基于嵌入式 Linux 的远程抄表系统集中器的实现①

朱仕亮　何克右(武汉理工大学 计算机科学与技术学院 湖北 武汉 430063)

摘　要： 通过分析比较国内外各种现有远程抄表系统的优点和不足，结合当前计算机技术和通信技术的发展和作者的实践经验，实现一种基于 S3C2440 处理器和嵌入式 Linux 的设计方案；具体说明了该方案的系统架构，并详细介绍了集中器嵌入式软件实现的关键技术。

关键词： 远程抄表；集中器；嵌入了 Linux

Implementation of Remote Meter Reading System Concentrator Based on Embedded Linux

ZHU Shi-Liang, HE Ke-You

(Department of Computer Science and Technology, Wuhan University of Technology, Wuhan 430063, China)

Abstract: This paper firstly introduces the background and current situation of Remote Meter Reading Systems, and points out their merits and shortcomings. Then, a design and implementation of a Remote Meter Reading System's concentrator based on embedded Linux is illustrated in detail according to the author's development experience in this field. This paper contains the architecture of the whole Remote Meter Reading System, the structure of the concentrator and its software design method.

Keywords: remote meter reading system; concentrator; embedded Linux

1　引言

远程抄表系统，又称为自动抄表系统，是一种利用传感技术、通信技术和计算机技术，对“三表”(即电表，水表，燃气表)的计量数据进行自动读取并传送到远程信息中心进行数据统计、分析和计算的计算机应用系统。早在上世纪六十年代，美国电话电报公司(AT&T)便开始了自动抄表系统的实验开发，但是这种基于电话系统的解决方法由于成本高，未能大规模在实际中应用。随着技术的进步，成本的降低，大规模的使用远程抄表成为可能，特别在那些无法进行人工抄表的场合[1]。我国从上世纪九十年代开始研发用于远程抄表的智能电子表，现在市场上已经出现了各种技术类型的远程抄表系统。这些抄表系统一般由以下三个典型的部分构成：1)表计信息采集模块，负责将表具计量信息数字化并储存和传输 2)网络通信模块，负责收集采集模块采集到的数据并传送给远程的信息处理中心。3)信息处理中心，负责对抄表数据进行统计，分析和计算，以及提供用户查询等功能。目前，网络通信模块的主要载体是集中器，其硬件平台大部分还是单片机，软件平台大多采用单流程的循环控制系统或者带有简单的操作系统，他们的优点是开发成本低，能够满足一些基本的功能要求，但主要存在以下不足：1)硬件平台依赖性强，不具通用性，不利于软件的开发，升级和移植；2)单流程的循环控制系统，其响应命令的平均时间较长，使用不方便。3)硬件平台简单导致了它只能使用简单的嵌入式操作系统，这使得软件的开发难度加大，缺乏广泛的应用接口支持，难以适应长期的发展需要[2]。

2　系统整体结构

采用 ARM9 内核的 S3C2440 处理器和嵌入式 Linux 操作系统设计的高性能集中器可以很好地解决上述问题。整个抄表系统分为三个部分：信息中心服务器(一般为供能公司，如燃气公司等)， 集中器，网络表。如图 1 所示。

① 收稿时间:2010-02-04;收到修改稿时间:2010-03-22

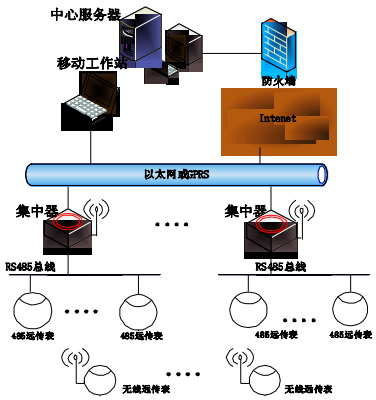

图 1 远程抄表系统整体结构图

该系统的表具都是智能表，集成了通信和数字化计量模块，能够响应集中器的控制命令，我们称之为远传表。他们和集中器的通信方式有两种，一种是基于 RS485 总线。这种情况下，如果不加 485 中继器，普通 485 总线只能挂 32 块表具。一种基于 CC1100 模块的短距离自组网络无线通信模式，这种情况下，中继器可以连接的表具数目受现场条件对无线信号的影响程度不同而不同。无论是哪种通信方式，对集中器来说，他们都是串口通信。采用远传表的好处是不需要专门配置采集模块，使得整个系统集成度更高。集中器通过网络接口接入小区以太网，可以响应来自移动工作站以及远程信息中心的控制命令，并且可以主动向远程信息中心报告表具异常信息。移动工作站用于管理小区范围的用户抄表，并向远程信息中心提供抄表数据。远程信息中心也可以直接和集中器通信，实时抄收、监控表具的运行情况。可以看出，集中器是整个远程抄表系统的通信中枢，对系统的稳定性起到关键的作用。

3 集中器的硬件平台

根据集中器的现实需求和扩展需求，我们采用基于 ARM9 内核 S3C2440 处理器的 mini2440 开发板搭建硬件开发平台。S3C2440 是一款专门针对功耗和成本敏感的应用系统而设计，它基于 ARM920T 内核，标准 0.13μm CMOS 工艺，核心电压 1.3v，外围设备电压最高 3.3v，带外扩 SDRAM 控制器，3 路串口控制器，2 个 USB 主机控制器，1 个 USB 设备控制器，触摸屏接口等丰富的接口控制器。本系统中的手持配置工具和 GPRS 模块和集中器都是通过串口进行通信的，需要注意的是，S3C2440 不具备完整 RS232 串口控制，它的 UART0 和 UART1 只有 TxD,RxD,RTS 和 CTS 四个功能引脚，UART2 只有 TxD 和 RxD 两个功能引脚[3]。但是一般的 GPRS 模块都需要载波检测引脚 DCD 来判断 GPRS 是否在线，以及 DTR 来控制 GPRS 模块的状态，如启动和关闭 GPRS 模块。本例中使用的 GR47 模块便需要 DCD 和 DTR 引脚才能正常操作[4]，解决办法是，从 S3C2440 中挑选两个空闲的 I/O 口来充当 DCD 和 DTR，然后把这两个引脚和 UART 控制器的其他现有引脚一块接到 MAX3232 串口扩展模块上，然后再把 GPRS 模块接在这个扩展模块上即可。程序中可以写一个针对这两个引脚的驱动程序来操作 GPRS 模块。

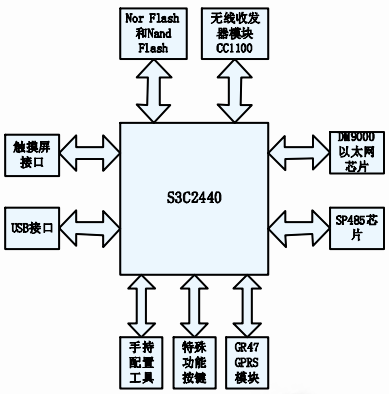

图 2 集中器的硬件接口模块组成

连接表具和集中器的 485 网络芯片采用半双工的 SP485 芯片，成本低，数据传输速率高达 5Mbps，完全可以满足表具计量数据规模的通信量。如果表具采用无线方式，则使用 CC1100 无线收发模块，这是一款为自动抄表系统设计的具有极低功率的 UHF 无线收发器，可以工作在 315/433/868/915MHz 的 ISM/SRD 波段，根据中国的 ISM 频率划分，我们这里选择 433MHz 这个中心频率。

手持配置工具完成表具条码扫描输入，配置集中器所管辖的表具信息。它与集中器通过串口进行通信。预留的按键，可以用于通知集中器手持设备的接入和取出以及系统复位等功能。

Nor Flash 主要用于开发阶段的调试，开发完成后可以不用 Nor Flash。Nand Flash 主要用于操作系统和程序的存储，相当于 PC 机上的硬盘。触摸屏是为将来把集中器设计为可视化操作而预留的。

4　集中器的软件设计方案

4.1 集中器的功能需求

集中器是系统中的通信和信息翻译的桥梁，在本系统中，它主要实现的功能有以下几点：

1) 通过短距离无线射频或者 RS485 与表具通信，获取表具的运行数据并做相应的数据分析处理。此外，还需要对表具进行必要的控制。

2) 通过GPRS或者以太网与远程信息中心服务器通信，接收并分析服务器发过来的命令数据帧，同时向服务器上报表具运行的异常情况。

3) 手持配置工具通过串口接入集中器，初始化或者更新集中器所管辖的表具信息，以及设置集中器的运行参数，如设置集中器需要连接的服务器的 ip 地址和端口号等。此功能主要用于集中器的现场安装以及后期的简单维护。

4) 以上所有通信都需要遵循一定的协议准则，因而集中器就必须根据这项协议来分析收到的通信帧，并在发起通信的时候组装通信帧。

4.2 任务的划分

软件系统的整体方案是基于嵌入式 Linux 的多线程与多进程设计方案。根据集中器的功能需求，我们可以将系统划分为以下三类任务：1)抄表任务，包括定时抄表任务和周期抄表任务；2)通信帧解析任务；3)网络通信任务，包括发送和接收两个任务；4)集中器配置任务，响应手持设备对集中器的设置操作。在本系统中，各任务间具有如图 3 所示的数据联系关系。可以看出，除了定时抄表任务和集中器配置任务之外，其他任务都要操作通信帧队列，因而他们之间是数据紧密相关的，应当设计为多线程。需要注意的是，在 Linux 中，同一进程的多个线程共享一个定时器，而定时抄表任务和周期抄表任务都依赖定时器来触发其工作。所以，如果把定时抄表任务也设计为线程，那么就需要用软件设计技巧来模拟两个定时器的功能[5]。但这样做就使得简单的功能却实现得比较复杂，不具可读性，因而我们把定时抄表任务作为独立的进程来运行。集中器配置任务需要和手持配置工具进行通信，通信协议与集中器和服务器的通信协议格式类似，可以和帧解析任务共享一些分析校验通信帧的可重入函数，故这里把集中器配置任务设计为线程。这样，整个任务结构就如图 4 所示。

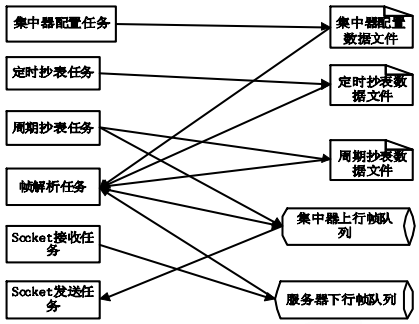

图 3 任务间的主要数据联系

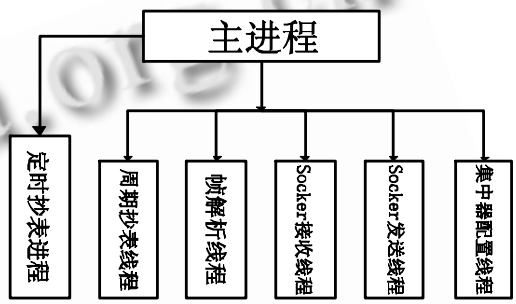

图 4　进程与线程的划分结果

4.3 各任务的实现

4.3.1 网络通信任务

主进程在完成必要的初始化信息后，首先产生一个子进程，用于定时抄表，然后创建周期抄表线程和帧解析线程，最后连接远程服务器，连接成功后，创建一个专门用于发送数据的套接口发送线程和一个专门用于接收数据的套接口接收线程，之后主进程就等待所有线程的结束并判断每个线程是否正常结束。如图 5 所示。如果套接口在中途发送异常，全局套接口号将被置为负数，然后发送线程或接收线程将通过信号告知主线程重新连接并在成功连接后更新全局套接口号。这种设计使得集中器完全是作为客户端(client)运行的，不具备监听外部连接的功能。之所以这样设计是由于集中器一般都是使用局域网内部的 ip 地址，通常不具备监听来自局域网之外的连接请求的条件。其次，集中器不具备监听功能也简单地防止了对集中器的一些恶意登陆，提高了安全性。由于使用的是线程，发送和接收可以在不同的线程中完成，都可以使用阻塞型 I/O 函数，代码功能清晰，避免了使用多路选择函数(如 select 函数)带来的复杂性[6]。在使用 GPRS 模块连接服务器的系统中，主线程连接远程服务器的过程就是连接 GPRS 的过程，只是接收线程和

发送线程共用的是 GPRS 模块的串口而不是套接口。网络通信线程的流程如图 6 所示。

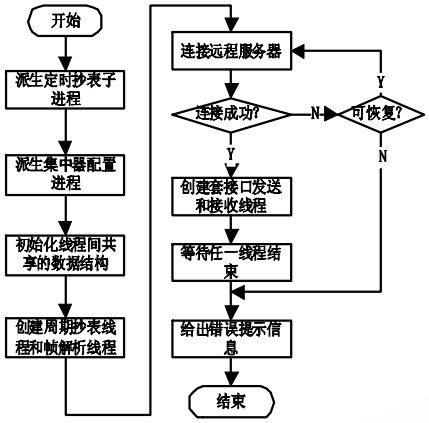

图 5 主进程的流程图

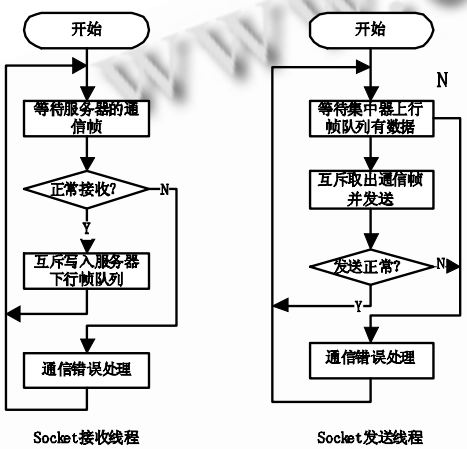

图 6 网络通信任务流程图

4.3.2 帧解析任务

帧解析线程完成对服务器命令帧(即服务器下行帧)的解释并执行相应的操作，然后再把操作的结果生成一个集中器的响应帧(即集中器上行帧)返回给服务器。帧解析线程是所有线程中功能最多的，完成了集中器大部分的功能，包括集中器的参数设置，对表具的实时控制，更新表具参数等，也是与其他线程交互最多的线程，所以帧解析线程涉及到多个同步关系，如图 3 所示。帧解析线程在服务器下行帧队列有数据时取出一条通信帧，然后依次分析帧起始符是否正确、校验码是否正确，如果没有异常即开始执行帧功能码所表示的任务，最后返回执行结果给服务器，完成一次帧解析过程，再次进入等待服务器下行帧队列的数据。其流程如图 7 所示。由于集中器和表具的通信是主从应答式的，表具发给集中器的通信帧一定是对集中器的命令的响应，故只要在抄表函数中校验一下该帧是否合法并且没有错误即可，不需要根据帧的功能分发任务，所以表具和集中器之间不需要帧解析任务。

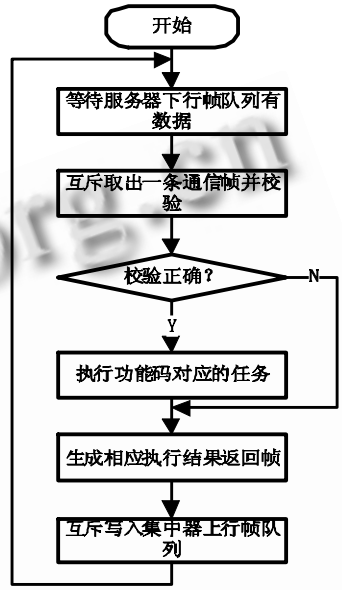

图 7 帧解析任务流程图

4.3.3 抄表任务

抄表任务负责收集集中器所管辖的所有表具的运行数据并保存在相应的文件中。根据应用需求，如 3.2 节所述，系统中有两个抄表任务，一个是周期抄表线程，另一个是定时抄表进程。周期抄表线程由抄表闹钟触发抄读集中器管辖的所有表，并对抄收结果的各个数据项进行分析，如若发现有异常情况，比如表具欠费等，产生一个报告表具异常的通信帧给网络通信线程，然后把抄到的数据存放在周期抄表文件中，等待下一个抄表闹钟的到来。另一个是定时抄表进程，它在定时抄表闹钟触发抄读所有表，但并不分析抄到的数据，只是把它们原样保存在一个定时抄表文件中，待服务器在后来查询。定时抄表进程可以只响应闹钟信号，屏蔽(mask)其他所有能屏蔽的信号，简化程序设计。如 3.2 节所述，同一进程的所有线程共享同一个定时器，所以，帧解析线程在获得服务器更新抄表周期的请求后，直接在本线程中设置闹钟即可，但需要保证只有周期抄表线程响应闹钟信号，因为闹钟信号具体发给哪个线程是不确定的[5]。抄表任务的流程

如图 8 所示。

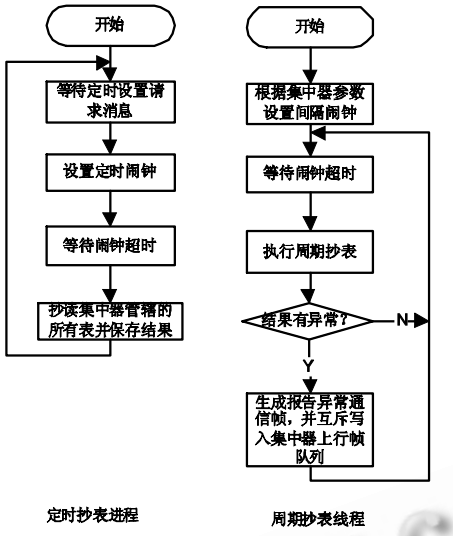

图 8　抄表类任务的流程图

4.3.4 集中器配置任务

集中器配置任务监视串口上有没有设备接入，当串口有接入数据可读时，读取并尝试拼凑一个完整的通信帧，在成功拼凑并校验通过后，执行该帧的功能，然后给配置设备一个返回信息，之后再次进入等待串口数据的状态。该任务也可以设计为睡眠等待按键信号，有按键后才打开串口并与手持设备通信，配置完成后，再次按键则关闭串口，任务进入等待下一次按键中。该任务流程简单，没有用图示。

4.4 线程间的同步和通信协议设计

多任务程序必须处理好任务间的同步和互斥问题，不同线程对同一数据进行不恰当的竞争存取会导致数据的不一致，互斥是一种特殊的同步。线程同步的手段一般有信号量、互斥量、读写锁、和条件变量。为了让中心服务器随时可查上一次周期抄表和定时抄表的结果，我们把这些数据都以文件的形式保存起来。本系统中多线程对通信帧队列的同步问题可以使用经典的“生产者消费者”问题的多生产者子类型解决。以集中器上行帧队列的同步为例，首先定义通信帧循环队列数据结构：

```
typedef struct {
    SVRF_RPT_RM_T    * base_p;
int   front; //头指针，弹出数据的位置
int   rear; //尾指针，新元素插入的位置
}SQ_T;
```

其中， base_p 是指向实际存储数据的数组，front 和 rear 分别表示队列中首元素和尾元素在这个数组中的下标索引。SVRF_RPT_RM_T 是自定义的集中器和服务器的通信帧协议结构类型。定义如下：

```
typedef union{
    u_char top_half[TOPHALF_LEN];
    struct {
     u_char _prefix[PREFIX_LEN];//起始符
     u_char _header[HEADER_LEN];//帧头
     u_char _funcode[FUNCODE_LEN];//功能码
     u_char _errcode[ERR_CODE_LEN]; //错误码
     u_char _datalen[DATLENCODE_LEN]; //帧长
     u_char _rptno[NO_LEN]; //集中器编号
     u_char _extra[0];
//Linux 中，长为零的数组实际上起到指针的作用，//但不占空间
   } whole;
}SVRF_RPT_RM_T;
```

帧的前半部分是固定的，是一些控制信息，中间是通信数据，最后是对整个帧的奇偶校验码和结束符，结构中没有体现，包括通信数据一块，实际存储在_extra 指向的空间中。为了方便访问结构体(struct)内联合体(union)的变量，可以在结构体中定义宏：#define frmextra whole._extra 来访问。由于通信帧之间数据长度相差太大，有的只是简单的应答帧，帧长很短，有的却需要返回集中器管辖的所有表的抄表数据，所以 SVRF_RPT_RM_T 的最后只定义了一个零长的数组指向通信数据实际存放的位置的指针索引[7]。定义两个分别表示队列有数据和队列已空的信号量用于生产者(包括帧解析线程和周期抄表线程)和消费者(socket 发送线程)的同步：

```
sem_t sem_svrfrmq_empty; //队列空闲位置计数
sem_t sem_svrfrmq_full;  //队列可用通信帧计数
```

初始化 sem_svrfrmq_empty 为队列的大小，sem_svrfrmq_full 为 0。定义互斥操作队列的互斥量 pthread_mutex_t mutex_bk2svrfrmq。生产者线程在 sem_svrfrmq_empty 信号上等待，消费者线程在 sem_svrfrmq_full 信号上等待。等待通过后，使用互斥量互斥更新通信队列。

(下接第 178 页)

(上接第 152 页)

5 结语

通过使用 ARM9 内核的处理器和嵌入式 Linux 操作系统，集中器系统的可靠性得到很大的提高，软件开发的速度也得到了加快，同时也降低了设计复杂功能的难度并具有极高的可扩展性和可移植性。初步的实践证明是非常可行的。

参考文献

1 Tom D, Tamar Kin. Automatic meter reading. Public Power. 1992,50(5):80－84.

2 和晓军,刘磊.网络电能表集中器的设计与实现.计算机与现代化, 2009,7:8－91

3 Samsung Electronics. S3C2440A User's Manual. 2004,332－353.

4 Sony Ericsson. Application Note Using AT Commands to control TCP/IP stack on GR47/GR48. 2003,6－9.

5 Stephen A. Rago 著, 尤晋元等译. Unix 环境高级编程(第二版).北京:人民邮电出版社, 2006.333－334,335－337.

6 Butenhof DR. Programing with POSIX Thread, 1997:20－24.

7 Linux 设备驱动开发详解.北京:人民邮电出版社, 2008.65－67.