USB 主机芯片 CH375 的应用参考

通过 CH375 控制其它 USB 设备

通过 USB 连接两个单片机系统

版本：1B

http://wch.cn

1、概述

USB 接口芯片 CH375 可以工作于主机方式或者设备方式。

在 USB 主机方式下，单片机可以通过 CH375 模仿计算机与其它 USB 设备通讯，例如：模仿计算机从其它 USB 设备中采集数据、模仿计算机控制 USB 打印机、从 USB 存储器中获取数据等。

另外，还可以用 CH375+CH372（或两个 CH375）在两个单片机系统之间构成 USB 对连，用于在两个单片机系统之间提供主从式数据通讯。

2、控制 USB 打印机

当前的主流打印机都采用 USB 接口，如果单片机系统需要连接 USB 打印机，就需要通过 CH375 等 USB 主机芯片实现。本节是关于这方面的说明，适用于符合 USB 打印机类规范的各种 USB 打印机。

2.1. 与并口打印机比较

对于单片机而言，操作 USB 打印机与操作并口打印机的主要区别在于：

①、USB 打印机需要一个 USB 设备枚举初始化过程（用 CH341 的 USB 打印机可省去该过程）。

②、并口打印机的数据输出是以字节为单位逐个输出的，需要 STB、ACK 以及 BUSY 信号的配合，而 USB 打印机的数据输出是以数据块为单位通过 DATA-OUT 事务完成的，一次最多可以写入 8、16、32 或 64 字节（不同打印机其值不同，CH341 是 32），由于 USB 传输本身具有状态应答以及忙状态处理，所以输出数据块的过程已经自动处理应答和忙状态。其中，USB 打印机的 DATA-OUT 事务相当于并口打印机的 STB，DATA-OUT 事务的 ACK 应答相当于并口打印机的 ACK，DATA-OUT 事务的 NAK 应答相当于并口打印机的 BUSY。

③、并口打印机的端口状态通过 PEMPTY、SELECT、ERROR 等信号输入，而 USB 打印机通过控制传输读取状态值。状态值为 1 字节(8 位)数据，其中：位 5(Paper Empty)为 1 说明无纸，位 4(Select)为 1 说明打印机联机，位 3(Not Error)为 0 说明打印机出错。

④、USB 打印机中的软复位 SOFT-RESET 相当于并口打印机的 INIT 信号，USB 打印机中的枚举初始化成功相当于并口打印机的 SLCT-IN 信号。

例子 C 语言程序是 CH375PRT.C，该程序可以做到让单片机象并口一样输出数据给 USB 打印机，该程序并未考虑打印格式以及打印描述语言。以下是其中的主要程序：发送、查询、初始化。

2.2. 向打印机输出数据

下面的子程序用于输出不大于 64KB 的数据块给 USB 打印机

```
void send_data( unsigned short len, unsigned char *buf ) {  /* 主机发送数据块,一次最多 64KB */
    unsigned char l, s;
    while( len ) {  /* 连续输出数据块给 USB 打印机 */
        toggle_send( tog_send );  /* 数据同步 */
```

```
        l = len>endp_out_size?endp_out_size:len;  /* 单次发送不能超过端点尺寸 */
        wr_usb_data( l, buf );  /* 将数据先复制到 CH375 芯片中 */
        s = issue_token( ( endp_out_addr << 4 ) | DEF_USB_PID_OUT );  /* 请求 CH375 输出数据 */
        if ( s==USB_INT_SUCCESS ) {  /* CH375 成功发出数据 */
            tog_send = ~ tog_send;  /* 切换 DATA0 和 DATA1 进行数据同步 */
            len-=l;  /* 计数 */
            buf+=l;  /* 操作成功 */
        }
        else if ( s==USB_INT_RET_NAK ) {  /* USB 打印机正忙,正常情况下应该稍后重试*/
/* 如果未执行 SET_RETRY 命令则 CH375 自动重试,所以不会返回 USB_INT_RET_NAK 状态 */
            /* s=get_port_status( );  如果有必要,可以检查是什么原因导致打印机忙 */
        }
        else {  /* 操作失败,正常情况下不会失败 */
            clr_stall( endp_out_addr );  /* 清除打印机的数据接收端点,或者 soft_reset_print() */
/*          soft_reset_print();  打印机出现意外错误,软复位 */
            tog_send = 0;  /* 操作失败 */
        }
/* 如果数据量较大,可以定期调用 get_port_status()检查打印机状态 */
    }
}
```

2.3. 从打印机输入状态

下面的子程序用于通过控制传输查询 USB 打印机的端口状态，适用于 CH375A 芯片

```
unsigned char get_port_status_X( ) {  /* 查询打印机端口状态,返回状态码,如果为 0FFH 则说明操作失败 */
/* 返回状态码中: 位 5(Paper Empty)为 1 说明无纸, 位 4(Select)为 1 说明联机, 位 3(Not Error)为 0 说明出错 */
    buffer[0]=0xA1; buffer[1]=1; buffer[2]=buffer[3]=buffer[4]=buffer[5]=0; buffer[6]=1; buffer[7]=0;
        /* 以上是控制传输获取打印机状态的 SETUP 数据包的内容 */
    wr_usb_data( 8, buffer );  /* SETUP 数据总是 8 字节 */
    if (issue_token_X((0<<4)|DEF_USB_PID_SETUP,0x00)==USB_INT_SUCCESS) { /* SETUP 阶段 DATA0 操作成功 */
        if (issue_token_X((0<<4)|DEF_USB_PID_IN,0x80)==USB_INT_SUCCESS) { /* DATA 阶段 DATA1 接收成功 */
            rd_usb_data( buffer );  /* 读出接收到的数据,通常只有 1 个字节 */
            wr_usb_data( 0, buffer );  /* 发送 0 长度的数据 DATA1 说明控制传输成功 */
            if (issue_token_X((0<<4)|DEF_USB_PID_OUT,0x40)==USB_INT_SUCCESS) /* STATUS 阶段操作成功 */
                return( buffer[0] );  /* 返回打印机的状态码 */
        }
    }
    return( 0xFF );  /* 返回操作失败 */
}
```

2.4. 初始化打印机

下面的子程序用于初始化 USB 打印机，分析描述符加载配置

```
unsigned char init_print() {  /* 初始化 USB 打印机,完成打印机枚举 */
#define  p_dev_descr        ((PUSB_DEV_DESCR)buffer)
#define  p_cfg_descr        ((PUSB_CFG_DESCR_LONG)buffer)
    unsigned char status, len, c;
    status=get_descr(1);  /* 获取设备描述符 */
```

```
    if ( status==USB_INT_SUCCESS ) {
        len=rd_usb_data( buffer );  /* 将获取的描述符数据从 CH375 中读出到单片机中,返回描述符长度 */
        if ( len<18 || p_dev_descr->bDescriptorType!=1 ) return( UNKNOWN_USB_DEVICE );
        if ( p_dev_descr->bDeviceClass!=0 ) return( UNKNOWN_USB_DEVICE );  /* 不符合 USB 规范 */
        status=set_addr(3);  /* 设置打印机的 USB 地址 */
        if ( status==USB_INT_SUCCESS ) {
            status=get_descr(2);  /* 获取配置描述符 */
            if ( status==USB_INT_SUCCESS ) {  /* 操作成功则读出描述符并分析 */
                len=rd_usb_data( buffer );  /* 将获取的描述符数据读出到单片机中,返回描述符长度 */
                if ( p_cfg_descr->itf_descr.bInterfaceClass!=7  /* 不是 USB 打印机或不符合规范 */
                    || p_cfg_descr->itf_descr.bInterfaceSubClass!=1 ) return( UNKNOWN_USB_PRINT );
                endp_out_addr=0;
                c=p_cfg_descr->endp_descr[0].bEndpointAddress;  /* 第一个端点的地址 */
                if ( (c&0x80)==0 ) {  /* OUT 端点 */
                    endp_out_addr=c&0x0f;
                    endp_out_size=p_cfg_descr->endp_descr[0].wMaxPacketSize;  /* 端点最大长度 */
                }
                else if ( p_cfg_descr->itf_descr.bNumEndpoints>=2 ) {  /* 接口有两个以上的端点 */
                    c=p_cfg_descr->endp_descr[1].bEndpointAddress;  /* 第二个端点的地址 */
                    if ( (c&0x80)==0 ) {  /* OUT 端点 */
                        endp_out_addr=c&0x0f;
                        endp_out_size=p_cfg_descr->endp_descr[1].wMaxPacketSize;
                    }
                }
                if ( endp_out_addr==0 ) return( UNKNOWN_USB_PRINT );  /* 不是 USB 打印机或不符合规范 */
                status=set_config( p_cfg_descr->cfg_descr.bConfigurationValue );  /* 加载 USB 配置 */
                return(status);
            }
        }
    }
}
```

3、连接两个单片机系统

如果应用系统中有很多地点分散的未连网终端设备，而工作人员又需要定期从终端设备中获取现场数据，那么可以有三个方案：

①、终端设备采用单片机和 CH375，平时保存数据在终端内部的存储器中，当工作人员插入 U 盘后，终端设备将数据转存到 U 盘中，完成现场数据收集。

②、终端设备采用单片机和 CH375 并配备一个 U 盘，平时直接保存数据到 U 盘中，工作人员定期取走保存了数据的 U 盘，换上另一个空 U 盘，供终端设备继续保存数据。

③、终端设备采用单片机和 CH372，平时保存数据在终端内部的存储器中，工作人员使用手持式数据采集器，采集器由单片机和 CH375 以及存储器构成，采集器中的单片机通过 CH375 模仿计算机从终端设备中获取数据，下面就讨论这种应用，两个单片机通过 USB 交换数据。

以上方案中，终端设备都具有 USB 通讯能力，所以不但可以连接 U 盘或者采集器，也可以直接连接计算机提交数据。

注意：如果终端设备不是采用 CH37X 芯片，那么表格以及例子中的端点号可能不同。

3.1、建立连接

	说明	在主机端 CH375	在设备端 CH372/CH375
	上电复位后， 设置芯片工作模式	set_usb_mode(5 或 6) 等待 USB 设备连接	set_usb_mode(2) 等待中断
建立连接	USB 物理连接时	收到设备连接中断	
	模仿标准的 USB 配置， 对于 CH37X 是可选步骤	get_descr(1) set_addr(2) get_descr(2) set_config(1)	CH37X 内置固件自动完成， 如果不是用 CH37X 芯片， 那么可能需要单片机实现
	自定义其它初始化	其它初始化 USB 配置	
以下进行正式通讯			

3.2、主机端发送数据

主机端主动发送数据给设备端，设备端在端点 2 进行数据接收

如果设备端尚未取走前一次接收到的数据，则在主机端执行下一次 ISSUE_TOKEN 时一直等待，主机端不产生操作完成中断，直到设备端准备好接收下一次数据或者通讯错误。

	说明	在主机端 CH375	在设备端 CH372/CH375
奇数次发送	准备将发送的数据	单片机用 WR_USB_DATA7 将数据写入主机端的 CH375，一次最多 64 字节	等待中断
	奇数次是输出 DATA0	SET_ENDP7(80H)	
	高位 2 指接收端的端点号，低位 1 启动 OUT 事务	ISSUE_TOKEN(21H)	
		当设备端收到数据产生中断时，主机端也会产生操作完成中断	收到数据后产生中断
			GET_STATUS 得状态码 02H，说明端点 2 已经收到数据
	收到主机端发出的数据		RD_USB_DATA 从 CH372 或 CH375 读出数据给单片机
偶数次发送	准备将发送的另一组数据	WR_USB_DATA7 写入数据	等待中断
	偶数次是输出 DATA1	SET_ENDP7(C0H)	
	向端点 2 启动 OUT 事务	ISSUE_TOKEN(21H)	
		当设备端收到数据产生中断时，主机端也会产生操作完成中断	收到数据后产生中断
			GET_STATUS 得状态码 02H，说明端点 2 已经收到数据
	收到主机端发出的数据		RD_USB_DATA 读出数据
以上是数据传输的主要过程，以下是辅助过程			
错误	如果主机端 ISSUE_TOKEN 操作完成中断返回错误，请按下行处理		
	复位设备端的端点 2 接收	CLR_STALL(02H)	内置固件自动完成

3.3、主机端接收数据

主机端从设备端接收数据，设备端在端点 2 进行数据发送

如果设备端尚未准备好数据，则在主机端执行 ISSUE_TOKEN 时一直等待，主机端不产生操作完成中断，直到设备端准备好数据或者通讯错误，主机端才产生操作完成中断。为了防止主机端一直等待下去，

通常应该事先通知设备端准备好数据，或者在初始化主机端后执行 SET_RETRY(0XH)命令，防止在设备端忙时主机端不断重试并且一直等待。

	说明	在主机端 CH375	在设备端 CH372/CH375
奇数次接收	准备将发送的数据		单片机用 WR_USB_DATA7 将数据写入设备端的 CH372 或 CH375，最多 64 字节
	奇数次是输入 DATA0	SET_ENDP6(80H)	等待中断
	高位 2 指接收端的端点号，低位 9 启动 IN 事务	ISSUE_TOKEN(29H)	
		收到设备端发出的数据后产生操作完成中断	当主机端收到数据产生中断时，设备端也会产生发出数据中断
	收到设备端发出的数据	RD_USB_DATA 从主机端的 CH375 读出数据给单片机	GET_STATUS 得状态码 0AH，说明端点 2 已经发出数据
			UNLOCK_USB 解除锁定
偶数次接收	准备将发送的另一组数据		WR_USB_DATA7 写入数据
	偶数次是输入 DATA1	SET_ENDP6(C0H)	等待中断
	向端点 2 启动 IN 事务	ISSUE_TOKEN(29H)	
		收到设备端发出的数据后产生操作完成中断	发出数据后产生中断
	收到设备端发出的数据	RD_USB_DATA 读出数据	GET_STATUS 得状态码 0AH，说明端点 2 已经发出数据
			UNLOCK_USB 解除锁定
以上是数据传输的主要过程，以下是辅助过程			
错误	如果主机端 ISSUE_TOKEN 操作完成中断返回错误，请按下行处理		
	复位设备端的端点 2 发送	CLR_STALL(82H)	内置固件自动完成

3.4、实际程序

以 C 语言为例，源文件是 CH375LNK.C。本例中只使用 CH372/CH375 的主端点 2 进行数据收发，没有使用端点 1。

设备端的程序与连接计算机时的处理过程是相同的，当 USB 设备连接到主机端 CH375 芯片时无需特殊处理，除非上位机与下位机另有约定，标明设备端的步骤只是为了配合说明整个数据流程。

3.4.1、基本操作

```
unsigned char wait_complete() {  /* 查询方式，主机端等待操作完成，返回操作状态 */
    while( CH375_INT_WIRE );  /* 查询等待 CH375 操作完成中断(INT#低电平)，可参考打印机程序进行超时处理 */
    CH375_WR_CMD_PORT( CMD_GET_STATUS );  /* 产生操作完成中断，获取中断状态 */
    return( CH375_RD_DAT_PORT() );
}
unsigned char endp6_mode, endp7_mode;
void set_usb_mode( unsigned char mode ) {  /* 设置 USB 工作模式 */
    CH375_WR_CMD_PORT( CMD_SET_USB_MODE );
    CH375_WR_DAT_PORT( mode );
    endp6_mode=endp7_mode=0x80;  /* 复位 USB 数据同步标志 */
    while ( CH375_RD_DAT_PORT()!=CMD_RET_SUCCESS );
```

```
}
```

3.4.2、数据同步

USB 的数据同步通过切换 DATA0 和 DATA1 实现：在设备端，CH372/CH375 可以自动切换；在主机端，必须由 SET_ENDP6 和 SET_ENDP7 命令控制 CH375 切换 DATA0 与 DATA1。

主机端的程序处理方法是为 SET_ENDP6 和 SET_ENDP7 分别提供一个全局变量，初始值均为 80H，每执行一次成功事务后将位 6 取反，每执行一次失败事务后将其复位为 80H。

```
void set_endp6() {
    CH375_WR_CMD_PORT( CMD_SET_ENDP6 );
    CH375_WR_DAT_PORT( endp6_mode );
    endp6_mode^=0x40;
    delay2us();
}
void set_endp7() {
    CH375_WR_CMD_PORT( CMD_SET_ENDP7 );
    CH375_WR_DAT_PORT( endp7_mode );
    endp7_mode^=0x40;
    delay2us();
}
```

3.4.3、数据读写，单片机读写 CH372 或者 CH375 芯片中的数据缓冲区

```
unsigned char rd_usb_data( unsigned char *buf ) {  /* 读出数据块 */
    unsigned char i, len;
    CH375_WR_CMD_PORT( CMD_RD_USB_DATA );
    len=CH375_RD_DAT_PORT();  /* 后续数据长度 */
    for ( i=0; i!=len; i++ ) *buf++=CH375_RD_DAT_PORT();
    return( len );
}
void wr_usb_data( unsigned char len, unsigned char *buf ) {  /* 写入数据块 */
    CH375_WR_CMD_PORT( CMD_WR_USB_DATA7 );
    CH375_WR_DAT_PORT( len );  /* 后续数据长度，len 不能大于 64 */
    while( len-- ) CH375_WR_DAT_PORT( *buf++ );
}
```

3.4.4、主机操作

```
unsigned char issue_token( unsigned char endp_and_pid ) {  /* 执行 USB 事务 */
/* 执行完成后，将产生中断通知单片机，如果是 USB_INT_SUCCESS 就说明操作成功 */
    CH375_WR_CMD_PORT( CMD_ISSUE_TOKEN );
    CH375_WR_DAT_PORT( endp_and_pid );  /* 高 4 位目的端点号，低 4 位令牌 PID */
    return( wait_complete() );  /* 等待 CH375 操作完成 */
}
void host_send( unsigned char len, unsigned char *buf ) {  /* 主机发送 */
    wr_usb_data( len, buf );
    set_endp7();
    if ( issue_token( 0x21 )!=USB_INT_SUCCESS ) ERROR();
}
```

```
unsigned char host_recv( unsigned char *buf ) {  /* 主机接收，返回长度 */
    set_endp6();
    if ( issue_token( 0x29 )!=USB_INT_SUCCESS ) ERROR();
    return( rd_usb_data( buf ) );
}
```

3.4.5、主机端的主程序简单示例

```
main() {
    unsigned char data_to_send[250], data_by_recv[250];
    unsigned char  len; unsigned short i;
    set_usb_mode( 6 );  /* 设置USB主机模式，如果设备端是CH37X，那么5和6均可 */
    while ( wait_complete()!=USB_INT_CONNECT );  /* 等待设备端连接上来 */
    for ( i=0; i<250; i+=64 ) host_send( 64, &data_to_send[i] ); /* 发送250字节的数据给设备端 */
    host_send( 0, NULL );  /* 假定，发送空数据给设备端就能通知设备端发送数据 */
    for ( i=0; i<250; i+=len ) len=host_recv( &data_by_recv[i] ); /* 从设备端接收250字节的数据 */
}
```