2014年3月18日10:51:45

昨天发现一个非常严重的问题，主要研究了一下C8051F320的spi总线功能，出现了一个致命的错误。Spi总线所使用的IO口有4条或者3条，但不是任意的IO口。通过交叉开关IO口的说明，发现不同的特殊功能寄存器有不同的有限选择IO的功F能，优先权越高，能选择的IO口越少。SPI总线排在第二的位置，只能选择固定的IO口，这与板子的所设计的接口不一样。不知道这些外国人是怎么想的。我看了C8051F320的参考SPI程序，使用的是和手册上一样的IO口，但是这个usb收音机就不是啊，IO口接错，但是使用的是SPI的程序，真是坑爹。好像还可以用IO口模拟SPI总线来控制SI4702,这样感觉就大财小用了。

现在找到一个si470x的参考程序，它不是用自带的SPI功能，而是用普通IO口模拟3线控制芯片的，但是它的si470x的时钟信号是晶振直接提供的，我的板子是单片机提供的，这有有些问题，我还得先把时钟高清楚才行。

单片机内部具有一个稳压器，能将5v电压转换成3.3v，方便给其余3.3v外设供电，默认稳压寄存器REG0CN是将3.3v电压输出的，可以控制不使能稳压。如果对这个寄存器什么都不做，就默认使能稳压功能。芯片的Regin引脚要和usb的vbus线相连，这样就能将usb的5v电压转换成3.3v的电压了。

由OSCXCN寄存器选择外部振荡器，配置为0x61,选择晶体做为外部振荡器，频率选择32khz到84khz，实际上就是32.768kHz的晶振。

内部具有12Mhz的振荡器。使用4倍时钟乘法器将内部的12MHZ转换成usb的48mhz频率。

配置过程如下：

1. 向CLKMUL写入0x00来复位时钟乘法器 CLKMUL=0X00
2. 用MULSEL(CLKMUL^7)位选择时钟乘法器的输入 CLKMUL|=0X80
3. 延时大于5us
4. 用MULINIT位初始化时钟乘法器（CLKMUL|=0XC0）
5. 查询等待MULRDY=>’1’ while(!(CLKMUL&0X020));

用CLKSEL来选择时钟，CLKSEL=0X02;将usb时钟选择为4倍时钟乘法器48mhz，系统时钟为乘法器的一半24MHz

比较器：

比较器的输入应该被配置为模拟输入，P1^0和P1^1为比较器输入，应该P1MDIN = 0xFC;

现在不知道拿比较器来干什么

怎么给si4702提供时钟的？怎么在p1^7输出时钟

EIE1是扩展中断允许寄存器，包括usb中断、比较器中断、PCA中断、和定时器3中断

2014年3月19日14:48:39

总算明白优先权交叉开关是怎么回事了。当选择一种数字资源时（如spi,uart等），还没有分配的引脚中最低位分配给该资源。比如CP0A可以分配的引脚有所有的P0口，p1.0和p1.1,当这些引脚都没有分配给别的资源时，就将最低的引脚分配给CP0A,也就是讲P0.0分配给该引脚。如果p0.0已经分配给了别的资源，那么就要用跳过寄存器跳过这个引脚，重新分配别的引脚给该资源。

要使能CEX0(PWM的输出)到P0.0引脚，要设置XBR1=0X41，选择使能交叉开关，pca只有cex0链接到端口。

刚刚编写了PCA输出pwm波的程序，验证了交叉开关的功能和之前想的一致。本来IO口是没有分配给任何资源的，现在要分配给PCA的CEX0，那么就会将最低的IO（p0.0）分配给它，验证后确实是p0.0输出了pwm，能将led实现呼吸灯的功能。然后我将p0.0跳过，用跳过寄存器P0SKIP=0X01就将p0.0跳过了，pca就选择了下一个IO口即P0.1，连上后果然是p0.1输出了pwm，但是要注意用P0MDOUT寄存器将p0.1设置为推完输出。

USB收音机中定时器0用在PCA是时钟源，定时器2用在ADC转换器

PCA计数器寄存器是PCA0L和PCA0H

PCA0CPMn是捕捉比较寄存器

PCA0CPLn和PCA0CPHn是捕捉模块寄存器所要装入的数值人，如占空比

向这两个寄存器写入数值时应该先写入低字节

PCA计数器和比较器发生匹配时将产生一次PCA中断

2014年3月25日22:34:30

总算明白交叉开关是怎么回事了，设置了跳过引脚并不是说这个引脚就不用了。如果某个资源要使用这个引脚，但是把这个引脚设置了跳过，那么就将这个下一个引脚分配给该资源。之前看手册上说的spi功能的引脚分配图就只有固定的分配给该资源，以为spi就只能使用这几个引脚，跳过就不知道怎么办了，因为别的资源可以使用多个引脚，但spi在图上就只是固定的引脚是紫色的。其实这样是错的，事实上被跳过的引脚就相当于有东西插到缝里了，别的东西自然就向一边移动了。交叉开关也是一样，把p0全部跳过，那么spi就使用p1口，并不是就没办法了。

要是我有一台示波器就好了

SPI工作在三线方式，NSS不被交叉开关映射，本来p1.7是分配给nss的但是是用的 三线模式，因此p1.7就不分配给nss了。

（作为比较器的输入引脚要被跳过，因此CP0A就不能用交叉开关了，直接用CPT0MX来分配引脚。本来在交叉开关里面CP0A的优先级要比pca的要高，但是CP0A要被跳过，因此p1.7就要分配给pca的CEX0，这个8为的pwm输出的pca在p1.7产生时钟给si4702.）错了

CP0A是比较器的输出引脚，p1.1和p1.0才是输入引脚应该被跳过。因此CP0A还是分配给p1.7的。可是这又与图产生了冲突，因为图上是rclk连到p1.7的。

高手啊，有点明白了，它原来是想用这个比较器来产生32.768的时钟啊。比较器的两个输入p1.0和p1.1连在了外部晶振的两端，比较器的输出RCLK是连到P1.7的。这样收音机芯片的时钟就与pca输出的pwm无关了。

这下所有IO都与电路图对应了

定时器0的8位自动重装值关系到定时器的定时时间。重装值写入TH0中，假设定时频率是f,系统时钟是SYSCLK,预分频为48。在CKCON寄存器中设置分频为系统时钟的1/48.那么重装值TH0=SYSCLK/48/f。1/f就为定时周期，会按照这个时间进入中断函数，如果将这个定时器时钟分配给PCA，那么f就为PCA的时钟。

现在又一个问题，定时器0使用系统时钟的1/48。usb使用4倍内部时钟，pca使用定时器0溢出，也是间接使用内部时钟。只有比较器使用外部的32.768晶振，到底有什么用呢，为什么不直接将这个晶振放到收音机芯片，整这么麻烦干嘛，

2014年3月27日00:29:54

si4702的时钟是有比较器提供的，这下知道比较器是干什么用的了，但是pca又是干什么的就不太明白了，程序里说的是用来初始化usb，使能收发和中断。只有明天研究了。

“联合”是一种特殊的类，也是一种构造类型的数据结构。在一个“联合”内可以定义多种不同的数据类型， 一个被说明为该“联合”类型的变量中，允许装入该“联合”所定义的任何一种数据，这些数据共享同一段内存，已达到节省空间的目的（还有一个节省空间的类型：位域）。 这是一个非常特殊的地方，也是联合的特征。另外，同struct一样，联合默认访问权限也是公有的，并且，也具有成员函数。

sfr16用于按16位存取的特殊功能寄存器

在工程中加入C51S.LIB文件后警告就少了2个，具体也不知道为什么

原来程序里的.asm汇编程序是由.C的汇编c混合程序自动生成的。在工程里面右击F320\_TimeCritical.c选择第一项option，然后右边勾选Assemble SRC File

2014年3月29日15:04:09

R0-R7在[数据存储](http://zhidao.baidu.com/search?word=%E6%95%B0%E6%8D%AE%E5%AD%98%E5%82%A8&fr=qb_search_exp&ie=utf8&eid_gfrom=151)器里的[实际地址](http://zhidao.baidu.com/search?word=%E5%AE%9E%E9%99%85%E5%9C%B0%E5%9D%80&fr=qb_search_exp&ie=utf8&eid_gfrom=151)是由[特殊功能寄存器](http://zhidao.baidu.com/search?word=%E7%89%B9%E6%AE%8A%E5%8A%9F%E8%83%BD%E5%AF%84%E5%AD%98%E5%99%A8&fr=qb_search_exp&ie=utf8&eid_gfrom=151)PSW里的RS1、RS0位决定的。  
  
using 0时设置 RS1=0，RS0 =0，用第0组寄存器，R0--R7的在[数据存储](http://zhidao.baidu.com/search?word=%E6%95%B0%E6%8D%AE%E5%AD%98%E5%82%A8&fr=qb_search_exp&ie=utf8&eid_gfrom=151)区里的[实际地址](http://zhidao.baidu.com/search?word=%E5%AE%9E%E9%99%85%E5%9C%B0%E5%9D%80&fr=qb_search_exp&ie=utf8&eid_gfrom=151)是00H-07H。R0（00H）....R7（07H）  
  
using 1时设置 RS1=0，RS0 =1，用第1组寄存器，R0--R7的在[数据存储](http://zhidao.baidu.com/search?word=%E6%95%B0%E6%8D%AE%E5%AD%98%E5%82%A8&fr=qb_search_exp&ie=utf8&eid_gfrom=151)区里的[实际地址](http://zhidao.baidu.com/search?word=%E5%AE%9E%E9%99%85%E5%9C%B0%E5%9D%80&fr=qb_search_exp&ie=utf8&eid_gfrom=151)是08H-0FH。R0（08H）....R7（0FH）  
  
using 2时设置 RS1=1，RS0 =0，用第2组寄存器，R0--R7的在数据存储区里的实际地址是10H-17H。R0（10H）....R7（17H）  
  
using 3时设置 RS1=1，RS0 =1，用第3组寄存器，R0--R7的在数据存储区里的实际地址是18H-1FH。R0（18H）....R7（1FH）

使用8位的MOVX指令时，EMI0CN为访问地址的高8位，由 R0或R1给出低8位。

MOV EMI0CN, #12

MOV R0， #34

MOVX A, @R0 将地址0x1234的内容装入累加器A

程序里使用EMI0CN和R1来访问xdata

一次采样数据为4个字节，左右通道各两个。通过定时器2中断函数来选择左右通道，每进入一次中断函数就换一个通道。本程序选择的是采样结果左对齐的方式，因为采样结果有效位数是10，因此要用两个字节来保存结果，采用左对齐，剩余位就不用了。结果存储在外部数据存储器里面，即xram。使用EMI0CN和R1来作为XRAM的地址。在XRAM里每256位即一个字节为一个页，现将采样结果低字节ADC0L存入第0页，然后EMI0CN自增，选择第2页，将采样高字节ADC0H存入其中。

采样率为96khz

2014年3月29日23:50:34

正数的原码反码补码都是一样的。负数的补码就是反码加1。负数的反码就是除符号位即最高位为1外，其余位取反。

对于usb的专用寄存器不能直接赋值，而有专门的函数POLL\_WRITE\_BYTE()来实现，这个函数是根据手册上要求来写的。读寄存器的函数也一样。

对控制端点状态寄存器的访问是通过先写usb寄存器INDEX来实现的，由INDEX来选择要控制的端点。

2014年3月30日23:02:45

现在要开始研究usb方面的了。我使用bushound软件捕获设备与电脑之间传输的数据。

首先电脑发送获取设备描述符的命令，设备描述符有18字节。

然后获取配置描述符，但电脑没有要求全部的配置描述符，只要求传输前9字节就行了。

接下来就是从设备获取完整的145字节的配置描述符。

出现了一个GET\_DESCRIPTOR 的请求，80 06 02 03 09 04 04 00

不知道这个请求是干什么用的。

标准描述符类型有5个，其余是类特殊描述符

标准描述符：设备描述符（1），配置描述符（2），字符串描述符（3），借口描述符（4），端点描述符（5）。

类特殊描述符有：HID(0x21), REPORT(0x22) ,PHYSICAL(0x23)

wValue 是2字节的。为描述符的类型和索引。低字节为索引，高字节为类型。在usb中这种两个这几的变量是低字节在前，高字节在后。比如wValue为0x00 0x01,则可判断为设备描述符 。为0x02 0x03,其中0x03为类型，是字符串描述符，0x02为索引，为产品字符串。

字符串描述符所包含的描述符：语言ID(0)，厂商字符串(1),产品描述符(2),产品序列号(3).

根据前面的推理，之前的疑问应为字符串描述符中的产品描述符。可是捕获到的结果为12 03 46 00 ，这是不全的。因为接下来电脑又发送请求要获取这个描述符，但不再是4字节了，而是完整的18字节。应该就是“FM RADIO”这个字符串描述符了。

在各种描述符配置好后，电脑就不停地发送GET\_STATUS命令。这个命令从未用过，不知道什么意思。

在进行FLASH操作之前必须向FLASH锁定和关键寄存器FLKEY写入正确的关键码。就是按顺序向FLKEY写入0XA5，0XF1。

SI4702选择3线模式，SEN,GPIO3，SDIO引脚必须在RST的上升沿为低电平。

我发现usb中断处理函数里面没有对端点1和端点2处理的函数，端点1和2都是hid端点。端点是是音频流端点。

刚刚研究了一下C8051F320的uart功能，把参考程序烧进去，连上我的pl2302的usb转串口模块，再打开串口软件，发现跟普通51单片机一样能使用串口功能啊。这下我就能模仿圈圈的开发技巧了，就是讲usb调试过程通过串口发送到电脑，这样就能更清楚地知道usb工作的过程了，极大的有利于开发。但有一点遗憾的是我做板子上没有留串口的接口，我只能在开发板先调试了。

2014年4月3日13:52:18

无语了，刚刚编译程序，提示不能生成hex文件，也不知道出了什么问题，搞不懂，最近我一直都是在看程序，就没有改过，最多就是添加中文注释。现在不知道除了什么问题。我重新建了个工程，把程序考过去，重新编译还是不行。幸好我有备份。我得从原来的程序上修改了。

发现把P0MDIN设置了就不能使用UART了，这是怎么回事？

原来 P0MDIN为配置P0口是否为模拟输入，为0的话就将对应口设置为模拟输入，这时其弱上拉、数字驱动器和数字接收器都被禁止。之前将p0.4和p0.5都设置为0了，uart就不能用了，要将其设置为1才行。

我的这个uart是使用stdio功能的，也就是添加头文件stdio.h，然后将uart初始化后就能直接使用stdio.h中定义的printf函数来发送数据到串口了。

2014年4月3日17:15:50

搞了一下午串口终于成功移植到usb收音机工程里面了。

通过串口发现没有运行设置地址的函数。后来发现又有了。

电脑通过端点0中断发送设置地址命令，设备接收到这个命令后，进入Handle\_setup()函数，根据接受的数据分析进入设置地址函数Set\_Address()，但是没有直接给地址寄存器写值，而是设置Ep\_Status[0]为EP\_ADDRESS,然后会再次进入端点0输入中断函数，进入Handle\_Setup()函数，这个函数首先判断端点0是否为EP\_ADDRESS,如果是，则执行真正的设置地址功能，调用POLL\_WRITE\_BYTE函数写一个字节，即给FADDR设置为电脑发送命令中wValue中的值。

2014年4月4日11:52:41

今天测试的时候，结果电脑不能正常识别设备了，我就只添加了几个printf函数就出现了这样的问题，我把printf函数注释掉后还是不行。通过串口发现总线挂起了，这到底是怎么回事。

2014年4月4日13:27:42

靠，现在串口模块又罢工了。这个东西总是不稳定，记得去年在单片机实践的时候做串口通信，就是有时能用有时不能用。非常恼火。

2014年4月4日13:50:35

先串口被电脑识别了，但是串口助手软件里看不到结果。

通过串口发现，在枚举的时候，收到了set\_interface，set\_configuration这两个现在还不知道原理的标准请求，

还有两个类请求：get\_cur和set\_Idle这两个也不知道是什么意思

电脑发送set\_cofiguration请求，在wValue中为配置的标识，这个值与配置描述符的bConfiguration匹配就选择这个配置，因为可以有多个配置。注意这个值要是为0是没有配置的，仍处于地址状态，应该不为0.

竟然对端点1和2的配置是在set\_configuration函数里配置的。当收到Set\_Configuration命令后就会对端点1,2,3进行配置。端点1为不分割，输入端点，端点2为是使能双缓冲，中断传输。端点3为同步输出端点。

通过这个Set\_Configuration函数明白了Ep\_Status[4]这数组的含义了，数组中的四个字节分别表示端点0,1,2,3的状态，在配置好后为EP\_IDLE状态，即为空闲状态，若没有配置好，则为EP\_HALT，即为停止状态。

同时还有USB\_State来表示usb的状态，配置好后为DEV\_CONFIGURATION,没有配置好为，DEV\_ADDRESS状态。

当 一个请求结束后，要让设备知道这次的端点0数据已经处理完了，应该写：

if (Ep\_Status[0] != EP\_STALL) POLL\_WRITE\_BYTE (E0CSR, rbSOPRDY);

类特殊音频请求是通过默认端点即端点0来通信的。音频控制接口使用的是端点0。

在audio1.0协议里终于找到了音频类特殊请求了：

SET\_CUR 0X01

GET\_CUR 0X81

SET\_MIN 0X02

GET\_MIN 0X82

SET\_MAX 0X03

GET\_MAX 0X83

SET\_RES 0X04

GET\_RES 0X84

SET\_MEM 0X05

GET\_MEM 0X85

GET\_STAT 0XFF

2014年4月11日16:17:13

现在在研究音频协议，非常复杂，各种请求，各种描述符。在配置描述符中的input terminal 描述符中将bTerminalID和wTerminalType改成了audio2.0协议上指定的值。否则枚举会出错。在feature unit 描述符中将左右通道音量控制从原来的0x0000改成0x0200,即从0改为2，这样在枚举时就会有音量请求。之前就奇怪为什么只有静音请求而没有音量的请求。在写请求都是音频类特殊请求，在get\_cur,get\_max,get\_min等中实现。

现在将这些请求的代码按照get\_cur的模式写进去了，这下枚举也成功了。将音量的最小值最大值和分辨率都成功的返回到电脑了。

音量的请求都有两次，因为左右声道各有一次。

2014年4月12日15:15:04

现在通过呀呀usb\_hid软件监测到设备不断地向电脑发送数据，报告ID为12,后面的内容全部为0xff.不 知道出了什么问题，然后我测试原版的程序发现还是同样的问题。

2014年4月12日19:34:28

发现这个spi读写好像没有用，读出来的值始终是ff。难道是交叉开关那里出错让spi引脚出错了。

2014年4月13日00:04:04

我的程序得换成不用自带的spi，改用模拟3线模式。因为我看到程序里，根本就没有对设备进行控制的函数，如调频率，条音量，静音等。而另一个参考程序就有。还有hid部分是在是搞不懂他是怎么写的。他的好像对设备的控制都是通过断点0来实现的，可是上位机却找不到相应的函数。报告里面分了好几十个不同的报告，通过不同的报告ID来实现，非常复杂，通过set\_report和get\_report还发送和接受，可是有找不到怎样调用这两个函数。

我的 程序还有很多地方要修改啊。

2014年4月13日21:38:28

现在我的想法是将原来程序里的控制si4702的程序全部换掉，获取和发送报告的程序也不要了，直接通过端点1,2来控制。改了好久啊。要对交叉开关进行修改，要禁止SPI功能，并把原来控制spi的引脚跳过。输入要设置成数字io，输出SDIO要为开漏，其余为推挽。

2014年4月14日10:20:52

刚刚发现，读数据的时候SDIO要设置为开漏，写数据的时候SDIO要为上拉。也就是开漏为输入，上拉为输出。（相对于单片机而言）

有一个问题，之前也发生过，但没有注意。就是在一个函数里调用pritnf函数时，函数最开头有变量定义，printf函数不能写在变量前，否则会出错。写在变量后面就没有问题。

2014年4月14日14:51:14

发现单片机对si4702的读写不成功，想撤回原来的，不要对芯片的控制程序了。发现设备已经不能正常枚举为usb设备了。又不知道哪里出了错。又要重新来了。

2014年4月14日15:14:00

刚刚在下载软件里勾选了当断开设备时给设备上电。从字面上来看，这个是很正常的啊。可是发现串口又不能发送数据了。我以为是串口出问题了，又重新插入。又短接rxd和txd,发现串口没有坏，难道是单片机坏了。搞了好就都没有搞出名堂来。最后才发现勾选了这个东西。取消勾选发现就正常了。真是搞不懂啊。

2014年4月14日15:49:31

发现一个大秘密啊。在做for的递减循环的时候，中间变量i最好不要用

unsigned char类型。比如for( i = (numbits-1); i >= 0; i-- )，本来原程序是用的char型的，我把它改成unsigned char型的了，这样这就变成了一个死循环。i到0后又变成ff了，永无止境。正确的方法是使用char型变量。之前就一直在这个函数了卡住了。不知道是怎么回事，还一直以为是延时函数的问题。当我重新写这个循环的 时候，在for的主函数里就用printf函数输出i，才发现了问题。不然还一直没法解决。

出现这个问题的根本原因是unsigned 型的变量都是大于0的。

2014年4月14日16:14:11

太高兴了。总算有进步了。刚刚读取了si4702芯片所有寄存器的值，通过串口观察，第一个寄存器的值为0x1242,对照数据手册发现该寄存器的默认值就是0x1242.。这不仅证明了芯片是好的，而且还焊接在正确，并且对芯片的读写都是正确的。啊，太高兴了。

还说明了一个问题，就是我不用换方案了。之前想的是要是还不行要么是芯片坏了，要么是焊接问题。Qfn的芯片确实难焊接。只有换成tea5767芯片了，或者买一个si4702模块。这下都不用了。

2014年4月16日15:49:13

妈的，我要疯了，想换成tea5767芯片，发现程序不能移植，说是超出变量定义空间，先把这个搁置。研究hid，把圈圈的HID移植过去，发现又不听地有clear feature请求，这个请求又不知道是什么意思。真是难搞啊。

换成光盘里的hiduart例子的报告描述符就正常了，圈圈的就不正常。真是恼火。

2014年4月16日19:06:12

真是有很大的运气在里面啊，现在又将圈圈的hid报告描述符移植到我的 设备里面了，这下就正常了，没有奇怪的请求了。

离成功又近一步了，我用之前圈圈开发板上做的上位机软件来调试，往我的设备发送数据，结果发现串口调试返回了端点2中断信息。这说明通过端点2来传输hid数据是成功的。但是还没有在端点2输出中断函数里返回应答信号，因为上位机还不知道数据是否成功接收到了。

2014年4月18日12:57:14

现在又遇到新问题了，将auotoseek函数移植到我的工程里面，编译无错，一旦链接就有错，不能生产hex文件。错误提示使用了错误的外部符号，指的是NEX\_LED\_MODE,要是取消掉autoseek这个函数就没有问题了，真是搞不明白。还有，编译结果显示data区已经有260多个字节了，是不是超了呢。

应该是inserpreset函数出了问题。

2014年4月18日17:32:50

进过刚刚是试验，果然是data区超了，不能大于255字节，data是内部RAM，只有255字节。存的是局部变量，中间变量。就是在函数里定义的变量多了才导致的，而这与函数多少无关，程序是存储在外部存储器中的，也就是code区。刚刚我把autoseek函数里调用的其他函数都注释掉了，变量没哟动，发现编译后data还是超过255，然后再将里面的变量给注释掉，现在只有235了，也成功生成了hex文件。现在是明白了问题所在了。可是链接结果老师提示我NEXT\_LED\_MODE外部符号错误。让我一直都找不到解决办法。

Code存储类型，编译器会将其存放在代码空间flash中，信息在执行过程中是不能自我改变的，不会有信息写入到这个区域。在这段空间里存放指令代码和其他非易变信息。

Xdata是外部数据存储器，存放不经常改变的数据，读取速度较data慢，但空间比data大。

原来参考程序写得非常复杂，首先是test\_auotseek函数，里面主要有个autoseek函数，里面有个循环，使设备不但地搜台。Autoseek里面有si470x\_seek函数，这个才是真正的搜索函数，auotseek里面还有insertpreset函数，用来判断rssi，并插入频道到一个有15个usigned int型的数组里面。在auotoseek里面还有一个15个值的数组，用来存放rsii，搞得非常复杂。导致变量过多，超过了单片机的内部数据存储器。现在我只要si470x\_seek函数，收到一个台就通过hid传给电脑。不用保存到单片机的存储器里。

2014年4月18日21:32:52

有发现一个很奇怪的问题。今天下午在我去掉auotoseek等函数的时候编译出data只有230多，到晚上就240多了，非常仔细地检查了整个程序，没有多余的变量定义，怎么会就多占用了几个字节的data呢，现在data是非常宝贵的，不敢随便乱用。在卡住一段时间后，当我再main含数的c文件里编译却发现编译出只有233字节的data了。真是奇怪。之前还有一种问题就是在不同的c文件里链接，结果警告的数量都是不一样的。

现在基本上已经将HID功能写好了，通过上位机向设备发送8字节的数据，通过端点2输出中断函数来接收数据，并通过端点1发送一个数据包，发送后会进入端点1输入中断，这与之前的想法都是一样的。只要你向端点写入数据就会产生输入中断。若电脑向设备发送数据就会产生输出中断，这时在中断函数里判断是哪种中断，进入相应的数据处理函数。

2014-4-20 12:13:46

奇怪啊，在定时器2中断函数里加入printf函数后链接发现data就超过250了，就不能生成hex文件了。将printf函数注释掉后就只有230多了，奇怪啊。这个函数为什么会出现这样的问题呢，又没有定义变量。

我靠，我在main函数里将一些初始化函数注释掉后更奇怪的事情发生了。Data变成了260多了。这是在逗我吗。

现在的问题是，执行了turn\_on\_stream函数后并没有发现执行了端点3输入中断函数。

2014年4月21日18:46:24

现在开始研究电脑对音频信号的采集播放等问题了。说实话，一点信心都没有，差点都要放弃了，还在网上搜索要是毕业设计做不出来会不会毕不了业。因为又遇到问题了。其实我实在是太悲观了，没有做到不卑不亢。从一开始我就遇到了很多棘手的问题，到现在不都一一解决了吗，很少有问题让我持续很多天都解决不了。从前天就开始上位机编写了，发现音频这块是完全不会，参考程序也看不懂，弄得我快崩溃了。发现参考程序里面有个waveInOpen的函数，在翻遍了几乎所有的c++编程的书后发现在《visual c++开发实战1200例子》的第550个实例里有一个现实输入音频信号频谱图的程序，它能采集音频信号，将其福利叶变换，然后画在对话框里面。其中对音频信号的采集就采用了waveInOpen函数。可是这本pdf的书里面就只有很少的一点点代码，然后只好在网上搜索这本书的光盘资料。最终在CSDN找到了程序。放在桌面却打不开，双击dsw文件却提示找不到文件路径。网上搜索各种解决方法无果。只好自己建一个工程，将里面的程序用记事本打开，一点一点的复制。可是程序太复杂了，而且我已经好久没有搞MFC程序了，要复制一个工程太困难了。然后继续在网上搜索看在别的地方又没哟程序。结果还是没有找到。后来将程序从桌面放到了G盘，这次在打开DSW文件就能打开了，真是奇怪。太不容易了。结下来就成功看到了程序的实现过程。

首先要引用库文件。

#include <MMSystem.h>

#pragma comment(lib, "winmm.lib")

这个可以再设置、链接里面添加。也可以在工程里面添加。这两句程序有的是添加在stdafx.h里面，有的是添加在xxxdlg.h文件里面，估计都差不多。

参考程序是建立了一个没有父类的类，里面有对设备的各种处理。但是圈圈的程序里面对设备的处理都是在xxxdlg类里实现的。两者各有自己的好处和缺点。前者方便管理，程序清晰明了。后者最大的优势就是能输出调试结果到文本框里面。圈圈的程序里面有个文本框，可以输出调试信息，就像在单片机里面利用串口输出调试信息一样，可以再任何地方插入输出字符串到文本框里。如果单独在cpp文件里建类，这个类没有父类，就无法输出字符串到文本框里。也许能实现，但是我不知道怎么实现。也许是什么窗口指针的方法吧。

接下来就是定义各种变量了，包括定义句柄，数据类型的结构体，缓冲区和缓冲区结构体等。

要补充一点：

   WINDOWS下音频的采集，播放有三种模式：

        1）通过高级音频函数、媒体控制接口MCI[1、2]设备驱动程序；

        2）低级音频函数MIDI Mapper、低级音频设备驱动（WaveXAPI）；

        3）利用DirectX中的DirectSound；

我所所要使用的是第二种。参考程序就是使用的第二种。

音频缓冲使用malloc函数来分配，释放使用free函数。

在打开设备之前还要用waveInGetNumDevs函数查看输入音频设备的数量。里面还要判断设备的ID。发现WAVEINCAPSA结构体里面有MID和PID，这个是否就是和HID一样的VID和PID呢。因为整个复合设备就只有一套设备ID.

**WAVEINCAPS waveInputCapabilities; 定义输入设备属性的ID，结构体里包含了id，音频格式，通道数量等数据。**

**调用waveInGetDevCaps()函数得到设备的性能，第一个参数是设备ID，即编号，第二个就是waveInputCapabilities，通过这个函数就能得到ID值，存到waveInputCapabilities里面。**

**2014年4月21日23:18:342**

**今晚效率不高啊。帮谢兵改完报告后发现程序崩溃了，闪退，之前又白做了。当刚刚重新再原来的程序上做了个测试，检查有多少个音频设备。我在对话框里添加了一个用于测试的按钮。在按钮的单击函数里添加相关函数。**

**DWORD numWaveInputDevices = waveInGetNumDevs();**

**//Scan through each input device to see if we can find the FM Radio**

**for (DWORD i = 0; i < numWaveInputDevices; i++)**

**{**

**WAVEINCAPS waveInputCapabilities;**

**//Get the device capabilities of the currently indexed device**

**if (waveInGetDevCaps(i, &waveInputCapabilities, sizeof(waveInputCapabilities)) == MMSYSERR\_NOERROR)**

**{**

**AddToInfOut("faxian");**

**}**

**}**

**当然在stdafx.h函数里要添加相应的库。**

**这个函数先获取有多少个银品牌设备，然后获取每个设备的属性，如果成功获取，就在对话框里输出字符串。测试结果正常，在我没有插入我的设备的时候，没点一次按钮只能出现一次字符串，插入后每次会出现两次，说明这个过程是成功的 。**

**原来获取音频设备的编号的函数是int CFMRadioDevice::GetAudioDeviceIndex()，在这个函数里有前面将的各种获取设备数量，性能等函数，还要对设备的字符串名字进行判断。**

**这个要明天好好研究。**

**通过前面获取到的编号就要带入waveInOpen函数的第二参数里，就能打开设备了。看来还不太难。**

**2014年4月22日09:16:24**

**加上音频相关数据定义后程序就不能执行，编译链接都正常。**

**发现我添加了一个功能，同一个程序一次只能打开一个。一般情况下，同一个exe文件可以多次打开，生成多个窗口。我就设置了如果这个程序还没有关闭，又打开它，会提示程序已经打开，并且关闭刚刚打开的程序。现在的情况是，我已经关闭了程序，再次打开的时候就提示程序已经打开。难道是关闭的不够彻底。现在只有将这个功能删掉了。**

**发现利用字符串来确定设备的方法不太好。参考程序利用了strcmpy函数来比较两个字符串，实际上是逐个比较ASCLL码，相同就返回0，前一个小于后一个就返回负数，大于就返回正数。这样有的字符串比参考的大就会被错误地选择。我觉得还是比较ID比较合适。**

**可是刚刚测试发现读取到的ID全是ffff，这是怎么回事呢。难道USB音频设备的wMid和wPid都是FFFF.我把买的usb声卡插入，使用软件只**

**是多读出了一组ffff。如果真是我这样想的话，用ID来判断还真不行。**

**现在我用AddToInfOut函数将每一个设备的字符串输出到文本框里，发现了FM RADIO.看来还是成功的。**

**调试结果:**

**发现设备 (11:40:37)**

**FFFF**

**FFFF**

**麦克风 (4- FM Radio)**

**发现设备 (11:40:37)**

**0001**

**0065**

**麦克风 (Realtek High Definition**

**发现设备 (11:40:37)**

**FFFF**

**FFFF**

**麦克风 (2- Generic USB Audio De**

**结果表明，usb音频设备的ID全是ffff，不能用来区分不同的音频设备，还是得用字符串来实现。**

**发现usb音频设备的字符串会因你插入的接口不同而不同，之前是麦克风 (4- FM Radio)**

**，在我换了一个接口后变成了“麦克风 (5- FM Radio)”。这样还不好判断啊。**

**研究发现strnset函数能将一个字符串用指定的字符串替代指定的前多少个。这样我就可以用一个字符串将因不同接口造成的字符串里面的数字不同的数字前面都替代掉，这样就管你插入的是哪个接口，转换后的字符串都相同。**

**还有汉字是两个字节，strnset参数为2，只能替代一个汉字。**

**AddToInfOut(waveInputCapabilities.szPname,true,true); //输出转换前的字符串**

**strnset(waveInputCapabilities.szPname,'x',9); //将字符串的前9个字节用x替代，这样讲不会因为你usb接口的不同而字符串不同**

**AddToInfOut(waveInputCapabilities.szPname,true,true); //转换后的字符串**

**if (strcmp(waveInputCapabilities.szPname,"xxxxxxxxx- FM Radio)")==0 )**

**{**

**//Set the current index to i, and set i to numWaveInputDevices to break**

**//out of the for loop**

**//index = i;**

**//i = numWaveInputDevices;**

**AddToInfOut("找到设备");**

**}**

**}**

**这一步实现了。换了两个接口都成功找到设备了。**

**关于刚刚的问题是操作系统的问题，不同的操作系统会使字符串不同，按照我的这个程序，要想成功运行就只能在win7上面了。**

**声音输出的声卡的选择让系统自动选择，采用回调函数的方法。**

**InitializeStream**

**使用临界区对象实现线程同步：**

首先要定义一个临界区对象，CRITICAL\_SECTION gWaveCriticalSection;

**使用EnterCriticalSection和LeaveCriticalSection对数据进行保护 ，如果两个线程对同一个数据进行处理。就要将数据写入这两个函数之间，当一个线程正在处理数据时，另一个线程开始运行，当第一个线程没有执行LeaveCriticalSection时，第二个线程是不能访问数据的。**

**将共享资源加入者两个函数，任何时候只能有一个线程能访问该资源。**

**m\_FreeBlock++;**

**m\_FreeBlock %= BLOCK\_COUNT; //m\_FreeBlock对20取余数并赋值给m\_FreeBlock，保证了从0到19这20给数据块循环操作**

**参考程序在音频数据的处理中，不管是音频输入还是输出都是利用输出缓冲数据结构。**

**Waveinstart（）和waveoutwrite（）是对应的，前者开始录音，后者播放。**

**参考资料上这样说：**

**在录音和放音时，分配内存缓冲区的同时相应分配WAVEHDR数据块结构，然后将缓冲区的指针赋给数据块结构的成员变量lpData,这样当一个缓冲区填满后，也就是一个音频数据块填满了，通过消息机制就可以在消息函数中进行处理和播放，播放完后又可通过消息函数把缓冲区再送给音频输入驱动程序，继续进行采集和播放。**

**录音开始后，每当有采集数据填满数据块后，设备驱动程序就会发送消息MM\_WIN\_DATA(和WIM\_DATA一样)给用户窗口，相应的消息回调函数OnMmWimData()对声音数据进行处理，包括播放、存储或者网络发送，每当一个音频数据块播放完毕，设备驱动程序又会发出消息MM\_WOM\_DONE,相应的消息回调函数记录音频数据并经必要准备后重新发送给输入设备，以准备接收后续的采样数据。这样，最初为输入设备准备的音频数据块就在消息的控制下，在输入、输出设备间循环使用，无需人为控制实现了实时采集。处理和播放。**

**想起一个问题，关于static的**

**static BOOL bChange = TRUE;在一个函数里定义，每次执行这个函数的时候并不会给bChange赋真值。记得单片机里就有这么一个类似的程序。定时器中断里，给static 类型的i赋0值。然后i自增。每进入一次中断i++，并不是i就变成了0。这个道理还不明白。**

**在vc++里又遇到了。**

**现在不明白的是为什么要使用20块缓冲区。如果这20块缓冲区都使用了的话，那么就会调用20次streamaudio函数，可是实际上没有啊。还有一个问题是，音频输入没有采用回调机制，也就是不适用回调函数，那么怎么实现音频的自动采集呢。也没有看到线程啊。**

**2014年4月24日18:16:09**

**今天都没有怎么编程，变得越来越懒了，越来越看不起自己了。**

**现在编写的程序，要是没有插入设备，程序不能正常工作，并且关闭程序时，程序会崩溃，提示微软基础类停止工作，要是插入设备就正常了。**

**网上 资料：**

waveOutOpen函数最后三个参数的设置对消息处理方式起决定性作用，需要特别注意，通常我们用下列处理方法：  
 1. 使用窗口作为消息的接收者，则第四个参数设置为该窗口的句柄，则和这次播放有关的消息都将进入该窗口的消息队列，这时第五个参数为NULL，第六个参数为CALLBACK\_WINDOW，表明由窗口的过程来处理消息。  
 **2. 直接使用回调函数来处理消息，则第四个参数设置为该回调函数的指针，则和这次播放有关的消息都将由该函数处理，这时第五个参数为传入该函数的参数，第六个参数为CALLBACK\_FUNCTION，表明由指定函数来处理消息。**

3. 使用新的线程来处理消息，则第四个参数设置为该线程函数的指针，和这次播放有关的消息都将由该线程处理，这时第五个参           数为传入该函数的参数，第六个参数为CALLBACK\_THREAD，表明由线程来处理消息。  
 4. 如果你不需要处理消息，这后面三个参数分别为NULL,NULL,CALLBACK\_NULL

猜想：执行waveOutWrite开始播放声音，播放完成会产生WOM\_DONE消息，难道播放完会继续播放指定内存处的音频数据吗？而不用加载缓冲数据头结构，或者再次调用waveOutWrite函数播放吗？录音缓冲满了之后会产生WIIM\_DATA消息，然后要在回调函数里清除缓冲，再加入新的缓冲吗？这些疑问都找不到答案。网上找的资料都太肤浅了。就只说个要产生某某消息，又没有说产生消息后具体怎么处理数据。是自动处理吗？

这样，指定了这些参数后，你应该就能够打开音频输入设备了。下面要做的事情就是准备几个用做录音的缓冲区。常准备多个缓冲区，并在回调中循环使用。对于缓冲区，得使用waveInPerpareHeader准备一下头部，这个API比较简单，如果你是循环使用缓冲区，对每个缓冲区也只需要调用一次waveInPrepareHeader。为什么要使用一次就可以。参看waveInPerpareHeader说明就明白。此函数功能就是定位缓冲区的数据区地址，和数据大小。以便为系统所用。

一切准备好之后，就可以调用waveInAddBuffer和waveInStart开始录音了，只要你一调用这个waveInStart，录音就开始了，即使这个缓冲区录满之后你没有加入新的缓冲区进去，录音也不会停，只是这中间的语音数据全都丢了。当通过 waveInAddBuffer送入的缓冲区被录满后，Windows就会通过你在waveInOpen中指定的方式进行回调，在回调中把录好的语音数据取出来，并且，如果还想继续录音的话，得将下一个缓冲区添加进去。考虑到这个处理是有时间延迟的，而且音频对时间很敏感，一般都要先预加入若干个缓冲区，有人提出：比如，一共定义了8个缓冲区，而为了保险起见，最好保证任一时刻至少有3个缓冲区可被录音使用，那么在开始录音时，则先加入4个缓冲区，然后在回调中，如果当前录好的缓冲区第n个，则对第(n+4)%8调用waveInAddBuffer，这时，还有第(n+1)%8,(n+2)%8, (n+3)%8这三个缓冲区可用，即基本上就可以保证所录得音频中不会有断开的间隔。比如0，1，2，3这些个先加入，当0好的时候对4，5 ，6 ，7调用waveInAddBuffer。

如此这样何不：开始的时候把8个全部放入缓冲区，当一个缓冲区满后调用回调，处理后立即把这个缓冲区重用，继续添加到缓冲区队列中。不更简单明了。如下

mmReturn = ::waveInPrepareHeader ( m\_hRecord, pHdr, sizeof(WAVEHDR) ); //准备

mmReturn = ::waveInAddBuffer ( m\_hRecord, pHdr, sizeof(WAVEHDR) );//添加

注意这两步都是在回调，或者县城中处理的。

百度百科上面说：警告:请不要在[回调函数](http://baike.baidu.com/view/414773.htm)里面调用任何的waveOut系列的函数,否则一定会造成死锁(也就是程序失去响应).哪怕是waveOutUnprepareHeader,waveOutClose

既然这些函数多不能调用，那么回调函数里面写上面呢，播放完了又该干什么呢？

CALLBACK\_EVENT   dwCallback 参数栏是事件柄  
 CALLBACK\_FUNCTION  dwCallback 参数栏是CALLBACK过程地址  
 CALLBACK\_NULL   默认的设置，即无CALLBACK进程  
 CALLBACK\_THREAD   dwCallback 参数栏是线程ID  
 CALLBACK\_WINDOW   dwCallback 参数栏是窗口柄  
 WAVE\_ALLOWSYNC   如果该项被设置，一个同步的装置能被打开。如果在打开一个同步驱动时没有用该项，装置打开将会失败。  
 WAVE\_FORMAT\_DIRECT  如果设定该项

我总算找到满意的参考程序了，这个程序是在scdn上下载的，使用的是wavex函数，并且采用回调机制，能边录边播，还能录音成wav文件。还能显示波形，界面相当精美。

现在不管是否运行软件，话筒的声音都能传入到耳机。而且那个显示波形的软件也罢工了，应该是系统的问题。

我在显示波形的程序里添加了waveoutwrite函数，运行程序后，确实能听到话筒输入的声音了。可是关闭软件后却仍然能听到话筒里的声音。这里音频采集播放的消息机制就像多线程一样。音频部分是独立于软件界面运行的，如果没有关闭它，而只是关闭了界面是没有真正完全关闭的。

。。。。。。//如果输入数据缓冲区满，系统会发出MM\_WIM\_DATA消息，在该消息的处理函数中可以对声音数据进行处理，包括播放、存储或者网络发送，处理完毕要清空数据缓冲区再加入输入队列直至结束

一直搞不懂参考程序是怎么运行的。现在发现了OnTimer,SetTimer等相关函数。多半软件是使用SetTimer定时器来采集音频信号的。

百度：当你想每隔一段时间执行一件事的的时候，你可以使用它。 使用定时器的方法比较简单，通常告诉WINDOWS一个时间间隔，然后WINDOWS以此时间间隔周期性[触发程序](http://baike.baidu.com/view/1661832.htm)。通常有两种方法来实现：发送WM\_TIMER消息和调用应用程序定义的[回调函数](http://baike.baidu.com/view/414773.htm)。

Settimer是设置定时器的，Ontimer是响应Settimer消息的。当Settimer设置的时间到了，就会自动调动Ontimer()函数。

在输入音频的回调函数的WIM\_DATA消息里，使用下面的就不会程序崩溃

WAVEHDR \*pwh;

pwh=(WAVEHDR \*)dwParam1;

waveInPrepareHeader(hwi,pwh,sizeof(WAVEHDR));

waveInAddBuffer(hwi,pwh,sizeof(WAVEHDR));

刚刚试验了一下，发现耳机不会听到话筒的输入声音的，当我打开那个被我修改的频谱显示程序后，立马就能从耳机听到来自话筒的声音了。当我关掉软件后，过了几秒钟，就不能听到这个声音了。再次打开也能听到。

当我把打开输出设备，播放等的部分程序注释掉的时候，打开程序仍然能从耳机听到话筒的声音。

在waveInOpen函数里的第五个参数里，如果是(DWORD)this，就能在waveInProc回调函数里使用CUsbfmDlg\*pDlg=(CUsbfmDlg\*)dwInstance;

pDlg->StreamAudioIn();

如果第五个参数为(DWORD) AfxGetInstanceHandle,就不能这样用了，在waveInProc里，

WAVEHDR \*pwh;

pwh=(WAVEHDR \*)dwParam1;

waveInPrepareHeader(hwi,pwh,sizeof(WAVEHDR));

/waveInAddBuffer(hwi,pwh,sizeof(WAVEHDR));

真是搞不明白。

快要成功了，使用WAVE\_MAPPER打开设备，在回调函数的WIM\_DATA消息里调用streamaudioin()函数，在这个函数里有个输出字符串到文本框里的函数。初始化完成后，发现不停地有字符串输出到文本框，这说明，音频是在不停滴采集的。如果是插入我的usb设备还是不成功的，因为还没有音频信号输入。

在采集和播放的同步之间的问题上，实际是采用多缓冲的。但应该不能采集完一个缓冲就播放某个缓冲，不管前一个缓冲是否播放完成。如果采集过快，播放慢，就丢掉一些缓冲。若果播放过快，就停止采集，让采集放缓。参考程序就是采用这种思路的，只是当初没发现它是采用定时器来控制的，一直不明白是怎么运作的。

2014年5月15日09:55:30

Usb的等时传输，是不会进入中断函数的。一旦一个端点被配置为ISO输入方式，主机将每帧发送一个IN牌；数据在每帧中的位置可能不同。因此建议将IOS IN 端点使能为双缓冲方式。

2014年5月28日20:44:28

今天继续做我的USB音频设备。今天答辩完了，真是惊心动魄的答辩啊。刚开始老师看到我做的，不相信是我自己做的。他说如果我是作弊的话，肯定会不及格，如果是我自己做的，就可能会得优秀。然后我将ppt，没将几页，他就不让我讲了，问我问题。其中有个问题是，下载口怎么和设备共地。他看到我的设备上的下载口只有两个接口，他怀疑不能下载，我跟他解释，肯定能下载，我都下载了好多遍了，不信的话，我可以给你演示。然后给他讲了一大堆，他没有听懂。然后他让我回去拿电脑，要看我的电路图。然后我就冒着大太阳会到寝室，拿电脑，然后找王国谦拿下载器。累的一身汗水。回来后，给他看了我的设备的下载器，他说我这本身就是共地的。其实我的设备和下载器使用的是同一台电脑的USB输出的电源，肯定是供地的。然后他看了我的电路图。这时，他才相信这时我自己做的了。他觉得我自己做得还不错，想给我推优。我跟他解释我的设备还没有完全成功，但是他觉得我做的还是不错，他说如果我做成功的话，可以在淘宝上卖钱。然后我就说随便吧，他说看我的导师杨斯涵是否愿意了。并给我打了个高分，95分。整个下午就发现两个90分以上的，还有一个是季成伟，他98分，非常高端。后来杨斯涵给我打电话了，说问我愿不愿意推优，如果愿意，要把论文修改一下，我还是说我的作品不完美，她让我再改改。我就说试试吧。现在就是如果我在这几天内成功了，就能够成为优秀。现在我就在修改设备。

刚刚把串口连接线重新焊接到设备上了，利用串口观察数据状态，不知道怎么回事发现数据尽然传输到电脑了，端点3中断一直在中断了，感觉又离成功近了一步。

等时传输，不需要硬件控制握手，或者错误检测。因此偶尔会发生传输错误。但是参考程序给出的音频数据传输是在端点3输入中断中传输的，这是怎么回事。

对于音频等时传输，当设备接收到了主机发送给的令牌包后，中断服务程序会指向端点3中断（端点3已经配置为等时传输端点）。也就是说，音频数据的传输还是在中断中传输的。

协议规定，对于等时传输的接口，必须定义一个备用的0带宽的接口，这个备用接口不是等时传输接口。协议中强加了这个要求是因为，当主机不能提供等时传输端点的时候，可以使用备用端点，才能在设备枚举的时候成功。也就是说当主机不能提供等时传输端点的时候也能枚举出等时传输的音频设备。

音频流端点描述符中，端点属性是异步还是同步，这个是什么意思，有什么区别？

等时传输可以被配置为3种方式：异步等时模式，此时音频流使用的时钟和usb时钟相独立；同步等时模式，此时音频流时钟和usb时钟相关联；适应模式，此时音频流时钟根据缓冲要求来变换。

我这里选择的是协议里的同步传输，而参考程序里是异步传输，是否应该改过来试试。

现在数据是能输入进电脑了，但非常慢。完全播放不出来。

现在发现set interface 请求后 就turnon stream 了，怎么调用这个函数的啊。还有不停地调用端点3输入中断。

只有整型变量的地址才能防盗指向整型变量的指针变量中。如float a; int \*p;p=&a;这是错误的。

指针赋值有两种方法： （1）int a; int \*p=&a; (2)int a; int \*p; p=&a; 这两句等价

二维数组a[][],a代表首行地址，a+1代表第二行地址，a[1]为第一行地址，即&a[1][0].

第一行，第二列的地址为a[1]+2.a 代表二维数组首个元素的地址

a[0]+1和\*(a+0)+1都是&a[0][1]

\*(a+1)+2就是&a[1][2]

\*(a[0]+1)和\*(\*(a+0)+1)都是a[0][1]的值

\*(a+i)和a[i]是等价的。在一维数组里a[i]表示值，在二维数组里a[i]代表地址，a[i]+j表示a[i][j]的地址，\*(a+i)表示地址，和a+i是等价的

a+1和a[0]+1不同，前者表示第一行地址，后者表示第0行第二列地址

usb全速设备，主机每一毫秒产生一个帧起始包（SOF）。而高速设备，主机每125us产生一个帧起始包。

2014年5月31日21:58:37

总算发出声音来了，我将要返回主机的音频信号，通过端点3直接发送存储在单片机里面的音频信号，这个信号是一个有很多数据的数组。将这个数组里的数据直接通过端点3发送给到主机，打开软件，发现数据速率明显比之前的用AD转换快多了，而且大约每一秒发出滴的一声。之前是每二十多秒，发出噗的一声噪音。

2014年6月1日10:56:12

主机每帧发送一个IN令牌包，这个每帧就是每毫秒，因此在收到端点3输入请求后就应该将一秒钟要发送的数据传输给主机。由于采样率为96khz，每毫秒就要发送94\*4字节的数据给主句。之前以为没1/96000秒主机都会发送数据或者没有请求，直接发送数据。这些都是错误的，主机还是会每秒发送请求，在端点3的输入中断中可以看到。

一旦使用发送指定数据到主机的函数，设备就不运行了。搞了好久都不明白。原来是这几个变量在 搞怪啊：//xdata unsigned int FIFO\_INDEX;

//xdata unsigned char SINE\_TABLE\_INDEX\_L=0;

//xdata unsigned char SINE\_TABLE\_INDEX\_R=1;

不要它们就正常，要了就不运行。

这真不科学啊，添加了一个xdata unsigned char 的别的变量就行，要原来的就不行。真他妈的不科学啊。艹啊。将变量名字全部换了之后，一切正常。这个变量名不都是随便定义的吗，况且又没有和别的变量冲突，编译都完全通过。就是不能运行。

上位机音频缓冲区大了好像还不行，导致播放延迟。