

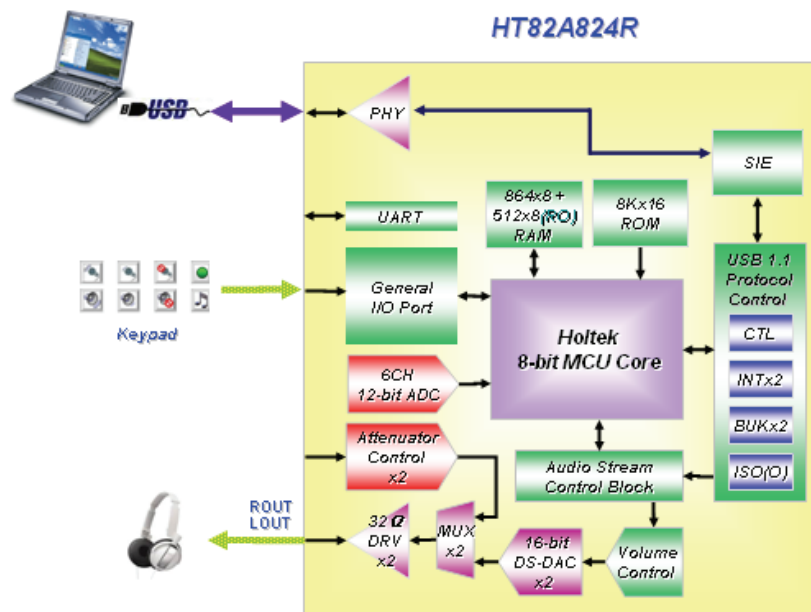
HT82A824R USB Speaker Application

D/N : HA0277E

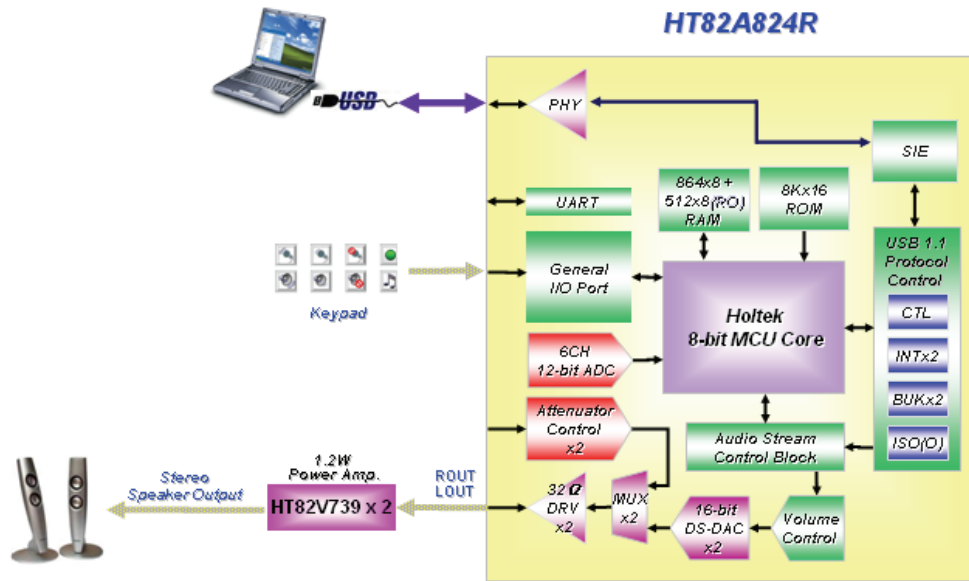
Brief Description

The HT82A824R is an enhanced USB speaker MCU. This text introduces how to use the HT82A824R to design for USB speaker products as well as discussing USB audio class criterion and programming structure description.

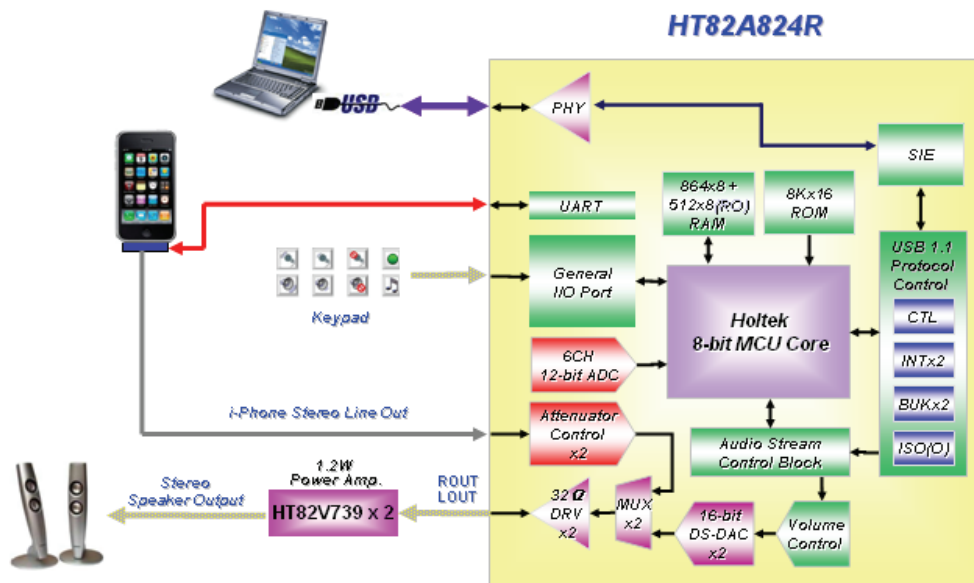
HT82A824R USB Headphone Application Diagram



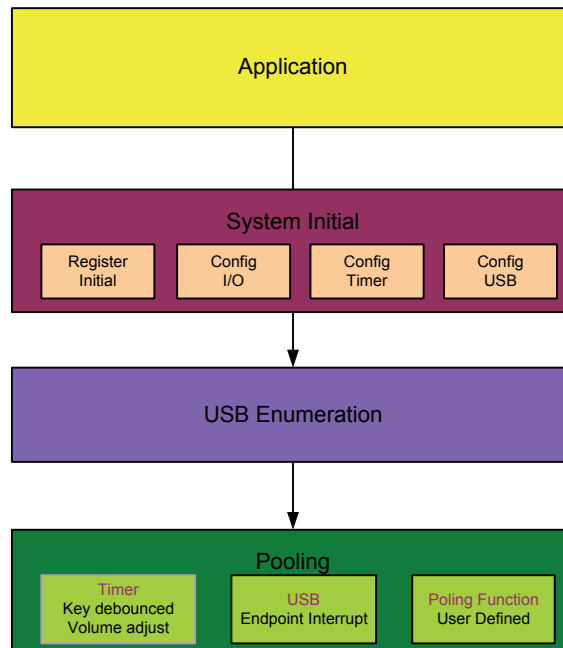
HT82A824R USB Speaker Application Diagram



HT82A824R USB/i-Phone Docking Speaker Application Diagram



Holtek USB Audio Programming Structure



Register Initial: Register initialised

Config I/O: I/O initialised

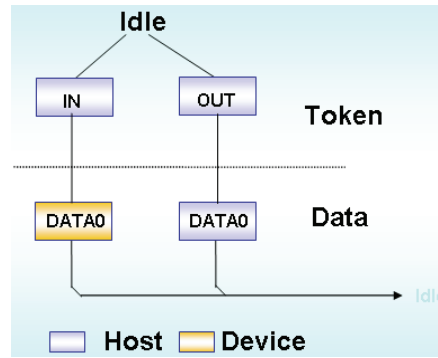
Config Timer: Corresponding timer registers initialised

Config USB: USB corresponding registers initialised. When set the USB function starts operation

USB Enumeration: Enumerate for USB

When the enumeration finished, there are three modules which the users can revise: timer interrupt, USB endpoint interrupt and poling function.

The HT82A824R USB audio uses full speed isochronous transmission mode, each frame is 1ms (USB criterion: $1\text{ms} \pm 500\text{ns}$). In terms of different frequencies, it transmits different voice data volumes. For example, when the audio frequency is 2-CH (48kHz /16-bit), the data volume is 192 bytes in 1ms. Based on this example, the Holtek USB phone microcontroller has two internal buffers, a Ping-Pong structure and the data volume is 192 bytes respectively. In all, it is 384 bytes. It uses SOF, start of frame, as a control signal for switching between the two buffers. As the table shows, USB isochronous in or out transmission does not need an ACK signal. It is different to the other three transmission modes, control, interrupt or bulk. That is to say, although the data has errors, it will not be transmitted again.



Isochronous In/Out Transmission

The HT82A824R has an isochronous out endpoint. Isochronous out is used to transmit the voice data in PC play media to the device. The voice data format is PCM. In case of audio frequencies it is 2-CH (48kHz/16-bit) and a frame transmits 192 bytes data so the Ping-Pong buffer is 384 bytes in all for this endpoint. The following illustration shows the descriptor's definition for the play media format. The descriptor defines the play frequency, a frame data volume, endpoint number and endpoint direction, etc. The endpoint number is set as 2. As follows, descriptive language supports both of the audio frequencies 48 kHz /16-bit/2-CH and 44.1kHz/16-bit/2-CH.

Its isochronous endpoint is implemented by hardware. For the programming engineer, it dose not waste amount of resources to manipulate the speaker voice data which can largely save time on software development. The customer can analysis the digital data by software by putting the data into RAM bank 1~4 using the internal hardware DMA.

```
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

To obtain the Microsoft's USB audio device logo, in the audio segment, it must meet the following two criteria:

- Sampling frequency: The USB audio device output sampling frequency must support 44.1k and 48k (or an integral multiple frequency)
- The output audio quality must meet the following spec (USB Speaker).

For premium desktop implementations:

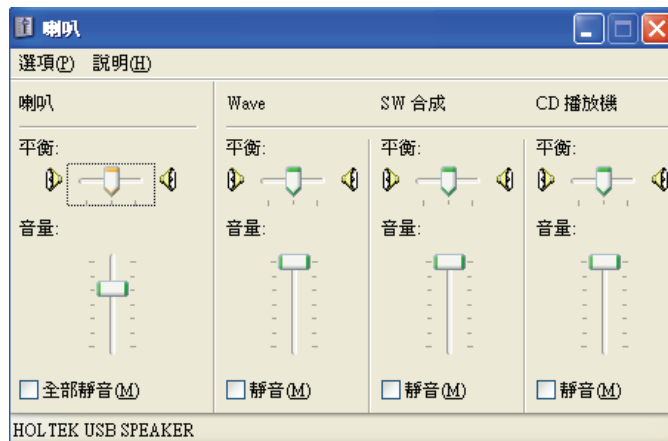
Device Type	Requirement	Value	Frequency range at 48kHz and above
Analog Speaker Output Jack (Example: 125mW into 8 Ohm load)	THD+N	≥ 75dB FS	[20Hz, 20kHz]
	Dynamic range with signal present	≥ 90dB FS A-weight	[20Hz, 20kHz]
	Magnitude Response	≤ ±.25dB ripple (.5dB peak to peak delta), 1dB at upper band edge, 3dB at lower band edge	[20Hz, 20kHz]
	Sampling frequency accuracy	0.02%.	
	Line output cross-talk	≥ 60dB	[20Hz, 15kHz]
	Full scale output voltage	≥ 1Vrms	
	Dynamic range with signal present during system activity	≥ 90dB FS A-weight	[20Hz, 20kHz]
Interchannel phase delay	30 degrees or 12.5 microseconds, whichever is greater	[20Hz, 20kHz]	

Note: The HT82A824R does not support Audio WHQL certification if the product has common speaker and external earphone output. For products with an earphone output, then for testing WHQL Audio, they will require an Audio Fidelity test. Here the Render Power Transition test item will result in a failed WHQL test. If the product does not have an external jack, then it does not have this test item.

USB Sound Effect Device Description

Volume Control

Speaker volume control : There is a volume control window in the microsoft operation system. When the user drags the control key, the USB adjusts the volume using the next endpoint 0 SET_CUR request function (SET_CUR request function content is shown in the following table), the volume set point loaded into the PC will be reflected into the IC digital volume control register (USVC), hence it can change the volume when playing.



The following table shows the speaker volume controlling request function. Whether controlling the speaker volume or the microphone volume, it is implemented using the SET_CUR character, wIndex. When the wIndex is equal to 0x0200, it means it is controlling the speaker volume.

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 Request Function Content

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

SET_CUR Request Function Sends 2-Bytes Data

When controlling the speaker volume, the request function is followed by 2-bytes of data. This data is the volume value. As shown in the following table, on the basis of the range set in the IC volume control register, the speaker volume controls the high bit. In addition, in bus enumeration, the operating system will set the speaker initialised volume value using the SET_CUR request function, generally to set the medium value.

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

Speaker Volume Control High Bit (Low bits all are 00H)

Transaction	F	SETUP	ADDR	ENDP	T	D	TP	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	CRC6	EOP	Idle	Time Stamp						
16906	-->	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	355.450 μs	00027.0482 3427									
Transaction	F	OUT	ADDR	ENDP	T	Data	ACK	Time	Time Stamp							
8132	S	0x87	7	0	1	00 00	0x4B	261.000 μs	00027.0485 2254							
Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time	Time Stamp							
8161	S	0x96	7	0	1	0 bytes	0x4B	731.467 μs	00027.0487 2914							

Speaker Volume Set USB Package Chart

The following table shows the speaker mute controlling request function. Whether controlling the speaker volume or the microphone volume is implemented using the SET_CUR character, wIndex. When wIndex is equal to 0x0200, it means controlling the speaker sound mute. Then the host computer will transmit 1-byte of data, when the value is equal to 0x01, it means it is mute.

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	

Mute Controlling Request Function

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

Mute Controlling Request Function Transmits 1-Byte Data

For bus enumeration, the operating system will firstly read the audio device volume parameter (maximum volume, minimum volume, change unit), after reading these parameter from the device, the operating system just controls the device volume exactly. The following table shows the description for reading the return value using the volume request function and the device. As shown in the table, the maximum volume is up to +6dB. The minimum is -32dB. All the values in question are changed according to the sound effect the device needs.

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

Request Function Read Volume Parameter

bRequest	wValue	wIndex	wVolume	Description
0x81	0x0200	0x0200	0xYYYY	Read the device current speaker volume
0x82	0x0200	0x0200	0xE000	Read the device current speaker minimum volume(-32dB)
0x83	0x0200	0x0200	0x0C00	Read the device current speaker maximum volume(+6 dB)
0x84	0x0200	0x0200	0x0100	Read the device current speaker volume change unit(1)

Request Function Read the Volume Parameter Return Value Example

Transaction	F	SETUP	ADDR	ENDP	T	D	TP	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	00	00	0x9F014	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	8D	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							

Read Speaker Minimum Volume USB Package Chart

Transaction	F	SETUP	ADDR	ENDP	T	D	TP	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	00	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	455.867 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						

Read Speaker Volume Change Unit USB Package Chart

Because the HT82A824R supports both 48K and 44.1K frequencies, the PC uses the next preset frequency SET_FREQ request function to change the device output frequency set when playing the voice document content with different sampling frequencies. It is possible to set the output frequency to be 48K or 44.1K by bit 3 in the MODE_CTRL controller.

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 Request Function Content

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

SET_FREQ Request Function Sends 3-BytesData

Programming Example:

(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

```

To change the computer USB sound effect device volume except using the application program internally to set up the window, another way is to use the Human Interface Device in the USB criterion to directly change the volume by using keys in the user device end. To change the volume by this way at the device end, the user must write the exact HID category and record description language. The following illustration shows how the speaker volume is controlled. Firstly, known as the following HID category Interface2_descriptor, it uses report ID 1 to transmit the media key, namely, volume increment or decrement, mute and so on. While report ID 3 is used to transmit the other types of data, namely the functions the user defines. When using report ID, the descriptor length is DW 0003FH (63-bytes). HID_end_point_descriptor states an endpoint descriptor in interface 2. This endpoint input is described as endpoint 1. The host computer inquires the data that we transmit, the time intervals are 48ms (30H in DW 03F30H). Because it is using the same input endpoint 1, when transmitting the media key or other types of data, different report IDs must be used to distinguish them. Therefore the user will know what type of data the device transmits to the PC, which is the reason for using report ID.

Descriptive Language Example:

```

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:

```

The hid_report_desc_table label is the full media key record descriptor. A report ID takes up one type. Now introduce the media key record descriptor. In the USB HID usage tables criterion we can find out that the volume increment usage ID is E9, the volume decrement's is EA and the mute's E2. The leading 09 means the following usage ID is one byte. When the following usage ID is 2 bytes, the usage ID leading byte is 0A.

The value DW 00175H means the number of bits the input media key uses. Taking 00175H as an example, it means that one key takes up one bit. The second value is DW 00295H. It means how many bits the above uses. Because the volume increment and volume decrement are the above-used, it is 00295H here and takes up 2 bits. We use 3 media keys at present, namely, volume increment, volume decrement or mute, they take up 3 bits in a byte. Matchup of its function and bit is as follows.

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

The function declared first in advance takes up the low bits. As shown in this example, a volume increment has been declared in advance, so it is bit 0 while volume decrement is bit 1, etc. Temporally, the 5 unused bits must fully describe the record descriptor to make a full byte.

When transmitting the media key first byte, the report ID is equal to 1, namely 01H. While transmitting the second byte, the report ID is the key function described in record descriptor. For example, when it is mute, the second byte is 04H. When it is volume increment, the second byte is 01H. When it is volume decrement, the second byte is 02H.

Descriptive Language Example:

```

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:
    
```

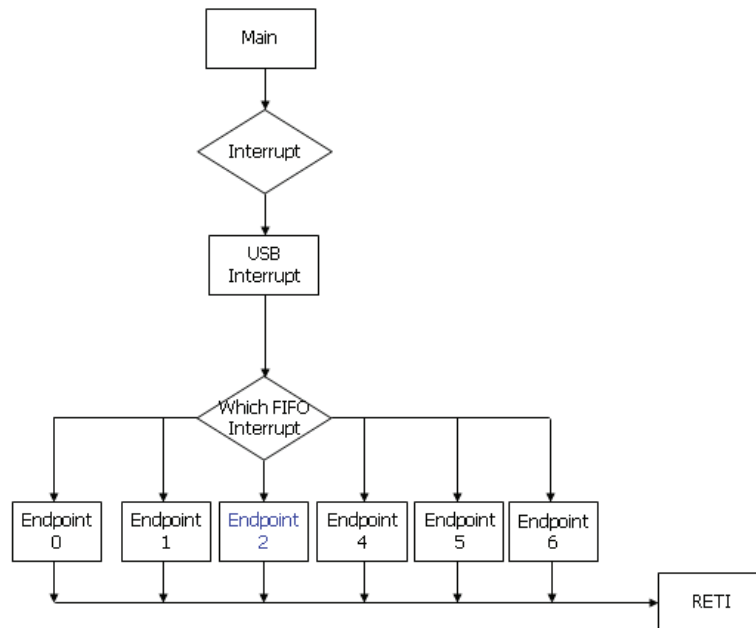
Holtek USB Audio Software Technology

The Holtek USB audio MCU SIE is Holtek's own design. The design concept not only includes compatibility but also considers how to make it easier for software engineer to compose the USB bottom layer procedure. This section will explain the most important parts with a written description and corresponding flow chart.

- USB Interrupt Manipulation Flow

When one USB SIE endpoint needs to transmit data, a USB interrupt will occur, it will enter the interrupt subroutine and a register determines which endpoint has generated the interrupt. Then enter the corresponding program sequence for handling.

Note: Endpoint 2 is an isochronous out endpoint.



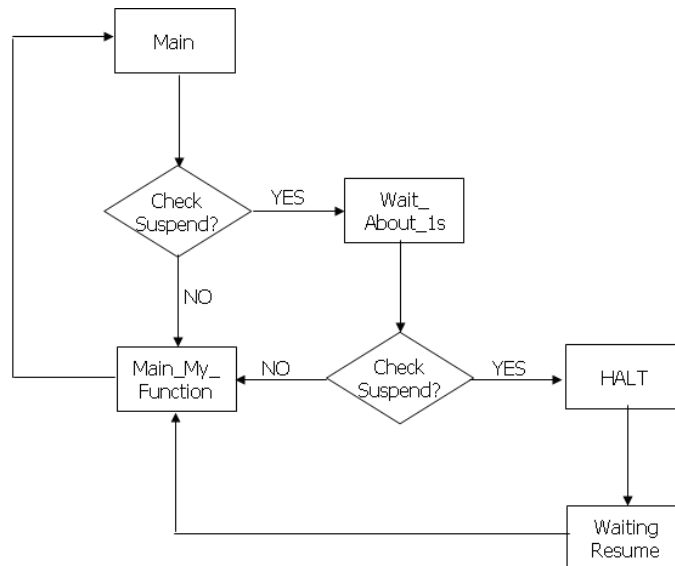
Programming Example:

```

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
  
```

- Main Program

In the main program, it will check the SUSPEND flag bit. When the SUSPEND flag bit is set high, after 1s, it checks the SUSPEND flag bit again. When the SUSPEND flag bit is still high, after setting corresponding flag bits, it enters the halt mode, waiting for the next PC resume signal. After a delaying of 1s, the SUSPEND flag bit is cleared and it continues operation with the main program.



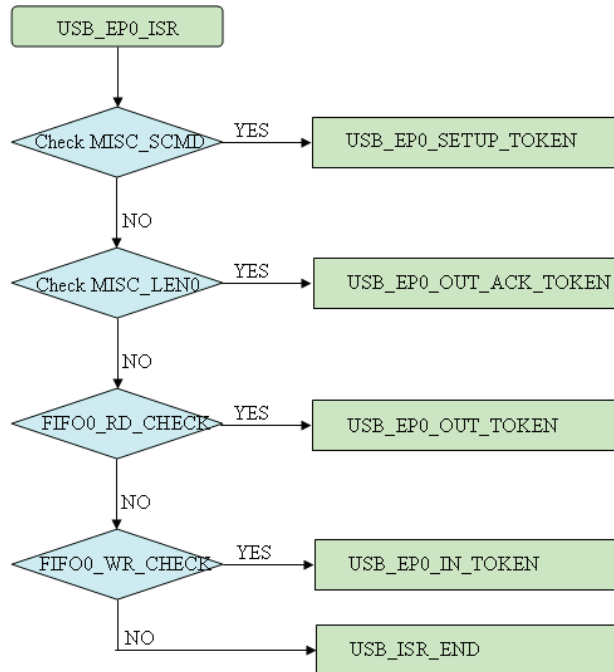
Programming Example:

```

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
  
```

- Endpoint 0 Main Flow

After entering the USB interrupt subroutine, the device determines whether it is endpoint 0 (control pipe), enters the endpoint 0 subroutine, then determines whether it is a SETUP Token, OUT ACK (Zero-Length), IN Token(Control Read) or OUT Token(Control Write).

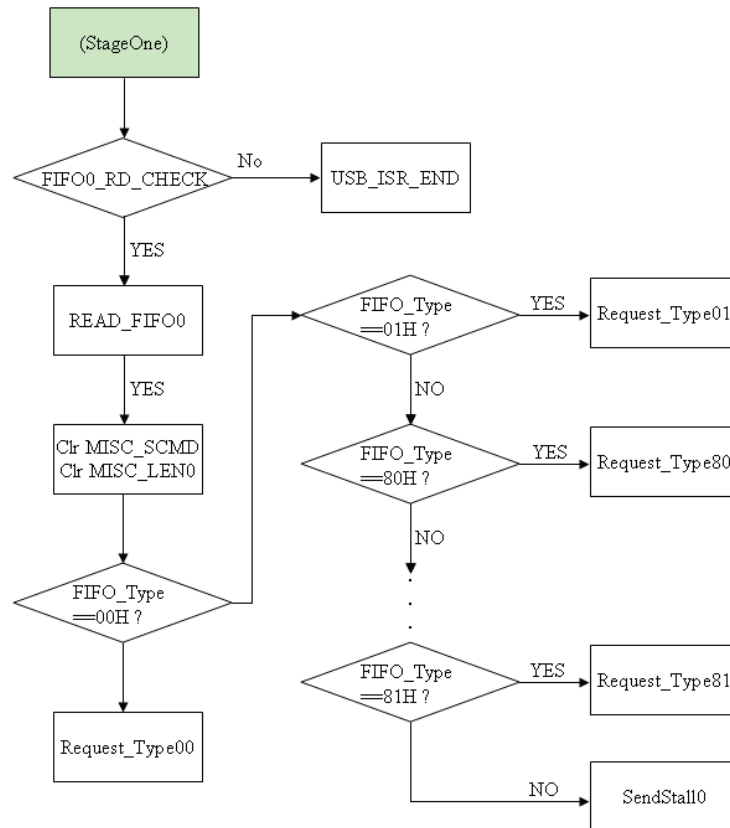


Programming Example:

```

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
  
```

- If it is endpoint 0 interrupt and it is SETUP Token, then the device begins to analyse the setup command (Stage One). There are 8 bytes in the setup command. Firstly it analyses the request_type, then jumps to the corresponding function. If there is no command support, it jumps to SendStall0.



Programming Example:

StageOne:

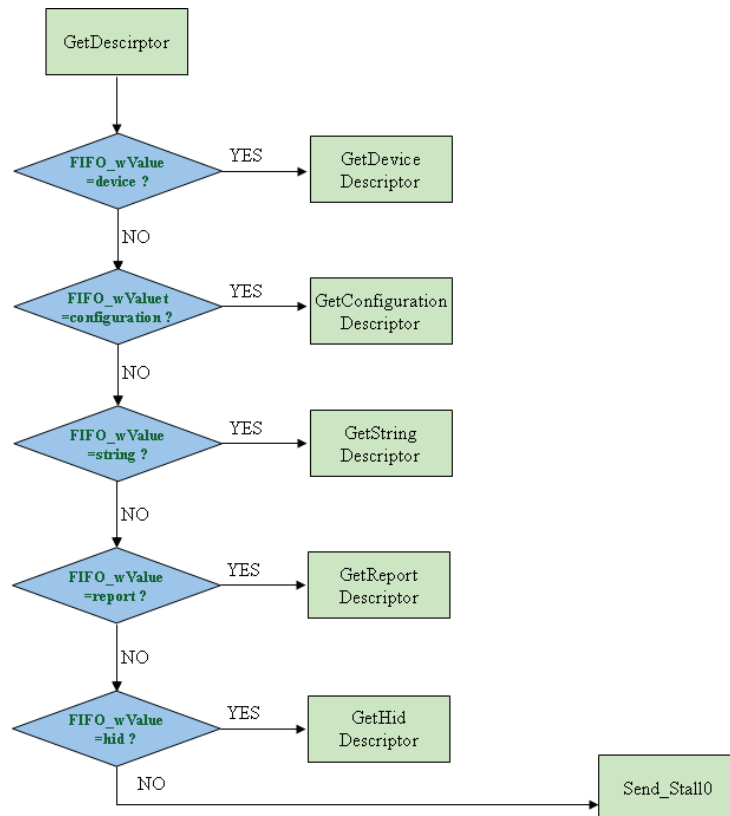
```

CALL FIFO0_RD_CHECK
SNZ bFlag_FIFO_Ready
JMP StageOne_End ;the EP0 FIFO RD is not ready no data is comin
CALL Read_FIFO0 ;Read EP0 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
    
```

- In bus enumeration, the host computer reads the descriptor from the device. The device transmits it to the host according to the different PC request states.



Programming Example:

```

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
    
```

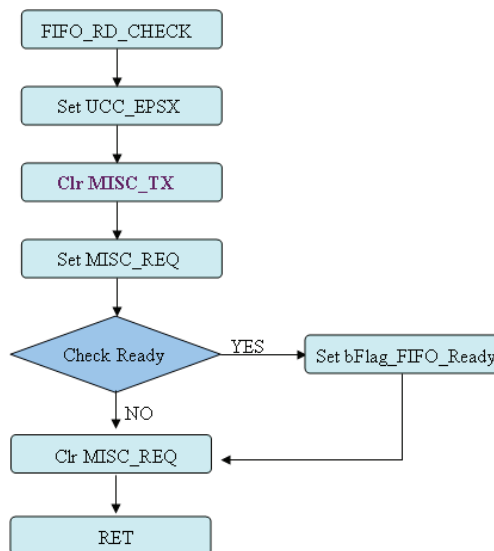
```

;-----
;Then test for HID class Descriptor
;-----
MOV A,FIFO_WvalueH      ;81 06 00 22
XOR A,report
SZ Z
JMP GetReportDescriptor

MOV A,FIFO_WvalueH      ;81 06 00 21
XOR A,HID
SZ Z
JMP GetHIDDescriptor
JMP SendStall0          ;can't parser

```

- When transmitting data with the host computer (control pipe and interrupt pipe), the firmware must check whether the FIFO is ready after which it will read or write to the FIFO.

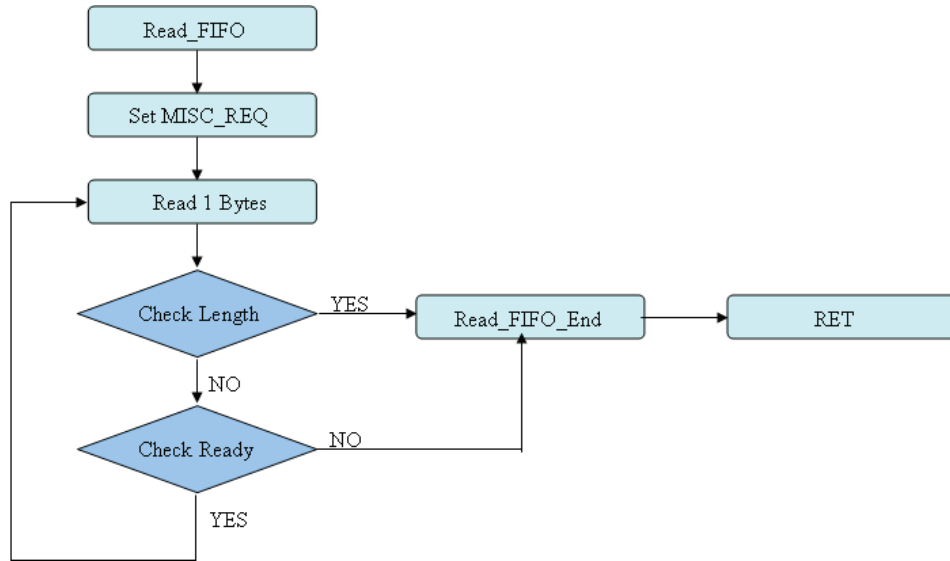


Programming Example:

```

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
  CALL Delay_3us
  SET R1.@MISC_REQ          ;set request
  CALL Delay_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

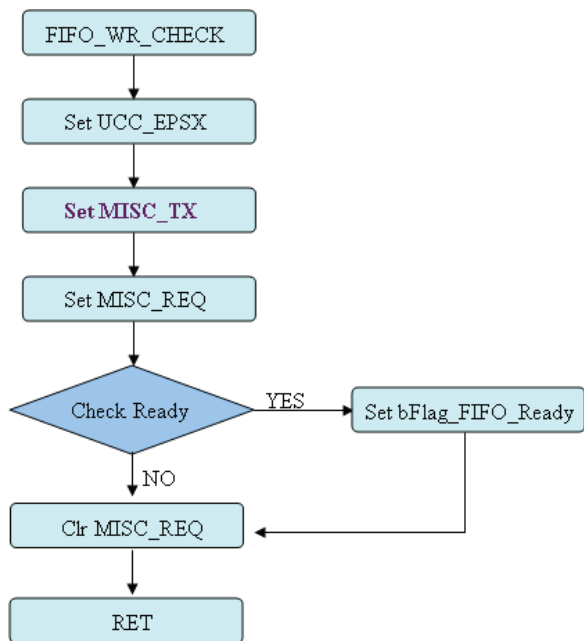
```



Programming Example:

```

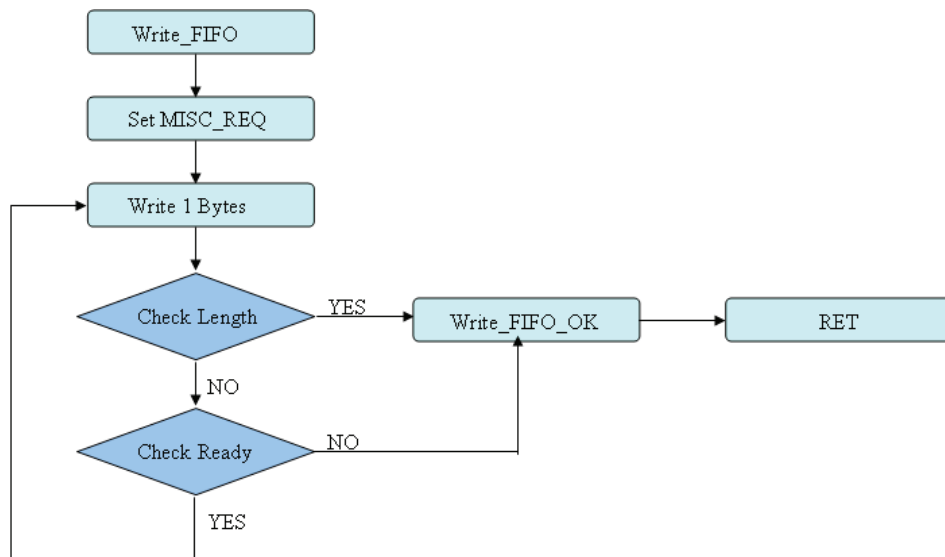
Read_FIFO_Loop:
MOV A, FIFO_TEMP
MOV MP1, A
MOV A, R1
MOV R0, A
INC FIFO_SendLen
INC MP0
MOV A, FIFO_SIZE
XOR A, FIFO_SendLen
SZ Z ;1=FIFO_SIZE=FIFO_SendLen
JMP Read_FIFO_End
MOV A, USB_MISC
MOV MP1, A
CALL Delay_28us
SZ R1.@MISC_Ready
JMP Read_FIFO_LOOP
JMP Read_FIFO_End
  
```



Programming Example:

```

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
  CALL Delay_3us
  SET R1.@MISC_REQ           ;set request
  CALL Delay_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
  
```


Programming Example:

```

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 代表傳完了
  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
```

Revision History

Revision: V1.10

Updated Date: 2011/09/14

Revision History:

Add the note for the Microsoft[®] WHQL Audio Requirement table content on page 5.