网络优先数字出版时间:2014-2-18
网络优先数字出版地址:http://www.cnki.net/kcms/detail/45.1075.N.20140218.1419.008.html
广西科学院学报 2014,30(1):8~11
Journal of Guangxi Academy of Sciences Vol.30,No.1 February 2014

基于 ARM 的 LPC2132 通用串口驱动程序设计
Design of Universal Serial Port Driver Based on ARM LPC2132

唐民钦,夏 军
TANG Min-qin,XIA Jun

(广西瀚特信息产业股份有限公司,广西桂林 541004)
(Guangxi Hunter Information Industry Co.,Ltd.,Guilin,Guangxi,541004,China)

摘要:【目的】使嵌入式系统软件能够适应多种不同通讯协议,并保持通讯模块的稳定性。**【方法】**基于 ARM 微处理芯片 LPC2132,提出一种通用的串口驱动程序设计方案:将通讯协议与串口驱动分离,把协议帧数据段分类处理,系统软件只需要定义协议帧数据段描述,而不需要和具体的通用驱动程序打交道即可实现通讯。**【结果】**该串口驱动程序在远程环境采集系统中经过长时间的测试和应用,运行稳定。**【结论】**该串口驱动程序设计方案提高了编码效率,简化了驱动软件设计。

关键词:串口驱动程序 ARM LPC2132
中图分类号:TP311.1 **文献标识码:**A **文章编号:**1002-7378(2014)01-0008-04

Abstract:【Objective】In order to make embedded system software fit different kinds of communication protocol and remain the stable of communication module, universal serial port driver is designed.**【Method】**Base on the ARM LPC2132 microprocessor chip, we propose the design of universal serial port driver. We separate the communication protocol from port driver and classify the protocol frame data segment. Therefore, the system only needs to define the description of protocol frame data segment. Real-time communication can be realized without communication with specific universal driver.**【Result】**This serial port driver is tested and applied stably in remote acquisition environment for a long time.**【Conclusion】**This serial port driver increases the efficiency of coding and simplifies design for driver software.

Key words:serial driver, ARM, LPC2132

【研究意义】随着现代信息的发展以及计算机的广泛使用,嵌入式微处理器系统已成 IT 界新的技术热点。在大量的工业控制系统中,各种不同厂家的设备采用不同的串口通信协议,给软件设计和系统配置带来不少的麻烦。因此,将协议帧数据段分类处理,把通讯协议与串口驱动分离,对提高编码效率,简化驱动软件设计具有重要意义。**【前人研究进展】**目前,通信协议对串口信号的数据格式,波特率等都有着严格的限制,基于嵌入式系统例如 Linux 和 VxWorks 的串口驱动设计,都只是针对串口的数据流操作[1],基于系统驱动底层设计的[2]。针对不同协议的系统仍然需要重新开发通讯协议驱动。**【本研究切入点】**建立统一的协议帧结构格式,使串口驱动通用化,简化不同协议的软件设计。**【拟解决的关键问题】**设计一种描述通讯协议帧数据结构,并以此数据结构为基础构架通用串口驱动程序框架。

收稿日期:2013-11-21
修回日期:2013-12-12
作者简介:唐民钦(1982-),男,工程师,主要从事嵌入式系统软件开发研究。

1 串口驱动程序设计方案

系统采用 Philips 公司的 LPC2132 芯片,它是一款支持实时仿真和嵌入式跟踪的 32/16 位 ARM7TDMI-S TM CPU 微控制器。程序设计采用了 ADS1.2 集成开发环境,驱动模块通过简单的参数修改即可支持多系统开发平台移植。

设计一种描述通讯协议帧的数据结构 CmdRx-

Status_t,把协议帧分解为 N 个接收数据段,以此数据结构为基础构架通用串口驱动程序框架,流程图如图 1,其具体工作过程如下:

(1)系统上电后首先调用 UartInit( ),初始化串口硬件接口,设置波特率,初始化中断控制器,申请接收内存。然后调用 InitUartCtrl( )完成数据段接收控制器 CmdRxCtrl_t 参数的初始化,包括第一个接收数据段的类型标志 flag,数据段的长度 len。

(2)设备进入中断服务程序调用 ProcessUartStatus( )函数开始接收数据。根据 CmdRxCtrl_t 接收控制器的参数 flag,len 判断接收数据是否与当前数据段相匹配。如果数据匹配并且本数据段接收完毕,则调用子程序 updataUartStatus( )更新 CmdRxCtrl_t 接收控制器的参数;数据不匹配则调用 InitUartCtrl( )复位接收控制器。

(3)子函数 updataUartStatus( )的功能是:提取 CmdRxStatus_t 中定义的下一个数据段的信息,更新 CmdRxCtrl_t 接收控制器的参数 flag,len,进行下一轮数据段的接收处理。如果 CmdRxStatus_t 中定义的所有数据段接收完毕,产生事件,将接收帧交由主函数 main( )处理,串口驱动处理完毕,等待下一帧数据的接收。

由于目前的串口数据量越来越大,速度也越来越快,造成丢帧现象,降低系统可靠性,所以还必须加上多级缓存处理,根据不同方案的资源多少,用户自定义配置多级缓存区大小[3]。

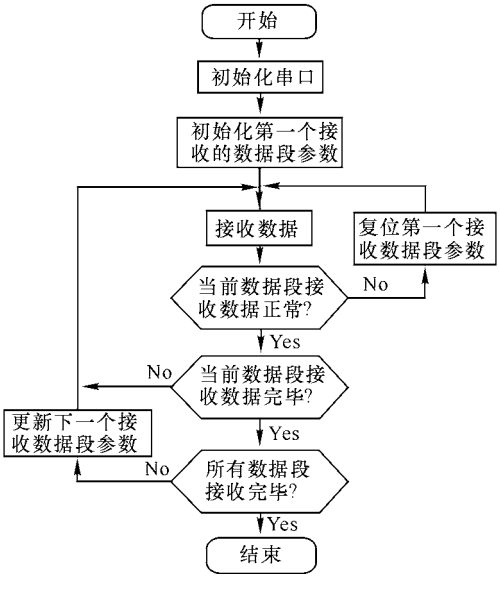

图 1 串口驱动程序流程

2 数据结构

根据前面一节通用串口驱动程序结构,在该驱动用到的数据结构类型定义如下。

2.1 数据段描述结构体

```
typedef struct
{   uint8   flag;
    uint8   len;
    uint8 const * data;
} CmdRxStatus_t;
```

flag 字段表示数据段类型(标志,定长,非定长等);len 字段表示当前数据段的长度。当为非定长数据段时,此字段数值为之前接收的某段数据段标号;data 字段表示数据段待比较数组的指针,只有 flag 为标志数据段时,即 flag=CMP_DATA 才有效。

Flag 字段对应的宏定义

```
#define CMP_LEN       0 //比较定长
#define CMP_DATA      1 //比较数据
#define CMP_DYN_LEN   2 //比较动态长度
```

2.2 帧的缓存结构体

```
typedef struct
{   uint8   Len;
    uint8   CmdBuf[CMD_BUF_MAX];
} CmdType_t;
```

Len 字段表示帧数据的长度;CmdBuf[ ]字段表示帧的缓存数据(CMD_BUF_MAX 为缓存的最大长度)

2.3 驱动控制结构体

```
typedef struct
{ uint8   status;          //
  uint8   flag;            //
  uint8   len;             //
  uint8   count;           //
  uint8   Cmdcount;        //
    uint8   Size;          //
    uint8   Entries;       //
    CmdType_t  *pStart;    //
    CmdType_t  *pEnd;      //
    CmdType_t  *pWriteIn;  //
    CmdType_t  *pReadOut;  //
} CmdRxCtrl_t;
```

驱动控制结构体是串口驱动的核心数据结构,分为两个部分:

(1)数据段接收控制字段,包括:status 字段表示当前处理数据段标号;flag 和 len 字段缓存当前数据段的类型和长度,count 字段为当前数据段接收数据计数;Cmdcount 字段表示帧长度计数。

(2)多级缓存控制字段,包括:Size 字段表示多级缓存队列的大小;Entries 字段表示写入缓存的条目数;pStart 字段表示多级缓存开始指针标记;pEnd 字段表示多级缓存结尾指针标记;pWriteIn 字段表示待写入缓存指针;pReadOut 字段表示待读出处理的缓存指针。

3 串口驱动的实现

根据上述的串口驱动程序设计方案,该驱动具体的实现过程如下:

(1)串口初始化函数:void InitUartCtrl(CmdRxCtrl_t * pUartCtrl,const CmdRxStatus_t * pStatusTbl,CmdType_t * pCmdQueue,uint16 QueueMax),完成对驱动控制结构体相关字段的初始化。具体如下:

数据段接收控制初始化。

```
pUartCtrl->status=0;
pUartCtrl->flag=pStatusTbl[0].flag;
pUartCtrl->len=pStatusTbl[0].len;
pUartCtrl->count=0;
pUartCtrl->cmdcount=0;
```

多级缓存控制初始化。

```
pUartCtrl->pStart=&pCmdQueue[0];
pUartCtrl->pEnd=&pCmdQueue[QueueMax];
pUartCtrl->pWriteIn=&pCmdQueue[0];
pUartCtrl->pReadOut=&pCmdQueue[0];
pUartCtrl->Size=MaxLen;
pUartCtrl->Entries=0;
```

(2)更新下一接收数据段相关参数函数:uint8 updataUartStatus(CmdRxCtrl_t * pUartCtrl, const CmdRxStatus_t * pStatusTbl,uint8 StatusMax)。StatusMax 为数据段总数,工作过程为:确认当前数据帧是否接收完毕。如果数据帧接收完毕,则判断多级缓存是否溢出,溢出做丢弃处理,未溢出,则更新多级缓存控制参数;如果数据帧未接收完,更新下一个数据段相关控制参数,包括数据段类型和长度,数据段接收数据计数清 0;进入下一次数据段接收循环。

返回值对应的宏定义

```
#define RX_SUCCESS     0  //帧接收成功
#define RX_CONTINUE    1  //继续接收数据
#define RX_FULL        2  //缓存满
#define RX_ERROR       3  //接收数据异常
```

(3)接收数据处理函数:uint8 ProcessUartStatus(uint8 cmdbyte,CmdRxCtrl_t * pUartCtrl,const CmdRxStatus_t * pStatusTbl,uint8 StatusMax)。工作过程为:确认多级缓存是否存满。满则直接退出;不满则根据不同数据段类型,比较判断当前数据是否异常。数据异常,复位第一个接收数据段的控制参数;数据正常,则判断当前数据段数据是否接收完毕,接收完毕调用 updataUartStatus 函数更新;未接收完毕继续接收,进入下一次循环。返回值与 updataUartStatus 函数相同。源代码如下:

```
if(pUartCtrl->Entries >= pUartCtrl->Size)
{//缓存满
    return RX_FULL;
}
if(CMP_DATA == pUartCtrl->flag)
{//如果当前状态需要比较数据,比较本次收到的数据
    if(cmdbyte! =pStatusTbl[pUartCtrl->status].data[pUartCtrl->count])
    {//比较数据不相等
        ResetUartStatus(pUartCtrl,pStatusTbl);
        return RX_ERROR;
    }
}
pUartCtrl->pWriteIn->CmdTbl[pUartCtrl->cmdcount]=cmdbyte;
pUartCtrl->cmdcount++;    pUartCtrl->count++;
if(pUartCtrl->count < pUartCtrl->len)
{//当前状态未取完数据
    return RX_CONTINUE;
}
else
{//当前状态下收取数据完毕
    return updataUartStatus(pUartCtrl,pStatusTbl,StatusMax);
}
```

(4)帧结构数据的抽象描述。以图 2 的帧结构举例说明。协议定义:帧头数据为:0x55aa,结束标志位 0xaa55,帧号长度校验字段都为 1 字节。对应的帧结构数据的抽象描述如下:

```
uint8 const SOP[]={0x55,0xaa};
uint8 const FCS[]={0xaa,0x55};
const CmdRxStatus_t  Frame[6]=
{                          //
```

```
    {CMP_DATA      ,2,SOP   },//帧头,标号(0
    {CMP_LEN       ,1,NULL  },//帧号,标号(1
    {CMP_LEN       ,1,NULL  },//长度,标号(2
    {CMP_DYN_LEN ,2,NULL  },// 数据,2 为长度段的标号,标号 3
    {CMP_LEN       ,1,NULL  },// 校验,的标号(4
    {CMP_DATA      ,2,FCS   }// 帧头,标号(5
  };                          //
```

帧头	帧号	长度	数据	校验	结束标志

图 2 动态长度数据帧结构

(5)读取帧数据缓存接口函数:uint8 ReadUartRxQueue(CmdRxCtrl_t * pUartCtrl, CmdType_t * pCmd)。工作过程为:确认 Entries 是否为 0,即判断缓存是否为空。空则返回;非空则读缓区,将 pReadOut 指向的缓存区数据,复制到 pCmd 缓存区,pReadOut 指向下一个待读缓存区,Entries 减 1。

4 结束语

本文阐述了一种新型的串口驱动程序的设计方法,它提供的开发框架把串口驱动程序独立出来,将关注驱动程序运行,转化为关注帧结构的数据抽象描述,极大地简化了驱动程序的开发。该串口驱动配置多级缓存区,避免丢帧导致系统可靠性降低的问题。改进后的驱动不但在一定程度上降低了开发的难度,为应用程序提供了更为方便的接口,也在很大程度上提高了串行设备的效率。

参考文献:

[1] 耿杰恒,王竹林,贾春宁. 基于 ARM9 和嵌入式 Linux 的串口驱动开发[J]. 科学技术与工程,2008,8(3):786-789.

[2] 吴雨舟,路唯佳,张平. 基于 MPC8270 的 VxWorks 下串口驱动程序开发[J]. 计算机工程,2007,33(17):275-277.

[3] 周立功. 深入浅出 ARM7-LPC213 * 214 * [M]. 北京:北京航天航空大学出版社,2005.

(责任编辑:陆 雁)

广西启动优质高产高糖糖料蔗基地建设

新闻时间:2014-02-18

2 月 13 日,记者从全区优质高产高糖糖料蔗基地建设试点工作动员会议上获悉,为提高广西蔗糖产业竞争力,广西启动优质高产高糖糖料蔗基地建设试点工作,到 2015 年 1 月底将建成 50 万亩。

据了解,这 50 万亩示范基地将实现经营规模化、种植良种化、生产机械化、水利现代化,力争平均亩产达 8t,蔗糖分达 14% 以上。

试点工作将在自治区初步认定的 40 个糖料蔗生产核心基地县(市、区)片区范围内,选择水源充足、地势平缓、基础设施较好、条件较为成熟的区域开展。崇左市和农垦系统近年来在土地流转及节水灌溉方面建设基础较好,因此成为此次试点的集中区域,分别承担 20 万亩和 16 万亩的建设任务;其余南宁 4 万亩,柳州、来宾各两万亩,防城港、贵港、北海、钦州、百色、河池各 1 万亩。

示范基地建设主体为制糖企业、农业企业、专业合作社(种植大户),以及自治区农垦集团公司等 4 大类。各地将因地制宜,探索建立制糖企业直接经营、制糖企业与农户合作经营、农业企业投资经营、糖料蔗种植专业合作社经营、种植大户(家庭农场)经营等建设模式。

(摘自《广西日报》)