Enddevice**系统总体概要设计**

**历史记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **时间** | **版本** | **修改人** | **详情** |
| 2017/06/05 | V1.0 | 苏友江 |  |
|  |  |  |  |
|  |  |  |  |

# 硬件简介

## 微控制器选型

MCU:采用STM32L100RBT6，该微控制器主要应用于超低功耗的场合，系统时钟最高可达到32MHz，目前终端运行在32MHz下，基本参数如下表所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 内核 | flash | ram | eeprom | urart | I/O |
| Cortex M3 | 128k | 16k | 2k | 3 | 51 |

注：虽然官方手册写该芯片RAM只有10k，但是实际可以使用16k，按原厂技术支持说法

是后面6k比较不稳定。

## 外部晶振

HSE用户外部时钟源采用12M晶振，RTC LSE外部时钟源采用外部32.768k晶振。以下是MCU实际运行参数：

<SystemCoreClock :32000000>

<SYSCLK\_Frequency:32000000>

<HCLK\_Frequency :32000000>

<PCLK1\_Frequency :32000000>

<PCLK2\_Frequency :32000000>

## Uart串口使用

COM1：用于debug调试打印，默认波特率115200

COM2：用于连接zigbee模块/WIFI/PLC等

COM3：预留

## SPI总线使用

目前只使用SPI2：用于连接w5500以太网芯片/nrf24L01无线2.4G通信模块。以下是相关GPIO定义：

|  |  |
| --- | --- |
| CS | PB12 |
| SCK | PB13 |
| MISO | PB14 |
| MOSI | PB15 |
| w5500reset | PC6 |

## IIC总线使用

IIC采用GPIO模拟实现（很多人反馈stm32的内置IIC有问题，IIC访问过程中被中断有可能出现死机问题），用于和外部RTC芯片DS1338通信。

|  |  |
| --- | --- |
| SCL | PB10 |
| SDA | PB11 |

## 电能ADC采样

采样电路采用信号放大芯片TLC2274，共有4个放大器。

电能采样基本流程，交流电频率大约为50Hz，周期为20ms，要在一个交流电周期中采样32个点，采样周期必须为625us，因此必须设置定时器的周期为625us，当启动定时器后，就会每隔625us采集一个AD值，最终根据采样的AD值计算出电压、电流和功率等。

总共有三路ADC通道用于电能采样，GPIO分配如下：

|  |  |
| --- | --- |
| 漏电压采样 | PC1 |
| 电压采样 | PC2 |
| 电流采样 | PC3 |

## 开/关灯和调光

开/关灯是通过GPIOC的PIN4引脚来对继电器的控制，最终来实现对路灯的控制，目前RTU09C采用的是常闭的接法，主要原因是保证继电器故障后，路灯可亮着。PIN4低电平位开灯，而高电平为光灯。

RTU09C支持两种调光方式：DAC和PWM。

DAC是通过GPIOA的PIN4引脚实现DAC转换，可以输出0~3.3v，再经过电路转换最终实现0~10V的调光。

PWM是通过GPIOA的PIN6引脚实现脉冲宽度调制，最终输出5V/400Hz的PWM调光。

## 其它I/O使用情况

终端LED运行指示灯通过GPIOB的PIN9实现LED按照1s的周期闪烁。

# 软件概述

Enddevice的flash空间由BOOT、APP和远程升级存储区。Flash中各个程序空间分配如下表所示：

|  |  |  |
| --- | --- | --- |
| BOOT(8kbytes) | APP(60kbytes) | 升级存储区(60kbytes) |
| 8000000~0x8002000 | 8002000~0x8011000 | 8011000~0x8020000 |

Enddevice软件开发环境：keil MDK 4.54/keil MDK5.14集成开发环境，通常是用它来编译程序和在线跟踪调试（可以用j-link或者st-link等），软件的编辑和阅读通常采用source insight。

## 软件框架

## 代码结构

源代码工程目录结构图如下：



Src源代码文件目录结构如下所示：



## BOOT介绍

BOOT程序主要用于APP程序升级和引导，终端上电后，程序先运行BOOT程序。首先，初始化相关的模块，包括RCC系统时钟和UART1等。然后，延时3s等待COM1接收到字符，如果收到任意字符，则进入Y-modem进行APP程序升级。否则进入APP程序的引导。具体运行流程如下图所示：

RCC时钟配置

配置UART1，用于下载程序

是

检测串口3s内是否收到任意数据？

根据Y-modem升级应用程序

否

根据BootFlag引导指定的应用程序

## APP程序框架介绍

----------

Communication

Control

Misc

Measure

Embeded OS(lib-freertos)

HAI(src\_mdl)

RTU Framework

Driver Uart\GPIO\LED\EEPROM\RTC …(bsp-platform)

HAL(bsp-board)

**APP**

----------

**MDL**

----------

**HAL**

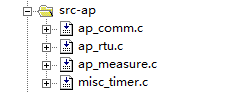
----------

Hardware

**PHY**

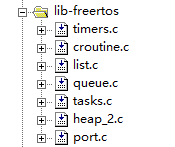
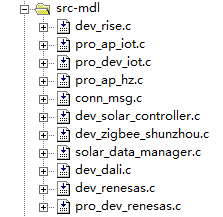
## APP层级结构和源文件介绍

APP层：功能逻辑应用层，包括实现通信、控制、计量等功能

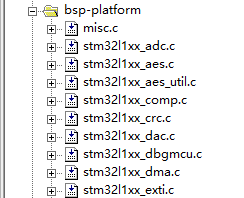
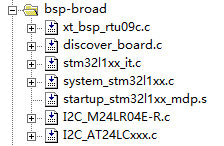


RTU09C总共有三个线程，按照优先级从高到低排列，分别在ap\_measure.c、ap\_commm.c和ap\_rtu.c三个源文件中。Misc\_timer.c中创建了两个软件定时器用于处理系统一些杂事，如喂看门狗、led运行指示灯、计时和电能累计等。

MDL层：中间接口层。包括系统框架、OS系统、硬件接口



HAL层：硬件抽象层。包括各种芯片和设备的驱动。



PHY层：物理层。就是实际的数字高低电平。

## APP文件结构



应用层：src-ap,prj

中间层：src-mdl，lib-freertos,src-os

驱动层：bsp-broad,bsp-platform，src-drv

## 嵌入式实时操作系统freeRTOS

FreeRTOS是一个迷你的实时操作系统内核。作为一个轻量级的操作系统，功能包括：任务管理、时间管理、信号量、消息队列、内存管理、记录功能、软件定时器、协程等，可基本满足较小系统的需要。

相对μC/OS-II、embOS等商业操作系统，FreeRTOS操作系统是完全免费的操作系统，具有源码公开、可移植、可裁减、调度策略灵活的特点，可以方便地移植到各种单片机上运行，其最新版本为8.2.1版。

在RTU09C中目前使用的freeRTOS版本是V7.5.2，主要使用了freeRTOS中的多任务管理、消息队列和软件定时器功能。

# APP各个模块介绍

软件模块介是按照CPU的运行顺序来依次介绍。

## 初始化

1. 版级支持包BSP初始化，包括系统时钟、GPIO、RTC、ADC、DAC和PWM等；
2. 看门狗初始化；
3. 注册各个AP应用，调用ApInit初始化各个AP，创建各个AP消息队列和线程，创建定时器处理杂项业务；
4. 启动freeRTOS操作系统任务调度；

## 电能测量模块-ap\_measure

电能测量线程的任务优先级最高，保证了电能测量的实时性。通过启动定时器，在定时器中断中程序中，启动ADC和DMA去采样AD值，再从采样的AD值计算电能，基本流程图如下所示：

计算交流电压值

启动采样定时器

计算漏电电压

Lamp index

计算电流值

计算有功功率

计算视在功率

计算功率因素

否

采样点数是否到达35个点？

是

去除AD采样中的直流分量

检测外部是否断电，如果断电，则保存电能数据

计算交流电的频率

一个测量周期完成

## 通讯连接模块-ap\_comm

Comm线程主要工作是消息的接收和派送。

接收的工作方式：从串口接收数据，并且经过解析和转换，最终通过消息队列，传送给ap\_rtu线程。

派送的工作方式：从消息队列中获取消息，并且经过转换和封装，最终通过串口发送给PLC模块或者zigbee模块。

# 通信模块介绍

## WIFI模块-ESP8266

通过COM2连接，默认波特率115200，无校验。目前软件只支持一路TCP连接，配置成功后，进入透传模式。

## 以太网模块-w5500

采用SPI2 连接，最多支持8个socket。

## zigbee模块-四信

通过COM2连接，波特率115200.

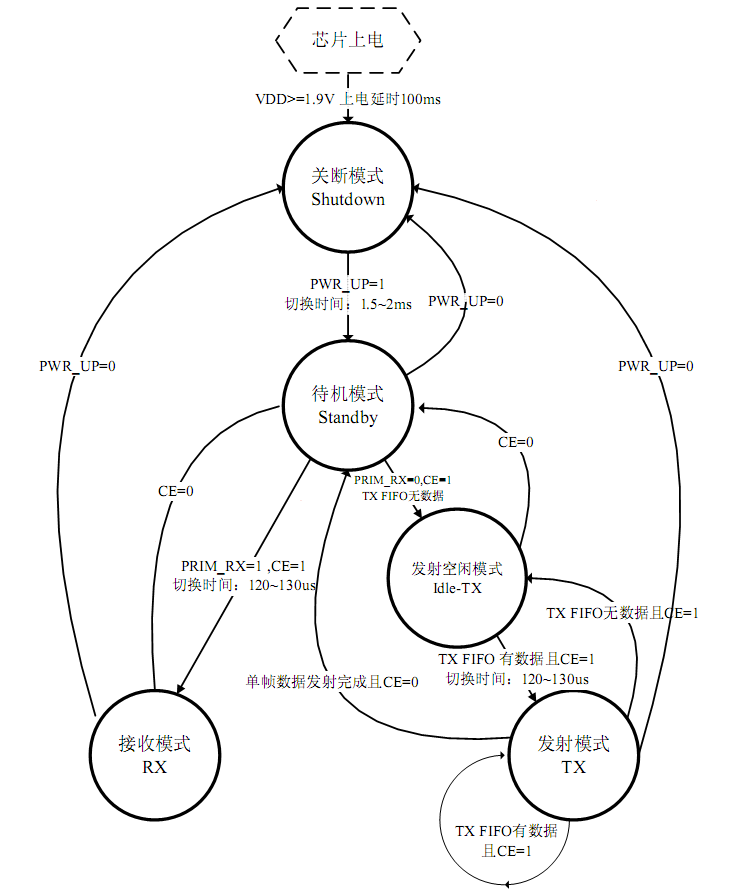
# nRF24L01网络设计

通过nRF24L01无线模块，实现简单的树型网络，主要用于低功耗自组网无线传感网节点。

## Nrf24L01介绍

nRF24L01是由NORDIC生产的工作在2.4GHz~2.5GHz的ISM 频段的单片GFSK无线收发器芯片。以下是参数特性：

* 宽电压工作范围，1.9V~3.6V，输入[引脚](https://baike.baidu.com/item/%E5%BC%95%E8%84%9A)可承受5V电压输入；
* 4线SPI通讯端口，通讯速率最高可达8Mbps；
* 工作频率范围为 2400MHz-2525MHz，共有 126 个 1MHz 带宽的信道；
* 发射功率可设置，有0:-12dbm到 7:7dbm；
* 数据速率：2Mbps/1Mbps/250Kbps；
* 6个数据通道（6个接收地址），满足多点通讯和调频需要；
* 数据包每次可传输1～32Byte的数据；
* 可配置为 Shutdown、Standby、Idle-TX、TX 和 RX五种工作模式；



注：nRF24L01是属于半双工通信，发送时无法接收数据，发送完必须马上切换到接收模式下。

## 组网

nRF24L01地址支持3～5bytes的长度，为了节省ram采用3bytes长度（AW\_3）。本网络主节点、路由节点和终端节点都只使用两个接收通道，因此有两个接收地址，一个定义为本机地址（网络中必须唯一，不能冲突）；另一个则是网络级别地址（主节点为一级网络，地址是0；直连主节点的路由节点为二级节点，地址是1；终端节点不中转数据没有网络级别地址，而是用地址ff作为广播地址，暂时没用）。

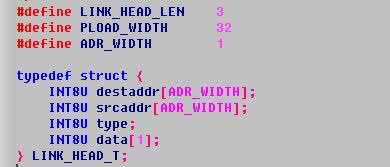
为了保证终端节点低功耗，因此终端节点不能一直处于接收模式，而是要定时进入待机模式或关断模式。这就导致了网络的工作模式，当主节点有数据要主动发送给终端节点，就不能马上发送，而是要等收到终端的心跳包后，立即发送待发送的数据包给终端。（终端每次发送数据后都会先切换为接收模式，大约等待10ms的时间，在切换为待机模式）

### 数据格式

nRF24L01发送接收数据最大只能32bytes，以下是帧格式定义:

|  |  |  |  |
| --- | --- | --- | --- |
| Head | | | User data |
| 目标地址1byte | 源地址1byte | 数据类型1byte | 用户数据29bytes |

以上的地址只是一个byte，但是nRF24L01的地址至少也要3bytes，因此把nRF24L01高两个byte固定为0x99, 0x99。



### 入网请求（type：0x01）

路由节点和终端节点在入网之前，不知道主节点的本机地址，但是知道主节点的网络级别地址是0x00，因此可以把入网请求帧发送给主节点。当入网成功后，主节点会告诉终端节点和路由节点关于主节点的本机地址，后续的通信都是采用本地机制。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Head | | | User data | | | |
| 0x00（一级网络地址） | 本机地址 | 0x01 | 路由地址 | 流水号 | 设备类型 | 路由次数 |

注：

* 目标地址：一级网络地址、二级网络地址…等，先发一级网络入网请求，没回应再发二级网络，以此类推循环，直到入网成功；
* 路由地址：每次经过路由器中转，则填入本路由器的本机地址；
* 流水号：终端/路由器每次发起新的入网请求流水号加1，取值1～255循环。（避免同一次请求，多个路由中转，导致主节点多次应答）；
* 设备类型：中端：0x00；路由器：0x01；
* 路由次数：发起入网请求时为0x01，每次路由中转则加1；

主节点接收到终端、路由器的入网请求，先从在线链表和离线链表中查找是否有相同的地址节点，如果有就更新节点信息，否则就会从空闲链表获取一个空闲节点，用来创建新的网络节点。

### 心跳请求（type：0x01）

路由节点和终端节点加入网络成功后，在没有其它交互数据时，每隔30s左右会发送心跳包。心跳包的用户数据部分暂时没有使用。