|  |  |
| --- | --- |
| 产品名称Product name | 密级Confidentiality level |
| ARM9集中器嵌入式软件 |  |
| 产品版本Product version | Total pages 共页 |
| V1.00 |

大型公共建筑能耗监测系统集中器

嵌入式软件详细设计方案

**产品名称：ARM9集中器嵌入式软件**

**编 写：靳占军 宋宝善**

**日 期：2016.03.10**

**烟台航天德鲁节能科技有限公司**

Revision Record 修订记录

| Date  日期 | Revision Version  修订 版本 | Sec No.  修改 章节 | Change Description  修改描述 | Author  作者 |
| --- | --- | --- | --- | --- |
| 2015.04.08 | 初始版本 |  |  | 靳占军、宋宝善 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 1 引言

## 1.1 背景

公司之前已开发的大型公共建筑能耗监测系统集中器（ARM9集中器），在新疆乌鲁木齐大型公建工程项目中已安装使用，在项目使用过程中发现了如下问题：远程通信不稳定、多厂家终端设备兼容不便、现场调试不便（必须在GPRS网络成功情况下远程桌面到服务器才可以调试）、无远程调试手段等问题。

公司能耗检测部急需大型公共建筑能耗监测系统集中器，但当前的ARM9集中器远不能满足需求。

## 1.2编写目的

本详细设计方案，旨在详细规划ARM9集中器嵌入式软件的设计思路、层次划分、模块间关系及具体实现方式，为项目相关人员提供方案评审和具体实施依据，并为项目后期维护提供依据。

## 1.3 参考资料

《ARM9-集中器嵌入式软件-需求单》

《ARM9-集中器嵌入式软件-立项报告》

《住建部大型公建能耗监测系统技术导则》

# 2．设计概述

## 2.1 需求概述

1. 和上位机通讯，支持RS485总线和GPRS/3G无线通讯，RS485总线支持有线组网。通讯采用XML文本格式。
2. 抄表周期设置，支持每天定点（最多24个时间点，精确到分）和按周期抄表（以分钟为单位，最小1分钟）两种方式，可设置。
3. 支持水、电、气、热四种设备集中自动抄表，每种表都支持多种厂家协议。
4. 支持客户按需设定需求的抄表数据项，满足不同客户需求。
5. 抄表数据可自动向服务器推送，同时支持上位机按时间主动请求数据。
6. 有完善的日志记录，便于过程查询；一些重点故障，可通过上位机查询，以便远程了解集中器状态。
7. 支持任意透传，以方便系统远程调试、维护。
8. 支持远程任意操控数据库，支持远程下发Linux系统命令，以方便远程控制。
9. 数据加密传输。

## 2.2 运行环境概述

1）硬件基于老版ARM9集中器硬件, 结构示意图如图2.1所示，使用米尔科技的ARM9核心板，控制器使用AT91SAM9x35，DDR2容量128Mbytes，Nand Flash 128Mbytes；外配有Jtag接口、RTC实时时钟等部件。集中器主板设有GPRS接口、上行485接口、1通道下行485接口、4通道MBUS通道接口，以及6只按键。



图2.1 当前ARM9集中器硬件结构示意图

2) 集中器运行Linux嵌入式操作系统，方便调用XML文件、ASE加密、MD5校验、SQL数据库等封装库的集成调用。

## 2.3 条件与限制

1) 老版ARM9集中器硬件没有外扩外部存储部件，抄表数据只能存储在Flash中，但Flash空间有限（除系统占用，剩余约40MBytes），历史数据存储量有限。

2) 没有LCD显示设备，虽然硬件设有6个按键，但是因无显示接口，不能支持集中器按键调试。

3) 老版ARM9集中器没有可参考文档，没有最终版源代码，没有IO驱动源代码，且我们没有具有较多Linux嵌入式开发经验的工程师。

# 3.详细设计方案

## 3.1 整体设计思路

ARM9集中器嵌入式软件设计分为“应用层”和“底层支持层”两个部分。首先在硬件平台的基础上做好底层支持，即底层支持层；在做应用层的功能设计开发时，只需要调用“底层支持层”提供的封装接口即可。这种分层设计便于系统设计中分工合作，并可以将可能发生的问题限制在局部，当有异常发生时，不至于改一发而动全身，方便系统调试和维护。

就集中器功能需求，“底层支持层”主要需要完成以下内容：

1. GPRS网络组建，及GPRS网络管理，当网路异常时可自动连接；
2. GPRS网络数据接收与发送支持；
3. 上行RS485接口数据接收与发送支持；
4. 下行RS485接口数据发送与接收支持；
5. MBUS总线接口数据发送与接收支持；
6. 所有接口的数据接收超时控制；
7. 上/下行485接口数据方向控制、4通道MBUS切换 所需要的IO驱动支持；
8. 集中器抄表过程设计，抄表过程可中断设计，便于网络远端随时控制占用抄表通道；并且不同表类型的抄表接口统一。

Linux集中器设计通信端口众多，设计中尽量统一不同设备但功能相同的函数接口，通过“设备编号”的形式进行自动区分，方便应用层使用。全部采用可重入设计，便于相同函数在不同线程中的同时调用运行。



图3.1 集中器设计分层划分

就集中器的功能需求及硬件平台来说，应用层与底层支持层的衔接，主要有上行Xml格式数据的收发和下行抄表数据的收发、解析、存储等处理两部分，具体如图3.2所示，至于GPRS网络组建与管理等，与应用层无关，只要“底层支持层”能够稳定支持GPRS网络数据收发，并合理管理网络，当网络异常时可自动重新组网即可。



图3.2 与应用层有关的底层支持层接口

在集中器中，接口有上行485接口、GPRS接口、下行抄表485接口、下行MBUS接口，都使用UART设备，其中GPRS相关需要分为管理和数据收发两个部分，接口种类较多，为便于管理，我们人为的对各个设备进行表3.1中定义，在本设计中根据设备号区分不同设备进行管理，对不同设备进行操作。

表3.1 集中器中各个接口设备号宏定义列表

|  |  |  |
| --- | --- | --- |
| 设备名称 | 设备号宏定义 | 说明 |
| 上行485 | #define UP\_COMMU\_DEV\_485 1 |  |
| 上行GPRS | #define UP\_COMMU\_DEV\_GPRS 3 | 为方便GPRS网络管理与数据收发，将GPRS定义这3钟虚拟设备。 |
| GPRS模块AT指令 | #define UP\_COMMU\_DEV\_AT 4 |
| GPRS网络数据 | #define UP\_COMMU\_DEV\_ATIPD 5 |
| 下行抄表485 | #define DOWN\_COMMU\_DEV\_485 6 |  |
| 下行MBUS | #define DOWN\_COMMU\_DEV\_MBUS 7 |  |

下面首先分别定义图3.2中所示数据收发接口形式，之后在下文介绍具体实现方式：

* 上行xml数据发送：

void FileSend(uint8 dev, FILE \*fp)，其中dev指定设备号，可使用UP\_COMMU\_DEV\_485和UP\_COMMU\_DEV\_GPRS；\*fp为要发送文件的文件标识符。此函数通过dev指定设备自动将\*fp指向的文件全部发送出去。如果使用RS485接口，自动进行数据收发方向的控制。

* 上行xml数据接收：

uint8 FileGetch(uint8 dev,uint8\* data,uint16 OutTime)，其中dev指定设备号可使用UP\_COMMU\_DEV\_485和UP\_COMMU\_DEV\_GPRS；指针\*data指向接收数据存储区，Outtime为等待时间，范围为0-65535，单位为ms，如果OutTime=NULL，则一直等待直到数据到来。使用此函数，如果在OutTime时间内指定设备有数据到来，则返回接收到的数据，如在OutTime时间内无数据到来，则返回超时错误。

* 下行数据发送：

void DownDevSend(uint8 dev,uint8\* buf,uint32 n)，其中dev指定设备号，可使用DOWN\_COMMU\_DEV\_485和DOWN\_COMMU\_DEV\_MBUS；指针\*buf指向待发送数据起始位置，n为发送数据字节数。调用此函数可将从buf地址开始的n字节数据，通过dev指定设备发送出去。如果使用RS485接口，自动进行数据收发方向的控制。

* 下行抄表数据接收：

uint8 DownDevGetch(uint8 dev,uint8\* data,uint16 OutTime)，其中dev指定设备号，可使用DOWN\_COMMU\_DEV\_485和DOWN\_COMMU\_DEV\_MBUS；指针\*data指向接收数据存储区，Outtime为等待时间，范围为0-65535，单位为ms，如果OutTime=NULL，则一直等待直到数据到来。使用此函数，如果在OutTime时间内指定设备有数据到来，则返回接收到的数据，如在OutTime时间内无数据到来，则返回超时错误。

## 3.2 系统线程划分

集中器运行Linux系统，采用多线程设计，如图3.3所示，共设置有图中10个线程，其中箭头表示线程功能的依托关系，箭头指向线程功能实现需要依托箭头起始线程。



图3.3 集中器线程划分

* 上行485接口数据接收、处理线程，用于实时检测上行485接口是否有数据到来，有数据到来则写入相应队列，供上行485总线xml数据接收解析处理线程调用。
* GPRS接口数据接收、处理线程，用于实时检测GPRS模块是否有数据输出，如有数据则写入GPRS管理队列和网络数据接收队列，然后由GPRS网络管理线程负责GPRS网络管理相关，由GPRS网络数据接收、处理线程负责处理GPRS网络数据。
* 下行485接口数据接收处理线程，与MBUS接口数据接收处理线程，用于实时检测有没有抄表数据返回，如有则将数据写入相应队列，供抄表及数据处理线程解析、处理、存储等。
* 长数据上推处理线程，用于抄表历史数据的或表基础信息的主动上报。

## 3.3 底层支持层

### 3.3.1 各接口数据处理方式

* **数据接收**方**式**

上行485接口、GPRS接口、下行485接口、MBUS接口数据的接收，全使用非阻塞select函数的方式实现，接收到数据后将数据存入对应的数据队列，供相应线程读取数据使用。

各个接口数据接收流程如图3.4所示，随时等待接收串口数据到来，数据到来后将数据写入对应的数据队列，并发送相应的数据接收信号量。



图3.4 各个接口接收数据流程图

* **数据接收超时处理**

各个接口数据写入对应数据队列之后，其他线程从对应数据队列中读取数据，如【3.1整体设计思路】部分所述，对应函数为uint8 FileGetch(uint8 dev,uint8\* data,uint16 OutTime)和uint8 DownDevGetch(uint8 dev,uint8\* data,uint16 OutTime)，通过设备号dev决定读取数据的队列，OutTime为等待数据超时时间。数据读取及超时处理机制如图3.5所示。



图3.5 数据读取及超时处理机制流程图

* **数据发送**

如【3.1整体设计思路】部分所述，数据发送主要由void FileSend(uint8 dev, FILE \*fp)和void DownDevSend(uint8 dev,uint8\* buf,uint32 n)两个函数完成，应用层准备好将要发送的文件或数据，调用相应函数，并指定发送端口即可完成数据的发送。在数据发送时，相应功能函数应自动根据发送数据字节数及波特率，自动决定延时时间，保证数据发送完整，并具有较高的实时性。流程如图3.6所示（方向切换针对485接口）。



图3.6 数据发送流程图

### 3.3.2自动抄表过程

为满足上位机不定时透传/抄读集中器下挂设备的需求，集中器自动抄表任务设计为可中断式。为实现对不同设备的集中抄读，在抄读每一只设备前都根据本设备的具体参数进行通断或端口设置。抄表任务流程如图3.7所示。

在图3.7中可以看出，在抄读每一只设备之前系统先请求通道信号量，请求到信号量后才执行设备抄读，否则一直等待通道信号量的到来，抄读完本设备后立即释放通道信号量。这种设计方式可保证每只设备的抄读过程不被打断，但相邻两只设备的抄读过程允许打断，在全抄表过程中允许上位机透传或实时抄读操作设备。

在抄读每一只设备前，都会根据本设备的基本信息设置通道、串口参数等，但在设置这些参数前首先检测当前与将要设置的是否相同，只有在不相同的情况下才进行设置，相同则不再设置，这种方式既可以保证如果抄表过程被中断，参数被修改后系统可以自动恢复正常继续抄表，又可以保证执行效率（不同通道切换时等待总线稳定需3秒）。另外，为了进一步提高全抄表效率，集中器应自动按照通道号对设备信息进行划分，全抄表时按照通道进行抄表，以防全抄表时通道不停地切换。



图3.7 自动抄表任务流程图

### 3.3.3 GPRS管理方式

集中器支持GPRS无线网络TCP/IP组网，GPRS网络管理线程主要完成GPRS模块的初始化、网络组建、登录服务器、网络维持等工作。因GPRS网络是短连接网络，占用的通信链路超过一定时间没有数据收发则会自动切断，我们为了维持网络的长连接，采用定时发送心跳帧的方式维持。当集中器长时间接收不到心跳帧响应时，可自动重新组建网络，详细流程如图3.8所示。



图3.8 GPRS网络管理流程图

### 3.3.4 GPRS网络数据接收

GPRS网络数据接收处理独立于GPRS网络管理线程，由专门的线程负责。根据GPRS模块ATC手册，当GPRS网络有数据到来时，GPRS模块会主动发出标志信息，以西门子MC52I模块为例，当网络有数据到来时模块会主动发送“\r\n^SISR: 1, 1\r\n”信息提示，当集中器收到该提示信息后即可通过读取指令“AT^SISR=<srvProfileId>, <reqReadLength>”读取网络数据，GPRS模块返回“^SISR: <srvProfileId>, <cnfReadLength>[, <remainUdpPacketLength>]”，其中cnfReadLength即为当前网络中的数据字节数，集中器可通过GPRS模块从网络读取这些字节，并进行其他处理。集中器中GPRS网络数据接收处理流程如图3.9所示。



图3.9 GPRS网络数据读取流程图

## 系统应用层

### 远程升级断点续传

ARM9集中器支持远程升级功能，当集中器程序有更新时，在远端即可实现对集中器的更新，并支持断点续传功能，具体流程如图3.10所示。

当开发人员编译好一个新的嵌入式程序后，将其MD5校验值、字节数告诉维护人员，并将新程序打包成.tar.bz2格式的压缩包交给维护人员。

服务端开启远程升级过程，向集中器下发升级准备命令（帧索引为0），把新程序的字节数、MD5校验值、压缩包分解后的帧数告诉集中器。

集中器收到升级准备命令后，首先比较新旧文件的MD5校验值，如果不一致，向服务端返回“MD5不一致”应答；如果一致，向服务器返回“MD5一致”应答。

服务端如果接到“MD5一致”应答，终止升级过程；如果服务端接到“MD5不一致”应答，则向集中器下发“询问缺帧情况”命令。

集中器根据初始参数中的“总帧数”一项，查询当前缺少的帧，将其组织成字符串列表发送回服务端。

服务端根据“缺帧列表”，每间隔一段时间（通常是300毫秒）就向集中器下发一帧数据。由于xml文件属于纯文本格式，不能直接发送二进制数据（上下位机在解析二进制数据时可能会出现错误），而如果将每字节对应的ascii码直接发送，则会把原始的数据长度增加一倍（100%），传输效率很低，所以在发送原始数据（二进制）之前，要将二进制数据编码为base64格式，base64格式编码只增加原始长度的33.33%。每条数据帧包含的信息有：本帧原始数据（二进制）的字节数，本帧的索引号（从1开始），本帧原始二进制数据的CRC16（modbus-RTU）校验值。集中器根据这些信息判断当前接收到的数据是否正确，如果正确将其缓存，如果不正确就将其丢弃。

服务端在发送完所有的缺帧后，再次查询集中器的缺帧情况，如果返回的列表不为空，则再次重复上述过程，直到集中器返回的缺帧列表为空为止。

当服务端接收到空的缺帧列表后，向集中器下发“合并”命令。集中器在接到“合并”命令后，按照帧索引的顺序依次将解码到的数据写入临时文件，并改其后缀名为“.tar.bz2”。再用shell命令将其解压，计算解压到的文件的MD5校验值，如果与服务端发来的值一致，则将其更新至本地数据库，并用新程序替换旧程序，向服务端发送“合并成功”应答，并重启系统；如果MD5值与服务端发来的不一致，则向服务端返回“合并失败”应答。

服务端如果接收到“合并成功”应答，则终止升级过程；如果接收到“合并失败”应答，则重新开始上述的升级过程，如果连续3此都收到“合并失败”应答，则终止升级过程，并由维护人员排查集中器或者网络等出现的问题。



图3.10 集中器远程升级流程图

### 3.4.2 数据库管理

数据库管理的主要内容有：

1. 系统参数管理：读取、配置主要的系统参数。当程序启动后，读取系统参数将其放入全局数组，方便上层模块调用，不必与数据库频繁交互。
2. 仪表地址信息管理：程序启动后将数据库内的表地址信息读取到全局链表内，方便抄表模块读取，避免了频繁读取数据库。仪表信息维护方面，前期只支持全部更新，后期完善支持单个或某些表地址的修改及增删，但前期局部修改及增删可以用数据库透传协议实现。
3. 仪表数据项配置：由于单体能耗系统应用于不同的场合，不同场合可能客户关注的数据种类是不一样的。比如热量表，有的客户需要看累积热量即可，而有的客户可能关注当前的流速、进出口温度、累积流量、累积热量等，所以灵活配置抄表数据项对于后期软件的维护作用很关键。数据项配置表的结构见《数据库设计方案》。当程序启动后，将数据项配置表的信息读入全局链表，每次要向数据库插入抄表历史数据时，都要读取此链表。
4. 历史数据表管理：根据不同的仪表类型，将抄表结果存入历史数据表内；在进行历史数据上传时，先将所有符合查询时间点条件的历史数据，按照仪表类型分别存入不同的全局链表里，然后逐条读取、上传历史数据。

数据库管理模块的基本思路是，逐行将从数据库读取的数据压入一个全局链表内。当需要对其遍历时，传入一个回调函数，这个函数的参数必须要用对应的数据行结构体指针，其他参数可自行定义。

注意：现在的数据库管理模块对于全局变量没有访问控制，即不是线程安全的，后期一定要加上访问控制，使这个模块更加安全、稳定。

### 3.4.3 协议管理

具体的协议格式详见《ARM9-集中器嵌入式软件-通讯协议设计方案》。

协议管理模块分为：1) 临时xml文件管理。 2) 协议分配：根据服务端下发的协议，执行相关动作。

1. 临时xml文件管理：当前组网设备（GPRS、RS485或者是以太网）从服务端读取（GPRS或者RS485)到协议数据后，检测xml的头和尾，把完整的一帧xml数据写入到临时xml文件里，再将其发送到协议分配模块。程序分配了15个临时文件给上行、下行通讯使用，如果通讯线程较少，则15个足够用。如果以后加通讯线程，需视情况增加临时文件的数量。当程序要使用临时文件时，先请求一个临时文件信号量（如果当前15个临时文件信号量都被占用，则请求的线程将被阻塞，等待其他线程释放其中一个临时文件时，请求线程再对其进行竞争），在使用后一定要释放掉临时文件信号量，给其他的线程使用。
2. 协议分配：从临时xml文件管理读取文件内容到内存中（使用c语言库libxml2），以树的结构存储（xml文本也是树结构）。然后读取帧头，获得此帧的基本信息：发送方编号（由服务端维护，意为在当前的能耗监测系统中，所有服务器的有效编号）、接收方（由服务端维护，本集中器只存储一个有效集中器号）、协议类型编码、操作编码。根据协议类型编码，调用不同的协议处理函数；根据不同的操作类型，对应的协议处理函数执行不同的操作。由于协议类型很多，所以程序中设计一个函数指针数组，用于索引协议处理函数，这样可以根据协议编码很方便的进行相关函数的调用，省去了啰嗦的***switch-case***语句。

### 3.4.4 Shell透传协议

用glibc的库函数popen来打开一个管道，将服务端发来的shell命令原封不动的传给popen，然后用fread读取管道中的数据到缓存中，最后返回到服务端。

### 3.4.5 抄表透传命令协议

集中器常规的抄表类型为水、电、气、热四种仪表。对于其他需要集中器采集或控制的设备，需要服务端直接发送抄表或控制命令，比如温湿度传感器、风速传感器，这些终端设备需要连接到一个数模转换模块，集中器通过读取数模转换模块对应通道上的数字量来获取当前终端的读数。而对于中央空调等控制系统，也通过抄表命令透传协议直接由服务端发送控制命令给集中器，集中器将命令再转发给对应的通道上，实现对终端设备的控制。

上位机下发透传命令时，信息包括：通道号、串口配置参数（串口号、波特率、停止位、数据位、校验位）、仪表类型和仪表编号、透传命令。透传命令每个字节之间用空格（” ”）分隔，而串口参数用逗号（”,”）分隔。使用glibc的strtok\_r函数分离出具体参数，strtok\_r函数的原型是：char \*strtok\_r(char \*str, const char \*delim, char \*\*saveptr)，其中str指向源字符串，delim指向分隔符集合（每个字符都算一个单独的分隔符，而不是一个分隔串），saveptr用于存储每次处理后（strtok将分隔符替换为'\0'）的字符串首地址（初始化为NULL即可）。strtok每次都将处理后的字符串首地址返回给调用者，所以要按照相应参数的字段域个数，设置一个指针数组，用于存储字符串首地址序列。

解析到透传参数后，根据通道号，请求对应设备的信号量，并设置对应通道的串口参数并切换到目标通道上（为稳妥起见，设置三遍串口参数）。根据通道号，选取对应的接受队列，首先将其清空以避免老数据扰乱抄表数据。然后发送透传命令帧。读取返回结果时，设置超时等待时间为1秒，避免超时等待太短而错过仪表返回数据。将仪表返回的数据返回给服务端。最后释放对应设备的信号量。

### 3.4.6 数据库透传协议

数据库透传协议是在shell透传协议的基础上封装而来，当服务端发来一条sql语句后，集中器组织一条shell命令:“sqlite3 db\_name “sql””，按照shell透传的方式将结果返回到服务端。

### 3.4.7 历史数据上传协议

首先初始化一个消息队列的文件描述符g\_uiQmsgFd，用以操作消息队列中的消息。当接收到服务端下发的查询历史数据命令或者每次抄表完成后，都要向此队列中发送上传历史数据的消息，消息中包括协议类型、抄表时间点、网络设备号（GPRS、 RS485，当使用RS485组网时，不允许集中器自己上推数据，会引起总线数据的混乱）等信息。单独设置一个上传数据线程，用于接收上传数据的信号，并进行数据处理，最后将数据上传到服务器。

当上传数据线程接收到上传数据信号时，根据消息体内的协议类型，调用上传数据函数，并将抄表时间点、网络设备号信息传递给被调用函数。

上传处理函数首先根据时间点将水电气热四表的目标时间点的历史数据读取到四个链表中，再计算出各自的帧数、四表的总帧数、四表的总行数等参数。

逐个对各类型的表进行上传，上传时使用自定义函数wr\_his\_xml组织xml的内存内容，通过这个函数可以很方便的组起一帧上传数据。wr\_his\_xml使用简单的状态跳转模式来根据不同的数据类型写入不同的xml段，其状态跳转图如下：

当前是最后一帧?

3.11 数据上传xml文件创建状态跳转图

当前帧的数据保存到临时xml文件后，就将其发送到对应的网络设备上，并等待服务端发回的应答信号量，阻塞等待时间为10秒钟。当连续5次都接收不到服务端的应答后，终止上传过程，否则继续上传下一帧的数据，直到全部数据上传完毕。

### 3.4.8 仪表地址信息更新协议

服务端分多帧下发表地址信息，集中器逐条将地址信息读取、压栈（压栈前先清空原栈内的表地址信息），当全部读取成功后，清空原有的表地址信息，再将栈内的地址逐条写入到数据库，而表地址信息栈在读取xml文件时就存储了全部地址信息，所以此时不必重新再从数据库读取一遍地址信息。

### 3.4.9 基本参数配置协议

逐个将读取到的参数信息写入链表中。当服务端下发的所有参数都读取完成后，调用自定义函数set\_sysconf将系统参数写入数据库。写入数据库时，程序首先检查当前配置数据表中有没有要配置的数据行，如果有，则更新其配置值；如果没有，则将当前配置插入配置数据表。

注：虽然“设置集中器抄表模式和抄表周期参数协议”在协议定义中独立于“基本参数配置协议”，其实在程序处理时都调用同一个自定义函数write\_sysconfig。

### 3.4.10 RS845组网

当前版本的ARM9集中器有两个RS485通道，一个用于下行抄支持485通讯方式的仪表数据（/dev/ttyS1），一个用于与上行服务端交互数据（/dev/ttyS2）。

设置一个单独的线程管理服务端发来的数据，将上行485芯片的方向设置为“接收”，并设置相应的串口参数（由于受限于当前硬件，上行485口最高速度只能设置为19200bps）。线程不断的检测当前设备是否有数据到达，如果有数据到达，则将数据缓存到上行485队列中。再设置一个线程，检测上行485队列中是否有完整的xml通讯帧，如果有，就调用协议处理函数对其进行相应处理。

当要向上行485设备发送数据时，首先将上行485芯片的方向设置为“发送”，再将要发送的数据写入对应的文件描述符。当全部数据发送完毕后，将上行485芯片的方向设置为“接收”，以接收来自服务端的数据。

### 3.4.11 四表合一及多厂家表协议兼容

针对每种类型的仪表，设置一个协议索引数组，数组的索引值就是下发表地址信息中的协议类型。协议索引数组中的内容包括：串口设置函数指针数组索引值、数据标识符、前导符个数、通讯总线类型（MBUS、 RS485），电表的协议索引数组还包括通讯协议类型（modbus、645）。通过这些信息，结合地址信息中的协议类型值，就能正确调用当前仪表对应的串口参数、总线类型、抄表帧的组帧、应答帧的解帧等处理函数。

每当添加一种新的厂家的仪表时，只需添加其相应的参数到协议索引数组中，再针对此种仪表单独写一套组帧、解帧函数即可，后期维护的工作量大大减轻。

对于各种类型的仪表，都有几个重要的数据项要提取，针对每种类型的仪表设置一个结构体，将读取到的数据存储到这个结构体中再交给数据库处理函数，即可完成模块之间的交流，在一定程度上固定了抄表数据格式。

### 3.4.12运行日志管理

因当前集中器存储空间有限，设置日志文件的最大值为5M（5\*1024\*1024字节），每插入一条日志前，都检查日志文件大小是否超限，如果超过5M，则将其删除。

使用自定义函数write\_log\_file(char\* buffer, unsigned buf\_size)进行日志的插入，buffer为要插入的日志内容，buf\_size为本条日志的字节数。插入日志时，write\_log\_file自动写入当前时间信息，而当前日志的文件名、函数名、在代码中的行号则由调用者提供。