(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103115665 A
(43)申请公布日 2013.05.22

(21)申请号 201110364710.9

(22)申请日 2011.11.17

(71)申请人 王明哲
地址 150001 黑龙江省哈尔滨市南通大街145号哈尔滨工程大学21号254室
申请人 黄丽莲

(72)发明人 黄丽莲 王明哲

(51)Int.Cl.
G01H 17/00(2006.01)

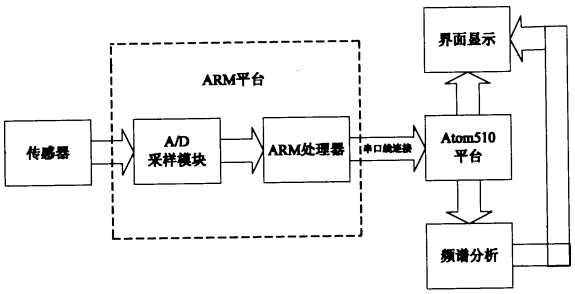
权利要求书1页 说明书13页 附图2页

(54)发明名称

基于ATOM与ARM的在线式实时振动检测仪

(57)摘要

本发明提供一种用于检测设备运行状况的在线式实时振动检测仪，是基于新兴的英特尔ATOM平台和比较成熟的ARM嵌入式技术所开发的。该发明首先通过ARM平台上的A/D模块进行数据采集处理，将振动模拟信号转化为数字信号，然后将数字信号通过串口传输到ATOM平台上，在此平台上使用图形界面对进行频谱分析后的数据进行显示，便于诊断。其优点是：无需人工巡检，提高工作效率，实时性好，具有良好的人机交互界面，处理能力强，简单易用，适用于环境恶劣下的设备振动检测。



CN 103115665 A

1. 基于因特尔凌动 ATOM 和友善之臂 ARM mini2440 的在线式实时振动检测仪，是一种在线式实时检测系统，包括传感器、ARM 平台和因特尔凌动 ATOM 平台。其技术特征在于传感器与友善之臂 mini2440 平台相连，友善之臂 mini2440 平台与因特尔凌动 ATOM 平台相连，传感器将设备振动信号转换成电信号，ARM 平台上的 A/D 模块对电信号进行采样，并通过友善之臂 mini2440 处理器的控制传输到因特尔凌动 ATOM 平台，因特尔凌动 ATOM 平台对接收到的数据进行频谱分析，并在图形化界面上显示。

2. 根据专利要求 1 所述的在线式实时振动检测仪，其技术特征在于其所述的因特尔凌动 ATOM 平台在 Linux 操作系统下管理数据和开发人机界面，并可远程控制友善之臂 mini2440 处理器改变采样速率，对设备状态进行实时分析和现场诊断。

3. 根据专利要求 1 所述的在线式实时振动检测仪，其技术特征在于其所述的因特尔凌动 ATOM 平台和友善之臂 ARM mini2440 平台之间通过串口线连接，实现对振动信号的实时处理。

基于 ATOM 与 ARM 的在线式实时振动检测仪

技术领域

[0001] 本发明涉及一种检测设备运行状况的在线式机械振动检测仪。

背景技术

[0002] 目前，国内外机械故障的诊断，多数采用振动诊断，大致可以分为以下四类：

[0003] 1) 传统的在线式检测仪，综合了先进的计算机技术、传感器技术和网络技术，具有早期预测故障的能力。但是，由于系统构成复杂，操作比较繁琐，费用昂贵，故应用上受到限制。

[0004] 2) 传统的离线式检测仪，具有适用范围广、能够对各类设备进行诊断分析的特点，是传统的在线式的良好补充；然而由于体积大、不便携带，实时性差。

[0005] 3) 基于单 CPU(Central Processing Unit) 的手持式数据采集器，该类设备功能比较简单。单 CPU 系统受其硬件资源、算法和速度的限制，主要完成振动信号时域特征的测量，并且测试精度低，实时性差，数据存储容量小，一般无故障诊断功能，只能借助于 PC 机对采集过来的数据进行处理。

[0006] 4) 基于主从式双 CPU 的手持式数据采集、分析器。该类系统可以充分利用前端 CPU 控制能力强的优势和后端 CPU 强大、快速的数据处理能力，可以完成对振动信号的数字滤波、时域特征检测、频域特征检测，但是缺乏友好的人机交互界面，也缺少有效的数据管理与存储功能，对复杂的多任务处理就会显得吃力。

[0007] 现有在线式操作复杂，费用昂贵；手持式缺乏人机交互界面，不能在线随时检测设备运行中出现的机械故障，特别是当设备工作环境恶劣，人力资源有限的情况下，手持式的应用更受到了限制。因此不能保证机械设备安全运行，易造成机械设备的损坏，给企业带来严重的经济损失。

[0008] 基于上述背景，本发明提出一种在线式实时振动检测仪。

发明内容

[0009] 本发明的目的在于提供一种可抵抗恶劣环境，具有良好的人机交互界面、简单易用、实时性好、节省人力资源、处理能力强的振动检测仪，针对机械的运行状态进行实时在线监测。

[0010] 本发明的目的是这样实现的：基于 ATOM 和 ARM 的在线式实时振动检测仪，包括传感器、ARM 平台和 ATOM 平台。其特征在于传感器与 ARM 平台相连，ARM 平台与 ATOM 平台相连，传感器将设备振动信号转换成电信号，ARM 平台上的 A/D 模块对电信号进行采样，并通过 ARM 处理器的控制传输到 ATOM 平台，ATOM 平台对接收到的数据进行频谱分析，并在图形化界面上显示。

[0011] ATOM 平台在 Linux 操作系统下管理数据和开发人机界面，并可远程控制 ARM 处理器改变采样速率，对设备状态进行实时分析和现场诊断。

[0012] ATOM 平台和 ARM 平台之间通过串口线连接，实现对振动信号的实时处理。

[0013] 有益效果

[0014] 本发明对比已有技术具有以下创新点：

[0015] 利用主从式双 CPU 的系统实现在线检测设备的运行状况，实时性好。使用 ATOM 平台作为主处理器，可以充分利用 ATOM 平台运行的操作系统管理数据和开发人机界面，操作简单。

[0016] 本发明对比已有技术具有以下显著优点：

[0017] 1. 适用于设备工作环境恶劣的振动检测；

[0018] 2. 无需人工巡检，降低工作量，节省时间，大大提高工作效率；

[0019] 3. 实时性好、简单易用；

[0020] 4. 良好的人机交互界面、处理能力强。

附图说明

[0021] 图 1 是系统整体框图

[0022] 图 2 是软件设计模块图

[0023] 图 3 是 Linux 内核配置界面

具体实施方式

[0024] 1. 硬件部分

[0025] 本发明的整体原理方框图如图 1 所示，由传感器、ARM 平台和 ATOM 平台组成。

[0026] ATOM 平台采用 Intel ATOM Z510 处理器，搭配 Intel us15w 单芯片系统控制器，相比传统 x86 双桥片架构功耗更低，配置更加灵活。UP-Atom510 外扩常用接口和控制器，包括 5 个 USB 接口，100M 网卡，1000M 网卡，PS2 键盘鼠标接口，音频接口，串口以及 VGA 接口，同时底板上还有 PC104 总线接口。

[0027] 工作过程如下：机械设备的振动信号由振动传感器转换为电信号，经 A/D 转换芯片转换成数字信号，直接送入 ARM 处理器进行存储。将存储数据经过串口线传输到远程 ATOM 平台，在 Linux 系统下对数据进行傅里叶变换，进行频谱分析，并将处理结果送人机界面进行显示，通过频谱图了解机械设备运行状况，进而进行诊断。

[0028] 2. 软件部分

[0029] 2.1 软系统的总体设计

[0030] 软件部分采用模块化程序设计方法（即系统的总体设计由各个子程序完成）和 C 语言编程，使得程序结构清晰，便于以后进一步扩展系统的功能。整个程序由数据采集、与串口通讯模块、频谱分析和频谱显示模块构成。系统的主模块图如图 2 所示。

[0031] 2.2 开发环境的搭建

[0032] 安装嵌入式 Linux 内核及设备驱动全部源代码（光盘安装后建立完备的开发环境）。交叉编译的工具被放置到 /opt/host/armv4l 目录下。步骤如下：

[0033] (1) 进入开发环境下的内核源码树，然后 make menuconfig。

[0034] (2) 按照应用需要对内核选项和驱动模块进行选择和配置（配置界面如图 5.8 所示）。

[0035] (3) 完成自己的设置后，保存配置，退出。

[0036] (4) 执行 make zImage 编译生成自己定制的内核映像文件，该文件会被自动复制到 /tftpboot/ 目录下以供烧写。

[0037] (5) 如果需要重新编译内核，可以进入 Linux 内核目录，重新编译内核映像文件 zImage。先进入内核源码树，然后键入下列命令：

[0038] make clean

[0039] make dep

[0040] make zImage

[0041] 内核配置界面如图 3 所示

[0042] 2.3 编译应用程序，生成可执行文件，然后需要将其存放到开发板 Linux 的 ramdisk 文件系统中。这就需要重新制作 ramdisk 文件系统映像，并烧写 flash。ramdisk.image.gz 为 LINUX 的文件系统映像的压缩文件。具体制作过程如下：

[0043] (1) 将 ramdisk.image.gz 移至 /tftpboot 目录下，在同目录下解压为 ramdisk.image。

[0044] (2) 然后用命令 mount-o loop ramdisk.image/mnt 将 ramdisk.image 文件系统映像文件 mount 到 /mnt/cdrom 目录中。

[0045] (3) 进入 /mnt/cdrom/bin/ 目录，复制应用程序的目标文件 MiniM 到该目录下。

[0046] (4) 关闭 /mnt/cdrom/ 窗口，umount/mnt/cdrom。

[0047] (5) 进入 /tftpboot 目录，在终端内键入命令 gzip ramdisk.image 压缩文件系统的映像文件成 ramdisk.image.gz，并存放到 /tftpboot/ 目录下。

[0048] 3A/D 数据采集与存储模块

[0049] 3.1A/D 转换模块初始化

[0050] ARM 开发平台上的 A/D 型号为 SD16，要使 SD16 能够正常采集模拟量并进行转换，需要对 SD16 进行初始化，初始化主要是对一些相关寄存器的设置。在控制寄存器 SD16CTL 中写入控制字 0X0114，打开 SD16，可以进行转换，即设定 SD16 内核时钟源为系统子时钟，内部参考电压产生器关闭，不消耗功耗，关闭溢出中断允许位，选择 SD16 的时钟源分频因子；在控制寄存器 SD16CCTLx 中写入控制字 0X008E，即设定转换模式为序列通道多次转换模式，过采样速率设定为 256，同时指定了用于保存结果的转换存储寄存器为 SD16MEM0-SD16MEM2：当转换正常结束时，转换结果写入选定的存储寄存器 SD16MEMx，相应的中断标志位 SD16IFGx 置位。

[0051] 3.2 模数转换模块的初始化程序

[0052] 本装置选择序列通道多次转换模式。该模式对有顺序的多个通道做多次转换。转换结果将按照顺序在转换结果存储寄存器存放。对该序列的转换一直进行直到软件停止为止。模数转换模块的初始化程序如下：

[0053]

#include<msp430x42x.11>

[0054]

```
void SDl6init(void)
{
    SDl6CTL=SDl6REFoN+SDl6SSEL0;
    //SDl6REFON 为 1，选择 1．2V 内部参考电压
    NSDl6SSEL0 为 01， 选择子系统时钟 SMCLK 为时钟源
    SDl6CCTL0=SDl6SNGL+SDl6IE+SDl60SRx+SDl6SC：
    //SDl6SNGL 为 0，选择序列转换模式
    //SDl6IE 为 1，表示使能中断
    HSDl60SRx 为 oo，表示选择 256 过采样率
    //SDl6sC 为 1，表示使能 ADC
     SDl61NCTL0 F SDl6INTDLY_01
     //sDl61NTDLY 0 位 oo，表示第四个采样中断
)
```

[0055] 3.3 开关量采集的软件设计

[0056] 本装置设计了 8 路开关量采样通道，占用 ARM 单片机的 P1 口的 P1.0、P1.1、P1.2、P1.3、P1.4、P1.6 端口和 P2 口的 P2.0、P2.1 端口。采集开关量时，可直接通过扫描判断 P1、P2 口输入寄存器每一位的高、低电平信号，来确定引脚电平，即开关量输入。初始化时，需要在设置 P1、P2 口的方向寄存器中，设置 P1、P2 口为输入状态，使用的语句为：

[0057] P1DIR ＝ 0x00 ;// 设置 P1 口为输入

[0058] P2DIR ＝ 0x00 ;// 设置 P2 口为输入

[0059] 扫描判断 P1 口（P2 口同理）输入寄存器每一位的高、低电平信号，使用的语句为：

[0060] if(P11N&0x00// 判断 P1.0 引脚电平

[0061] KKin0 ＝ I ;// 如果为高，则开关输入变量 KKin0 为 1

[0062] else KKin0 ＝ 0 ;// 如果为低，则开关输入变量 KKin0 为 0

[0063] 判断其它位的语句同理，要采集 8 路开关输入量，需要把 8 个引脚上的电平全部扫描一遍。

[0064] 3.4 采样控制模块

[0065] AD 采样器 THSI206 的工作好坏直接影响采样数据的准确性，该模块主要完成对采样方面的工作，包括发送采样命令、接受采样数据和中断处理，这些处理主要集中在采样函数 sample_one_point() 里完成，其主要代码如下所示

[0066]

```
Void sample_one_point(unsigned char *pbuffer)
{
    P2=0x8f  / / 模拟开关地址设置
    convst=0;  / / 发送采样命令;
    convst_ad=0;
    convst_ad=1;
    convst_ad=1;
   Pbuffer[0]=Xbyte(PORT_AD);//读取数据
   Pbuffer[1]=Xbyte(PORT_AD);
   Pbuffer[2]=Xbyte(PORT_AD);
   Pbuffer[3]=Xbyte(PORT_AD);
}
```

[0067] 模拟开关的地址选择通过 P2 口设置,AD 采样器的采样命令通过向 CONVST 引脚信号置一个负脉冲完成。数据的读取通过 XBYTE 指令完成。低四位与 8255 的选通引脚相连,它实际上是将 8255 置于不选通状态,同时 MOVX 指令 P3.7 引脚上出现 RD 有效信号 1,RD 又与 THS206 的 RD 相连,而且 THS206 的触发级别为 4,这样就连续读出四路采样结果数据。再改变模拟通道地址,这样就读出总共 32 路采样结果数据。

[0068] 3.5 数据存储模块 :

[0069] 数据存储模块主要任务是完成在采样存储工作方式下的 FLASH 存储器的数据的读取、载人和编程。数据读取函数是 rbyte32(),其主要代码如下所示 :

[0070]

```
Void rbyte32(unsigned long p_flash,unsigned char *pbuffer)
{
    unsignecl char i;
    whlle(flash_ready：=0)j
    for(1=0ji<32：i++)
    (
      XBYTE[PORTA_8255]={unsigned char)P_flash;
      xBYTE[PORTB_8255]=(unsigned char)(P_flash>>8);
      xBYTE[PORTC_82551=(unsigned char);
      ((p_flash>>l 6)&0X00000003);
}
```

[0071] 数据读载入函数是 w32byte-sector(),其主要代码如下所示：

[0072] 下面是代码程序：

[0073]

```
Void w32byte—sector(unsigned long p_flash,unsigned char *pbuffer)
{
    unsignea char P_sectorbyte;
    P_sect0lbyte=(unsigned char)P_flash;
    for(i=0;i<32;i++)
    {
    XBYTE[PORTA_8255]=p_sectorbyte;
    P0=pbuffer[i];
    nce_flash=0;
    nwe_flash=0;
    nce_flash=1;
    nwe_flash=1;
     P_sectorbyte++;
}
```

[0074] 当载人 8 个 32 路信号的采样点数据后，FLASH 的 256 字节缓冲区被写满，必须将这 256 字节的数据编程到 FLASH 中去，其编程的平均时间为 10ms，大于采样间隔 2.5ms，所蹦在编程完成以前就要开始下一次的采样，而且在一次编程期间，FLASH 缓冲器中的 256 个字节数据不能被修改，否则这 256 字节的数据就要重新编程，所以就要利用单片机内部所设置的数据缓冲区来暂时存储采样数据。在 FLASH 编程期间，系统将采样 4 个点（32 路），数据缓冲区的大小可以存放 5 个点（32 路）的采样数据，还剩一个点的数据存放空间，由于 FLASH 的载人时间纳秒级，远小于采样间隔，所以可以在一个采样间隔内将暂存的 4 个点数据载人 FLASH，释放单片机数据缓冲区的空间。

[0075] 3.6 与串口的通信模块

[0076] 与串口通信模块主要完成单片机通过并行接口和计算机进行采样数据的传送。在数据发送模式和采样发送模式下都需要向 ATOM 平台传送采样数据，但两种模式下的并行接口处理略有不同，下面是两个传送函数的程序代码：

[0077] 数据发送工作模式：

[0078]

```
void sendpc—one_point(unsigned char+p_wbuffer）
{
    unsigned char i;
    for(i=0； i<32； )
    {
    xBYTE[PORTB_8255]=pwbuffer[i]；
    while(ndatastkepp==1);
    if(nwyir_epp==0)
      {
       i=0；
       nwait_epp=l；
      }
}
```

[0079] 4 串口通信模块

[0080] 4.1 系统启动后，ATOM 平台与 ARM 平台将以事先约定的通信协议和数据传输协议进行通讯。其中，通信协议是为保证上下位机通讯成功而达成的握手协议；数据传输协议是上下位机间的通讯数据包格式的约定，分为上传数据包格式和下传数据包格式。上传数据包是下位机从现场采集的数据打成包后通过串口传输到上位机的；下传数据包是上位机往下位机传送的配置信息指令包。

[0081] (1) 通信协议

[0082] 上位机通过 COM 串行端口设备与下位机进行通信，利用串口线来完成数据的传输和接收的功能，并且相互之间达成自定义的简单通信协议。本系统采用 COM1 端口来实现通信，只需对串口实行一定的参数配置，即可完成数据的传输功能。其简单的通信协议。

[0083] (2) 数据包传输协

[0084] 数据传输协议是上下位机的通讯数据包格式的约定，是下位机从现场采集的数据打成包后通过串口传输到上位机的。数据包由三个子包构成，分别是：通道 1 快变数据子包、通道 2 快变数据子包、缓变数据子包。其中 1、2 通道快变数据子包中格式化存放的快变数据即指传感器采集的振动信号加速度值，缓变数据指下位机采集卡的电池容量。注意，数据包格式的约定包头和包尾都还需要有校验码，以防数据包读取出错。本数据包格式包头

校验码是 AA，包尾校验码是两个 DD，DD。当然为了保证数据包的连续性每个包还需要有包号和通道号。数据包大小为 12 个字节，具体约定格式见下表 1

[0085] 表 1 数据包格式表

[0086]

包头校验码	包号	通道号	第一通道数据	第二通道数据	缓变数据	包尾校验码
AA	2 字节	1 字节	2 字节	2 字节	2 字节	DD DD

[0087] 现对串行端口操作过程作简要介绍：

[0088] 4.2 串口的初始化

[0089] 串行数据传输模块包括串行口初始化子程序和数据传输子程序，各子程序分别如下。其中数据传输采用查询方式，也可以方便地改为中断方式。

[0090] ATOM 平台接收数据所用 C 语言程序包括初始化

[0091] 子程序和接收子程序。各子程序分别如下：

[0092]

```
void init_com1 (void) / *初始化子程序 * /
{
outportb(0x3fb ,0x80) ; / *线控制寄存器高位置 1 ,使波特率设置有效 * /
outportb(0x3f8 ,0x18) ; / * 波特率设置 ,与单片机波特率一致为 4800bps * /
outportb (0x3f9 ,0x00) ;
outportb ( 0x3fb , 0x03) ; / * 线控制寄存器设置 ,8 位数据位 ,1 位停止位 ,无奇偶校验 * /
outportb(0x3fc ,0x03) ; / * Modem 控制寄存器设置 ,使 DTR 和 RTS 输出有效 * /
outportb(0x3f9 ,0x00) ; / * 设置中断允许寄存器 ,禁止一切中断 * /
}
void receive- data (void) / * 查询方式接收数据子程序 */
{
while ( ! kbhit () )
{
```

[0093]

```
while ( ! (inportb (0x3fd) &0x01) ) ;/ * 若接收寄存器为空 ,则等待 * /
printf (" %x " ,inportb (0x3f8) ) ; / * 读取结果并显示 */
}
getch () ;
}
```

[0094] 两平台间采用在通信工业中应用最广泛的 RS-232 串行接口，并约定通信双方共同遵守的通信协议和数据包传输协议

[0095] 4.3 打开串口，获取串口句柄。

[0096] 通信程序从 CreateFlle 函数处指定串口设备及相关的操作属性。若打开串口成功则返回一个有效句柄，该句柄将被用于后续的通信操作，并贯穿整个通信过程，否则返回

INVALID_HANDLE_VALUE。程序中设计的 CreateFile 函数如下：

[0097]

```
hCom=CreateFile(TEXT("COM1"),
GENERI_READ|GENERIC_WRITE,
0,
NULL,
OPEN_EXISTING,
0,
NULL
);
```

[0098] createFile 函数中有几个值得注意的参数设置：第一个参数 IPzFileName 指向通讯端口后加一个冒号，如制定“COM1：”。因串口为不可共享设备，第三个参数串口共享方式应设为 NULL；第四个参数创建方式必须为 OPEN 一 EXISTING，即打开已有的串口。对于第五个参数的安全属性，其值为 0。

[0099] 4.4 串口设置。

[0100] 打开串口后需对串口配置好正确的波特率、字符长度等方可通讯。可以通过 1/0 设备控制（IPCTL）调用来配置串口驱动程序，还有种可采用的方法，即使用 GetCommstate 和 SetConnnstate 来配置串行端口。调用 GetCommstate 函数把串行口的初始配置填充到一个 DCB 结构中，修改串行口的配置，应该先修改 DCB 结构，然后 SetCommstate 函数用指定的 DCB 结构来设置串行口。除了在 DCB 中的设置外，程序一般还需要设置 1/0 缓冲区的大小和超时。用 I/O 缓冲区来暂存串行口输入和输出的数据，如果通信的速率较高，则应该设置较大的缓冲区。调用 SetupComm 函数可以设置串行口的输入和输出缓冲区的大小，从 DCB 结构可知，SetCommstate 能设置多种状态，很多属性可使用默认值，只需根据需要修改必须的区域。程序中只对 DCB 结构件，的 BaudRate 波特率、Bytesize 字符民度、parity 效验位、StopBits 停比位等参数进行配置，其中 Bytesize、parity、StopBits 定义了串行数据字节传输格式。本设计采用的波特率为！ 15200bPs，字符传输长度为 8 位，无效验位，梅字节一位的传输格式。

[0101] 4.5 串口的读写

[0102] 正如使用 CreateFilc 来扫‘开串行端口，可以使用 ReadFile 与 WriteFile 函数读写串亨了端口。当然在读写串「」端口的前提下，必须保证 CreateFile 已经成功地打开了串行端口，否则将读写失败。

[0103] 社一串口端口写数据仅需这样调用 WriteFile 函数：

[0104]

```
writeFile()
{
    int rc;
    DWORD cByte;
    BYTE ch;
    ch=TEXT(‘S’):
    Re=WriteFile(hCom,&ch, l, &cBytes.,NULL);
}
```

[0105] 其中，hcom 为已经打开的串口句柄，cBytcs 被设置为实际写入的字节数量，将字母“S”写入已经打开的串行端口。若成功，将返回 TRUE。

[0106] 读串行端口也很简单，调用过程如下：

[0107] int rc；

[0108] DWORD cByte；

[0109] BYTE ch；

[0110] ch ＝ TEXT(‘S’)：

[0111] re ＝ WriteFile(hCom,&ch,1,&cBytes.,NULL)；

[0112] ch 将读入一个字符，cByte 被设置为实际读取字竹的数量。若 ReadFIle 函数调用成功，也将返回 TRUE。

[0113] 4.6 关闭串口。

[0114] 调用 CloseHandle() 以关闭一个串行端口，CloscHandle 只有一个参数，就是调用 CreateFile。打开一个端口时返回的句柄。系统启动后，数据采集管理线程被创建，其线程处理函数就每隔 1s 循环调用通信模块，严格按照通信协议从下位机接收采集到的数据包。

[0115] 5 频谱分析模块

[0116] 系统中采用的信号处理方式有 FFT 变换、自相关分析、功率谱分析、倒频谱分析以及时域参数指标分析等。现在的信号分析仪都无法处理连续的信号，而要通过 A/D 采样将连续信号离散化，然后再对这组已知长度的时间序列进行分析处理，从而达到用系统完成信号分析的目的。现将系统中调用的几个信号分析算法函数罗列如下：

[0117] (1)FFT 变换 :voidffi(doublex[],doubley[],intn,intsign)，能对一个有 n 个成员的数组进行傅立叶变换或傅立叶逆变换，当 sign 的值设为 1 时，这个函数的计算结果是傅立叶变换，当 sign 值为一 1 时，计算结果是傅立叶逆变换。

[0118] (2) 相关分析和功率谱分析 :voidpmpse(doublex[], intn, intm, intnffi, int win, doublefs, doubler[], doublefreq[], doublesxx[], intsdb)，用 Weleh 平均周期图法来计算信号的功率谱和相关函数，其中输出参数 r 用来存放经过相关分析后的值，而 sxx 用

来存放经过功率谱分析后的值。

[0119] (3) 倒频谱分析 :noat*PCT_PC(intn,float*xr),对一个有 n 个成员的数组进行倒频谱分析,函数返回值就是倒频谱分析后的结果。上述是本系统中一些数据处理函数,在信号分析和处理中被调用,以对信号进行分析和变换,是波形分析模块中的几个主要分析函数。ATOM 端系统中的波形分析模块通过数据管理模块访问存储在数据库中的结晶器振动信号,调用各种信号处理方法,对这些信号进行分析处理,以期能完成对结晶器振动的检测和分析。

[0120] 6 频谱图显示模块

[0121] 仪器视图所显示的波形都是快速变化的,直接在设备上绘图会导致屏幕快速刷新,产生大量闪烁不适合观察,因此在构建视图类的时候均采用了位图缓冲的方法。就是在画图时候不直接对屏幕绘图,而是先在将所需要画的图形全部绘制到内存的位图中,然后再通过函数把整个位图拷贝到屏幕上来,这样每次更新视图的时候,屏幕只会刷新一次,从而解决了闪烁的问题。该软件绘图区主要有 CGraphStatic 类完成。CGraphStatic 类的基类是 CStatic,它提供了一个 Windows 静态控件的功能。一个静态控件可以用来显示一个文本字符串、方框、矩形、图标、光标、位图或增强的图元文件,也可被用作标签、方框或用来分隔其它的控件。CGraphStatic 类主要完成了以下操作 :

[0122] ◆规划绘图区的大小

[0123] ◆设置横轴和纵轴的属性

[0124] ◆对光标的绘制和各种操作

[0125] ◆将实际波形数据映射成像素数据

[0126] ◆绘制栅格及各项相关信息

[0127] 对每一个视图而言,虽然绘图的内容不同,但基本方法是一致的,均是通过外部调用函数 Invalidata() 发送 WM_PAINT 消息后调用当前视图类的 OnPaint() 函数驱动绘图。

[0128] 频谱图的显示是基于 CStatic 类的 DrawTrace(CDC*pDC, int i) 完成,最多可以同时显示三条不同的 Trace 曲线。频谱图的实现过程是 :由 CSocket 的派生类 CMySocket 类从底层硬件接收到实时数据,再由 CVI_SADlg::OnReceiveCommData 函数将接收到的有效数据进行 FFT 分析和谱估计,处理好后的数据再转存到 ResultData.DisplayResult[0], CGraphStatic 类进行绘制。

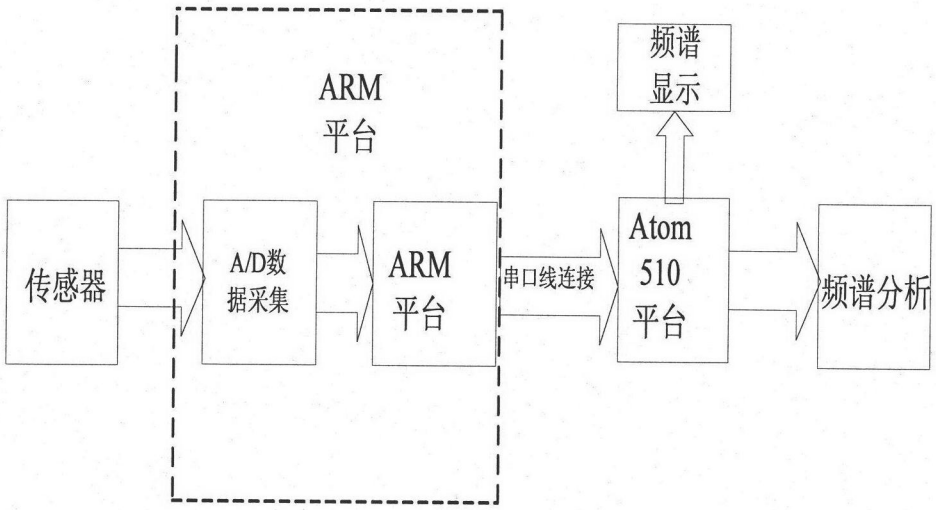

ARM
平台
频谱
显示
传感器
A/D数
据采集
ARM
平台
串口线连接
Atom
510
平台
频谱分析

图 1

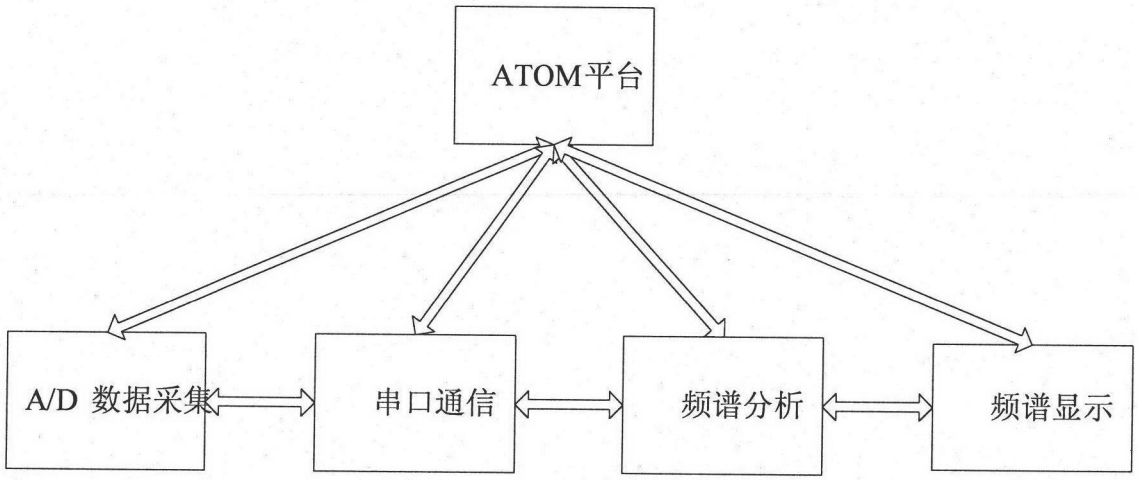

ATOM平台
A/D 数据采集
串口通信
频谱分析
频谱显示

图 2

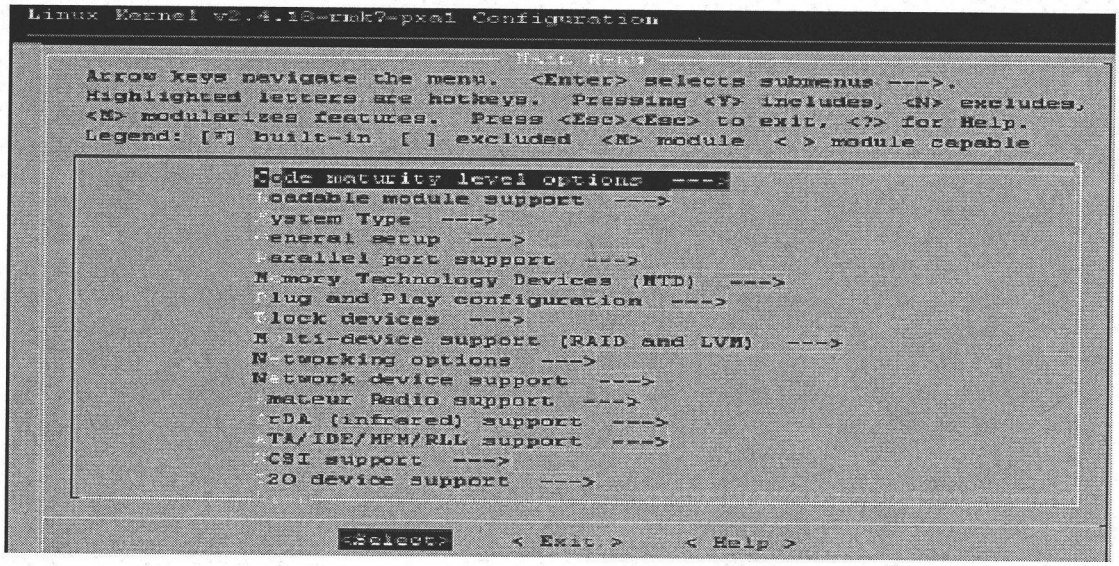

Linux Kernel v2.4.18-rmk7-pxa1 Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help.
Legend: [*] built-in [ ] excluded <M> module < > module capable
Code maturity level options --->
Loadable module support --->
System Type --->
General setup --->
Parallel port support --->
Memory Technology Devices (MTD) --->
Plug and Play configuration --->
Block devices --->
Multi-device support (RAID and LVM) --->
Networking options --->
Network device support --->
Amateur Radio support --->
IrDA (infrared) support --->
ATA/IDE/MFM/RLL support --->
SCSI support --->
I2O device support --->
<Select> < Exit > < Help >

图 3