

STM32F407 的 FreeRTOS 多机通信系统设计

陈旅超¹, 蒲泽坤¹, 沈勇²

(1. 云南民族大学 电气信息工程学院, 昆明 650504; 2. 昆明欧力达机电有限公司)

摘要: 为了实现通过 PC 上位机控制单片机主机并通过主机控制从机的目的, 针对 STM32F407 芯片设计了一款基于 FreeRTOS 的多机通信系统。该系统基于 FreeRTOS 实时操作系统, 以 STM32F407 为核心, PC 上位机通过以太网向单片机主机发送信息, 同时命令单片机主机通过 CAN 总线控制单片机从机, 实现多机通信。本文将对 CAN 通信、以太网通信等各个部分进行分析并设计程序软件。实验结果表明, 本次设计的多机通信系统能够成功向单片机从机发送信息并接收来自从机的信息, 可以实现多个控制模块之间的数据传输, 多机之间相互协作, 共同完成目标任务。

关键词: STM32F407; FreeRTOS; 多机通信; TCP/IP; CAN

中图分类号: TN919; TP274

文献标识码: A

Design of Multi-machine Communication System Based on FreeRTOS for STM32F407

Chen Lvchao¹, Pu Zekun¹, Shen Yong²

(1. School of Electrical and Information Engineering, Yunnan Minzu University, Kunming 650504, China;

2. Kunming Oulida Electromechanical Co., Ltd.)

Abstract: In order to realize the purpose of controlling the single chip computer through the upper computer of the PC, and controlling the slave computer through the single chip computer, a multi-machine communication system based on FreeRTOS for STM32F407 chip is designed. The system is based on FreeRTOS real-time operating system, with STM32F407 as the core, the PC upper computer sends information to the single chip computer host through Ethernet, and CAN command the single chip computer host to control the single chip computer slave through CAN bus, to achieve the purpose of multi-machine communication. This paper will analyze and design program software for CAN communication, Ethernet communication and other parts. The experiment results show that the design of the multi-machine communication system can successfully send information to the slave microcomputer and receive the information from the slave microcomputer, can realize the data transmission between multiple control modules, so as to achieve the mutual cooperation between the multi-computer to complete the target task.

Keywords: STM32F407; FreeRTOS; multi-machine communication; TCP/IP; CAN

0 引言

单片机是指将完整的计算机系统集成在一块芯片上, 具有多种数据处理功能, 在各种系统(特别是嵌入式系统)中应用效果良好。单片机具有体积小、效率高、经济安全等优点^[1], 广泛应用于家电行业、交通运输行业、金融行业、通信行业等众多行业。如今, 人们对单片机的控制、通信等技术以及功能的要求越来越高, 传统的单片机单机控制已经无法满足市场需求^[2]。通过单片机实现多机协同, 已成为科技发展的必然趋势, 具有重要的实用价值和参考价值^[3-5]。

本设计使用 FreeRTOS 操作系统, 通过操作系统中的各个消息队列以及各种通信方式完成 PC 上位机对单片机主机以及从机的控制。本文设计该多机通信的目的有

以下几点: ①加快项目运行速率。多个单片机之间各自运行复杂项目的一部分工作, 最终通过通信方法实现相互协调工作, 远快于单个单片机运行项目。②缩短项目开发时间。针对复杂项目, 如果将项目拆分成多个简单项目并进行设计要远快于单独设计一个复杂项目。③便于性能测试。多机通信可以更加方便地完成各个模块的性能测试, 更加迅速地找到性能不足之处。

1 FreeRTOS 操作系统设置

FreeRTOS 任务线程的配置用于管理多个单片机之间通信。本文设置两个任务, 分别用于实现本机任务以及控制与从机之间的通信。使用 CubeMX 能够快速完成 FreeRTOS 嵌入式系统的快速配置。图 1 所示为分配完毕的任务线程。

Tasks and Queues		Timers and Semaphores		Mutexes		Events		FreeRTOS Heap Usage	
Config parameters		Include parameters		Advanced settings		User Constants			
Task Name	Priority	Stack Size	Entry Func.	Code Gene.	Parameter	Allocation	Buffer Name	Control Blo.	
defaultTask	osPriorityN...	512	StartDefaul...	Default	NULL	Dynamic	NULL	NULL	
PLC	osPriorityB...	1024	PLC_Task	As weak	NULL	Static	PLCBuffer	PLCContr...	
CNC	osPriorityH...	512	CNC_Task	As weak	NULL	Static	CNCBuffer	CNCContr...	

图1 FreeRTOS 线程设置

同时,需要设置用于传递信息的信息队列,这些消息队列被用于接收来自 PC 上位机或者单片机从机的信息。具体配置如图 2 和图 3 所示。

Queues					
Queue Name	Queue Size	Item Size	Allocation	Buffer Name	Control Block Name
PLCMsg	4	uint32_t	Static	PLCMsgBuffer	PLCMsgControlBl...
CNCMsg	8	uint32_t	Static	CNCMsgBuffer	CNCMsgControlBl...

图2 消息队列配置

Mutexes		
Mutex Name	Allocation	Control Block Name
PLCMutex	Static	PLCMutexControlBlock
CNCMutex	Static	CNCMutexControlBlock

图3 互斥量配置

完成操作系统配置后,初始化各个模块,在 StartDefaultTask 函数中运行程序,完成所有软件程序的驱动初始化,将 TCP/IP 通信程序放入 for 循环中,保证可以实现运动控制器与上位机之间的实时通信过程。具体实现程序如下(其中 CncDaemon 程序用于实现实时通信过程):

```
void StartDefaultTask(void * argument){
    FRESULT res;
    /* init code for USB_DEVICE */
    MX_USB_DEVICE_Init();
    vRegisterSampleCLICommands();
    res=fileSystemInit();
    if(res!=FR_OK){
        WritLen((uint8_t *)fileError,strlen(fileError));
    }
    settings_init();
    /* init code for LWIP */
    MX_LWIP_Init();
    TFTP_ContextInit();
    osDelay(20);
    tftp_init(&mytftp);
    /* USER CODE BEGIN StartDefaultTask */
    PLCEventHandle = osEventFlagsNew (&PLCEvent _ attrib-
    utes);
```

```
CNCEventHandle = osEventFlagsNew (&CNCEvent _ attrib-
    utes);
    PLCHandle=osThreadNew(PLC_Task,NULL,&PLC_attrib-
    utes);
    CNCHandle = osThreadNew (CNC_Task, NULL, &CNC _ at-
    tributes);
    /* Infinite loop */
    for(;;){
        //Check conn
        CncDaemon();
        ProcessMessage();
        osDelay(5);
    }
    /* USER CODE END StartDefaultTask */
}
```

系统整体实现流程图如图 4 所示。单片机主机接收到数据后,判断是否执行通信任务,如果触发通信任务,则向单片机从机发送信息并接收返回的信息,同时将返回的信息发送至上位机中。

2 与上位机之间的通信软件设计

本文设计的多机通信系统与 PC 上位机之间通过以

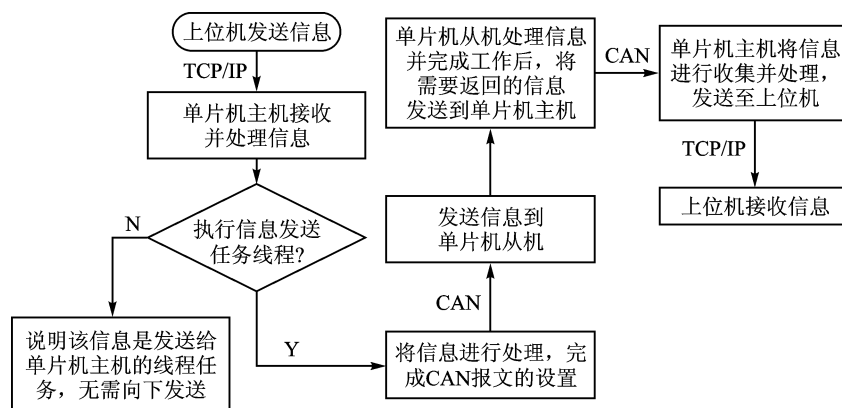


图4 系统整体实现流程图

太网进行连接,需要完成的目标包括修改 IP 地址以及端口号、完成数据的接收以及需要发送的数据采集。

2.1 修改 IP 地址以及端口号

文中使用 TCP/IP 协议进行连接,并修改连接的 IP 号以及端口号。具体程序如下:

```
if(conn != NULL){
    ret=netconn_delete(conn);
    LWIP_DEBUGF(API_LIB_DEBUG,("netconn_delete = %d\n",ret));
    conn=NULL;
}
```



```

conn=netconn_new(NETCONN_TCP);
if(conn!=NULL){
    ret=netconn_connect(conn,&settings.OctopusIP,settings.
OctopusPort);
    if(ret!=ERR_OK){
        LWIP_DEBUGF(API_LIB_DEBUG,(connect_failed));
    }
    else{
        netconn_set_recvtimeout(conn,RETIMEOUT);
        LWIP_DEBUGF(API_LIB_DEBUG,(Connect_Ok));
        reg_no=0;
    }
    return ret;
}

```

其中 IP 地址以及端口号的修改如下:

```

IP4_ADDR(&settings.IpAddress,DEFAULT_IP_0,DEFAULT_IP_1,
DEFAULT_IP_2,DEFAULT_IP_3);
IP4_ADDR(&settings.netMask,NETMASK_ADDRESS_0,
NETMASK_ADDRESS_1,NETMASK_ADDRESS_2,NET-
MASK_ADDRESS_3);
IP4_ADDR(&settings.Gateway,GATEWAY_ADDRESS_0,
GATEWAY_ADDRESS_1,GATEWAY_ADDRESS_2,GATE-
WAY_ADDRESS_3);
IP4_ADDR(&settings.OctopusIP,Octopus_ADDRESS_0,Octo-
pus_ADDRESS_1,Octopus_ADDRESS_2,Octopus_ADDRESS_3);

```

2.2 数据接收及发送程序设计

数据接收及发送程序应用在 FreeRTOS 实时操作系统的主循环之中,启动运动控制器之后一直处于运行状态,每隔 10 ms 左右被任务线程调用一次。该部分共完成 3 部分任务:其一是保证实时接收来自上位机的命令信息,并将其写入寄存器内;其二是将来自运动控制器的信息实时传输给上位机;其三是对通信连接的稳定性作出反应,如果连接失败或者超时则反馈对应的信息。具体程序如下:

```

struct netbuf * recbuf=NULL; //接收到的信息
static int dog=RE_DOG_TIME/RETIMEOUT;
static int time=0;
static bool offline=true;
if(dog<0){
    offline=(CncConnect()!=ERR_OK);
    dog=RE_DOG_TIME/RETIMEOUT;
}
if(offline){
    dog--;
}
if(!offline){
    ret=netconn_recv(conn,&recbuf);//数据接收

```

```

switch(ret){
case ERR_OK:
    //todo:use recbuf to do something!
    //getrespond,and send respond
    ProssGrblCmd(recbuf);//数据处理
    time=0;
    if(send_buf!=send_buf1){
        send_buf=send_buf1;
    }
    else{
        send_buf=send_buf2;
    }

    sendLen=ProssRespond(send_buf,TCP_SEND_SIZE);
    ret=netconn_write(conn,send_buf,sendLen,NETCONN_
MORE);//发送信息
case ERR_TIMEOUT:
    time++;
    if(time>180){
        offline=true;
        time=0;
        LWIP_DEBUGF(API_LIB_DEBUG,("ERR_TIMEOUT
\r\n"));
    }
    break;
}

```

其中,变量 dog 用于判断连接出错的次数。若错误次数超出预先设置的值,则重新与 PC 上位机进行连接,减少冗余。

2.3 发送数据采集

通过 netconn_write() 向上位机发送数据不止需要得到数据内容还需要数据长度,因此在采集数据的同时还需要采集该数据信息的长度。具体程序如下:

```

pbuff=send_event->EVENT_data;
pf=(uint8_t *)&CAN_IN1B;
for(i=0;i<EVENTValueSize;i++){
    *pbuff=*pf;
    pbuff++;
    pf++;
    len++;//数据长度
}

```

3 与从机之间通信的程序设计

本设计使用 CAN 通信获取从机中的数据,并且将 CAN 通信与 FreeRTOS 相结合,当触发相关任务 PLC_Task 时,将开始进行信息的接收以及传递。

3.1 CAN 通信配置

首先需要设置过滤器,过滤器可以用于选择使用哪个

FIFO 作为邮箱进行数据的存储。本设计中使用标识符列表模式设置过滤器,并且设置两个过滤器使得在通信过程中只能通过扩展帧以及数据帧。具体程序如下:

```
CAN_Filter_init. FilterActivation=ENABLE; //过滤器使能
CAN_Filter_init. FilterFIFOAssignment = CAN_FilterFIFO0;
//选择使用哪个 FIFO, 可为 CAN_FilterFIFO0, CAN_FilterFIFO1;
CAN_Filter_init. FilterMode=CAN_FILTERMODE_IDLIST;
//标识符列表模式
CAN_Filter_init. FilterScale=CAN_FILTERSCALE_32BIT;
CAN_Filter_init. FilterBank=FilterNumber;
CAN_Filter_init. FilterIdHigh=FilterID>>16;
CAN_Filter_init. FilterIdLow=FilterID & 0xffffUL;
FilterID=ExtId_2<<3|CAN_ID_EXT|RTR;
CAN_Filter_init. FilterMaskIdHigh=FilterID>>16;
CAN_Filter_init. FilterMaskIdLow=FilterID & 0xffffUL;
HAL_CAN_ConfigFilter(&hcan1, &CAN_Filter_init);
```

下一步是设置 FIFO 接收函数。若接收到信息,则将信息发送到 PLCMsgHandle 队列尾部,并返回是否成功入队的信息。

本设计中,将 CAN 的 FIFO 接收回调函数与 PLCMsgHandle 队列结合,将收到的数据写入 PLCMsgHandle 中,若有信息入队,则根据信息内容选择不同的变量进行存储。具体程序如下:

```
BaseType_t xStatus;
BaseType_t xHigherPriorityTaskWoken;
HAL_StatusTypeDef status=HAL_CAN_GetRxMessage(hcan,
CAN_RX_FIFO0, &RxMsgHd, (uint8_t *) &RxMsg. CanRxMsg. B[0]);
if(status == HAL_OK){
    RxMsg. ExtId=RxMsgHd. ExtId;
    xStatus = xQueueGenericSendFromISR (PLCMsgHandle,
(void *) &RxMsg, &xHigherPriorityTaskWoken, queueSEND_TO_BACK);
}
```

3.2 CAN 通信接收程序设计

首先将数据内容放入宏定义变量 CAN_IN1B_LOW 以及 CAN_IN1B_Hi 中,然后将 CAN 通信中获得的数值赋值给 send_even。具体程序如下:

```
xStatus=xQueueReceive(PLCMsgHandle, &RxMsg, 0);
if(xStatus==pdPASS){
    //处理接收数据
    //copy to IB
    CAN_IN1B_LOW=RxMsg. CanRxMsg. D[0];
    CAN_IN1B_Hi=RxMsg. CanRxMsg. D[1];
    CAN_UPDATA=! CAN_UPDATA;
}
```

3.3 CAN 通信发送程序设计

在 HAL 库中,若想通过 CAN 发送数据,只需要使用 HAL_CAN_AddTxMessage() 程序即可,在使用前先设置需要发送的数据。需要对标准标识符以及扩展标识符进行选择,本设计中选择使用扩展标识符。同时,设置消息类型为一帧 8 位。选择时间触发模式,该模式用于插入时间戳。

```
TxMsgHd. IDE=CAN_ID_EXT; //扩展帧模式
TxMsgHd. ExtId=G200CMD_CANID; //扩展标识符
TxMsgHd. DLC=8; //数据长度
TxMsgHd. StdId=CAN_ID_STD; //标准标识符
TxMsgHd. IDE=CAN_ID_EXT;
TxMsgHd. RTR=CAN_RTR_DATA; //帧类型
TxMsgHd. TransmitGlobalTime=ENABLE; //发送时间标记
```

设置好需要发送的报文信息之后,便可以通过 HAL_CAN_AddTxMessage() 函数发送信息:

```
if(HAL_CAN_AddTxMessage(&hcan1, &TxMsgHd, (void *)
&CAN_out, &CAN_Tx_MailBox1)!=HAL_OK){
    Can_Send_error++;
}
```

4 实验测试及结果分析

通过 PC 上位机中的调试助手向单片机主机发送数据,并触发向 CAN 传输信息的任务,在测试中将任务触发条件设置为按动按键 KEY1,并设置从机在收到信息后将收到的信息返回至主机中,主机通过网口将信息发送至 PC 上位机中,同时将程序内部各模块接收及发送情况在串口中进行显示,结果如图 5 所示。

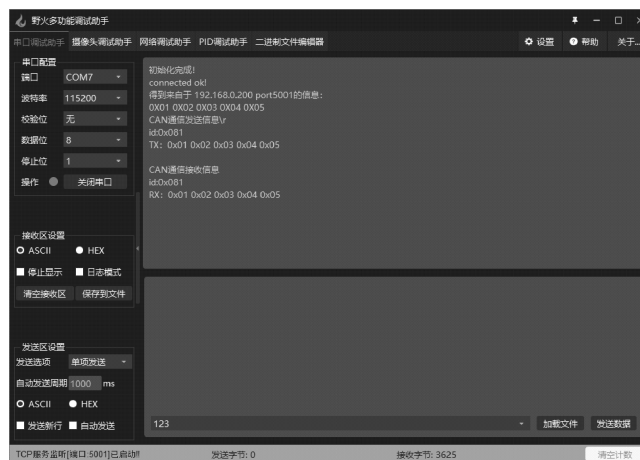


图 5 串口报文

同时,可以在网络调试助手中查看收到返回的信息,如图 6 所示。

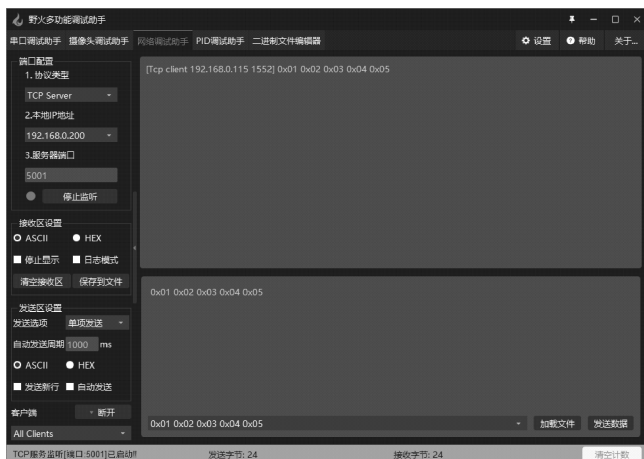


图6 网口接收信息

实验结果表明,当PC上位机通过网口向单片机主机发送数据“0x01 0x02 0x03 0x04 0x05”后,只需按下按键,单片机主机即可成功通过CAN通信向从机发送这些信息,并且将来自于从机的信息发送给PC上位机。可见,本设计成功通过FreeRTOS操作系统连接PC上位机,并与多个单片机进行数据传输。

5 结 语

针对多机通信的问题,本文通过设置FreeRTOS线程

任务,并将各个通信模块与FreeRTOS中的消息队列相结合,成功实现多机之间的通信任务,并且通过设置线程任务的方式控制多机之间的通信过程,提高了单片机的运行效率。本文设计的多机通信系统可移植性强,适用于各种需要多机配合完成的项目。本文只配置了一个任务用于与从机进行通信,如有需要还可以设置多个任务,实现与更多模块之间的通信。

参考文献

- [1] 张贺帅,张哲.以单片机为基础多机通信系统的设计对策分析[J].长江信息通信,2022,35(3):117-119.
- [2] 杜路泉,莫建麟,王玉晶,等.基于单片机的多机通信系统设计[J].宁德师范学院学报(自然科学版),2021,33(2):153-158.
- [3] 胡念祖.基于ARM和嵌入式Web服务器的自动售货机设计[D].西安:西安工程大学,2019.
- [4] 吴乐明.基于单片机技术的多机通信系统设计[J].中国新通信,2018,20(9):7-8.
- [5] 谭玉琪,王超,汤吉昀.基于Proteus的单片机多机通信的设计与仿真[J].电子设计工程,2019,27(3):164-168.

陈旅超(硕士研究生),主要研究方向为嵌入式运动控制。通信作者:陈旅超,2447475008@qq.com。

(责任编辑:薛士然 收稿日期:2023-05-08)

6 景中的物体。此外,如果给它一幅荷兰画家维米尔(Vermeer)的作品,AI可以扩展它的“戴珍珠耳环的女孩”,使人们对画中她所处的房间有一个可信的印象。

然而,作家(比如Elektor和其他知名媒体的编辑团队)对ChatGPT的推出感到震惊。这种AI可以以会话方式和多种语言与用户进行交互。到目前为止,关于碳化硅(SiC)MOSFET和氮化镓(GaN)晶体管优点的讨论,以及它们相对于硅MOSFET的优点,已经出现了高度精准的效果。因此,即使是小众话题,似乎也被很好地涵盖了。

虽然这个工具非常聪明,但它只知道训练之前相关话题的答案。因为它不是持续学习的,所以不会了解最新的时事或最新的K-pop乐队剧(shame)。另一个小问题

是,过了一段时间,答案似乎变得有点老套和公式化。

总之,嵌入式系统和AI越来越多地被用于创建智能、自主设备和系统中。嵌入式系统为AI算法的运行提供了硬件和软件平台,而AI算法使这些系统能够以更智能和更人性化的方式感知、分析和响应其环境。随着嵌入式系统和AI的能力不断提高,我们期待在机器人、医疗保健、交通运输等领域看到一系列令人兴奋的新应用。

本文授权来自本刊的合作伙伴Elektor媒体集团,如果希望免费订阅Elektor的英文在线内容,请访问<https://www.elektormagazine.com/>。

(责任编辑:芦潇静 收稿日期:2023-06-30)

IAR 全面支持兆易创新基于 Arm Cortex - M7 内核的超高性能 MCU

嵌入式开发软件和服务的全球领导者 IAR 与业界领先的半导体器件供应商兆易创新(GigaDevice)联合宣布,最新发布的 IAR Embedded Workbench for Arm 9.40 版本已全面支持兆易创新基于 Arm Cortex - M7 内核的超高性能 MCU 微控制器——GD32H737/757/759 系列,为开发人员提供高效的工具链。

GD32H737/757/759 系列超高性能 MCU 基于 600 MHz Arm Cortex - M7 内核,凭借双发射 6 级流水线架构,以及支持高带宽的 AXI 和 AHB 总线接口,可以获得更高的主频及处理性能,达到 1 552 DMIPS 和 2 888 CoreMarks 的优异结果。GD32H7 突破了 MCU 的性能边界,为复杂运算、多媒体技术、边缘 AI 等高级创新应用提供强大的算力支撑,进一步迭代拓展了兆易创新在超高性能领域的产品布局,并持续以强大供应链和高品质打造平台化优势,赋能开发者应对未来挑战。