文章编号:1671-4598(2019)01-0166-05 DOI:10.16526/j.cnki.11—4762/tp.2019.01.034 中图分类号:TP336 文献标识码:A

基于 STM32 的多串口并行传输系统设计

陈旭辉[1]，杨红云[2]
(1. 武汉纺织大学 数学与计算机学院，武汉 430200；
2. 湖北大学 教育学院，武汉 430062)

摘要：针对工程控制系统中多串口并发通信的需求，设计了一种以 STM32F429 为核心的多串口并行传输系统；该系统充分利用了处理器内部的 8 个串口和网络接口资源，串口利用 DMA 方式在缓存中循环接收，解决了并发接收时查询或中断接收方式可能导致的数据帧丢失以及处理器时间占用较多的问题；针对不同的协议帧，采用了超时判断的方式，避免了一旦串口接收数据长度不正确后导致的后续接收问题；网络传输采用了 lwIP 协议栈。该设计实现了同时采集 7 路读卡信息，并通过 1 路串口或网络向上位机上传数据的功能，测试表明多串口并发传输无数据帧丢失现象；该系统无外接串口扩充电路，减小了整个电路的体积，提高了系统的可靠性和稳定性。

关键词：多路串口；并行传输；STM32；lwIP；DMA

Design of Multi—channel UART Parallel Transmission System based on STM32

Chen Xuhui[1], Yang Hongyun[2]
(1. College of Mathematics & Computer Science, Wuhan Textile University, Wuhan 430200, China;
2. College of Education, Hubei University, Wuhan 430062 China)

Abstract: In order to meet the needs of multi—channel UART parallel communication in engineering application, this paper designed a multi—channel UART parallel transmission system based on STM32F429. The system makes full use of the eight UARTs and network interfaces inside the processor, DMA is used to receive UART data in a circular buffer, and this solves the problem of data frame loss and processor occupation time caused by querying or interrupting mode in concurrent reception. For different protocol frames, the method of timeout judgement is adopted to avoid subsequent reception problems once the UART receives incorrect data length. The lwIP protocol stack is used for network transmission. The design realizes the simultaneous acquisition seven channels card number and uploads data to the host computer through an UART or a network port, the test shows that there is no data frame loss during the transmission of the system. The system has no external UART circuit, which reduces the volume of the whole circuit and improves the reliability and stability of the system.

Keywords: Multi—channel UART; parallel communication; STM32; lwIP; DMA

0 引言

串口作为一种常用的串行通讯接口，由于其标准发布时间早、使用简单，使其在工控和测量设备以及部分通信设备中有着广泛的应用。很多模块设备提供一个串口来进行通讯，如导航定位模块、读卡器模块、打印模块和无线通信模块等。当一些应用需要同时接有多个串口模块时，如果系统串口有限，可以使用分时方式与各个模块进行通讯，但是在一些实时突发传输的场合，这种分时传输方式显然不能够满足使用要求，于是对系统的物理串口数目需求越来越多。在嵌入式应用系统中，很多嵌入式处理器的串口数目通常只有 1—3 个，如果有更多串口需求就需要使用软件或外部电路扩充串口。

串口扩充方法很多，对实时突发性没有要求时，可以通过模拟开关切换来分时使用串口[1]；节约成本可以使用通用 IO 口来模拟串口[2]；追求稳定性和控制简单可以使用一些专用的串口扩展芯片[3-4]；高速和更灵活的应用可利用 FPGA 芯片来实现[5]。

一些嵌入式处理器厂商也意识到多串口的需求，在处理器中提供了多个串口功能，相对于其他串口扩充的方式，在处理器中提供多串口的方式具有可靠性高、传输速率快和实现容易的特点，且外围电路结构简单。如 ST 公司的 STM32F427/429 就是其中之一。该系列芯片基于 ARM Cortex—M4 内核，最高主频能达到 180 Mhz，拥有 256K SRAM 和 512 K 以上的 FLASH，最多能同时提供 8 个串口、一个 USB 接口和一个以太网 MII/RMII 接口[6]。

系统要求采集 7 路读卡信息，上位机不可用时可独立工作，读卡后对比已录入的卡号给出相应的提示和控制动

收稿日期:2018-06-09； **修回日期**:2018-07-20。
基金项目:湖北省自然科学基金项目(2015CFB721)。
作者简介:陈旭辉(1974-)，男，湖北仙桃人，讲师，主要从事计算机系统结构、嵌入式系统方向的研究。
通讯作者:杨红云(1979-)，女，河南开封人，副教授，硕士生导师，主要从事物联网、学习分析、在线教育方向的研究。

作，并记录相应的信息。在有上位机时，可通过网络或串口将相关信息上传到上位机，并预留了 USB 接口。

1　系统硬件与软件结构

1.1　系统硬件结构

由于要求同时接收 7 路读卡信息，可选一个串口或网口与上位机通信，一共需要 8 个串口和一个网络接口。系统选用 STM32F429VIT6 芯片，由于传输距离有数十米左右，采用 RS485 接口传输，每个串口前面都接有 RS485 转换芯片，各由一个 GPIO 端口控制转换芯片的收发，RS485 转换芯片可选 max3485 等芯片，需要隔离功能时可选 ADM2582 等芯片。STM32F429VIT6 只有 MII/RMII 接口并没有 PHY（物理层）接口，需要外接 PHY 网络接口芯片，可选 LAN8742、DP83848 等芯片，这里选用的是 DP83848。USB 接口作为备选上位机通信方案，使用全速模式，无需外接接口芯片。系统另外外接了一片大容量 SPI FLASH，用于保存卡号和记录读卡等信息。系统结构和 8 个二线串口的引脚分配见图 1 所示。

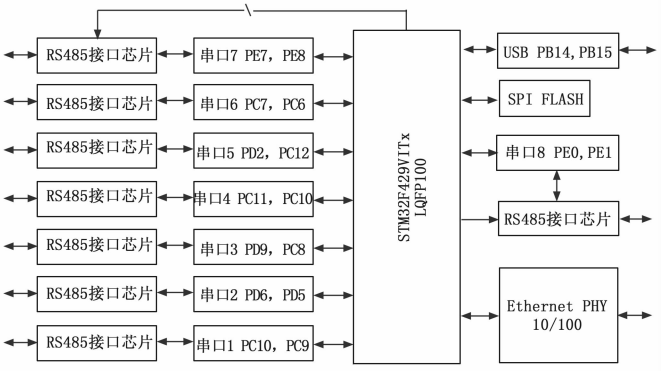

图 1　系统硬件结构图

1.2　系统软件结构

STM32F429 各个模块的初始化配置代码使用图形化软件配置工具 STM32CubeMX 生成，该软件可用图形化向导自动生成初始化代码[7]，库版本是 STM32Cube _ FW _ F4 _ V1.16.0。网络功能选用了轻量级 TCP/IP 协议栈 lwIP，由于系统需要多任务和消息队列等功能，使用了 FREERTOS 操作系统[8]，编译环境为 MDK－Arm 5.23。4 个线程分配如下：网络通信用一个线程；7 个串口接收使用 DMA 方式，共用一个线程；与 PC 通信串口收发各用一个线程。

2　软件设计

2.1　串口程序设计

STM32F429 的串口在其波特率的 16 倍采样率下，最大波特率可达 5.62 Mbit/s，在 8 倍采样率下最大波特率可达 11.25 Mbit/s，而且串口没有硬件 FIFO，每个串口只有一个接收和一个发送寄存器[9]，中断接收方式已经不能保证数据的可靠接收。即使在低波特率下，微控制器每接收一个字符都要进入一次中断处理程序，会使微控制器的处理时间大量浪费，再加上 8 个串口可能同时接收，这时就需要用到微控制器的 DMA 串口接收功能。串口发送一个字时，会存入到发送数据寄存器 TDR，然后由串口硬件逐位发送，此时跟控制器的处理时间没有关系；而串口发送字的字间间隔时间一般没有明显的时序要求，所以串口发送可以不采用 DMA 方式。

2.1.1　DMA 配置

STM32F429 拥有 2 个 DMA 控制器，每个控制器可与 8 个数据流中的一个连接，每个数据流可达 8 个通道（请求），从其 DMA 请求映射表上可知 8 个串口的接收可以配置到 2 个 DMA 控制器的不同通道上，表 1 为 8 个串口接收功能的 DMA 配置。如果发送也要使用 DMA 方式，在表 1 配置的基础上，只有 4 个串口可以使用 DMA 发送功能，见表 2。

表 1　8 个串口接收功能的 DMA 配置

串口号	1	2	3	4	5	6	7	8
DMA 号	2	1	1	1	1	2	1	1
数据流号	2	5	1	2	0	1	3	6
通道号	4	4	4	4	4	5	5	5

表 2　在表 1 基础上 4 个串口发送功能的 DMA 配置

串口号	1	3	5	6
DMA 号	2	1	1	2
数据流号	7	4	7	6
通道号	4	7	4	5

每个串口接收 DMA 请求的配置如下：

DMA 模式是循环方式，DMA 源地址是串口接收数据寄存器 RDR，属于外围设备，总是从该地址读，源地址不知递增，DMA 目的地址是 SRAM 存储器，从 RDR 读取后依次存放，所以目的地址要递增。数据宽度根据串口设置选择，如果数据和校验位一起不超过 8 位则选一个字节，否则选半字长。

2.1.2　循环队列

在串口和 DMA 初始化之后，运行 HAL _ UART _ Receive _ DMA 函数多次开启多个串口的 DMA 接收，其形参分别为各串口句柄、接收缓存区和待接收的数据长度。该函数非阻塞，运行后立即返回。随后开始 DMA 自动接收，接收的串口数据依次存放在接收缓存区中，超过待接收的长度后，循环覆盖接收缓存区。如果实现 HAL _ UART _ RxHalfCpltCallback 回调函数，会在收到待收数据的一半时调用，实现 HAL _ UART _ RxCpltCallback 回调函数，会在收到所有数据后调用。要获取其他接收到的数据长度时要使用 _ _ HAL _ DMA _ GET _ COUNTER（ _ _ HANDLE _ _ ）宏，该宏返回 DMA 中剩余要传输的长度，因此接收到的长度等于待接收的数据长度减去该宏返回的值，当接收到所有待接收长度的数据后，该宏返回 0，此时处理不当会丢失数据。

可以把串口接收 DMA 缓存看作一个循环队列，写该缓

存由 DMA 控制，用户无法干预，写满后自动从头开始，同时更新剩余要传输的长度。如果用户无法在缓存循环覆盖前读取数据，数据就会丢失，可根据情况调整 DMA 缓存长度 UART _ SIZE，即使发生覆盖，最近的长度为 UART _ SIZE 的数据是可读取的，这跟普通循环队列有所不同。一次 DMA 已传输的数据长度为 len＝UART _ SIZE－ _ HAL _ DMA _ GET _ COUNTER，用户维护已读数据长度 read 和已接收到的数据总长度 total，初始都赋值为 0，在 UART _ RxCpltCallback 中更新接收到的数据总长度 total ＋＝ UART _ SIZE，而该回调函数只在 DMA 接收完数据后调用。实现 ReadFromDMA 函数从指定串口 DMA 缓存区读取指定长度数据，并返回实际读取的数据长度，该函数非阻塞。

图 2 为循环接收缓存示意，表 3 列举了 UART _ SIZE ＝8 时 DMA 接收缓存处理实例。ReadFromDMA 函数首先判断总长度 total 和已读长度 read 的大小，如 read 大于等于 total，说明本次 DMA 接收数据长度还没到 UART _ SIZE，可读数据缓存区从 read Mod UART _ SIZE 开始，长度为 len－（read Mod UART _ SIZE），如表 3 中第 1 和第 2 数据行。如果 read 小于 total，要判断接收数据总的长度（total ＋len）－read 与缓存长度的大小，如果前者大，说明上次读后发生循环覆盖，说明有数据没有及时读取，已经覆盖了部分数据，可读数据缓存区从 len 后面开始，可能的数据长度就是 UART _ SIZE，如表 3 中第 4 和第 5 数据行；如前者小于等于后者，可读数据缓存区从 read Mod UART _ SIZE 开始，长度为 UART _ SIZE － （read Mod UART _ SIZE） ＋ len，如表 3 中第 3 数据行。实际 UART _ SIZE 可取 128、256 等，这样上述的 Mod 在 C 语言中的操作 read％UART _ SIZE 可替换为 read&（UART _ SIZE－1）。

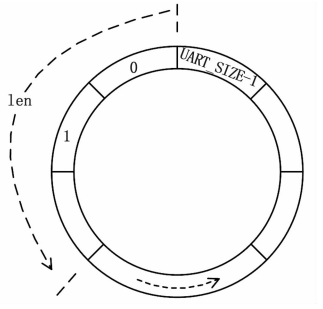

图 2 DMA 接收缓存

表 3 DMA 接收缓存处理实例(UART_SIZE＝8)

total	read	len	接收总长	可读起始地址	可读长度	数据丢失
0	2	4	4	2	2	否
8	11	6	14	3	3	否
16	11	1	17	3	6	否
16	11	7	23	7	8	是
80	20	5	85	5	8	是

2.1.3 超时处理

串口接收的数据帧可能是不定长或定长的，不定长的数据帧需要解析其协议，得到该帧数据长度，从而读取完整的一帧数据，定长的数据帧直接接收定长数据即可。不论定长或不定长的数据帧，当接收到一个不完整的数据帧时，如果没有帧的超时处理，会把后面一帧部分数据当作前面帧的一部分，会导致大量后续帧数据错误，甚至可能一直无法接收到一个正确的完整数据帧。

串口超时接收可以利用 HAL _ GetTick 函数实现，该函数返回系统运行的 ticks 数（1 毫秒），通过接收前后调用获取时间差以及待接收和实际接收的数据长度来判断是否超时。

2.1.4 帧协议处理

一般工程中的串口协议相对简单，但是在实际中经常会遇到一些问题。如首次收发失败，后续正常；收发失败后重试多次才能成功，甚至不可恢复；RS485 需要多发一个字节等等，这些问题很大部分都是软件引起的。如串口初始化不当可能导致首次收发失败，对 RS485 收发芯片的收发引脚控制时机不当导致需多发数据。下面通过工程中遇到的两个典型帧协议说明接收的处理方法，这里只讨论数据的完整接收，并不涉及到数据的校验，校验信息可以包含在数据中。

当数据帧是定长时，例如如表 4 所示的一种读卡器的协议，可以直接一次接收 18 个字节，然后判断结束标记字符。不加超时功能时，一旦有非 18 个字节数据，例如只收到 16 字节，数据不足，会用接收的下一帧数据的前 2 字节补入，这样导致后续读到的数据都有误。增加接收 18 个字节的超时功能后，如只收到部分数据，超时时间到会放弃该次接收的数据，后一帧数据正常则能恢复正常接收。这种方法启动超时的时刻需要保证是在接收到第一个字节后，如果不是，例如启动接收就开始超时计时，数据可能在超时结束的时候才收到部分数据，另外一部分数据就丢失了，如图 3 所示。如果某种情况下收到的数据多于 18 字节，如果前 18 字节不是完整数据，也会导致数据丢失。这些异常一般情况下出现的机率不大，但在收发方调试的情况下经常出现。

表 4 定长数据帧示例

16 位 ASCII 数据	0D 0A

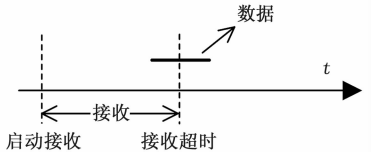

图 3 数据在超时即将结束时出现

系统采用的接收方法流程如图 4 所示，就是每接收到 18 个字节判断最后两个字节是否是 0D 0A，是则为接收到

完整一帧，否则丢弃前面一个字节，再接收一个字节，然后判断最后两个字节是否是 0D 0A，如此循环。每接收一个字节后判断是否超时，超时后丢弃所有接收的内容。算法保存最近接收的 18 个字节内容，没有使用循环队列，所以当收到的 18 个字节不是完整帧时，丢弃前面一字节，然后后面字节依次左移一字节。可见表 4 这种帧标记在数据后面的协议并不方便接收处理。

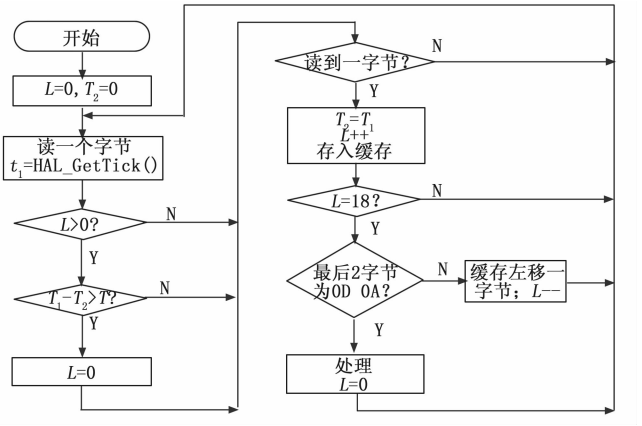

图 4　定长协议帧接收处理流程

当数据帧是不定长时，例如表 5 所示的协议，则处理流程如图 5 所示。比较定长和不定长数据帧的处理流程，可以发现并不是串口通信协议简单，处理就简单，一个好的串口协议才能简化程序的处理流程。

表 5　不定长数据帧示例

A5	5A	Length(data 长度)	Data

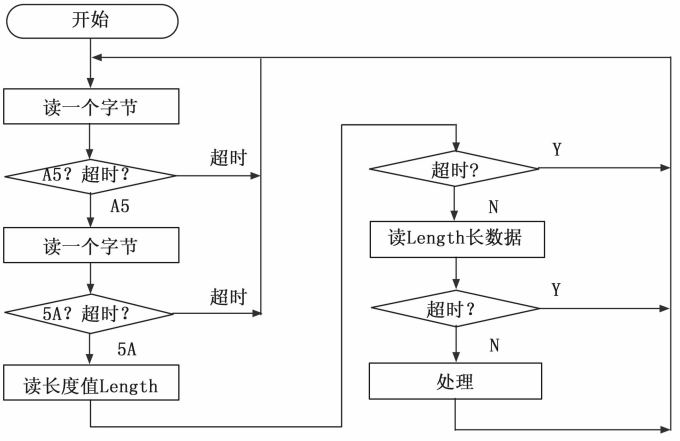

图 5　不定长协议帧接收处理流程

2.2　上位机通信串口

系统工作时，要求随时可以接收上位机的命令，所以单独使用一个线程接收上位机命令并处理。在接收到读卡信息后，有可能设置读卡信息实时上传，如果加上上位机的通信应答，可能有多条串口数据要发送，这些发送的动作可能在不同的线程中，而与上位机通信串口只有一个，由于线程的调度可能会导致一条数据发送了部分然后切换到另外一个线程发送数据，从而导致接收的数据混乱。这就要求串口发送必需保证一个数据帧完全发送完后才能发送另外一帧，这里采用一个独立的线程单独处理与向上位机发送数据，发送的数据通过消息队列排队，串口向上位机发送流程图如图 6 所示。

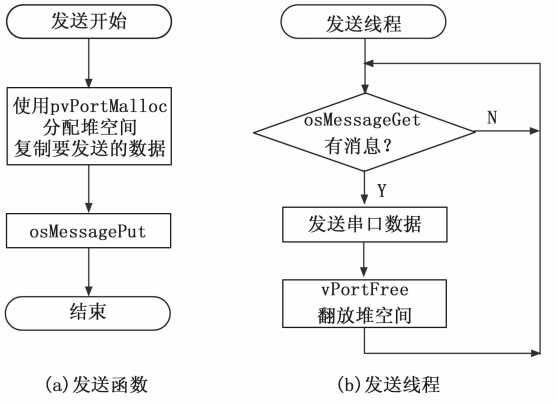

图 6　串口向上位机发送流程图

虽然 7 个串口可能同时接收数据，然后与上位机通信，但是每个串口的接收间隔却较长（刷卡间隔一般为秒级），在此间隔内所有数据都可以上传到上位机，对于 8 个串口，消息队列长度不小于 8 即可。消息队列中带的是数据对象的地址，入队列时需要各个串口使用不同的发送缓存区，在此使用了动态内存分配，消息取出后再释放。

2.3　网络传输设计

网络协议栈使用 STM32CubeMX 自带的轻量级 TCP/IP 协议栈 lwIP，是一个小型的开源协议栈，实现的重点是保持 TCP 主要功能的基础上减小对系统资源的占用，它需要的 RAM 和 ROM 在几十 K 以下[10]。lwIP 有 3 种编程接口，分别是 RAW、NETCONN 和 SOCKET。RAW 使用回调机制，编程接口不需要操作系统的支持，但其编程接口稍复杂；NETCONN 接口相对 SOCKET 接口来说更底层，SOCKET 接口被广泛熟知，使用更方便，这两种编程接口都需要有操作系统的支持。

这里使用 SOCKET 方式编程，采用 UDP 协议传输，网络通信线程流程如图 7 所示。网络通信速度相比串口要快很多，只用了一个线程来处理数据收发，多个串口读卡信息同样通过消息队列排队，再经过网络接口发送至上位机。

3　系统测试与分析

网络测试中发现如上电时不插网线，系统启动后再插入网线，网络功能无法使用。从现象看可能是网络状态改变时协议栈没有得到通知，通过下面步骤解决了这个问题。实现网络状态发生改变时的回调函数 ethernetif _ notify _ conn _ changed，在该函数中判断连接标记，如果不是 UP 状态，首先调用 netif _ set _ link _ up 函数，再更改连接标记。

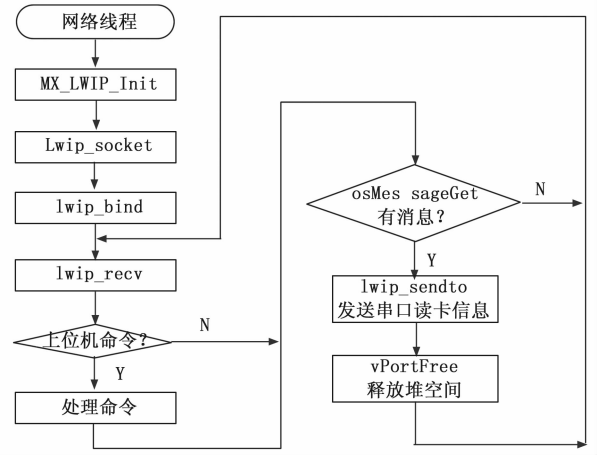

图 7 网络通信线程流程图

解决上述问题后发现在运行中插拔网线后，网络也无法正常工作。当插拔网线的时候，PHY 芯片会产生外部中断，在这个中断中重新初始化网络，外部中断要手工增加。首先将 DP83848 的第 7 脚 PWR _ DOWN/INT 配置为中断输出，当网络状态发生改变时，该引脚电平会被拉低，配置微控制器对应的引脚为下降沿触发中断，在中断处理函数中后读 DP83848 的中断状态寄存器，确定是网络状态发生改变时再调用 ETH _ MACDMAConfig 和 HAL _ ETH _ Init 函数，重新初始化网络。

本系统接收特点之一是 7 路串口数据可能同时到达，在这种情况下，DMA 接收方式可以及时读取并保存每个串口 RDR 中的数据。另一个特点是读卡都有一个秒级的间隔时间，每路读卡数据经波特率为 115200 b/s 的串口上传至上位机需要 2 毫秒不到的时间，另外由于发送队列的存在，不论多路读卡信息何时到，读卡间隔时间也足以满足数据上传。

读卡器波特率为 9600 b/s，系统 7 路串口也设为该速率。测试方案一为 7 路同时读卡，手工刷卡很难做到这一点，采用模拟读卡的方式，直接向读卡串口同时发送卡信息，间隔 2 秒，测试中 7 路读卡接收无数据丢失。测试方案二为等间隔读卡，依次轮流向 7 路串口发送卡信息，每次发送间隔 280 ms，上传到上位机的数据无丢失。测试方案三为随机多次模拟和人工刷卡，模拟读卡要设置好每路发送间隔时间大于 2 秒，多次测试 7 路读卡串口无数据丢失，上传到上位机的数据无丢失。

4 结束语

利用 STM32F429 片内的多串口和网络接口功能，设计了多串口并发通信系统，接收多路同时到达的数据，相对于其他串口扩充的方法，无外扩电路，速度高且各串口可工作于不同的速率，测试表明多并发串口传输无数据帧丢失。受限于了读卡器波特率和要求一路串口上传至上位机，并没有测试更高的传输速率，STM32F429 的 DMA 模块每通道有一个 16 字节的 FIFO，更高的串口传输速率可以考虑使用。该设计可供相似应用参考，也可作为网络多串口服务器和 USB 多串口采集系统参考。

参考文献：

[1] 孙 波，王 勇. 基于 MSP430 的串口扩展设计 [J]. 电子测试，2010 (4)：64 - 67.

[2] 郑志雄，胡爱兰. LPC1768 的全双工 UART 的软件模拟实现 [J]. 单片机与嵌入式系统应用，2013 (6)：25 - 28.

[3] 涂 清，杜列波，罗武胜. 基于 ARM9 的车载智能终端多串口扩展设计与应用 [J]. 计算机测量与控制，2011，19 (11)：2485 - 2487.

[4] 王心鹏，门雅彬，顾季源，等. 串口扩展芯片 XR16L784 在水文监测系统中的应用 [J]. 计算机测量与控制，2016 (3)：14 - 17.

[5] 刘 杰，臧 炜，梁晓鹏，等. 一种新型的 FPGA 实现 RS422 串口通信方法 [J]. 计算机测量与控制，2017 (3)：191 - 194.

[6] DS9405 _ STM32F427xx 和 STM32F429xx 单片机数据手册 [EB/OL]. 2018 (1). https：//www. stmcu. com. cn/.

[7] UM1718：STM32CubeMX for STM32 configuration and initialization C code generation [EB/OL]. 2018，http：//www. st. com.

[8] Reference Manual for FreeRTOS version 10. 0. 0 issue 1 [EB/OL]. 2017. https：//www. freertos. org/ Documentation/RTOS _ book. html.

[9] RM0090 _ STM32F405/415，STM32F407/417，STM32F427/437 和 STM32F429/439 单片机参考手册 [EB/OL]. 2018 - 04. https：//www. stmcu. com. cn/.

[10] LwIP—a Lightweight TCP/IP Stack—Summary [Z/OL]. http：//savannah. nongnu. org/projects/lwip/.

(上接第 165 页)

[4] 刘 宏，蒋再男，刘业超. 空间机械臂技术发展综述 [J]. 载人航天，2015，21 (5)：435 - 443.

[5] 荣 鹏. 美国快速响应卫星遥感相机电子学的技术特点 [J]. 航天返回与遥感，2016，37 (2)：9 - 17.

[6] 刘晓明，何煦红，张翼麟. 激光武器及导弹抗激光技术研究 [J]. 战术导弹技术，2014，4：1 - 4.

[7] Mao Chunjing，Guan Yong，David Jungwirth. Research and design of on—board dynamic SpaceWire router [J]. Journal of Electronics & Information Technology，2010，32 (8)：1904 - 1909.

[8] 朱晓燕，陶利民，张伟功，等. 面向卫星数据系统的 spacewire 应用模型仿真研究 [J]. 小型微型计算机系统，2015，36 (3)：616 - 620.

[9] 张科科，朱振才，夏 磊. 小卫星模块化设计技术分析 [J]. 航天器工程，2015，24 (6)：107 - 115.

[10] Sugawara Y，Sahara H，Nakasuka S，et al. A satellite for demonstration of Panel Extension Satellite (PET—SAT) [J]. Acta Astronautica，2008，63 (4)：228 - 237.