

基于FPGA的cy7c68013a双向通信实验

原创

置顶 [春哥笔记](#) 于 2018-05-11 20:47:35 发布 22909 收藏 105

分类专栏: [FPGA 嵌入式 USB CY7C68013](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/chengfengwenalan/article/details/80282946>

版权



[FPGA 同时被 3 个专栏收录](#)

33 篇文章 8 订阅

订阅专栏



[嵌入式](#)

2 篇文章 0 订阅

订阅专栏



[USB](#)

2 篇文章 0 订阅

订阅专栏

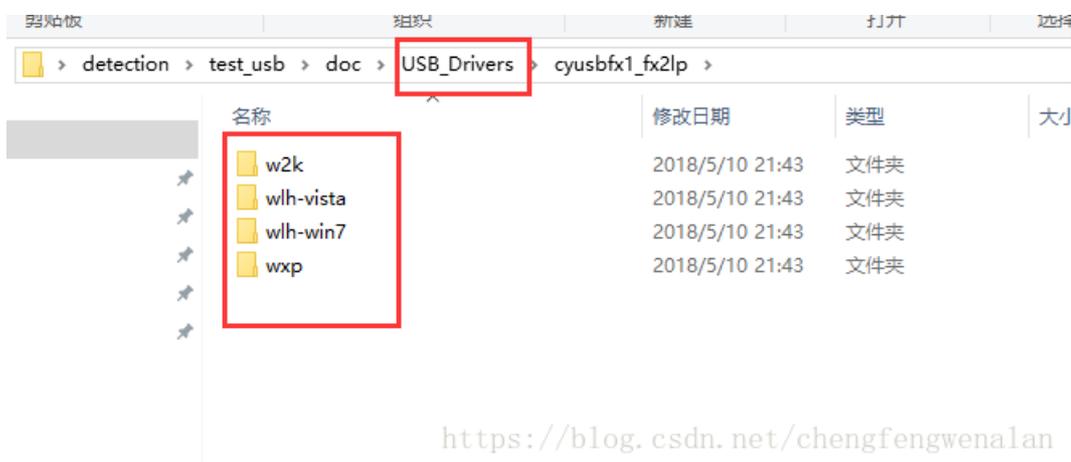
基于FPGA的cy7c68013a双向通信实验

本实验是基于FPGA的cy7c68013a的USB双向通信实验, 以前折腾过一段时间cy7c68013a, 没有入门时感觉好难, 入门了就会感觉很简单。本教程主要内容:

- 1.cy7c68013a的固件编写, 以及生成iic固件和下载固件。
- 2.cy7c68013a的slave模式, 以及他的读写时序
- 3.cy7c68013a的FPGA的上板测试, 包括发送和接受两部分

一、驱动

在进行试验前要先安装好cypress提供的usb驱动, 插上usb后, 电脑就会检测到未识别的设备, 这时打开设备管理器, 右键未识别的usb, 然后手动选择驱动, 驱动会在本教程最后的链接中给出



<https://blog.csdn.net/chengfengwenalan>


```
52
53 void TD_Init(void) // Called once at startup
54 {
55     CPUCS = ((CPUCS & ~bmCLKSPD) | bmCLKSPD1);
56     // CPUCS = 0x10; // CLKSPD[1:0]=10, for 48MHz operation, output CLKOUT
57     IFCONFIG = 0x43; // external clock source, IFCLK Tri-state, synchronous slave FIFO mode
58
59     PINFLAGSAB = 0x08; // FLAGA - EP2 Empty flag
60     SYNCDELAY;
61     PINFLAGSCD = 0xE0; // FLAGD - EP6 FULL flag 低有效 这个是标志信号，很重要
62
63     SYNCDELAY;
64     PORTACFG |= 0x80;
65
66     EP4CFG = 0x02; //clear the valid bits on ep4 and ep8
67     SYNCDELAY;
68     EP8CFG = 0x02;
69     SYNCDELAY;
70     EP2CFG = 0xA0; // OUT, 512-bytes, 4x, bulk 这个是设置IN和OUT端点
71     SYNCDELAY;
72     EP6CFG = 0xE0; // IN, 512-bytes, 4x, bulk
73
74
75     SYNCDELAY;
76     FIFORESET = 0x80; // activate NAK-ALL to avoid race conditions
77     SYNCDELAY; // see TRM section 15.14
78     FIFORESET = 0x02; // reset, FIFO 2
79     SYNCDELAY; //
80     FIFORESET = 0x04; // reset, FIFO 4
81     SYNCDELAY; //
82     FIFORESET = 0x06; // reset, FIFO 6
83     SYNCDELAY; //
84     FIFORESET = 0x08; // reset, FIFO 8
85     SYNCDELAY; //
86     FIFORESET = 0x00; // deactivate NAK-ALL
87
88
89     // handle the case where we were already in AUTO mode...
90     // ...for example: back to back firmware downloads...
91     SYNCDELAY; //
92     EP2FIFOCFG = 0x00; // AUTOOUT=0, WORDWIDE=0
93     // core needs to see AUTOOUT=0 to AUTOOUT=1 switch to arm endp's
94     SYNCDELAY; //
95     EP2FIFOCFG = 0x11; // AUTOOUT=1, WORDWIDE=1
96     SYNCDELAY; //
97     EP6FIFOCFG = 0x09; // AUTOIN=1, ZEROLENIN=0, WORDWIDE=1
98     SYNCDELAY;
99 }
```

<https://blog.csdn.net/chengfengwenalan>

固件只需要改这些参数，这里我都写好了，大家就不需要再改了，很容易看出我设置的时钟是48MHz，然后设置

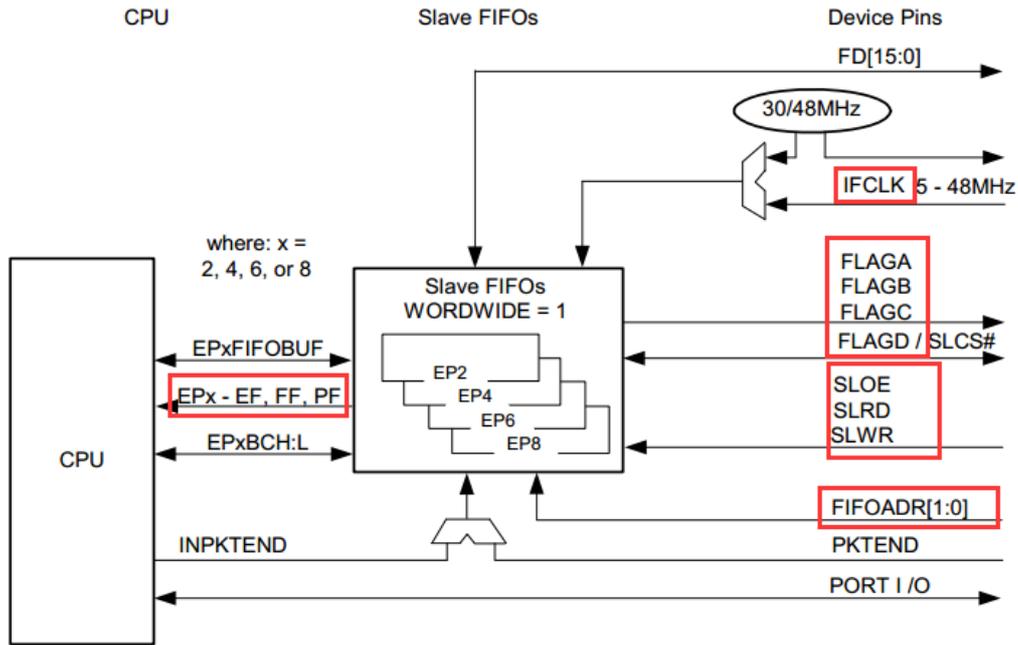
EP2为OUT端点，512字节，4缓冲，bulk（注意OUT,IN都是相对PC来说的，OUT表示PC-->cy7c68013a,IN则相反）

EP6为IN端点，512字节，4缓冲，bulk

flag_a 为EP2的EF，也就是空标志信号，为低时表示空，也就是没有数据过来，为高则表示有数据来了

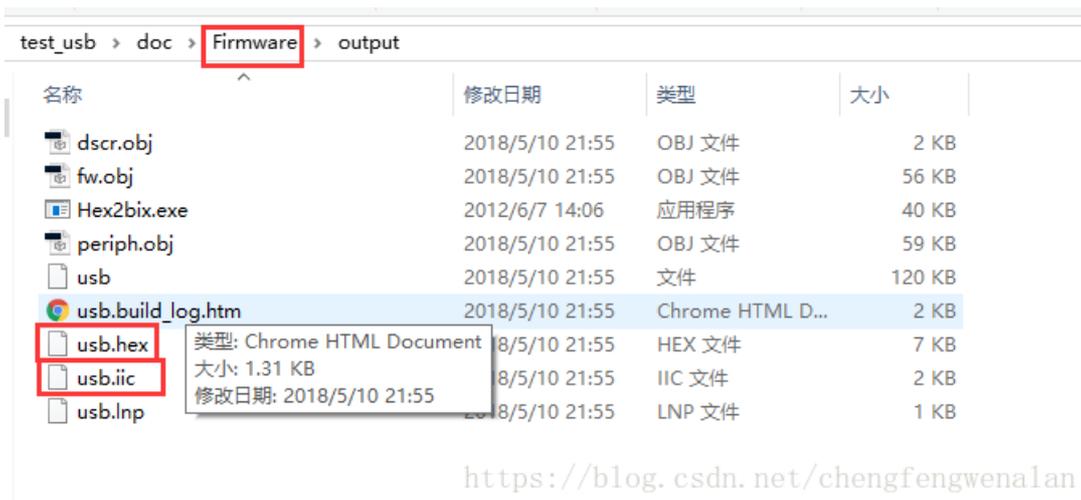
flag_d 为EP6的FF，也就是满标志信号，为低时表示写满了，这时再去写就是无效写了，为高则表示没有写满，可以继续写。

Figure 9-1. Slave FIFOs' Role in the EZ-USB System



<https://blog.csdn.net/chengfengwenalan>

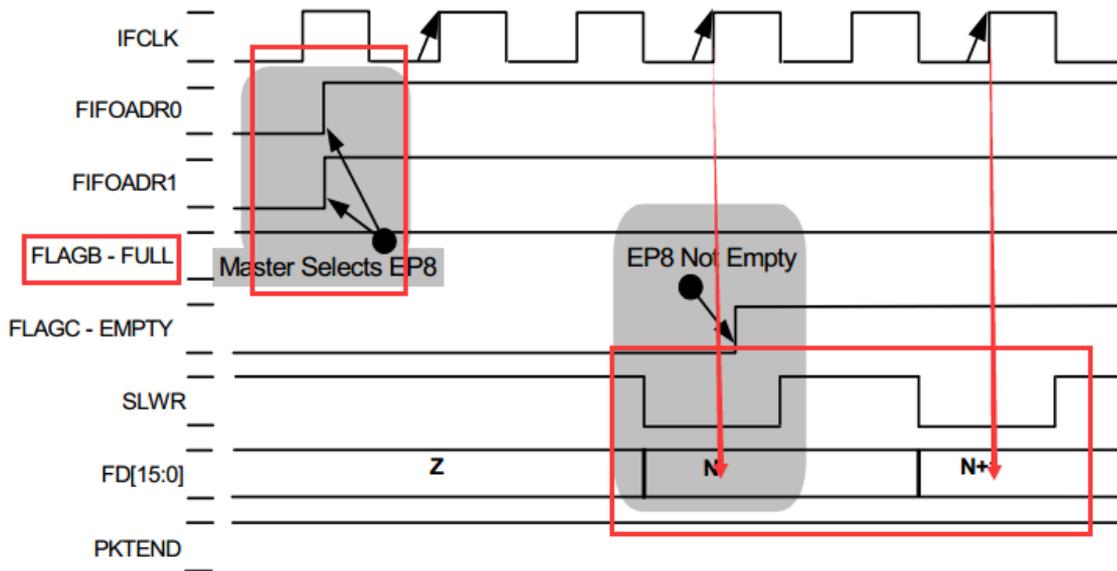
我写好的固件所在文件夹：固件源码什么的都在Firmware文件中



<https://blog.csdn.net/chengfengwenalan>

三、slave的写时序

Figure 9-12. Timing Example: Synchronous FIFO Writes, Waveform 1



<https://blog.csdn.net/chengfengwenalan>

有图很容易看出，再写之前要先把FIFOADR确认好，这个决定了你写的对象是谁

Table 9-2. FIFO Selection via FIFOADR[1:0]

FIFOADR[1:0]	Selected FIFO
00	EP2
01	EP4
10	EP6
11	EP8

<https://blog.csdn.net/chengfengwenalan>

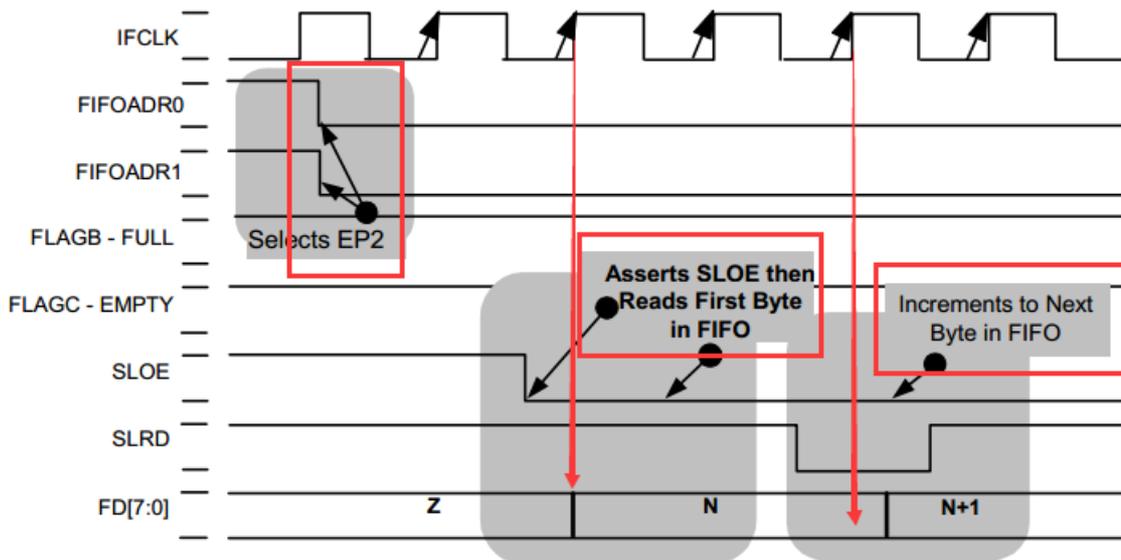
然后在fifo非满时（相应的FF标志位高），才可以进行写操作，这个时序很简单，就是拉低slwr信号就可以了，注意FD要与slwr对齐。

注意：写操作时，slwr与FD的数据都是FPGA来控制的，为了让cy7c68013a更好的采样，ifclk与clk反向之后再发送给cy7c68013a。

```
//=====
assign ifclk = ~clk;
```

四、slave读操作

Figure 9-17. Timing: Synchronous FIFO Reads, Waveform 1



<https://blog.csdn.net/chengfengwenalan>

读时序跟写也是类似的，再读之前先确定FIFOADR,然后拉低sloe，这时FD总线就会出现第一个数据，然后检测到slrd为低时，FD就会显示下一个数据。

五、FPGA与cy7c68013a通信

前面主要是一些准备工作，现在开始进入正式的通信过程，项目工程如下：

The screenshot shows a file explorer window for a project named 'test_usb'. The directory structure is as follows:

名称	修改日期	类型	大小
doc	2018/5/11 15:32	文件夹	
prj	2018/5/11 15:33	文件夹	
signal_tap	2018/5/10 20:17	文件夹	
sim	2018/5/9 21:23	文件夹	
src	2018/5/10 22:48	文件夹	

Additional annotations in red text provide context for the folders:

- doc: 驱动, API, 固件, datasheet
- prj: FPGA项目工程目录
- signal_tap: 调试
- sim: 仿真
- src: FPGA源码

<https://blog.csdn.net/chengfengwenalan>

```

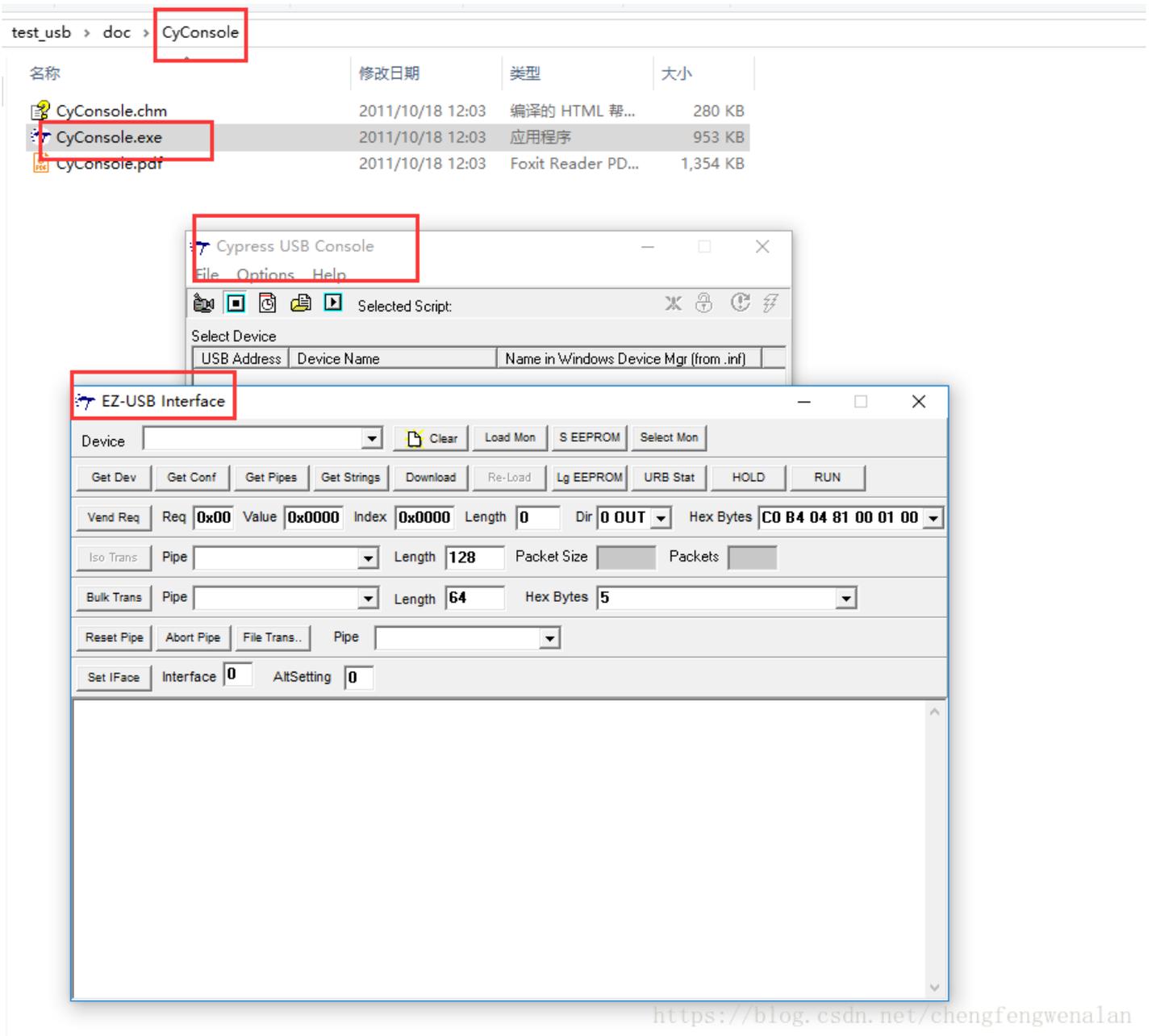
77
78 //state_n
79 always@(*)begin
80     case(state_c)
81         IDLE:begin
82             if(flag_a == 1'b1)begin //ep2非空，说明有数据来了，进入读状态
83                 state_n = READ;
84             end
85             else if(flag_d == 1'b1)begin //flag_d是EP6的FF，低有效，为高时表示该fifo现在空闲，往ep6写数据
86                 state_n = WRITE;
87             end
88             else begin
89                 state_n = state_c;
90             end
91         end
92         READ:begin
93             if(flag_a == 1'b0)begin //flag_a位低，说明数据已经读完了，进入空闲状态
94                 state_n = IDLE;
95             end
96             else begin
97                 state_n = state_c;
98             end
99         end
100        WRITE:begin
101            if(flag_d == 1'b0)begin //flag_d为低，说明写满了，进入空闲状态，这时cypress自己就会把这些数据打包好，然后发送给pc，
102                state_n = IDLE;
103            end
104            else begin
105                state_n = state_c;
106            end
107        end
108        default:begin
109            state_n = IDLE;
110        end
111    endcase
112 end

```

<https://blog.csdn.net/chengfengwenalan>

具体代码都已经有了详细注释了，自己去看看代码就知道了，这里就不详细说了

本教程所用的调试工具是官方的工具

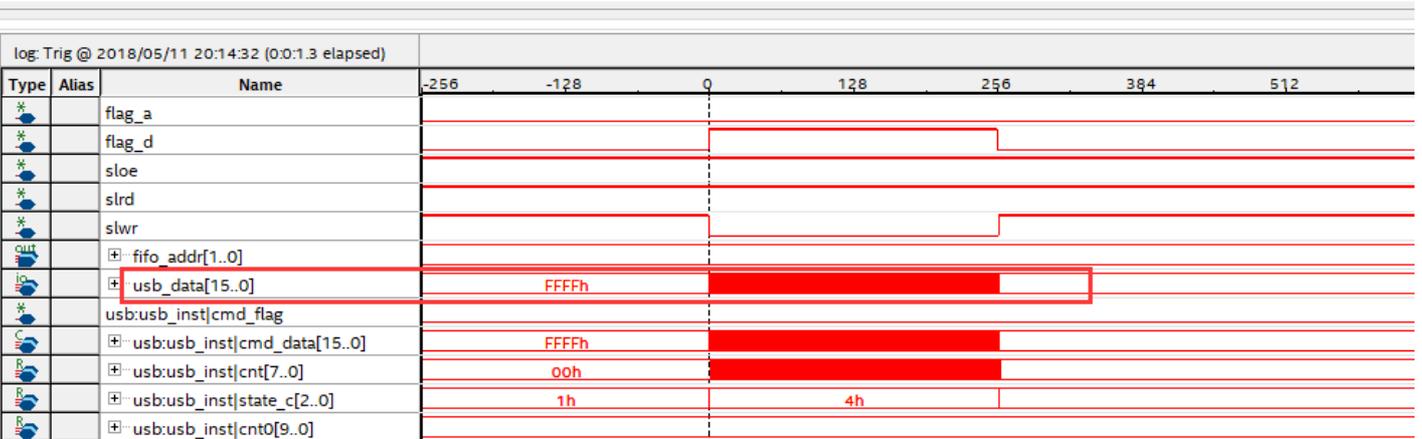


下面给出写的signal tap 的调试截图

写是一次写512个字节数据，0-255，注意usb的fifo是一次发送16位的，也就是2个字节。先发送低字节，然后再发送高字节，这我直接把低字节给赋值为0了

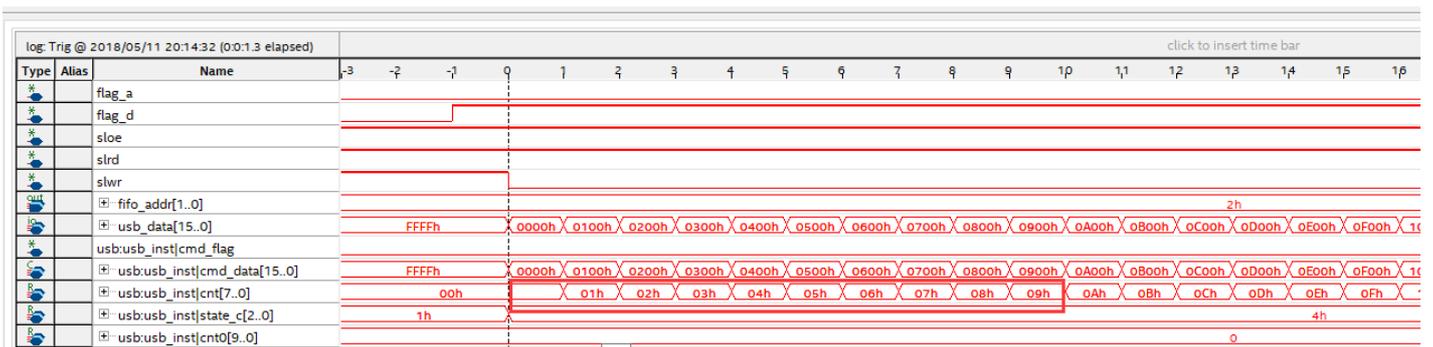
```
assign usb_data = (state_c[2] == 1'b1) ? {cnt, 8'h00} : 16'hzzzz; //先发送低字节，然后再发送高字节
```

trigger: 2018/05/10 22:52:05 #0		Lock mode: Allow all changes			
Type	Alias	Name	Data Enable	Trigger Enable	Trigger Conditions
		flag_a	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Basic AND
		flag_d	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		sloe	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		slrd	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		slwr	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
		fifo_addr[1..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Xh
		usb_data[15..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXh
		usb:usb_inst cmd_flag	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		usb:usb_inst cmd_data[15..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXh
		usb:usb_inst cnt[7..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXh
		usb:usb_inst state_c[2..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Xh
		usb:usb_inst cnt0[9..0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	XXXXXXXXXXb



<https://blog.csdn.net/chengfengwenalan>

前面局部放大图



<https://blog.csdn.net/chengfengwenalan>

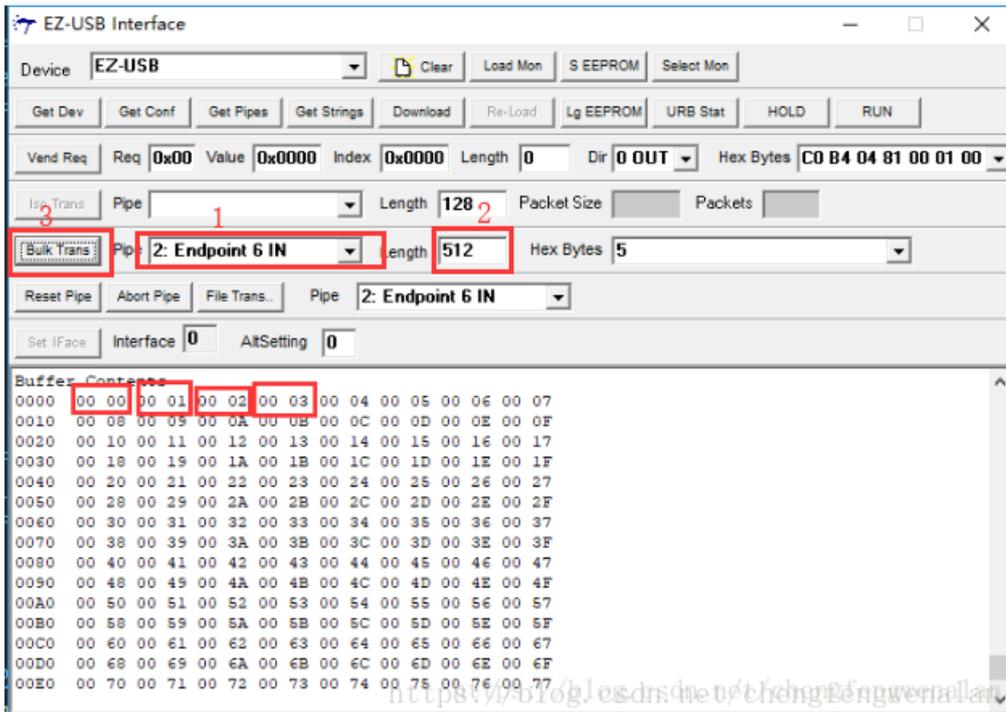
后面局部放大图，注意只有在flag_d为高时，slwr为低才是有效写，否则就是无效写，因为当flag_d为低时，表示写满了，这时fifo就会弃之后的写数据了（因为已经写满了，也装不下了是吧，总不能硬塞是吧，哈哈~）

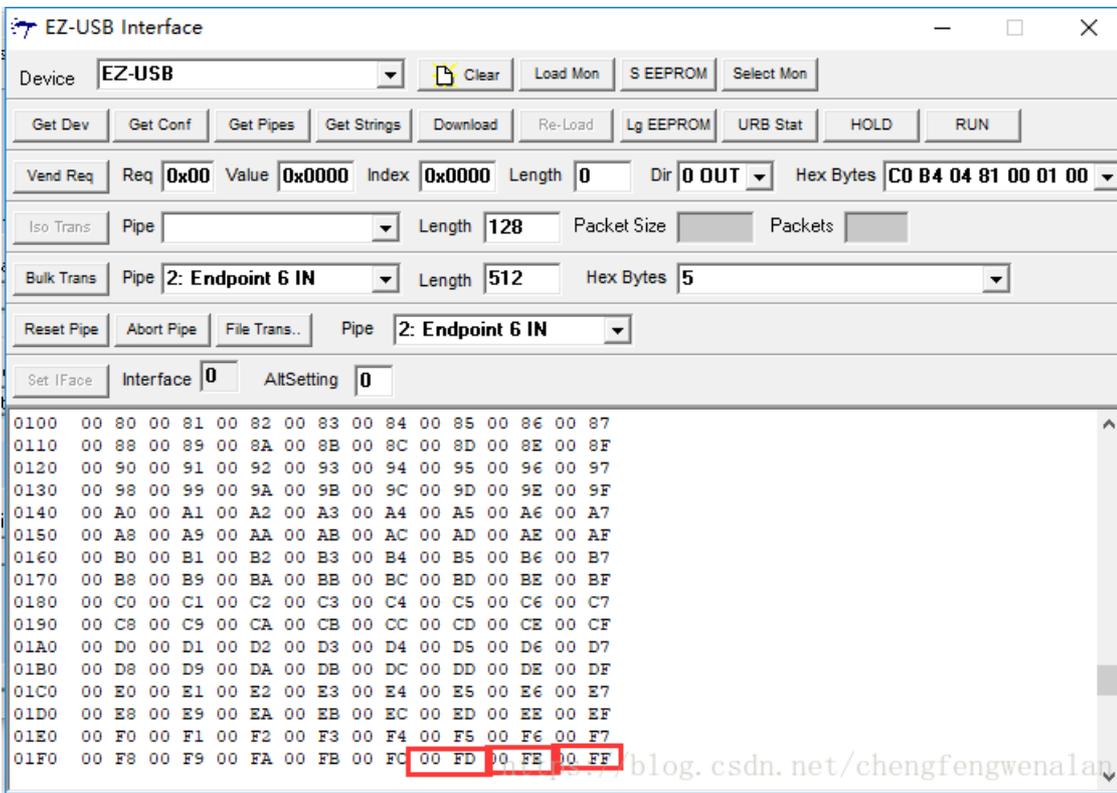
Alias	Name	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259
	flag_a																			
	flag_d																			
	sloe																			
	sldr																			
	slwr																			
	ffifo_addr[1..0]																			
	usb_data[15..0]	F100h F200h F300h F400h F500h F600h F700h F800h F900h FA00h FB00h FC00h FD00h FE00h FF00h 0000h																		
	usb:usb_inst[cmd_flag]																			
	usb:usb_inst[cmd_data[15..0]]	F100h F200h F300h F400h F500h F600h F700h F800h F900h FA00h FB00h FC00h FD00h FE00h FF00h 0000h																		
	usb:usb_inst[cnt[7..0]]	F1h F2h F3h F4h F5h F6h F7h F8h F9h FAh FBh FCh FDh FEh FFh 00h 01h																		
	usb:usb_inst[state_c[2..0]]																			
	usb:usb_inst[cnt0[9..0]]																			

<https://blog.csdn.net/chengfengwenalan>

PC端接受到的数据要2个字节一起读，因为usb是16位发送的，可以看出接受到的数据的确是0000-00FF

注意：pc接受数据按我标的编码顺序执行



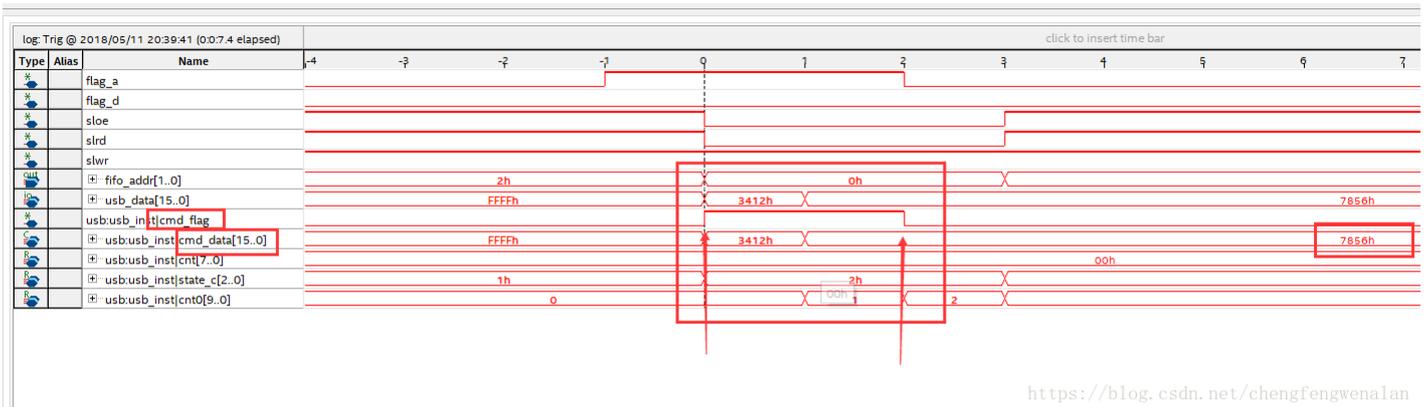
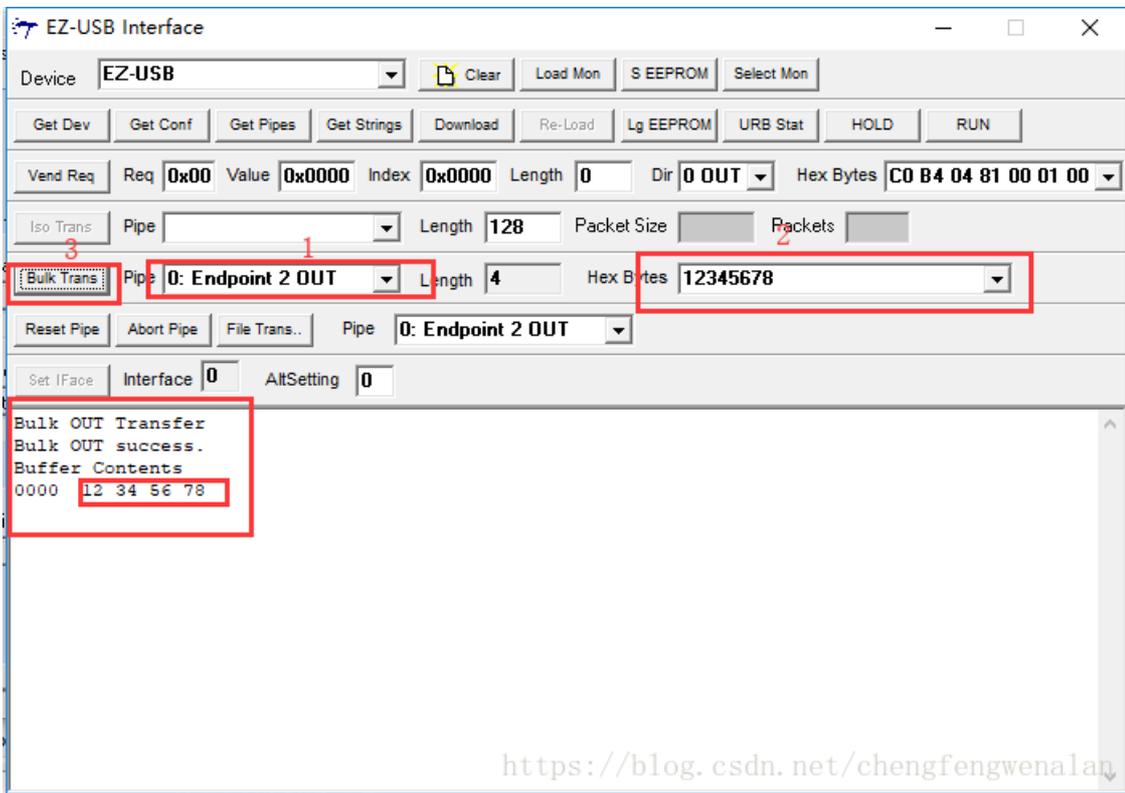


~~~~我是分割线~~~~

下面轮到“读”出场了

| trigger: 2018/05/10 22:52:05 #0 |       | Lock mode:                   | Allow all changes                   |                                     |                    |
|---------------------------------|-------|------------------------------|-------------------------------------|-------------------------------------|--------------------|
| Type                            | Alias | Name                         | Data Enable                         | Trigger Enable                      | Trigger Conditions |
| *                               |       | flag_a                       | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |                    |
| *                               |       | flag_d                       | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |                    |
| *                               |       | sloe                         | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |                    |
| *                               |       | slrd                         | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | 0                  |
| *                               |       | slwr                         | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |                    |
| out                             |       | fifo_addr[1..0]              | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | xh                 |
| io                              |       | usb_data[15..0]              | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | xxxxh              |
| *                               |       | usb:usb_inst cmd_flag        | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |                    |
| C                               |       | usb:usb_inst cmd_data[15..0] | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | xxxxh              |
| B                               |       | usb:usb_inst cnt[7..0]       | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | xxh                |
| B                               |       | usb:usb_inst state_c[2..0]   | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | xh                 |
| B                               |       | usb:usb_inst cnt0[9..0]      | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | XXXXXXXXXXb        |

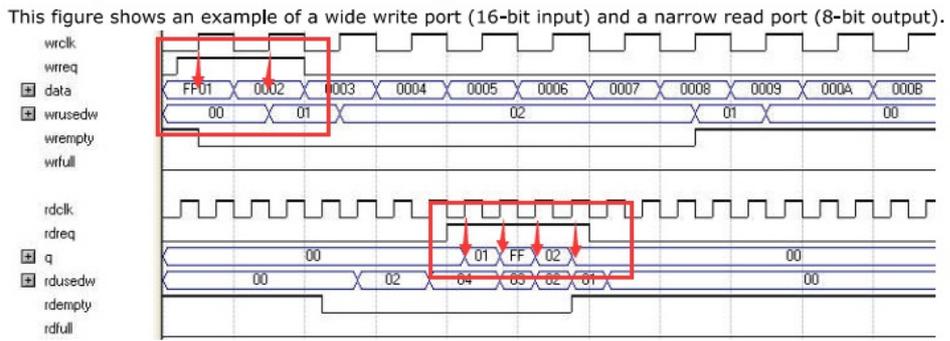
pc发送数据按1-->2-->3的步骤，可以看出我们发送了12 34 56 78 这4个字节



注意这里我是设置了cmd\_flag标志信号的，只有cmd\_flag为高时的cmd\_data的数据才是有效的，也就是pc发送过来的数据。

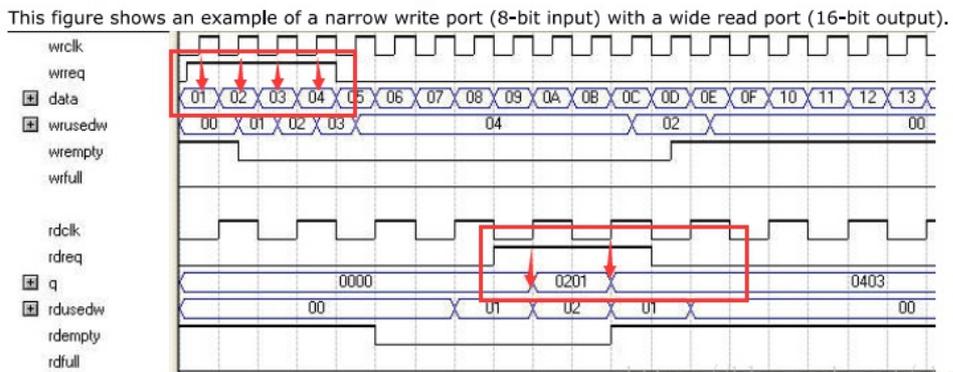
这里顺带插一嘴：

**Figure 7. Writing 16-bit Words and Reading 8-bit Words**



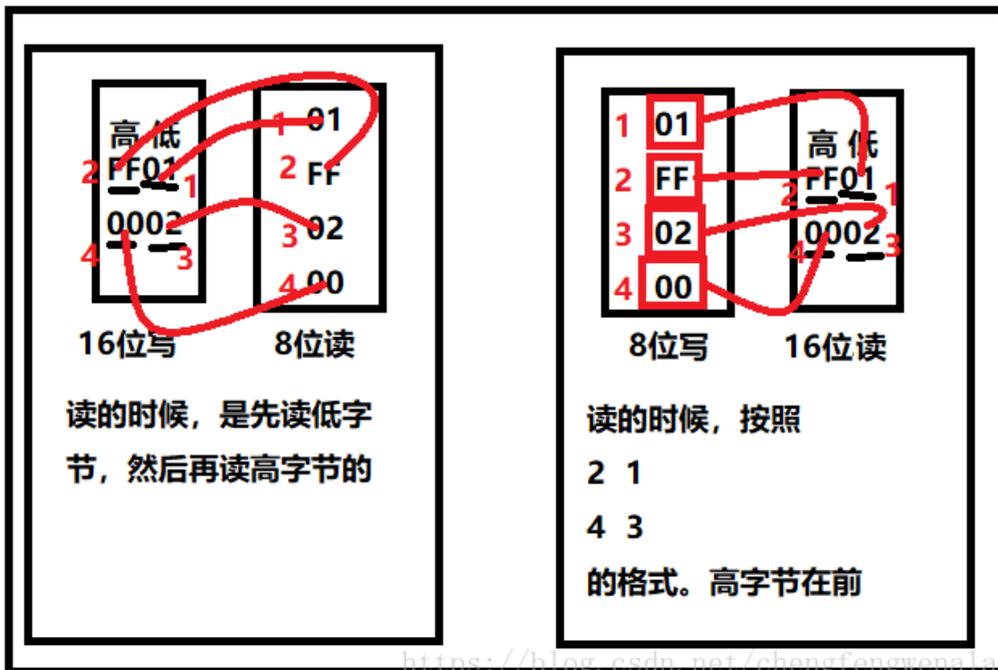
In this example, the read port is operating at twice the frequency of the write port. Writing two 16-bit words to the FIFO buffer increases the `wrusedw` flag to two and the `rdusedw` flag to four. Four 8-bit read operations empty the FIFO buffer. The read begins with the least-significant 8 bits from the 16-bit word written followed by the most-significant 8 bits.

**Figure 8. Writing 8-Bit Words and Reading 16-Bit Words**



<https://blog.csdn.net/chengfengwenalan>

alter 的 fifo ip 是可以读写位宽不一致的，具体看下面的图



<https://blog.csdn.net/chengfengwenalan>

由上图可以看出这个和usb是一样的格式，都是先发低字节，然后再发高字节。或者说先接受低字节，然后再接受高字节

至此本教程就全部介绍完了，讲的虽然有点简陋，但是基本的流程都已经讲到了，比起网上其他的教程还是要好不少的哈（自我陶醉一下），具体的看我的源代码。可以看出，使用cypress的usb还是比较简单的，因为usb协议什么的，他们都已经做好了，我们不需要考虑这些，要不然那一堆协议就看着头疼，具体的看我给的下载链接，我会把本教程所用到的工具，代码全部分享出来，也欢迎大家评论提问，不足之处还望指正~

## 六、福利

为了能及时回复大家，现在获取源码方式如下：

微信扫描下面的二维码关注【春哥笔记】公众号，回复“usb2”即可Get源码的获取方式：

