0426

++将数据传输改为双缓冲

需要对参数进行配置

void trans\_data1(void)

{

uint16\_t Send\_length=0,p\_size=0,i=0,trans\_time=0,less\_length=0,full\_time = 0,pack\_flag = 0,len = 0,usb\_in\_data\_remain = 0,usb\_in\_numofpackage=0,plus\_data=0;

// uint8\_t \*p = data\_final;

// less\_length = Length\_byte%64;

//Length\_byte = 150;

usb\_in\_numofpackage = Length\_byte / 64 + 1;

data\_final[0] = FRAME\_HEAD;

usb\_in\_data\_remain = Length\_byte;

//往两个缓存存入最初的64bytes数据

////////////////////////////

UserToPMABufferCopy(data\_final, ENDP1\_TXADDR, 64);

while(GetEPTxStatus(ENDP1)!= EP\_TX\_NAK);

SetEPDblBuf1Count(ENDP1, EP\_DBUF\_IN, 64);

UserToPMABufferCopy(data\_final, ENDP1\_TXADDR1, 64);

while(GetEPTxStatus(ENDP1)!= EP\_TX\_NAK);

SetEPDblBuf0Count(ENDP1, EP\_DBUF\_IN, 64);

///////////////////////////

usb\_in\_data\_remain -= 64;

plus\_data += 64;

/\*

首先在1、2两块缓存区，放入最开始的 64 Bytes数据。

第一次进入，交换缓存区，系统控制1，用户控制2，系统马上发送1，然后往缓存区2发放数据。

第二次进入，交换缓存区，系统控制2，用户控制1，系统发送2，然后往1存数据。

第三次进入，交换缓存区，系统控制1，用户控制2，系统发送1，然后往2存数据。

\*/

while(usb\_in\_numofpackage != 0)

{

usb\_in\_numofpackage --;

if(GetENDPOINT(ENDP1) & EP\_DTOG\_RX)

{

if(usb\_in\_numofpackage >= 0)

{

FreeUserBuffer(ENDP1, EP\_DBUF\_IN); //释放缓存区占用

}

if(usb\_in\_data\_remain > 0)

{

if(usb\_in\_data\_remain > VIRTUAL\_COM\_PORT\_DATA\_SIZE)

{

len = VIRTUAL\_COM\_PORT\_DATA\_SIZE;

}

else

{

len = usb\_in\_data\_remain;

}

SetEPTxValid(ENDP1);//发送缓冲区数据

UserToPMABufferCopy(data\_final + plus\_data, ENDP1\_TXADDR, len); //往空闲的缓存区放数据

while(GetEPTxStatus(ENDP1)!= EP\_TX\_NAK);

SetEPDblBuf0Count(ENDP1, EP\_DBUF\_IN, len);

//SetEPTxValid(ENDP1);//buf1发走

usb\_in\_data\_remain -= len;

plus\_data += len;

}

else

{

//SetEPDblBuf0Count(ENDP1, EP\_DBUF\_IN, 0);

SetEPTxValid(ENDP1);

}

}

else

{

if(usb\_in\_numofpackage >= 0)

{

FreeUserBuffer(ENDP1, EP\_DBUF\_IN);

}

if(usb\_in\_data\_remain > 0)

{

if(usb\_in\_data\_remain > VIRTUAL\_COM\_PORT\_DATA\_SIZE)

{

len = VIRTUAL\_COM\_PORT\_DATA\_SIZE;

}

else

{

len = usb\_in\_data\_remain;

}

SetEPTxValid(ENDP1);

UserToPMABufferCopy(data\_final + plus\_data, ENDP1\_TXADDR1, len);

while(GetEPTxStatus(ENDP1)!= EP\_TX\_NAK);

SetEPDblBuf1Count(ENDP1, EP\_DBUF\_IN, len);

// SetEPTxValid(ENDP1);

usb\_in\_data\_remain -= len;

plus\_data += len;

}

else

{

// SetEPDblBuf1Count(ENDP1, EP\_DBUF\_IN, 0);

SetEPTxValid(ENDP1);

}

}

}

Length\_byte = 1;

data\_number = 0;

Send\_length = 0;

}

void Virtual\_Com\_Port\_Reset(void)

{

/\* Set Virtual\_Com\_Port DEVICE as not configured \*/

pInformation->Current\_Configuration = 0;

/\* Current Feature initialization \*/

pInformation->Current\_Feature = Virtual\_Com\_Port\_ConfigDescriptor[7];

/\* Set Virtual\_Com\_Port DEVICE with the default Interface\*/

pInformation->Current\_Interface = 0;

SetBTABLE(BTABLE\_ADDRESS);

/\* Initialize Endpoint 0 \*/

SetEPType(ENDP0, EP\_CONTROL);

SetEPTxStatus(ENDP0, EP\_TX\_STALL);

SetEPRxAddr(ENDP0, ENDP0\_RXADDR);

SetEPTxAddr(ENDP0, ENDP0\_TXADDR);

Clear\_Status\_Out(ENDP0);

SetEPRxCount(ENDP0, Device\_Property.MaxPacketSize);

SetEPRxValid(ENDP0);

/\* Initialize Endpoint 1 \*/

/\*

SetEPType(ENDP1, EP\_BULK);

SetEPTxAddr(ENDP1, ENDP1\_TXADDR);

SetEPTxStatus(ENDP1, EP\_TX\_NAK);

SetEPRxStatus(ENDP1, EP\_RX\_DIS);

\*/

SetEPType(ENDP1, EP\_BULK);

ClearDTOG\_TX(ENDP1); // USB PERIPHERAL

ClearDTOG\_RX(ENDP1); // SW\_BUF for APPLICATION

SetEPDoubleBuff(ENDP1);

SetEPDblBuffAddr(ENDP1, ENDP1\_TXADDR, ENDP1\_TXADDR1);

SetEPDblBuffCount(ENDP1, EP\_DBUF\_IN, 64);

SetEPDblBuf0Count(ENDP1,EP\_DBUF\_IN,0);

SetEPDblBuf1Count(ENDP1,EP\_DBUF\_IN,0);

//ClearDTOG\_RX(ENDP1);

// ClearDTOG\_TX(ENDP1);

// ToggleDTOG\_TX(ENDP1);

SetEPRxStatus(ENDP1, EP\_RX\_DIS);

SetEPTxStatus(ENDP1, EP\_TX\_VALID);

/\* Initialize Endpoint 2 \*/

SetEPType(ENDP2, EP\_INTERRUPT);

SetEPTxAddr(ENDP2, ENDP2\_TXADDR);

SetEPRxStatus(ENDP2, EP\_RX\_DIS);

SetEPTxStatus(ENDP2, EP\_TX\_NAK);

/\* Initialize Endpoint 3 \*/

SetEPType(ENDP3, EP\_BULK);

SetEPRxAddr(ENDP3, ENDP3\_RXADDR);

SetEPRxCount(ENDP3, VIRTUAL\_COM\_PORT\_DATA\_SIZE);

SetEPRxStatus(ENDP3, EP\_RX\_VALID);

SetEPTxStatus(ENDP3, EP\_TX\_DIS);

/\* Set this device to response on default address \*/

SetDeviceAddress(0);

bDeviceState = ATTACHED;

}

#define ENDP1\_TXADDR (0x80)

#define ENDP1\_TXADDR1 (0xC0)

#define ENDP2\_TXADDR (0x100)

#define ENDP3\_RXADDR (0x110)

++传输前后对usb中断进行disable 和 enable的配置