

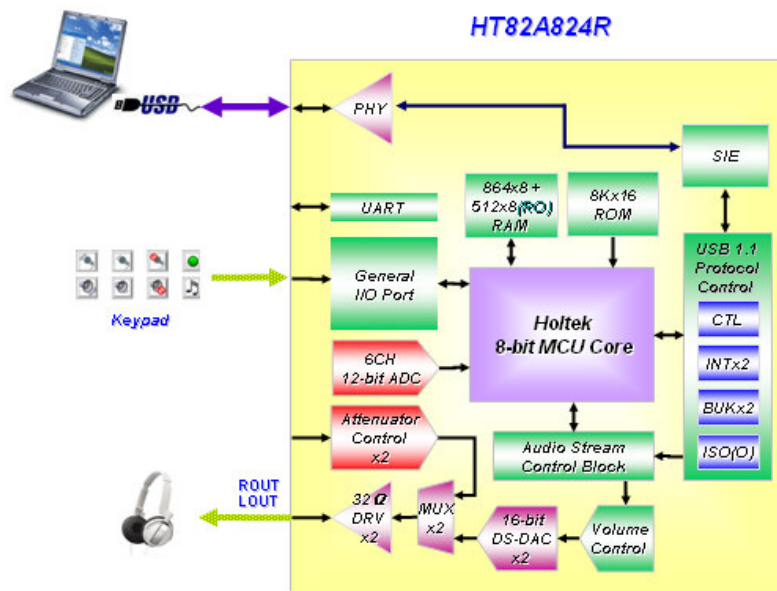
# HT82A824R 在 USB Speaker 的应用说明

文件编码: HA0277S

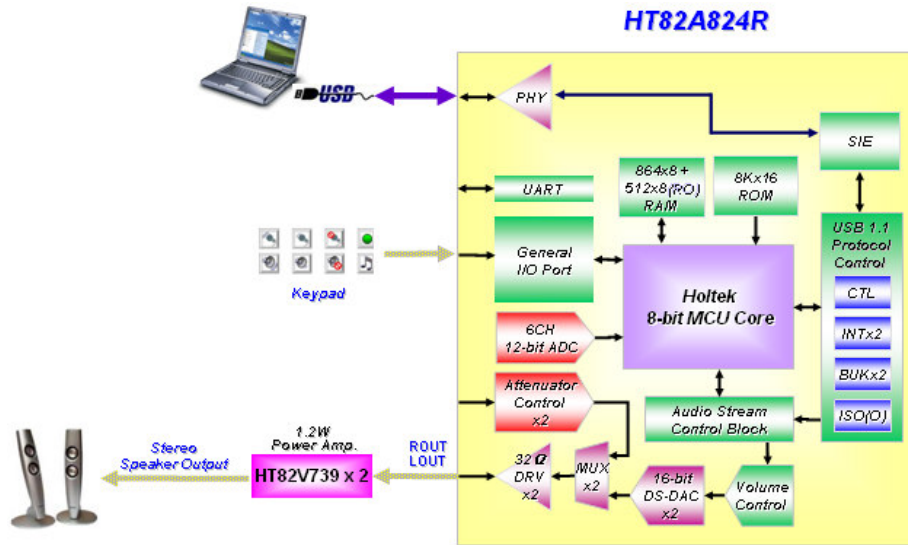
## 简介

HT82A824R 为 Enhanced USB Speaker MCU，本文将介绍如何使用 HT82A824R 设计 USB Speaker 相关产品，也探讨 USB Audio Class 的规范与程序架构说明。

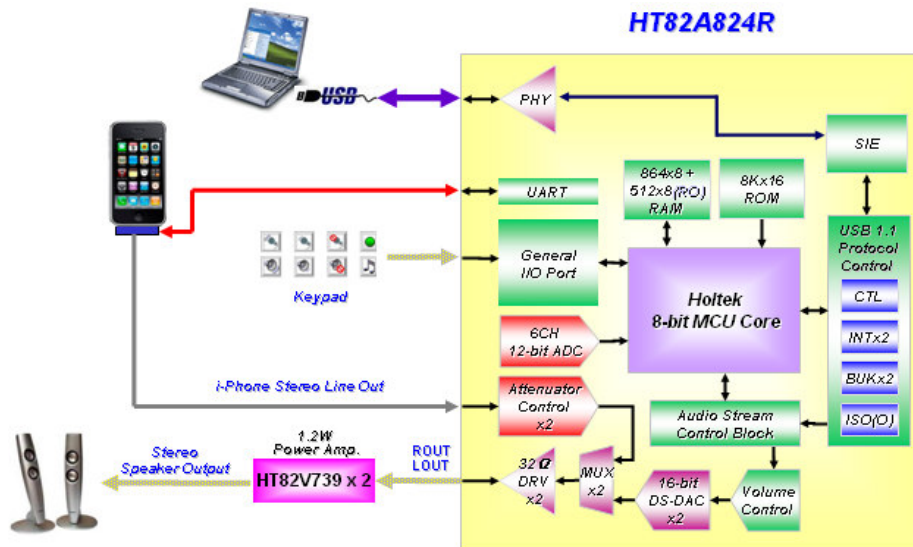
## HT82A824R USB Headphone 应用方框图



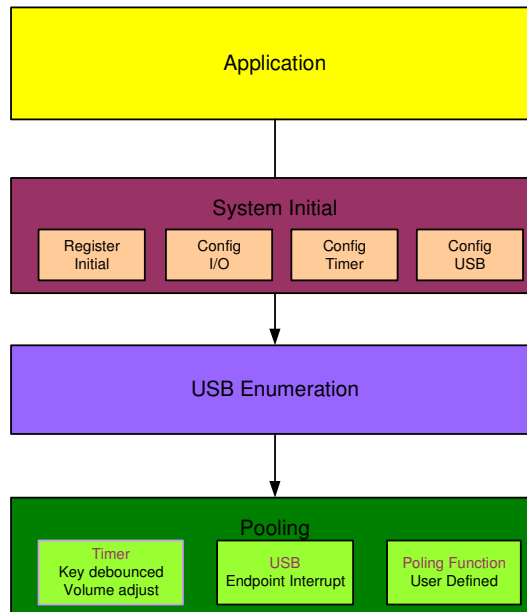
HT82A824R USB Speaker 应用方框图



HT82A824R USB/i-Phone Docking Speaker 应用方框图



## Holtek USB Audio 程序架构



Register Initial：寄存器初始化

Config I/O：I/O 初始化

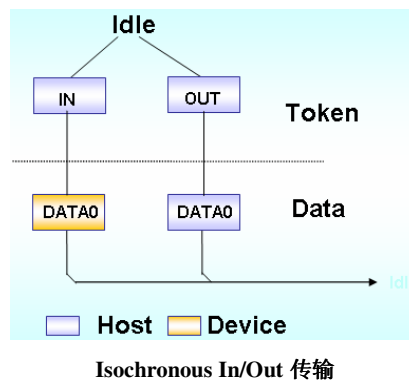
Config Timer：Timer 相关寄存器初始化

Config USB：USB 寄存器初始化，USB 寄存器设置完，USB Function 即会开始动作

USB Enumeration：此步骤为 USB 做列举的动作

当 USB 完成列举的动作后,有三个小模块 (Timer 中断、USB Endpoint 中断、Pooling Function) 是使用者可以自行修改并作运用的。

HT82A824R 的 USB Audio 使用 Full Speed Isochronous 传输模式，每个 Frame 为 1ms (USB 规范  $1\text{ms} \pm 500\text{ns}$ )，根据不同的频率会传送不同大小的语音数据量，例如 48kHz/16-bit/2-CH 时，1ms 的数据量为 192 Bytes，以此为例，盛群 USB Phone MCU 内部有两块 Ping-Pong 结构的 Buffer，各为 192 Bytes，总共为 384 Bytes，以 SOF (Start Of Frame) 做为切换两个 Buffer 的控制信号。由 Isochronous In/Out 传输的图可以发现 USB 的 Isochronous 传输不需要 ACK 信号，与 USB 的其它三种传输 Control/Interrupt/Bulk 不同，也就是说，就算是数据有错，也不会重传。



HT82A824R 有一个 Isochronous Out 端点 (Endpoint)。Isochronous Out 是用来将 PC 的播放媒体的语音数据通过 USB Port 传给装置，语音数据设置为 PCM 格式，假设是在 48kHz/16-bit/2-CH 情况下，一个 Frame 传 192 Bytes 数据，所以此端点的 Ping-Pong Buffer 大小总共为 384 Bytes，在描述符 (Descriptor) 关于播放语音格式的定义如下文字描述所示，在此描述符定义了播放频率、一个 Frame 的数据量、端点号码与端点的方向等；HT82A824R 的 Endpoint Number 设置为 2；如下的描述语言支持 48kHz/16-bit/2-CH 与 44.1kHz/16-bit/2-CH 两种格式。

HT82A824R Isochronous 端点都是由硬件来实现，对于写软件的工程人员，不必耗费大量资源来处理喇叭的语音数据，可以大幅节省软件开发的时间。如果客户需要分析 USB Audio 的数位数据，可通过 HT82A824R 内部硬件 DMA 的方式把数据放置在 RAM Bank 1~4，再由软件分析 USB Audio 的数据内容。

```
format_type_descriptor:
;support 48K/44.1K
DW 0240EH ;descriptor type(CS_INTERFACE), size of descriptor (add 44.1 KHz)
DW 00102H ;FormatType(FORMAT_TYPE_I), descriptorSubType(FORMAT_TYPE)
DW 00202H ;SubFrameSize(2 byte per slot), number of channel(2 channels)
DW 00210H ;SamFreqType(support 2 type), BitSolution(16 bits)(add 44.1 KHz)
DW 03F80H ;Sample Frequency(48000 Hz)
DW 000BBH ;
DW 03F44H ;Sample Frequency(44100 Hz)
DW 000ACH ;

end_point_descriptor:
DW 00509H ;descriptor type(END_POINT), size of descriptor
DW 00902H ;endpoint attributes(adaptive,isochronous), endpoint2(out direction)
DW 000C0H ;maxPacketSize(192 bytes)
DW 00001H ;Refresh(0), Interval(1ms)
DW 03F00H ;index string of this descriptor
```

### Microsoft® WHQL Audio Requirement

要取得微软的 USB Audio Device Logo, 在 Audio 这个部分,必需符合下列两个规范内容:

- 取样频率: USB Audio device 输出的取样频率必需支持 44.1k 与 48k (或其整数倍频率)
- 输出的 Audio 质量必需符合如下规格 (USB Speaker)

For premium desktop implementations:

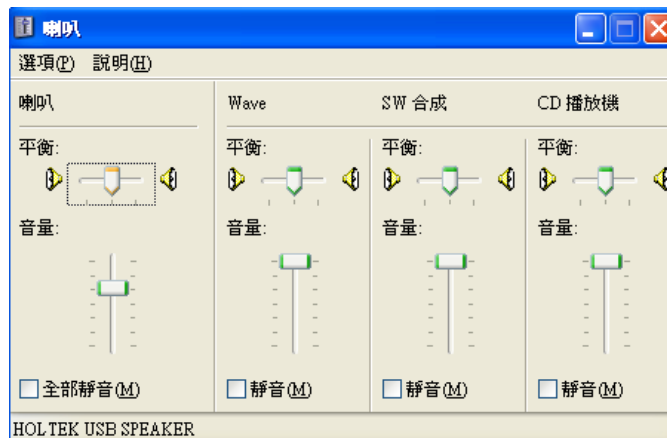
Device Type	Requirement	Value	Frequency range at 48kHz and above	
Analog Speaker Output Jack (Example: 125mW into 8 Ohm load)	THD+N	$\geq 75\text{dB FS}$	[20Hz, 20kHz]	
	Dynamic range with signal present	$\geq 90\text{dB FS A-weight}$	[20Hz, 20kHz]	
	Magnitude Response	$\leq \pm 2.5\text{dB ripple}$ (.5dB peak to peak delta), 1dB at upper band edge, 3dB at lower band edge		[20Hz, 20kHz]
	Sampling frequency accuracy	0.02%		
	Line output cross-talk	$\geq 60\text{dB}$		[20Hz, 15kHz]
	Full scale output voltage	$\geq 1\text{Vrms}$		
	Dynamic range with signal present during system activity	$\geq 90\text{dB FS A-weight}$		[20Hz, 20kHz]
	Interchannel phase delay	30 degrees or 12.5 microseconds, whichever is greater	[20Hz, 20kHz]	

注意事项: HT82A824R 不支持同时有 Speaker 与耳机插孔产品的 WHQL Audio 认证。支持耳机插孔输出的产品测试 WHQL Audio 时需要测试 Audio Fidelity。其中 Render Power Transition Test 测试项目会 fail 因而导致 WHQL 测试结果 fail。若产品不支持 jack 插孔输出则无此测试项。

### USB 音效装置说明

#### 音量控制

喇叭的音量控制: 在微软操作系统有一个音量控制的窗口, 当使用者拉动控制键时, USB 会透过端点 0 (Control Endpoint) 下一个 SET\_CUR 的请求函数来调整 (SET\_CUR 请求函数内容如下表格所示) 音量, PC 下载过来的音量设置值会反应到 IC 的 Digital Volume Control 寄存器 (USVC), 因此就能改变音乐播放时的声音大小。



下表为喇叭的音量控制请求函数，由 SET\_CUR 的 wIndex 字段来决定是要对喇叭还是麦克风做音量控制，wIndex=0x0200 表示要对喇叭做音量控制。

Offset	Field	Size	Value	Description
0	bmRequestType	1	0x21	
1	bRequest	1	0x01	SET_CUR
2	wValue	2	0x0200	DAC : VOLUME_CONTROL Master Channel
4	wIndex	2	0x0200 0x0600	D/A Feature Unit ID: 0x02(Lineout Lch and Rch Volume) Lower Byte : Audio Control Interface(0x00)
6	wLength	2	0x0002	Volume Control

#### SET\_CUR 请求函数内容

Offset	Field	Size	Value	Description
0	bMute	2	0xYYYY	The value is set by host

#### SET\_CUR 请求函数送出 2-Bytes 的数据

在对喇叭进行音量控制时，请求函数后会跟随 2-Bytes 的数据，这两个 Bytes 数据就是音量大小的值，下表的喇叭音量控制的高字节，是根据 IC 的音量控制寄存器所定义的设置范围。此外，在总线列举 (Bus Enumeration) 过程中，操作系统会以 SET\_CUR 的请求函数来设置喇叭的初始音量值，一般而言会设在中间值。

Volume Value	USB Audio Class Format
+6.0 dB	0x0C00
+5.5 dB	0x0B00
↓	↓
+0.5 dB	0x0100
+0.0 dB	0x0000
-0.5 dB	0xFF00
-1.0 dB	0xFE00
↓	↓
-12.0 dB	0xE800
-13.0 dB	0xE700
-14.0 dB	0xE600
↓	↓
-32.0 dB	0xC800

#### 喇叭音量控制的高字节(低字节皆为 00H)

Transaction	F	SETUP	ADDR	ENDP	T	D	Tr	R	bRequest	wValue	wIndex	wLength	ACK	Time Stamp
8108	S	0xB4	7	0	0	H>D	C	I	0x01	0x0200	0x0200	2	0x4B	00027.0482.2730
Packet	Dir	F	Sync	SETUP	ADDR	ENDP	CRC5	EOP	Idle	Time Stamp				
16908	-->	S	00000001	0xB4	7	0	0x16	233.330 ns	100.000 ns	00027.0482.2730				
Packet	Dir	F	Sync	DATA0	Data	CRC16	EOP	Idle	Time Stamp					
16909	-->	S	00000001	0xC3	21	01_00_02_00_02_02_00	0xAD01	233.330 ns	383.320 ns	00027.0482.2910				
Packet	Dir	F	Sync	ACK	EOP	Time	Time Stamp							
16910	<--	S	00000001	0x4B	233.330 ns	365.450 μs	00027.0482.3427							
Transaction	F	OUT	ADDR	ENDP	T	D	Data	ACK	Time	Time Stamp				
8132	S	0x67	7	0	1	00_00	0x4B	261.000 μs	00027.0485.2254					
Transaction	F	IN	ADDR	ENDP	T	D	Data	ACK	Time	Time Stamp				
8161	S	0x96	7	0	1	0 bytes	0x4B	731.467 μs	00027.0487.2914					

#### 喇叭音量设置 USB 封包图

下表为喇叭的静音控制请求函数,由 wIndex 字段来决定是要对喇叭还是麦克风作音量控制, wIndex=0x0200 表示要对喇叭作静音控制,接下来主机会送 1-byte 数据,如果值是 0x01 代表要静音。

Offset	Field	Size	Value	Description
0	bmRequestType	1	0x21	
1	BRequest	1	0x01	SET_CUR
2	wValue	2	0x0100	MUTE_CONTROL CHANNEL_0
4	wIndex	2	0x0200 0x0600	Mute for Lineout Volume   interface 0 Mute for MIC Recording Volume   interface0
6	wLength	2	0x0001	

静音控制的请求函数

Offset	Field	Size	Value	Description
0	bMute	1	0x01 0x00	TRUE (静音) FALSE

静音控制请求函数送出 1-Byte 的数据

在总线列举过程中,操作系统会先来读取 Audio Device 音量的参数(最大音量、最小音量、变化单位),当操作系统从装置读取这些参数后,才能对装置做正确音量控制,读取关于音量请求函数与装置的返回值说明如下表;从返回值的表格中可以得知,喇叭的音量最大为+6 dB,喇叭音量最小为-32dB,以上所谈到的这些值,可以依不同音效装置的需求而改变。

Offset	Field	Size	Value	Description
0	bmRequestType	1	0xA1	
1	bRequest	1	0x81 0x82 0x83 0x84	GET_CUR GET_MIN GET_MAX GET_RES
2	wValue	2	0x0200	VOLUME_CONTROL Master CH
4	wIndex	2	0x0200 0x0600	Lineout Volume   interface 0 MIC Recording Volume   interface 0 Lower Byte : Audio Control Interface (0x00)
6	wLength	2	0x0002	Volume Control

读取音量参数的请求函数

bRequest	wValue	wIndex	wVolume	说明
0x81	0x0200	0x0200	0xYYYY	读取装置目前的喇叭音量
0x82	0x0200	0x0200	0xE000	读取装置目前的喇叭音量的最小音量(-32 dB) .
0x83	0x0200	0x0200	0x0C00	读取装置目前的喇叭音量的最大音量(+6 dB) .
0x84	0x0200	0x0200	0x0100	读取装置的喇叭音量变化单位 (1)

读取喇叭音量参数请求函数的返回值范例

Transaction	F	SETUP	ADDR	ENDP	T	D	Tr	R	bRequest	wValue	wIndex	wLength	ACK	Time Stamp
6748	S	0xB4	7	0	0	D->H	C	I	0x82	0x0200	0x0200	2	0x4B	00026.7572.2352
Packet	Dir	F	Sync	SETUP	ADDR	ENDP	CRC5	EOP	Idle	Time Stamp				
13942	-->	S	00000001	0xB4	7	0	0x16	233.330 ns	100.000 ns	00026.7572.2352				
Packet	Dir	F	Sync	DATA0	Data	CRC16	EOP	Idle	Time Stamp					
13943	-->	S	00000001	0xC3	A.1 82 00 02 00 02 02 00	0xF014	233.330 ns	349.990 ns	00026.7572.2532					
Packet	Dir	F	Sync	ACK	EOP	Time	Time Stamp							
13944	<--	S	00000001	0x4B	233.330 ns	457.400 μs	00026.7572.3047							
Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time	Time Stamp					
6799	S	0x96	7	0	1	00 B0	0x4B	13.917 μs	00026.7576.0491					
Transaction	F	OUT	ADDR	ENDP	T	Data	ACK	Time	Time Stamp					
6800	S	0x87	7	0	1	0 bytes	0x4B	00026.7576.1326						

读取喇叭的最小音量的 USB 封包图

Transaction	F	SETUP	ADDR	ENDP	T	D	Tr	R	bRequest	wValue	wIndex	wLength	ACK	Time Stamp
6854	S	0xB4	7	0	0	D->H	C	I	0x84	0x0200	0x0200	2	0x4B	00026.7588.2182
Packet	Dir	F	Sync	SETUP	ADDR	ENDP	CRC5	EOP	Idle	Time Stamp				
14162	-->	S	00000001	0xB4	7	0	0x16	233.330 ns	100.000 ns	00026.7588.2182				
Packet	Dir	F	Sync	DATA0	Data	CRC16	EOP	Idle	Time Stamp					
14163	-->	S	00000001	0xC3	A.1 84 00 02 00 02 02 00	0x9614	233.330 ns	383.330 ns	00026.7588.2362					
Packet	Dir	F	Sync	ACK	EOP	Time	Time Stamp							
14164	<--	S	00000001	0x4B	233.330 ns	456.867 μs	00026.7588.2679							
Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time	Time Stamp					
6905	S	0x96	7	0	1	00 01	0x4B	13.833 μs	00026.7592.0231					
Transaction	F	OUT	ADDR	ENDP	T	Data	ACK	Time	Time Stamp					
6906	S	0x87	7	0	1	0 bytes	0x4B	518.317 μs	00026.7592.1061					

读取喇叭的音量变化单位的 USB 封包图

由于 HT82A824R 支持 48K / 44.1K 两种频率，因此当 PC 在播放不同的取样频率的音源文件内容时，会用下一个设置频率 SET\_FREQ 的请求函数来改变 Device 的输出频率设置，在 HT82A824R 的 MODE\_CTRL 寄存器 bit3 可以设置输出频率是 48K 还是 44.1K。

Offset	Field	Size	Value	Description
0	bmRequestType	1	0x22	
1	bRequest	1	0x01	SET_FREQ
2	wValue	2	0x0100	DAC : VOLUME_CONTROL Master Channel
4	wIndex	2	0x0200	D/A Feature Unit ID : 0x02
6	wLength	2	0x0003	Volume Control

SET\_FREQ 请求函数内容

Offset	Field	Size	Value	Description
0	bFreq	3	0xYYYYYY	The value is set by host 48K : 0x00BB80 44.1K : 0x00AC44

SET\_FREQ 请求函数送出 3-Bytes 的数据

程序范例:

```

(1)
;support 48K/44.1K ,2007-09-10
Request_TYPE22: ;22 01 00 01 02 00 03 00
    clr wdt
    ;set report
    MOV A,FIFO_REQUEST
    XOR A,SET_CUR
    SNZ Z
    JMP Request_TYPE22_End
    MOV A,FIFO_wIndexL
    XOR A,02H ;Endpoint 2
    SNZ Z
    JMP Request_TYPE22_End
    MOV A,FIFO_wLengthL
    XOR A,03H ;3-bytes data
    SNZ Z
    JMP Request_TYPE22_End
;Setting Frequency 48K or 44.1K
    mov a,22h
    mov nCmdIndex1,a
    jmp USB_EP0_ISR_END
Request_TYPE22_End:
    JMP SendStall0

(2)
Set_Sampling_Frequency:
;00BB80H=48k
;00AC44H=44.1K
    MOV A,FIFO_out1
    XOR A,80H
    SZ Z
    JMP Freq_48K_BYTE_1_2
    MOV A,FIFO_out1
    XOR A,44H
    SZ Z
    JMP Freq_44_1K_BYTE_1_2
    JMP ;Set_Sampling_Frequency_End

Freq_48K_BYTE_1_2:
    MOV A,FIFO_out2
    XOR A,0BBH
    SNZ Z
    JMP Set_Sampling_Frequency_End
    MOV A,FIFO_out3
    XOR A,00H
    SNZ Z
    JMP USB_EP0_OUT_TOKEN_End
    ;Set Sampling Frequency=48KHz
    CLR MODE_CTRL.FREQ
    JMP ;Set_Sampling_Frequency_End
    
```

```

Freq_44_1K_BYTE_1_2:
MOV A,FIFO_out2
XOR A,0ACH
SNZ Z
JMP          ;Set_Sampling_Frequency_End
MOV A,FIFO_out3
XOR A,00H
SNZ Z
JMP USB_EP0_OUT_TOKEN_End
          ;Set Sampling Frequency=44.1KHz
SET  MODE_CTRL.FREQ
JMP          ;Set_Sampling_Frequency_End
    
```

要改变计算机 USB 音效装置的音量大小，除了直接用 Windows 上的内建应用程序音量控制改变音量外，另外一种方式也可以用 USB 规范中的 HID (Human Interface Device)，直接在使用者的装置端以按键的方式，直接改变音量，要用此方法在装置端改变音量，必须撰写正确的 HID 类与报告描述语言，以下就以喇叭的音量控制来作说明。首先在如下的 HID 类 Interface2\_descriptor 得知，是使用 ReportID 1 是用来传输 Media Key，也就是 Volume Increment / Decrement、Mute 等功能；而 ReportID 3 是用来传输其它类型的数据，也就是使用者自定的功能。当有使用 Report ID 时，报告描述符的长度为 DW 0003FH (63 个 Bytes)。HID\_end\_point\_descriptor 是在 Interface2 申明一个端点描述符，此端点描述为输入端点 1，而主机来询问我们传送数据的时间间隔为 DW 03F30H 的 30H=48ms。由于使用同一个输入端点 1，所以当要传送 Media Key 或其它类型的数据时，装置必须用不同的 Report ID 来区分它们，这样才会知道装置送什么样的数据给 PC，这就是使用 Report ID 的原因。

描述语言范例:

```

Interface2_descriptor:
HID_class:
    DW 00409H ;INTERFACE descriptor , Size of this descriptor
    DW 00002H ;Index of this string , index of this interface
    DW 00301H ;HID , 1 endpoint
    DW 00000H ;Unused , Non-Boot Device
    DW 03F00H ;null string
HID_Desc:
    DW 02109H ;HID , Size of this descriptor
    DW 00110H ;HID spec rev #1.10
    DW 00100H ;bNumDescriptor , bCountryCode
    DW 03F22H ;Report Descriptor
          ;Use ReportID , Report ID 1 = Volume HID control
          ;Report ID 3 = Transform Other Data
IF UseReportID
    DW 0003FH ;63 bytes
ELSE
    DW 0001FH ;31 bytes
ENDIF
    
```

```

HID_end_point_descriptor:
    DW 00507H ;Endpoint descriptor , Length of this descriptor
    DW 00381H ;Interrupt , Endpoint 1 In direction
    DW 00008H ;wMaxPacketSize = 8 Bytes
    DW 03F30H ;48ms Interval
end_config_desc_table:
    
```

hid\_report\_desc\_table 的标签是 Media Key 的完整报告描述符。一个 Report ID 占用一个 Byte，现在来说明 Media Key 的报告描述符；在 USB HID Usage Tables 规范中可以查询，Volume Increment 的 Usage ID 是 E9、Volume Decrement 的 Usage ID 是 EA 及 Mute 的 Usage ID 是 E2，而前面的 09 代表后面的 Usage ID 是一个 Byte，如果后面的 Usage ID 是 2 个 Bytes，则 Usage ID 前面的 Byte 为 0A。值 DW 00175H 所表示的意思是所输入的 Media Key 占多少位，以 00175H 为例子，它表示一个 Key 占用一个位。第二个值为 DW 00295H 表示的意思是上面使用到几个 Bit，由于上面使用到 Volume Increment / Decrement 两种功能，所以在 这里为 00295H，占用两个 Bit。综上说明我们使用 Media Key，目前使用了三个键，而这三个键分别为 Volume Increment / Decrement / Mute，这三个键在一个 Byte 里占了三个 Bit，其功能与位对应关系如下表所示。

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Unused	Unused	Unused	Unused	Unused	Mute	Volume Decrement	Volume Increment

而且先申明的功能是先占用低位。以我们的例子来看，上面先申明 Volume Increment，所以它在位 0，而 Volume Decrement 则在位 1，以此类推。此时，有五个位未使用到 (Unused) 也必须将报告描述符完整描述，构成一完整字节。

我们在传送 Media Key 的第一个 Byte 为 Report ID=1，即 01H，而第二个 Byte 则为我们在报告描述符所描述的按键功能，举例来说，若为静音时，则第二个 Byte 则为 04H，若为音量增加键则第二个 Byte 为 01H，若为音量减少键则第二个 Byte 为 02H。

描述语言范例:

```

hid_report_desc_table:
    DW 00C05H           ://Usage Page(Consumer)
    DW 00109H           ://Usage Page(Consumer Control)
    DW 001A1H           ://Collection(Application)

    IF UseReportID
    DW 00185H           ://Report_ID(01)
    ENDIF
    DW 00015H           ://Logic Minimum(0)
    DW 00125H           ://Logic Maximum(1)
    DW 03F09H           ://Usage(Volume Increment)
    DW 03FE9H
    DW 03F09H           ://Usage(Volume Decrement)
    DW 03FEAH
    DW 00175H           ://Report Size(1): Data Length (1)bit
    DW 00295H           ://Report Count(2): Number of Data(INC,DEC)
    DW 02A81H           ://Input(Data,Variable,Absolute,No_Wrap,No_Preferred)
    DW 03F09H           ://Usage(Mute)
    DW 03FE2H
    DW 00195H           ://Report Count(1)
    DW 02E81H           ://Input(Data,Variable,Relative,No_Wrap,No_preferred)
    DW 00595H           ://Report Count(5)
    DW 00181H           ://Input(Constant)
    DW 03FC0H           ://End Collection
end_hid_report_desc_table:
    
```

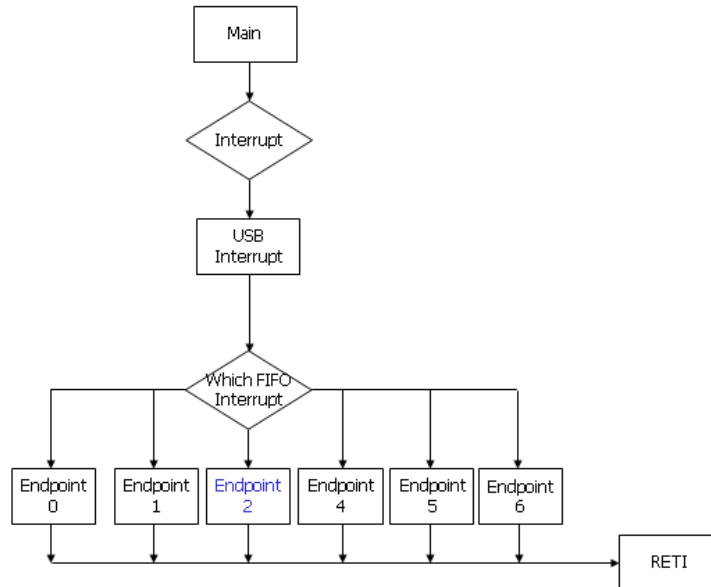
### 盛群 USB Audio 软件技术

盛群 USB Audio MCU 的 SIE 是盛群自行设计的，其设计理念除了兼容性外，也考虑到让 USB 软件工程师更容易撰写 USB 底层的程序，现就针对比较重要的部分以文字说明配合流程图如下：

- USB 中断处理流程

当 USB SIE 的某一个端点有数据需传输时，会产生 USB 中断，进入中断子程序后再由寄存器判断是由那个端点产生的中断，然后再进入相对应的程序代码处理。

说明: Endpoint 2 为 Isochronous out 端点



程序范例:

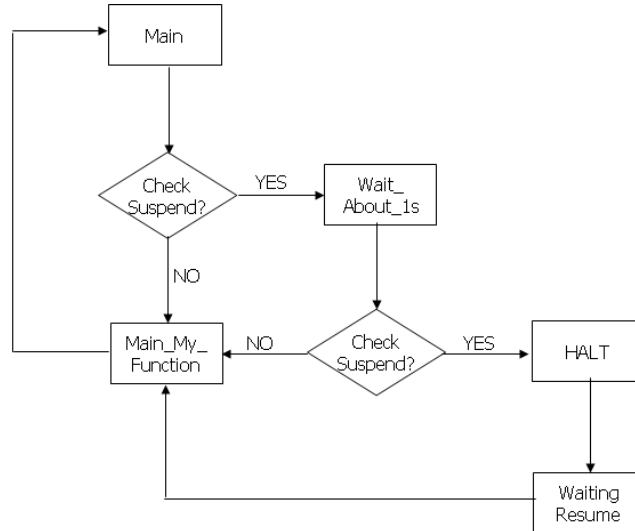
Check\_Access\_FIFO:

```

clr wdt
SZ     USR.@USR_EP0F
JMP    USB_EP0_ISR
SZ     USR.@USR_EP1F
JMP    USB_EP1_ISR
SZ     USR.@USR_EP2F
JMP    USB_EP2_ISR
SZ     USR.@USR_EP4F
JMP    USB_EP4_ISR
SZ     USR.@USR_EP5F
JMP    USB_EP5_ISR
SZ     USR.@USR_EP6F
JMP    USB_EP6_ISR
JMP    USB_ISR_END
  
```

• 主程序

在主程序中，会检查 SUSPEND (悬空模式) 标志位，如果 SUSPEND 标志位被设为“1”，会先延迟 1 秒钟后，再检查一次 SUSPEND 标志位，如果 SUSPEND 标志位仍为“1”，把相关标志设置好后，进入 HALT 模式，等待 PC 下 Resume 信号过来；如果延迟 1 秒后，SUSPEND 标志已经清除，则继续执行主程序。



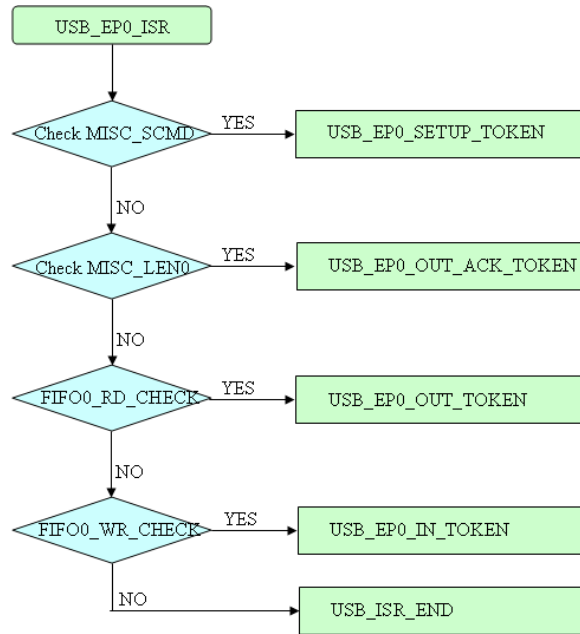
程序范例:

```

Start:
    call System_Initial
Main:
    SNZ USC.@USC_SUSP      ;check SUSPEND ?
    JMP Main_My_Function
    call wait_about_1s
    SNZ USC.@USC_SUSP
    JMP Main_My_Function
ToSuspend_again:
    SNZ USC.@USC_SUSP      ;check SUSPEND ?
    JMP Main_My_Function
    clr wdt
    clr TMR1C.4
    clr USB_LED_ON
    clr UCC.@UCC_USBCKEN
  
```

• 端点 0 主流程

进入 USB 中断后，判断是端点 0 (Control Pipe) 后，进入端点 0 的子程序，接下来判断是 SETUP Token、OUT ACK(Zero-Length)、IN Token(Control Read)或 OUT Token(Control Write)。

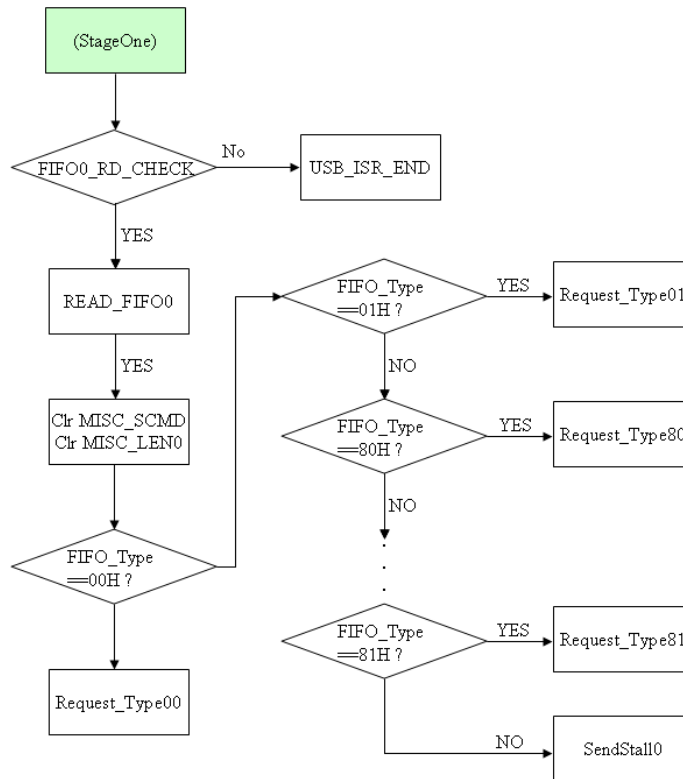


程序范例:

```

USB_EP0_ISR:
SZ     MISC.@MISC_SCMD           ;check setup token
JMP    USB_EP0_SETUP_TOKEN
SZ     MISC.@MISC_LEN0          ;check out ack token
JMP    USB_EP0_OUT_ACK_TOKEN
CALL   FIFO0_RD_CHECK
SZ     bFlag_FIFO_Ready
JMP    USB_EP0_OUT_TOKEN
CALL   FIFO0_WR_CHECK
SZ     bFlag_FIFO_Ready
JMP    USB_EP0_IN_TOKEN        ;else is in token
CLR    USR.@USR_EPOF           ;Fix OHCI Volume
JMP    USB_EP0_ISR_END
  
```

- 如果是端点 0 中断而且判断是 SETUP Token 时，则开始分析 Setup Command (StageOne), Setup Command 共有 8 Bytes，先解析第一个 Byte，也就是先解析 Request\_Type，再跳到对应的函数去执行，若没有支持的 command 则跳至 SendStall0。



程序范例:

StageOne:

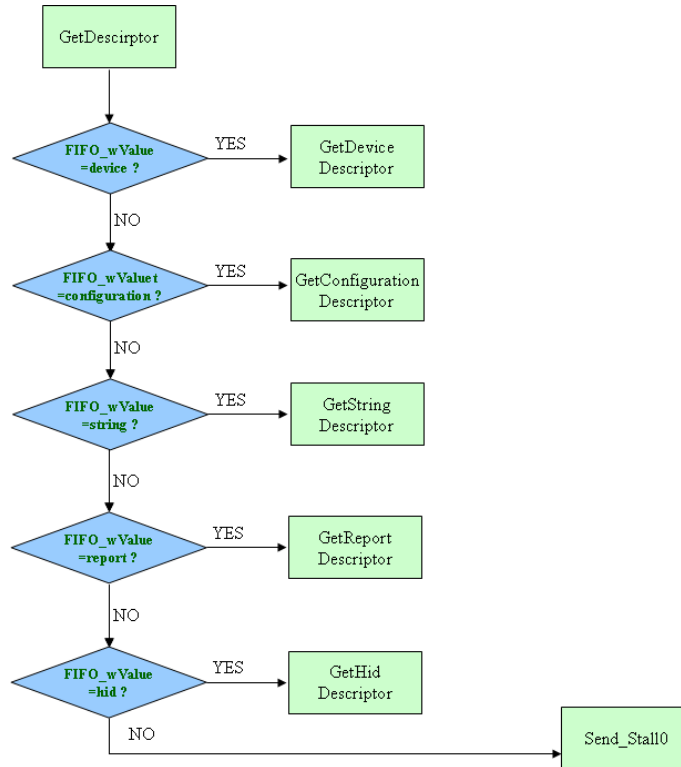
```

CALL FIFO0_RD_CHECK
SNZ bFlag_FIFO_Ready
JMP StageOne_End           ;the EPO FIFO RD is not ready 没有 data 进来
CALL Read_FIFO0           ;Read EPO Command
clr MISC.@MISC_SCMD
clr MISC.@MISC_LEN0
CLR USR.@USR_EP0F        ;Fix OHCI Volume
Clr wdt
nop
MOV A,FIFO_TYPE
XOR A,00H
SZ Z                     ;FIFO_TYPE=00H
JMP Request_Type00
MOV A,FIFO_TYPE
XOR A,01H
SZ Z                     ;FIFO_TYPE=01H
JMP Request_Type01
MOV A,FIFO_TYPE
XOR A,02H
SZ Z                     ;FIFO_TYPE=02H
JMP Request_Type02
MOV A,FIFO_TYPE
XOR A,80H
SZ Z                     ;FIFO_TYPE=80H
JMP Request_Type80
    
```

```

MOV A,FIFO_TYPE
XOR A,81H
SZ Z           ;FIFO_TYPE=81H
JMP Request_Type81
MOV A,FIFO_TYPE
XOR A,82H
SZ Z           ;FIFO_TYPE=82H
JMP Request_Type82
    
```

- 在 Bus Enumeration 的过程，主机会来读取装置的描述符，装置再根据 PC 所要求不同类型的描述符传给主机。



程序范例:

```

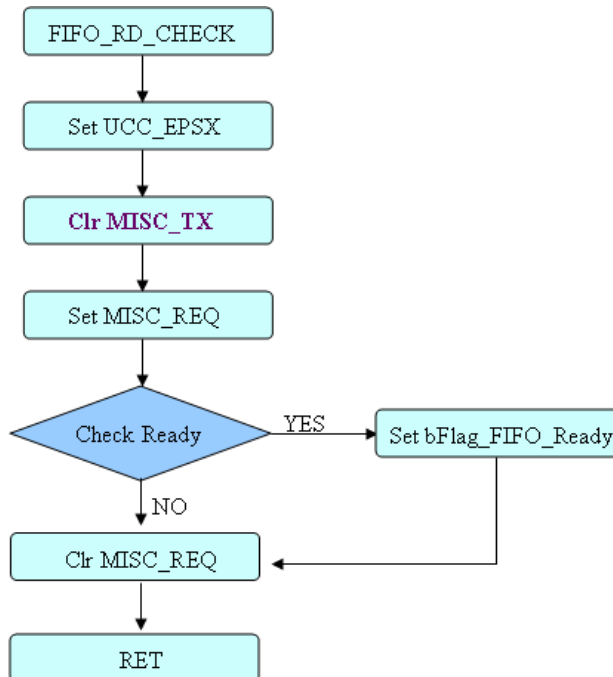
GetDescriptor:
clr wdt
CLR bFlag_RD_HTable
CLR bFlag_wait_control_out
MOV A,FIFO_WvalueH      ;80 06 00 01
XOR A,device
SZ Z
JMP GetDeviceDescriptor
MOV A,FIFO_WvalueH      ;80 06 00 02
XOR A,configuration
SZ Z
JMP GetConfigurationDescriptor
MOV A,FIFO_WvalueH      ;80 06 00 03
XOR A,string
SZ Z
JMP GetStringDescriptor
;-----
;Then test for HID class Descriptor
    
```

```

;-----
MOV A,FIFO_WvalueH      ;:1 06 00 22
XOR A,report
SZ Z
JMP GetReportDescriptor

MOV A,FIFO_WvalueH      ;:1 06 00 21
XOR A,HID
SZ Z
JMP GetHIDDescriptor
JMP SendStall0          ;:can't parser
    
```

- 当有数据要与主机做数据传输时(控制型端点与中断型端点), Firmware 需先检查 FIFO 是否已经准备好, 然后才能接着做读取或写入 FIFO 的动作。



程序范例:

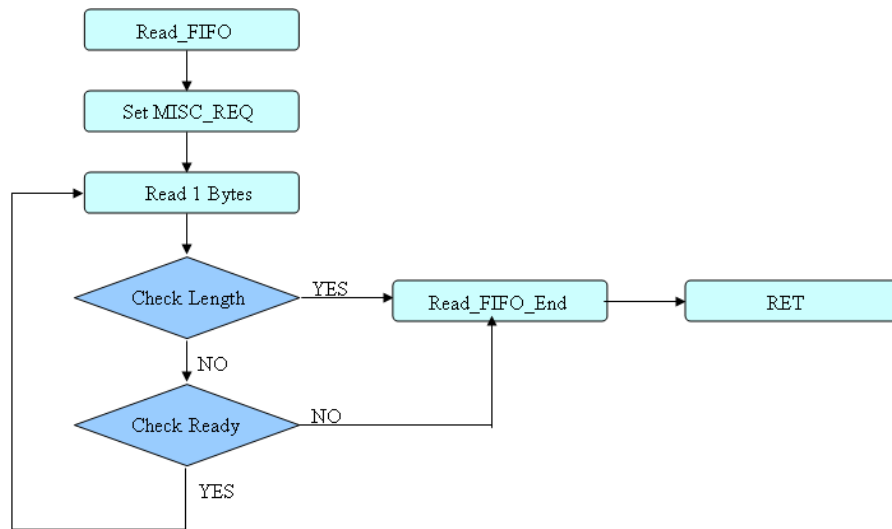
```

FIFO_CHECK:
    Clr wdt
    MOV FIFO_TEMP,A
    MOV A,USB_MISC
    MOV MP1,A
    MOV A,R1
    AND A,11111000b
    OR A,FIFO_TEMP
    MOV R1,A
    CALLDelay_3us
    SET R1.@MISC_REQ          ;:set request
    CALLDelay_28us
    SET bFlag_FIFO_Ready
    SNZ R1.@MISC_Ready
    CLR bFlag_FIFO_Ready     ;:if MISC.Ready = 1 -> bFlag_FIFO_Ready = 1
    SET bFlag_FIFO_LEN0
    SNZ R1.@MISC_LEN0
    CLR bFlag_FIFO_LEN0
    
```

```

clr MISC.@MISC_REQ
clr wdt
RET

```

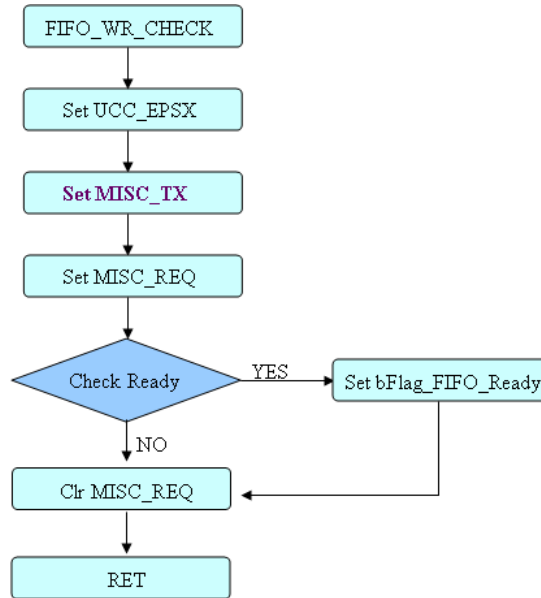


程序范例:

```

Read_FIFO_Loop:
MOV A,FIFO_TEMP
MOV MP1,A
MOV A,R1
MOV R0,A
INC FIFO_SendLen
INC MP0
MOV A,FIFO_SIZE
XOR A,FIFO_SendLen
SZ Z ;l=FIFO_SIZE=FIFO_SendLen
JMP Read_FIFO_End
MOV A,USB_MISC
MOV MP1,A
CALLDelay_28us
SZ R1.@MISC_Ready
JMP Read_FIFO_LOOP
JMP Read_FIFO_EndRead FIFO Flow

```

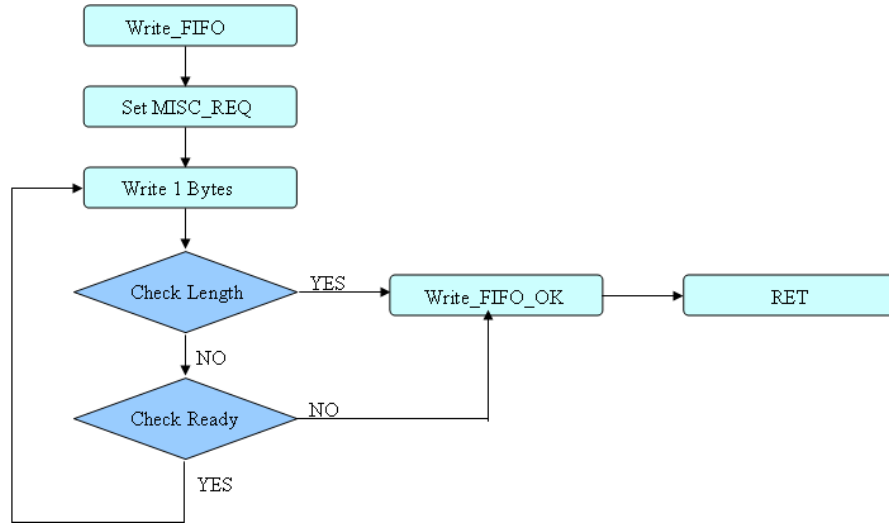


程序范例:

FIFO\_CHECK:

```

clr wdt
MOV FIFO_TEMP,A
MOV A,USB_MISC
MOV MP1,A
MOV A,R1
AND A,11111000b
OR A,FIFO_TEMP
MOV R1,A
CALLDelay_3us
SET R1.@MISC_REQ ;set request
CALLDelay_28us
SET bFlag_FIFO_Ready
SNZ R1.@MISC_Ready
CLR bFlag_FIFO_Ready ;if MISC.Ready = 1 -> bFlag_FIFO_Ready = 1
SET bFlag_FIFO_LEN0
SNZ R1.@MISC_LEN0
CLR bFlag_FIFO_LEN0
clr MISC.@MISC_REQ
clr wdt
RETWrite FIFO Check Flow
  
```



程序范例:

```

Write_FIFO_Loop:
  clr wdt
  MOV A,FIFO_SendLen
  XOR A,00H
  SZ Z
  JMP Write_FIFO_End
  MOV A,FIFO_TEMP
  MOV MP1,A
  MOV A,R0
  MOV R1,A
  DEC FIFO_SendLen
  MOV A,FIFO_SendLen
  XOR A,00H
  SZ Z
  JMP Write_FIFO_End      ;FIFO_SendLen=0 代表传完了
  INC MP0
  MOV A,USB_MISC
  MOV MP1,A
  call Delay_28us
  SZ R1.@MISC_Ready
  JMP Write_FIFO_Loop
Write_FIFO_End:
  Clr wdt
  JMP Write_FIFO_OKWrite FIFO Flow
  
```

## 版本记录

### 版本: V1.10

修改日期: 2011 年 08 月 29 日

修改内容: 新增第五页 Microsoft<sup>®</sup> WHQL Audio Requirement 单元的表格内容的“注意事项”说明。