文章编号：2096-1472(2018)-09-29-05
DOI:10.19644/j.cnki.issn2096-1472.2018.09.009

基于Nucleus操作系统实现TCP和UDP协议通信

汪 洋[1],禹 珉[2]
(1.武汉软件工程职业学院电子工程学院，湖北 武汉 430205；
2.武汉中原电子集团有限公司，湖北 武汉 430205)

摘 要：Nucleus操作系统凭借具有操作系统源码、调试时可跟踪至中断和寄存器级，以及支持多种处理器进行系统移植的优势，成为实时嵌入式产品系统优选,本文针对实时系统网络通信需求,提出利用操作系统外挂Nucleus NET模块实现TCP和UDP协议通信，并将源代码运用于多种模式组网终端产品，结果表明，数据语音通信及时顺达。

关键词：嵌入式系统；嵌入式操作系统；Nucleus；协议通信
中图分类号：TP316 **文献标识码**：A

Implementation of TCP and UDP Protocol Communication Based on Nucleus Operating System

WANG Yang[1],YU Min[2]
(1.*Department of Electronic Engineering,Wuhan Software Engineering Vocational College,Wuhan* 430205,*China*;
2.*Wuhan Zhongyuan Electronics Group CO.LTD,Wuhan* 430205,*China*)

Abstract: Nucleus operating system becomes the optimum real-time embedded system for the reason that its source code of the operating system can be traced to interrupt and register level during debugging,and can support multiple processors for system migration.In order to realize real-time system network communication requirements,this paper proposes to use plug-in Nucleus.The NET module implements TCP and UDP protocol communication and applies the source code to various modes of network terminal products.The results show that the data of voice communication is timely and compliant.

Keywords:embedded system;embedded operation system;Nucleus;protocol communication

1 引言(Introduction)

嵌入式操作系统是随着嵌入式系统的发展而出现的。早期嵌入式产品直接对处理器内部寄存器进行读写即可完成硬件接口驱动，没有专门的操作系统，应用软件直接建立在硬件上[1],软件通常以硬件的附属品的形式出现。从20世纪80年代，嵌入式软件进入操作系统的阶段：建立一个嵌入式实时应用程序变得非常直接,因为驻留在主机系统上的应用程序文件可以编译/汇编成目标文件并连接[2]。

本文首先介绍了Nucleus操作系统的一些基础知识，包括Nucleus PLUS和Nucleus NET。然后在这个基础之上介绍了如何使用Nucleus操作系统。接着介绍在Nucleus操作系统中，如何实现TCP和UDP协议[3]的通信。具体包括协议通信模型，以及相关的Nucleus系统函数。在本文的最后，还引用了两段作者在实际产品开发过程中编写的代码来举例说明如何实现网络通信这一功能。

2 Nucleus操作系统(Nucleus operation system)

Nucleus操作系统是美国ATI公司推出的嵌入式操作系统，属于工作实时、任务抢先、多线程操作内核[4]。它为实时要求较高的嵌入式应用而设计。Nucleus操作系统大约95%的代码由ANSI C语言编写,因此适用于大多数微处理器[4]。

2.1 Nucleus PLUS的简介

Nucleus PLUS是Nucleus操作系统的实时多任务内核，

主要负责任务管理、系统内存分配和回收各个硬件设备的驱动控制等。Nucleus内核所需空间很小，因此可以极大程度减小系统开销。其中，复杂指令集体系结构中占约20k字节，精简指令集体系结构中开销约40k字节，其内核数据结构1.5k字节[5]。Nucleus PLUS内核系统结构图,详见图1。

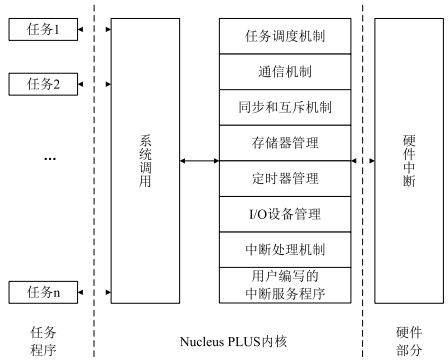

图1 Nucleus PLUS内核结构图

Fig.1 Nucleus PLUS kernel structure

通过内核系统结构图可以看出，系统调用两种资源，其一是用户编写的中断服务程序，其二是系统资源管理组件[6]，也称作软件组件。每一种组件类适于Windows机制下的动态链接库文件，它允许程序共享执行特殊任务所必需的代码和其他资源。各组件之间相互独立，一般只能通过外部接口进行访问而不能直接访问组件内部。Nucleus PLUS的具体组件列表,详见表1。

表1 Nucleus操作系统组件列表

Tab.1 List of components of the Nucleus operation system

编号	组件名称	简写	描述
1	Common Service	CS	通用服务组件
2	Initialization	IN	初始化组件
3	Thread Control	TC	线程调度组件
4	Timer	TM	定时组件
5	Active	AT	活动定时队列
6	Mailbox	MB	邮箱组件
7	Pipe	PI	管道组件
8	Queue	QM	队列组件
9	Semaphore	SM	信号量组件
10	Event Flag	EV	事件组件
11	Partition Memory	PM	存储分配组件
12	Dynamic Memory	DM	动态分配组件
13	I/O Driver	IO	输入输出组件
14	History	HI	历史组件
15	License	LI	许可证组件
16	Release	RL	版本控制组件

注：整理自《Nucleus_PLUS参考手册》[5]

这些组件能够为系统开发提供以下支持：任务管理(多任务和基于优先级和分时的任务调度)、任务间同步和互斥(通过信号量、事件组和信号实现)、通信(通过邮箱、队列和管道等)、存储器优化管理(提供动态和分区内存两种存储器管理机制)、实时时钟服务，以及中断管理服务。其中最常见的中断管理服务为利用定时器来处理周期性事件，以及任务的睡眠和挂起超时[6]。

软件工程师首先按照用户需求编写主任务代码，完成用户界面。对于每个任务调用组件并将中断任务放置于不同的中断接口服务程序，实现与Nucleus PLUS的内核的系统资源交互共享。

2.2 Nucleus NET的简介

前面介绍的Nucleus PLUS是Nucleus操作系统的核心，但并不支持网络功能。Nucleus NET是ATI公司为满足用户进行网络通讯要求设计开发的一个组件，与Nucleus PLUS联合使用。

Nucleus NET网络体系结构组织属于上下层单项隶属关系,每个下层只服务于它的上层,类似于子函数被调用，每个层有相对完整的功能,层与层之间接口独立。

Nucleus NET网络由以下四个层次[7]组成：

(1)物理层。其中硬件设备是指网卡、网线、集线器和中继器，信号是调制解调器通信信道中传输的原始比特流，位于网络结构中的最底层。

(2)数据链路层。Nucleus NET网络内核在数据链路层上，通过网桥，交换机等物理寻址方式将原始比特流转换为逻辑传输，为网络之间搭建、保持、发放数据，以及链路数据错误校正、重发等功能。

(3)网络层。数据准备好之后，路由器设备即第三层网络层就负责维护各个子网的正常有序运行，如逻辑编址、分组传输和路由选择。

(4)传输层。在Nucleus NET网络结构中是最高层，物理设备主要是网关，负责提供端到端的可靠通信。它提供通用的传输接口，这样高层用户就无需知道底层具体细节，就可以传输数据。传输层有两类不同的协议服务：TCP协议和UDP协议。前者属于面向连接的“传输控制协议”，后者属于面向无连接的“用户数据报协议”[8]。

3 Nucleus PLUS系统初始化(Nucleus PLUS system initialization)

由于系统最先运行“INT_初始化程序”，所以其中必须包括所有目标硬件复位向量地址、中断向量表、全局声明的数据结构、控制寄存器地址、系统变量和堆栈指针。

当“INT_初始化”程序完成后，控制权被转移到“INC_初始化”上。“INC_初始化”调用各个组件的初始化子程序。当各组件初始化均完成之后，“INC_初始化” 调用“应用_初始化”。

以上两个初始化均属于内核驱动，而“应用_初始化”函数才是程序员编写应用软件时的第一个函数。当Nucleus系统开始启动后，这个负责定义初始化应用程序环境的函数优先运行。“应用_初始化”函数所配备的指针,首址指向一片“自由净土”，是不被编译器或Nucleus PLUS占用的。

“应用_初始化”函数格式如下：void App_Ini(void*first_avail_memory)

一般情况下，在App_Ini函数中调用NU_Create_Memory_Pool函数、NU_Allocate_Memory函数和NU_Create_Task函数。通过这三个函数初始化应用程序环境。主要是创建内存区，分配内存区和建立第一个任务。在建立的任务中就可以编写自己的应用程序。

4　Nucleus NET初始化(Nucleus Net initialization)

在Nucleus NET的初始化相对于Nucleus PLUS的初始化要简单的多。主要由两个初始化函数NU_Init_Net函数和NU_Init_Devices函数来完成。这其中还有一个重要的NU_DEVICE结构体需要进行填充。NU_Init_Net函数主要负责初始化网络堆栈，包括设置内部数据结构和分配必要的资源。NU_DEVICE结构体中主要包含要使用的网络接口的一些参数。包括网络接口使用的驱动名称，设定驱动的操作，接口初始化程序的入口，接口的IP地址，接口的子网掩码，接口的中断，I/O设备起始地址和设备共享内存地址。配置完NU_DEVICE结构体后，调用该结构体的数据结构初始化函数和设备底层驱动程序中硬件初始化函数。

5　TCP协议通信(TCP protocol communication)

5.1　TCP协议简介

TCP传输控制协议(Transmission Control Protocol)是一种面向连接的、可靠的、基于字节流的通讯协议。TCP协议位于Nucleus网络层次的第四层传输层上。在Nucleus NET中通常使用C/S模式，完成建立连接、发送数据及接受数据工作。

5.2　TCP协议通信模型

在服务器端，Nucleus分四个步骤分别调用四个函数：首先通过NU_Socket函数“构造”一个套接字，其次用NU_Bind函数“绑定”本地地址结构与套接字。然后用NU_Linsten函数“监听”来自客户端的请求，当聆听到有远程客户发出连接请求后，第四步，Nucleus调用NU_Accept函数，“接受”连接请求，给请求者分配一个新的套接字。

在客户端，Nucleus也是调用NU_Socket和NU_Bind函数构造并绑定套接字，然后NU_Connet函数会接收到服务器发送的“接受”请求指令，它就为执行“连接”，具体细节就是为该请求构造一个表。只有连接完成后，才会中止请求任务。此时已做好传送数据的准备。

数据的发送NU_Send函数和接收NU_Recv函数对于服务器和客户端是一致的，数据传输完成后调用NU_Close_Socket函数关闭套接字。TCP协议通信流程图，详见图2。

6　UDP协议通信(UPD protocol communication)

6.1　UDP协议简介

UDP(User Datagram Protocol)用户数据报协议，也位于Nucleus网络层次的第四层传输层，属于无面向连接，无需建立和释放连接，服务器和客户端直接发送原始IP数据报文[9]。

6.2　UDP通信模型

在服务器端，首先使用NU_Socket函数创建一个套接字，创建完套接字后调用NU_Bind函数将本地地址与套接字进行绑定。绑定成功后，调用NU_Recv_From函数等待来自客户端的数据，同时也可以调用NU_Send_To函数发送数据给客户端。当数据传输结束后，可以调用NU_Close函数关闭连接。

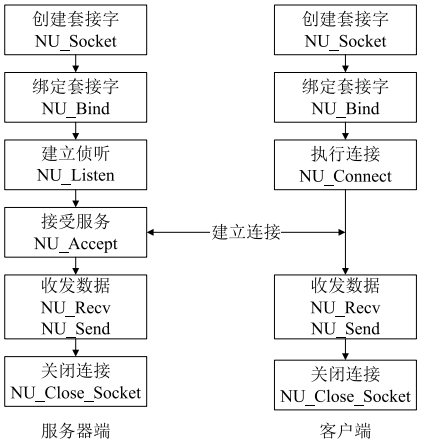

图2 TCP协议通信流程图

Fig.2 TCP protocol communication flow chart

在客户端，首先使用NU_Socket函数创建一个套接字。创建套接字成功后，就可以调用NU_Send_To函数发送数据和使用NU_Recv_To函数接收数据。数据传输完成后，可以调用NU_Close函数关闭连接。

在UDP协议中，服务器端不需要监听来自客户端的请求，客户端也不需要连接服务器端。这是与TCP协议最大的区别。这也造成UDP协议在传输过程只能够存在可能丢失的情况，但是其资源消耗小、处理数据块的优点[10]，通常用于传输音频和图像和普通数据。具体流程图见图3。

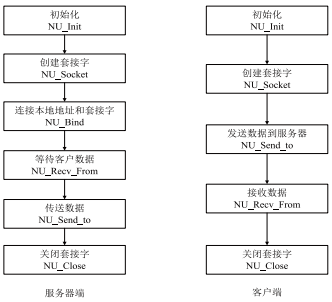

图3 UDP协议通信流程

Fig.3 UDP protocol communication process

7 代码实现(Code implementation)

如果要在硬件平台上使用基于Nucleus操作系统编写的嵌入式应用软件，就需要根据具体的硬件平台进行初始化。由于硬件平台的不同，初始化的过程虽然跟前文描述的差不多，但是在具体细节上存在很多的差异。特别是一些参数的配置上，不同的硬件平台有不同的参数配置。因此在下面的两段代码中，就没有提及Nucleus操作系统的初始化过程，而是直接描述网络通信模块。

7.1 TCP网络通信

以下是一段基于TCP协议，本机为服务器端的Nucleus NET代码。具体内容包含建立连接和收发数据两个任务。

```
status=NU_Create_Task(Main_Task……);//创建建立连接的任务[9]
status=NU_TCPRece_Task(……);//创建收发数据的任务[9]
void Main_Task(UNSIGNED argc,VOID*argv)//具体建立连接的函数
{STATUS status;
struct addr_struct servaddr;
status=NU_Init_Net(&Noncached_Memory);//网络堆栈初始化
if(status=NU_SUCCESS)
{//配置NU_DEVICE结构体
devices[0].dv_name="PQUICC_0";
devices[0].dv_flags=0;
devices[0].dv_init=PQUICC_Init;//初始化例程入口
memcpy(devices[0].dv_ip_addr,LocalIP,4);//本地IP
memcpy(devices[0].dv_subnet_mask,SubNet,4);//本地子网掩码
devices[0].dv_hw.ether.dv_irq=CN_DEMO_IRQ;//中断
devices[0].dv_hw.ether.dv_io_addr=CN_DEMO_IO_ADDR;
devices[0].dv_hw.ether.dv_shared_addr=0;
status=NU_Init_Devices(devices,1); //初始化设备
if ((TCP_Socket=NU_Socket(NU_FAMILY_IP,NU_TYPE_STREAM,0))>=0)//创建端口
{servaddr.name="servername";
servaddr.family=NU_FAMILY_IP;
servaddr.port=TCP_PROT;//TCP端口
memcpy(servaddr.id.is_ip_addrs,&LocalIP,4);//本地IP}
If(……)//套接字和地址进行绑成功
{Status=NU_Listen(TCP_Socket,50);开始第三步监听端口
If(status=NU_SUCCESS)
{Sockhost=NU_Accept(TCP_Socket,&elient_addr,0);//完成连接
If(Sockhost>=0)
{Status=NU_Broadcast_To_Queue(……);//队列通信
Status=NU_Close_Socket(TCP_Socket);关闭套接字}}}}}
void TCPRece_Task(UNSIGNED argc,VOID*argv)
{NU_Receive_From_Queue(……);//队列通信
While(con)
{NU_Fcntl(……);//设置套接字块标志
byte_recv=NU_Recv(……);收到数据
if(byte_recv=NU_NOT_CONNECTED)//如果连接断开
{con=0;
break;}
if(byte_recv>=0)//如果连接上
{……//处理接受的数据}
}
if(con==0)
{Status=NU_Terminate_Task(&TCPRece_Task);//中止任务}
NU_Suspend_Task(&TCPRece_Task);//挂起任务
}
```

7.2 UDP网络通信

以下是一段基于UDP协议，本机为服务器端的Nucleus NET代码。具体内容包含初始化和收发数据两个任务。

```
status=NU_Create_Task(……);//初始化任务
status=NU_UDPRece_Task(……);收发数据任务
void Main_Task(UNSIGNED argc,VOID*argv)
{STATUS status;
```

```
struct addr_struct  servaddr;
    status=NU_Init_Net(&Noncached_Memory);
if(status=NU_SUCCESS)
{devices[0].dv_name="PQUICC_0";
devices[0].dv_flags=0;
devices[0].dv_init=PQUICC_Init;//初始化例程入口
memcpy(devices[0].dv_ip_addr,LocalIP,4);//本地IP
memcpy(devices[0].dv_subnet_mask,SubNet,4);//本地子网掩码
devices[0].dv_hw.ether.dv_irq=CN_DEMO_IRQ;//中断
devices[0].dv_hw.ether.dv_io_addr=CN_DEMO_IO_ADDR;
    devices[0].dv_hw.ether.dv_shared_addr=0;
    status=NU_Init_Devices(devices,1);//初始化设备
//创建套接字
    if((UDP_Socket=NU_Socket(NU_FAMILY_IP,NU_TYPE_DGRAM,0))>=0)
{servaddr.name="servername";
servaddr.family=NU_FAMILY_IP;
servaddr.port=CN_DEFAULT_UDP_PROT;//UDP端口
memcpy(servaddr.id.is_ip_addrs,LocalIP,4);//本地IP
NU_Bind(UDP_Socket,&servaddr,0);
            }
}
}
void UDPRece_Task(UNSIGNED argc,VOID*argv)
{
//给已创建的UDP Socket添加广播属性
    NU_Setsockopt(VH_UDP_Socket,SOL_SOCKET,SO_BROADCAST,&optstatus,sizeof(INT16));
while(1)
{//调用NU_Recv_From服务接收UDP广播数据报
bytes_recv=NU_Recv_From(VH_UDP_Socket,recedata,CN_MAX_ETHER_UDP_LEN,0,&souraddr,&servlen);
    if(bytes_recv>0)
{……处理收到的数据}
    NU_Relinquish();//允许其他任务调用
}
}
```

在实际应用中，只要添加对硬件平台的相应驱动和对数据收发事件的处理，就可以编写出一个基本的使用TCP协议或UDP协议进行网络通讯的嵌入式应用软件，来完成所需达到的功能。

8　结论(Conclusion)

随着社会的进步和发展，嵌入式系统应用的范围越来越广。无论是工业、商业还是国防军工都有广泛的发展空间。而网络的应用，给嵌入式系统提供了更大的舞台，例如远程视频监控系统、远程语音通信系统等。而本文中提到基于Nucleus操作系统的TCP和UDP协议通信正是完成这些功能的基石。在这基础之上，只需要对通过TCP或UDP协议收发的数据按照一定的规则进行应用，就可以借助远程通讯实现更加强大的功能。也就是说，使得一款传统意义上的单机嵌入式产品具有了网络功能。

参考文献(References)

[1] CUI Kai,WANG Jie,ZHOU Kuanjiu,et al.Reliability of interrupt services for embedded systems[J].Journal of Tsinghua University,2016,56(8):878-884.

[2] CHEN Hongjun,LI Duo,YE Hua.Design and Realization of Embedded Termination of PLC Equipments' Remote Monitoring Based on 3G Technology Computer Science and Application,2015(5):186-193.

[3] WAN Hai,SUN Lei,WANG Tian,et al.Analysis of a method for attacking WTBlink layers in a train communication network[J].Journal of Tsinghua University,2016,56(1):42-50.

[4] 王伟.基于深度学习的网络流量分类及异常检测方法研究[D].中国科技大学博士论文,2018,06:25-27.

[5] vanquishasky.Nucleus实时操作系统分析报告[EB/OL].https://wenku.baidu.com/view/56560ba1b0717fd5360cdcad.html,2010-04-01.

[6] 汪碧华.嵌入式系统关键技术分析与开发应用[J].中国新通信2018,20(02):83.

[7] 刘宇帅,苏宇,王金波,等.航天嵌入式Linux实时性能优化研究[J].航天控制,2018,36(3):57-62;78.

[8] 吴泽智,陈性元,杨智,等.信息流控制研究进展[J].软件学报,2017,28(1):135-159.

[9] 黄啸,邓良,孙浩,等.基于硬件虚拟化的安全高效内核监控模型[J].软件学报,2016,27(2):481-494.

作者简介：

汪　洋(1974-)，女，硕士，高级工程师.研究领域：通信工程，信息系统.

禹　珉(1984-)，男，硕士，高级工程师.研究领域：信息系统，数据挖掘.