喇叭开关正极正常情况为 24V 高电平，如果检测到变为低电平，则判断为喇叭开关正极断路
44	喇叭地线断路	喇叭地线正常情况为 0V 低电平，如果检测到 IO 信号变为高电平，则判断为喇叭地线断路
45	提升开关正极断路	提升开关正极正常情况为 24V 高电平，如果检测到 IO 信号变为低电平，则判断为提升开关正极断路
46	提升接触器线圈地线断路	提升接触器线圈地线正常情况为 0V 低电平，如果检测到 IO 信号变为高电平，则判断为提升接触器线圈地线断路
47	提升接触器故	提升接触器线圈断电情况下，触头断开，触头出

	障	线端没有电信号，如果此时出线端还有电信号，说明触头没有断开，则判断为提升接触器故障 提升接触器线圈通电情况下，触头吸合，触头出线端应为 24V 电信号，如果此时出线端没有有电信号，说明触头没有吸合，则判断为提升接触器故障
48	下降开关正极断路	下降开关正极正常情况为 24V 高电平，如果检测到 IO 信号变为低电平，则判断为下降开关正极断路
49	下降电磁阀地线断路	下降电磁阀地线正常情况为 0V 低电平，如果检测到 IO 信号变为高电平，则判断为下降电磁阀地线断路
50	主接触器线圈正极断路	主接触器线圈正极正常情况为 24V 高电平，如果检测到 IO 信号变为低电平，则判断为主接触器线圈正极断路
51	主接触器触点正极断路	主接触器触点正极正常情况为 24V 高电平，如果检测到 IO 信号变为低电平，则判断为主接触器触点正极断路
52	主接触器故障	主接触器线圈断电情况下，触头断开，触头出线端没有电信号，如果此时出线端还有电信号，说明触头没有断开，则判断为主接触器故障 主接触器线圈通电情况下，触头吸合，触头出线端应为 24V 电信号，如果此时出线端没有有电信号，说明触头没有吸合，则判断为主接触器故障
53	制动器线圈正极断路	制动器线圈正极正常情况为 24V 高电平，如果检测到 IO 信号变为低电平，则判断为制动器线圈正极断路
54	制动器离线或线圈故障	制动器断电时，线圈接地端断开，此时线圈断开的一段的电压应该与制动器线圈正极相同为 24V，如果检测到断电时，IO 信号不为高电平，则判断为制动器离线或线圈故障

故障诊断功能函数：

故障诊断入口，调用故障实时在线检测函数和故障信息持续发送函数

```
/*****************************************************************************
*FunName:      void  ErrorTest(void)
*
*Discription: 故障诊断功能函数
*
*Notes:        故障实时在线检测，如有故障，则持续发送故障信息
*
*Define File: SystemBody.c
*
```

```
*****************************************************************************/
void  ErrorTest(void)
{
      ErrorRealtimeTest();//故障实时在线检测
      ErrorMessageContinueSend();//故障信息持续发送

}
```

故障实时在线检测示例函数

```
/*****************************************************************************
*FunName:     void  ErrorRealtimeTest(void)
*
*Discription: 故障实时在线检测函数
*
*Notes:       实时检测故障信息，如有故障则立即发送故障信息，故障恢复则发送恢复信息
*
*Define File: SystemBody.c
*
*****************************************************************************/
void  ErrorRealtimeTest(void)
{

      uint8_t *Tx_currentInfo = NULL;
      uint8_t temp_error = 0;
      uint8_t FlagCnt = 0;
      uint8_t BoundaryValue = 0;

      //1-13故障代码
      if (ScanData.allinfo.canData.canData1[3] != 0x00 && CanErrorFlag == 0)
      //1-13   CAN  Control error check   CAN数据中有故障信息且CAN通信正常
      {
            temp_error = ScanData.allinfo.canData.canData1[3];//记录当前故障
            if (CanErrorPast[temp_error] == 0)  //判断当前故障是否已经发送
            {
                  CanErrorPast[temp_error] = 1; //此故障为新发生的故障
                  . . . . . .
                  Send_Instruction(Tx_currentInfo);    //发送故障报警信息
                  releaseChar_ptr(Tx_currentInfo);
                  warnCounter = 1;     //故障信息持续发送计时
                  ErrorFlag[FlagCnt] = 1;  //  故障组标志置一

            }
      }  //如果CAN数据中没有故障信息
```

```
	else
	{
		if (ErrorFlag[FlagCnt] == 1) //判断是否有历史CAN故障组信息
		{
			uint8_t i = 1;
			. . . . . .
			ErrorFlag[FlagCnt] = 0;//	对应CAN故障组标志位置零

		}
	}
}
```

故障信息持续发送示例函数

```
/*****************************************************************************
*FunName:	void  ErrorMessageContinueSend(void)
*
*Discription: 故障信息持续发送函数
*
*Notes:		故障持续发送时，每隔1分钟发送一次所有的故障信息
*
*Define File: SystemBody.c
*
*****************************************************************************/
void  ErrorMessageContinueSend(void)
{
	if (warnCounter < 1) //故障报警计数为0时  则为1分钟计时
	{
		uint8_t *Tx_currentInfo = NULL;
		uint8_t temp_error = 0;
		uint8_t FlagCnt = 0;

		//1-13故障代码
		if (ErrorFlag[FlagCnt] == 1)
		//1-13   CAN  Control error check		判断CAN故障组是否有故障
		{
			temp_error = ScanData.allinfo.canData.canData1[3];
			CanErrorPast[temp_error] = 1;  //对应故障标志置一
			. . . . . .
			Send_Instruction(Tx_currentInfo);   //发送故障报警信息
			releaseChar_ptr(Tx_currentInfo);
			warnCounter = 1;//下一次计时
			ErrorFlag[FlagCnt] = 1;  //故障组标志置一

		}
```

```
        }
}
```

5.5 参数配置

功能：

参数配置功能主要设置智能终端正常运行的一些参数，智能终端通过串口与配置软件通信，实现出厂参数的配置。主要配置内容如下：

1) 叉车编号查询与设置

2) 温度报警值查询与设置

3) 电流报警值查询与设置

4) 运行时间查询与运行时间清零

5) 系统时间查询与系统时间同步

6) 温度传感器标定值设置

7) 系统参数查询与设置，包括通信方式选择和制动器控制选择

8) WIFI断开查询和设置

9) WIFI账号及密码查询与设置

10) GPRS端口查询与设置

11) 查询手机号

功能划分：

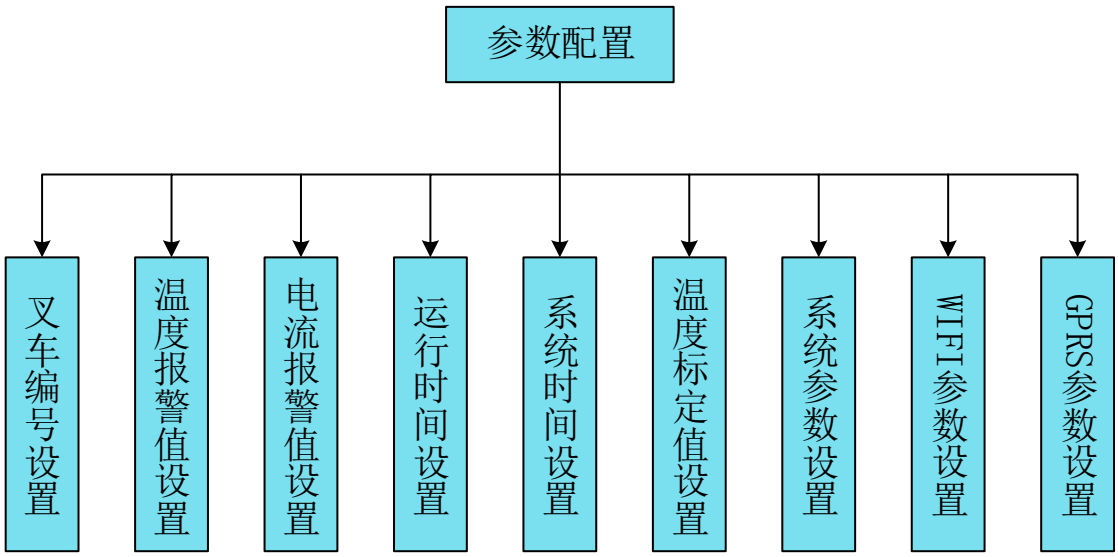

图8 参数配置功能划分

表12 全局变量

变量名	类型	含义	定义文件
DebugModFlag	Uint8_t	系统工作状态	Usart1Deal.c
ComModeSelect	Uint8_t	通信模式	Usart1Deal.c
ArresterControlEn	Uint8_t	制动器控制使能	Usart1Deal.c

表13 参数意义

DebugModFlag	变量含义
0	正常运行状态（默认状态）
1	实时信息状态
2	历史信息状态
3	配置模式

ComModeSelect	变量含义
1	单独WIFI通信方式
2	单独GPRS通信方式
3	WIFI和GPRS自动通信，WIFI优先
4	WIFI和GPRS同时通信

ArresterControlEn	变量含义
0	不控制和检测制动器
1	使能制动器控制和检测

参数配置功能函数：

参数配置函数的功能，根据单机配置指令编写，具体内容详见《单机配置指令》，函数主要功能是使系统进入配置模式，然后对各参数进行配置。示例函数如下：

```
/*****************************************************************************
*FunName:     void Usart1RxPackgeDeal(unsigned char * UsartBuff,unsigned char dataNum)
*
*In:          UsartBuff 接收数据缓存数组的首地址        dataNum 接收数据的字节数
*
*Discription: 解析和回应配置串口的数据
*
*Notes:       接收单机配置软件的数据，更系统工作模式并在配置模式下设置参数
*
*Define File: Usart1Deal.c
*
*****************************************************************************/
```

```
void Usart1RxPackgeDeal(unsigned char * UsartBuff, unsigned char dataNum)
{
	if (UsartBuff[0] == 'M')//串口模式选择
	{
		if (UsartBuff[1] == '?'&&dataNum == 2)//工作模式查询
		{
			USART1_SendMData("M+ok=", 5);//回应
			USART1_SendData(DebugModFlag + '0');
		}
		else
		{
			if (UsartBuff[1] == '='&&dataNum == 3)	//工作模式设置
			{
				if ((UsartBuff[2] - '0') < 4)
				{
					DebugModFlag = UsartBuff[2] - '0';//工作模式
					USART1_SendMData("M=", 2);//回应
					USART1_SendData(DebugModFlag + '0');
					USART1_SendMData("+ok", 3);
				}
			}
		}
	}
	else
		if (UsartBuff[0] == 'R') //历史状态下  查询历史数据
		{
			. . . . . .
		}
		else
			if (DebugModFlag == 3)  //进入配置模式
			{
				if (UsartBuff[0] == 'S')  //解析参数配置数据
				{
					switch (UsartBuff[1]) //参数配置类型
					{
					. . . . . .
					}
				}
			}
}
```

5.6 非功能部分设计

5.6.1 异常性处理

在程序中通过软件看门狗程序来处理软件运行异常，同时读取历史数据时进行再次校验，保证数据读取的正确性。

5.6.2 可靠性

经过长时间的测试，智能终端软件能够长时间正常运行；同时对软件进行了异常测试，软件处理并恢复正常运行。

6 硬件模块详细设计

6.1 数据采集模块

根据对现场需要采集的信息进行分析和统计，数据类型可以分为三种：开关量、模拟量和 CAN 通信数据。

开关量是指电动叉车控制所需的各种开关信号，所以开关信号主要分为两种，一种是 24V 电压信号，一种是 0V 电压信号。针对两种不同的开关信号，拟设计两种开关量检测电路。叉车中需要检测的开关量在 17 个以上，从节约处理器资源的角度考虑，可以通过增加转换电路使开关量检测占用资源少，并能够灵活扩展。

电动叉车需要检测的模拟量有电机温度和电机工作电流，温度范围在 0 到 100℃，电机温度测量拟选用 PT100 比较可靠，可以紧贴电机表面且不受电机的干扰。电机直流电流检测，电流范围在 0 到 100A，因为是大电流的检测，综合考虑可以用霍尔传感器检测比较可靠。

CAN 通信作为一种标准总线通信方式，应用已经非常广泛，可以参考现有的通信电路，具体的电路需要根据现场测试确定。

原理图如下：

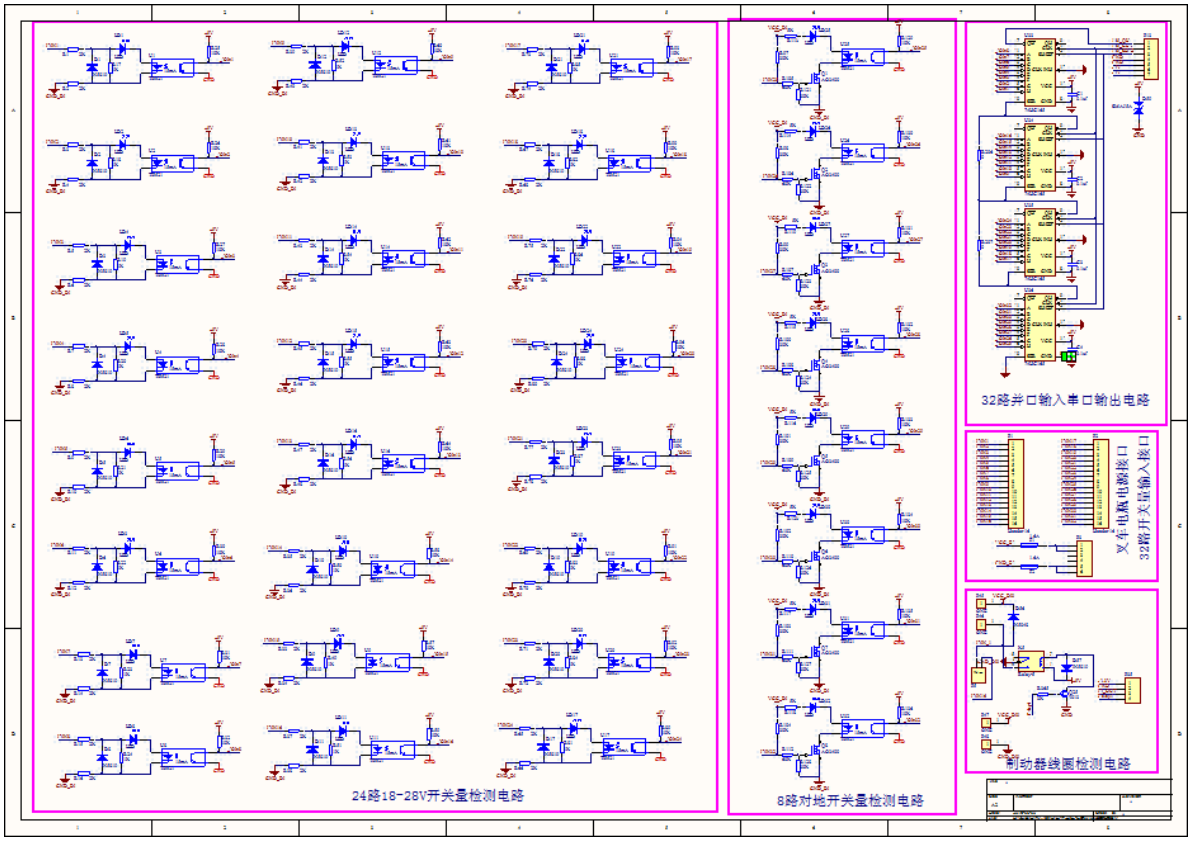

图9 数据采集模块原理图

6.2 电源供电模块

电动叉车的电源由 24V 的蓄电池提供，而诊断系统的工作设计需要提供足够功率的 5V 和 3.3V 电源，拟使用电源转换芯片将 24V 转换为系统需要的电压，同时还有考虑设计电源保护电路，防止大电流、大电压和浪涌信号冲击系统烧坏终端电路。

原理图如下

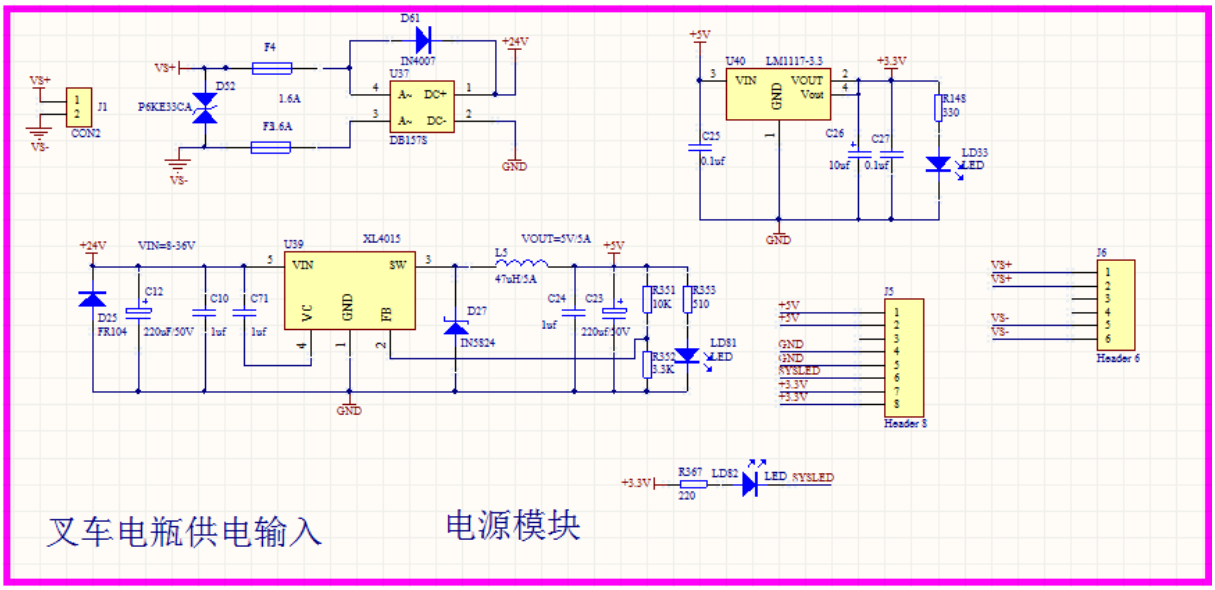

图10 电源供电模块原理图

6.3 GPRS通信模块

物联网设备的基本要求就是具有通信联网功能，拟设计故障诊断GPRS终端通信电路。GPRS通用无线分组业务，是一种基于GSM系统的无线分组交换技术，提供端到端的、广域的无线IP连接。通俗地讲，GPRS是一项高速数据处理的技术，方法是以"分组"的形式传送资料到用户手上。为保证通信的可靠性，GPRS通信模块采用高性能工业级嵌入式模块，内嵌自主知识产权的TCP/IP协议，可为终端提供高速，稳定可靠，永远在线的透明数据传输通道。

原理图如下：

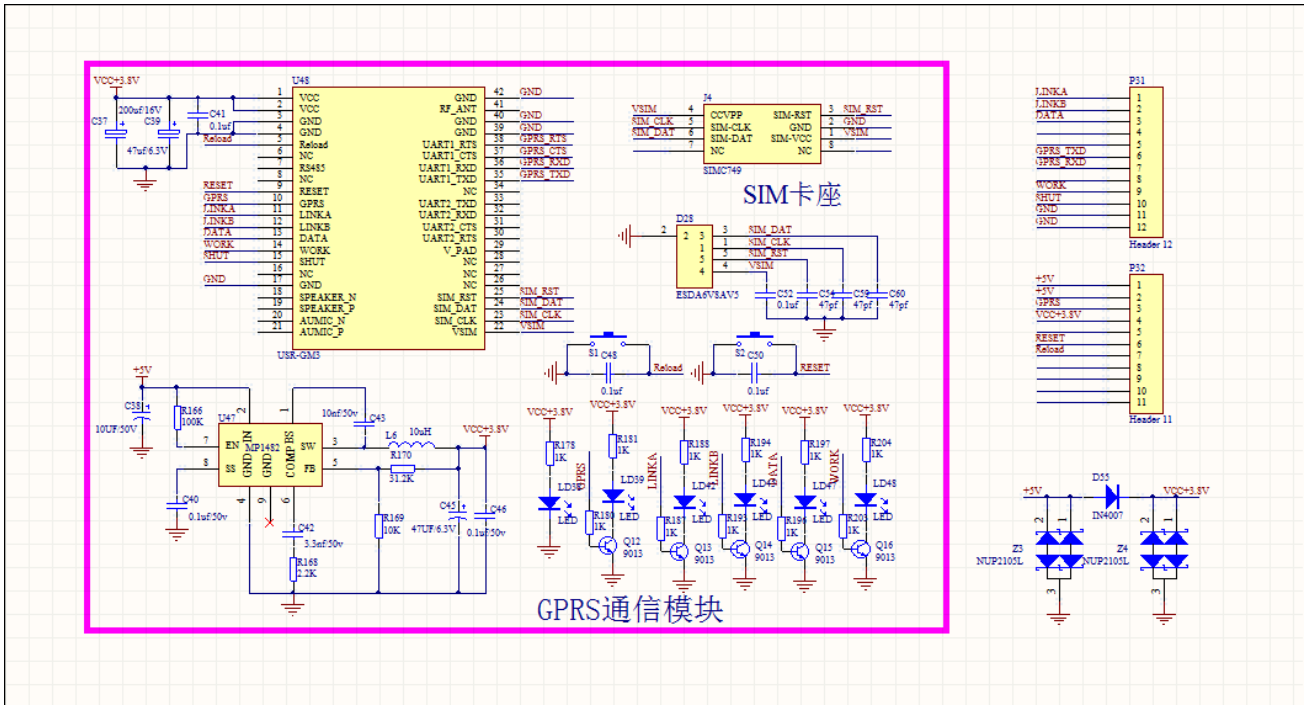

图11 GPRS通信模块原理图

6.4 WIFI通信模块

物联网设备的基本要求就是具有通信联网功能，拟设计故障诊断WIFI终端通信电路。WiFi是目前无线接入的主流标准，WIFI技术的成熟，为智能设备接入网络提供了便捷的方法。为实现局域范围内通信的可靠性，WiFi通信模块采用高性能工业级嵌入式模块，内置TCP/IP协议栈，能够使终端设备更好的加入无线网络，通过Internet 网络传输自己的数据。

原理图如下：

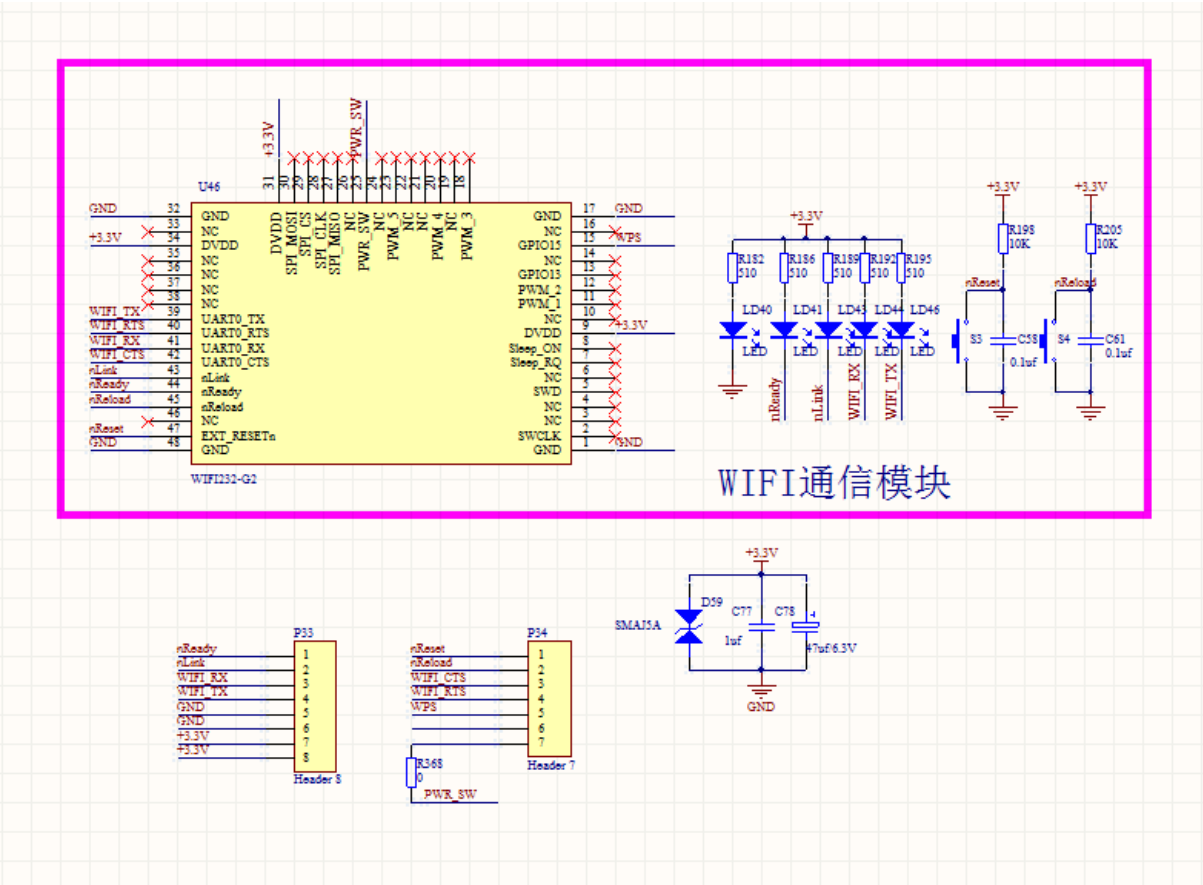

图12 WIFI通信模块原理图

6.5 处理器系统模块

智能终端离不开智能硬件的支持，拟设计主控电路选用高性能的嵌入式处理器，嵌入式处理器具有丰富的硬件资源，可以满足故障诊断终端需要的数据采集，如AD采集功能和CAN通信功能，硬件系统包括最小系统，外围供电电路，以及与智能终端的通信电路。处理器作为智能终端的核心组成部分，负责整个系统的数据采集、故障诊断和数据通信功能。为保障硬件系统的可靠和稳定运行，需要根据现场使用环境，合理规划设计硬件电路。

原理图如下：

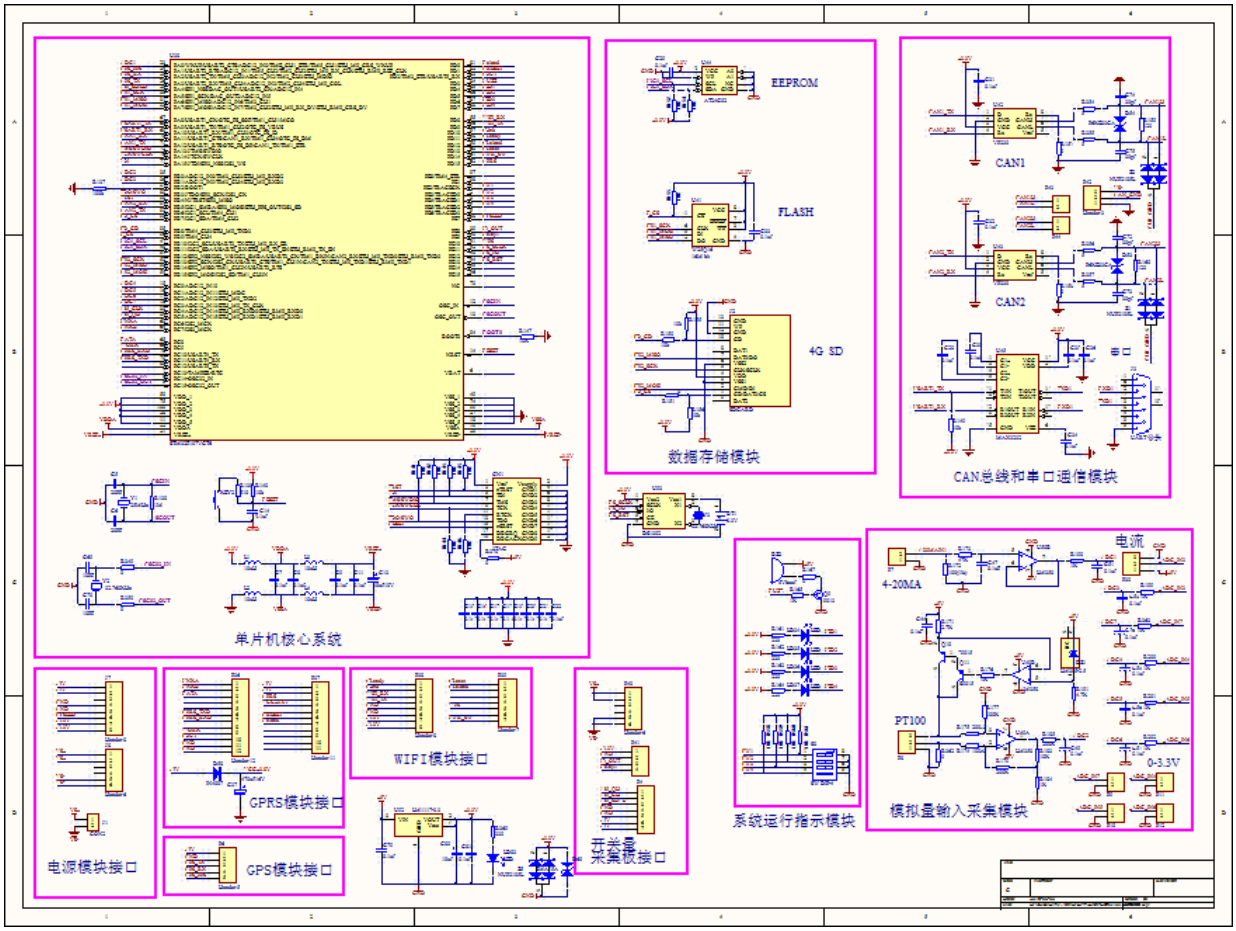

图13 处理器系统模块原理图