

```
*****  
;  
; file: spk_pcm8_4  
; Date: Sept 15, 2000  
; Description:  
;  
; USB Speaker, PCM8, 16 bytes/packet  
;  
; copyright 2000 Cypress Corporation  
*****
```

```
***** assembler directives *****
```

```
0000 CPU 64013
```

```
0000 XPAGEON
```

```
; processor registers
```

```
0000= Port0: equ 0h  
0001= Port1: equ 1h  
0004= Port0Int: equ 4h  
0005= Port1Int: equ 5h  
0008= port_mode: equ 8h  
0010= usb_address: equ 10h  
0011= end0_count: equ 11h  
0012= end0_mode: equ 12h  
0013= end1_count: equ 13h  
0014= end1_mode: equ 14h  
0020= global_int: equ 20h  
0021= endpoint_int: equ 21h  
0026= watchdog: equ 26h  
0024= timer_lo: equ 24h  
0025= timer_hi: equ 25h  
001F= usb_control: equ 1Fh  
00FF= control: equ FFh
```

```
00B8= endp0_dmabuff0: equ B8h  
00B9= endp0_dmabuff1: equ B9h  
00BA= endp0_dmabuff2: equ BAh  
00BB= endp0_dmabuff3: equ BBh  
00BC= endp0_dmabuff4: equ BCh  
00BD= endp0_dmabuff5: equ BDh  
00BE= endp0_dmabuff6: equ BEh  
00BF= endp0_dmabuff7: equ BFh  
00C0= endp1_dmabuff0: equ C0h  
00C1= endp1_dmabuff1: equ C1h  
00C2= endp1_dmabuff2: equ C2h  
00C3= endp1_dmabuff3: equ C3h  
00C4= endp1_dmabuff4: equ C4h  
00C5= endp1_dmabuff5: equ C5h  
00C6= endp1_dmabuff6: equ C6h  
00C7= endp1_dmabuff7: equ C7h
```

```
; mode encoding
```

```
0000= disabled: equ 00h  
0001= nak: equ 01h  
0003= stall: equ 03h  
0004= ignore: equ 04h  
000F= con_rd_ack: equ 0Fh  
000E= con_rd_nak: equ 0Eh  
0002= con_rd_stall: equ 02h  
000B= con_wr_ack: equ 0Bh  
000A= con_wr_nak: equ 0Ah  
0006= con_wr_stall: equ 06h  
0009= out_ack: equ 09h  
0008= out_nak: equ 08h  
0005= out_iso: equ 05h  
000D= in_ack: equ 0Dh  
000C= in_nak: equ 0Ch  
0007= in_iso: equ 07h
```

```
; request types
```

```
0000= get_status: equ 00h  
0001= clear_feature: equ 01h  
0003= set_feature: equ 03h  
0005= set_address: equ 05h
```

```

0006=      get_descriptor:      equ 06h
0007=      set_descriptor:      equ 07h
0008=      get_configuration:   equ 08h
0009=      set_configuration:   equ 09h
000A=      get_interface:      equ 0Ah
000B=      set_interface:      equ 0Bh
000C=      synch_frame:        equ 0Ch
0000=      device_status:      equ 00h
0000=      endpoint_status:    equ 00h
0000=      endpoint_stalled:   equ 00h
0001=      device_remote_wakeup: equ 01h

;descriptor types
0001=      device:              equ 01h
0002=      configuration:      equ 02h
0003=      string:             equ 03h
0004=      interface:         equ 04h
0005=      endpoint:          equ 05h
0022=      report:            equ 22h

; data memory variables
0011=      temp:                equ 11h
0012=      loop_counter:       equ 12h
0013=      bus_active:         equ 13h
0014=      suspend_port:      equ 14h
0015=      port_temp:         equ 15h
0016=      endp1_data_toggle:  equ 16h
0017=      endp0_data_toggle:  equ 17h
0018=      data_start:        equ 18h
0019=      data_count:        equ 19h
001A=      endpoint_stall:     equ 1Ah
001B=      CONFIG_NUM:        equ 1Bh
001C=      new_data:          equ 1Ch
001D=      pause_temp:        equ 1Dh

;***** interrupt vector table *****

0000      ORG      00h

0000 80 1B [05] jmp      reset          ; reset vector
0002 80 3D [05] jmp      bus_reset     ; bus reset interrupt
0004 80 56 [05] jmp      data_transfer ; 128us interrupt
0006 80 84 [05] jmp      lms_clear_control ; 1024ms interrupt
0008 80 8B [05] jmp      endpoint_zero  ; endpoint 0 interrupt
000A 80 79 [05] jmp      endpoint_one   ; endpoint 1 interrupt
000C 80 1A [05] jmp      error          ; not implemented in this version
000E 80 1A [05] jmp      error          ; not implemented in this version
0010 80 1A [05] jmp      error          ; not implemented in this version
0012 80 1A [05] jmp      error          ; not implemented in this version
0014 80 1A [05] jmp      error          ; not implemented in this version

0016 80 1A [05] jmp      error          ; general purpose I/O interrupt (not enabled)
0018 80 1A [05] jmp      error          ; not implemented in this version

;***** program listing *****

001A      ORG      1Ah

001A 00 [07] error: halt

;*****
;
;      Interrupt handler: reset
;      Purpose: The program jumps to this routine when
;               the microcontroller has a power on reset.
;
;*****

001B      reset:
001B 19 68 [04] mov      A, 68h
001D 30 [05] swap     A, dsp          ; sets data memory stack pointer
001E 19 00 [04] mov      A, 00h

```

```

0020 31 16 [05]      mov     [endp1_data_toggle], A
0022 31 00 [05]      mov     [endpoint_stalled], A
0024 31 13 [05]      mov     [bus_active], A
0026 2A 13 [05]      iowr   endl_count
0028 2A 04 [05]      iowr   Port0Int           ; no ints on port 0

002A 19 80 [04]      mov     A, 80h           ; set endpoint mode
002C 2A 1F [05]      iowr   usb_control

002E 19 0A [04]      mov     A, 0Ah           ; Port0 and Port1 in CMOS mode
0030 2A 08 [05]      iowr   port_mode

0032 19 07 [04]      mov     A, 07h           ; enable bus reset, lms, and 128us interrupts
0034 2A 20 [05]      iowr   global_int
0036 19 03 [04]      mov     A, 03h           ; enable endpoint interrupts
0038 2A 21 [05]      iowr   endpoint_int
003A 72 [04]         ei             ; enable interrupts

003B                do_nothing:
003B 80 3B [05]      jmp    do_nothing

;*****
;
;      Interrupt handler: bus_reset
;      Purpose: The program jumps to this routine when
;                the microcontroller has a bus reset.
;
;*****

003D                bus_reset:
003D 19 03 [04]      mov     A, stall         ; set to STALL INs&OUTs
003F 2A 12 [05]      iowr   end0_mode
0041 29 12 [05]      iord   end0_mode         ; check that mode was written
0043 10 0F [04]      and    A, 0Fh
0045 16 00 [05]      cmp    A, 00h
0047 A0 3D [05]      jz     bus_reset

0049 19 80 [04]      mov     A, 80h           ; enable USB address = 0
004B 2A 10 [05]      iowr   usb_address
004D 19 00 [04]      mov     A, disabled      ; disable endpoint1
004F 2A 14 [05]      iowr   endl_mode
0051 19 00 [04]      mov     A, 00h
0053 60 [04]         mov    psp,a
0054 80 1B [05]      jmp    reset

;*****
;
;      Interrupt handler: data_transfer
;      Purpose: If the 128us interrupt is enabled, the
;                program will jump to here every 128us.
;                In this program, the 128us interrupt transfer
;                an audio sample to the DAC every 64us.
;
;*****

0056                data_transfer:
0056 2D [05]         push   A                 ; store A register

0057 1A 1C [05]      mov     A, [new_data]    ; check for new data in FIFO
0059 16 FF [05]      cmp    A, FFh
005B B0 73 [05]      jnz    no_data_in_fifo

005D 1B C0 [06]      mov     A,[X + endp1_dmabuff0] ; get one audio sample
005F 2A 01 [05]      iowr   Port1           ; write to the DAC
0061 22 [04]         inc    X

0062 19 28 [04]      mov     A, 28h
0064                pause:
0064 27 1D [07]      dec    [pause_temp]
0066 25 [04]         dec    A
0067 B0 64 [05]      jnz    pause

0069 1B C0 [06]      mov     A,[X + endp1_dmabuff0] ; get another audio sample
006B 2A 01 [05]      iowr   Port1           ; write to the DAC
006D 22 [04]         inc    X

006E 40 [04]         mov    A, X
006F 16 10 [05]      cmp    A, 10h

```

```

0071 B0 77 [05]      jnz     done

0073                no_data_in_fifo:
0073 19 00 [04]      mov     A, 00h          ; all data in FIFO transfer
0075 31 1C [05]      mov     [new_data], A    ; clear new_data

0077                done:
0077 2B [04]          pop     A                ; restore A register
0078 73 [08]          reti

;*****
;
;      Interrupt handler: endpoint_one
;      Purpose: This interrupt routine handles the specially
;               reserved data endpoint 1. This interrupt
;               happens everytime endpoint 1 FIFO is written.
;               The routine reset the variables to indicate new
;               audio data has been received.
;*****

0079                endpoint_one:
0079 2D [05]          push A                ; store A register

007A 29 14 [05]      iord   end1_mode
007C 19 FF [04]      mov     A, FFh          ; receive new audio data
007E 31 1C [05]      mov     [new_data], A    ; set new_data
0080 1C 00 [04]      mov     X, 00h

0082 2B [04]          pop     A                ; restore A register
0083 73 [08]          reti

;*****
;
;      Interrupt handler: lms_clear_control
;      Purpose: Every lms this interrupt handler clears
;               the watchdog timer.
;*****

0084                lms_clear_control:
0084 2D [05]          push A
0085 19 41 [04]      mov     A, 41h
0087 2A 26 [05]      iowr   watchdog        ; clear watchdog timer
0089 2B [04]          pop     A
008A 73 [08]          reti

;*****
;
;      Interrupt handler: endpoint_zero
;      Purpose: This interrupt routine handles the specially
;               reserved control endpoint 0 and parses setup
;               packets. If a IN or OUT is received, this
;               handler returns to the control_read
;               or no_data_control routines to send more data.
;*****

008B                endpoint_zero:
008B 72 [04]          ei

008C 2D [05]          push A
008D 29 12 [05]      iord   end0_mode
008F 10 80 [04]      and    A, 80h          ; check if SETUP packet received
0091 A0 A5 [05]      jz     no_setup
0093 29 12 [05]      iord   end0_mode        ; check mode for valid SETUP
0095 10 9F [04]      and    A, 9Fh
0097 16 91 [05]      cmp    A, 91h
0099 B0 A7 [05]      jnz   bad_setup
009B 29 11 [05]      iord   end0_count      ; check count for valid SETUP
009D 10 CF [04]      and    A, CFh
009F 16 4A [05]      cmp    A, 4Ah
00A1 B0 A7 [05]      jnz   bad_setup
00A3 90 AD [10]      call   host2dev_devrecip ; SETUP ...goto parsing
00A5                no_setup:
00A5 2B [04]          pop     A
00A6 73 [08]          reti                ; IN or OUT, or SETUP completed
00A7                bad_setup:

```

```

00A7 19 03 [04]      mov     A,03h           ; stall following an bad SETUP
00A9 2A 12 [05]      iowr   end0_mode
00AB 80 A5 [05]      jmp    no_setup

;*****stage one..determine type of transfer (bmRequestType)

00AD                host2dev_devrecip:
00AD 19 01 [04]      mov     A, nak           ; clear the setup flag leave in Nak mode
00AF 2A 12 [05]      iowr   end0_mode
00B1 29 12 [05]      iord   end0_mode           ; retry write if needed
00B3 16 01 [05]      cmp    A, nak
00B5 B0 AD [05]      jnz    host2dev_devrecip

00B7 1A B8 [05]      mov    A, [endp0_dmabuff0] ; parse packet
00B9 16 00 [05]      cmp    A, 00h
00BB B0 E0 [05]      jnz    host2dev_intrecip

00BD 1A B9 [05]      mov    A, [endp0_dmabuff1]
00BF 16 01 [05]      cmp    A, clear_feature
00C1 A1 77 [05]      jz     clear_feat

00C3 1A B9 [05]      mov    A, [endp0_dmabuff1]
00C5 16 03 [05]      cmp    A, set_feature
00C7 A1 67 [05]      jz     set_feat

00C9 1A B9 [05]      mov    A, [endp0_dmabuff1]
00CB 16 05 [05]      cmp    A, set_address
00CD A1 5E [05]      jz     set_addr

00CF 1A B9 [05]      mov    A, [endp0_dmabuff1]
00D1 16 09 [05]      cmp    A, set_configuration
00D3 A1 9F [05]      jz     set_config

00D5                h2dd_stall:
00D5 19 03 [04]      mov    A, stall           ; send a stall to indicate that the requested
00D7 2A 12 [05]      iowr   end0_mode           ; function is not supported
00D9 29 12 [05]      iord   end0_mode           ; write retry
00DB 16 03 [05]      cmp    A, stall
00DD B0 D5 [05]      jnz    h2dd_stall
00DF 3F      [08]      ret

00E0                host2dev_intrecip:
00E0 1A B8 [05]      mov    A, [endp0_dmabuff0]
00E2 16 01 [05]      cmp    A, 01h
00E4 B0 F7 [05]      jnz    host2dev_endprecip

00E6 1A B9 [05]      mov    A, [endp0_dmabuff1] ; parse packet
00E8 16 0B [05]      cmp    A, set_interface
00EA A1 8F [05]      jz     set_intf

00EC                h2di_stall:
00EC 19 03 [04]      mov    A, stall           ; send a stall to indicate that the requested
00EE 2A 12 [05]      iowr   end0_mode           ; function is not supported
00F0 29 12 [05]      iord   end0_mode           ; write retry
00F2 16 03 [05]      cmp    A, stall
00F4 B0 EC [05]      jnz    h2di_stall
00F6 3F      [08]      ret

00F7                host2dev_endprecip:
00F7 1A B8 [05]      mov    A, [endp0_dmabuff0]
00F9 16 02 [05]      cmp    A, 02h
00FB B1 0D [05]      jnz    host2dev_endprecip2

00FD 1A B9 [05]      mov    A, [endp0_dmabuff1] ; parse
00FF 1F      [04]      XPAGE
0100 16 03 [05]      cmp    A, set_feature
0102 A1 67 [05]      jz     set_feat

0104 16 01 [05]      cmp    A, clear_feature
0106 A1 77 [05]      jz     clear_feat

0108 19 03 [04]      mov    A, stall           ; send a stall to indicate that the requested
010A 2A 12 [05]      iowr   end0_mode           ; function is not supported
010C 3F      [08]      ret

010D                host2dev_endprecip2:
010D 1A B8 [05]      mov    A, [endp0_dmabuff0]
010F 16 22 [05]      cmp    A, 22h
0111 B1 18 [05]      jnz    dev2host_devrecip

```

```

0113 19 0B [04]      mov     A, con_wr_ack
0115 2A 12 [05]      iowr   end0_mode
0117 3F   [08]      ret

0118                dev2host_devrecip:
0118 1A B8 [05]      mov     A, [endp0_dmabuff0]      ; read the first byte of the buffer
011A 16 80 [05]      cmp     A, 80h                  ; compare to 80h
011C B1 35 [05]      jnz    dev2host_intrecip

011E 1A B9 [05]      mov     A, [endp0_dmabuff1]      ; parse
0120 16 08 [05]      cmp     A, get_configuration
0122 A1 92 [05]      jz     get_config

0124 1A B9 [05]      mov     A, [endp0_dmabuff1]
0126 16 00 [05]      cmp     A, get_status
0128 A1 A6 [05]      jz     get_stat

012A 1A B9 [05]      mov     A, [endp0_dmabuff1]
012C 16 06 [05]      cmp     A, get_descriptor
012E A1 CD [05]      jz     get_desc

0130 19 03 [04]      mov     A, stall                ; send a stall to indicate that the requested
0132 2A 12 [05]      iowr   end0_mode                ; function is not supported
0134 3F   [08]      ret

0135                dev2host_intrecip:
0135 1A B8 [05]      mov     A, [endp0_dmabuff0]
0137 16 81 [05]      cmp     A, 81h
0139 B1 46 [05]      jnz    dev2host_endprecip

013B 1A B9 [05]      mov     A, [endp0_dmabuff1]      ; parse
013D 16 0A [05]      cmp     A, get_interface
013F A1 84 [05]      jz     get_interf

0141 19 03 [04]      mov     A, stall                ; send a stall to indicate that the requested
0143 2A 12 [05]      iowr   end0_mode                ; function is not supported
0145 3F   [08]      ret

0146                dev2host_endprecip:
0146 1A B8 [05]      mov     A, [endp0_dmabuff0]
0148 16 82 [05]      cmp     A, 82h
014A A1 4D [05]      jz     match
014C 3F   [08]      ret
014D                match:
014D 1A B9 [05]      mov     A, [endp0_dmabuff1]      ; get HID class descriptor
014F 16 06 [05]      cmp     A, get_descriptor
0151 A1 CD [05]      jz     get_desc

0153 1A B9 [05]      mov     A, [endp0_dmabuff1]      ; parse
0155 16 00 [05]      cmp     A, endpoint_status
0157 A1 B1 [05]      jz     endp_status

0159 19 03 [04]      mov     A, stall                ; send a stall to indicate that the requested
015B 2A 12 [05]      iowr   end0_mode                ; function is not supported
015D 3F   [08]      ret

;*****stage two..determine request type (bRequest)

015E                set_addr:
015E 92 72 [10]      call   no_data_control
0160 1A BA [05]      mov     A, [endp0_dmabuff2]      ; get address
0162 0D 80 [04]      or     A, 80h                  ; address enable bit
0164 2A 10 [05]      iowr   usb_address              ; set usb address register
0166 3F   [08]      ret

0167                set_feat:
0167 1A BA [05]      mov     A, [endp0_dmabuff2]
0169 16 00 [05]      cmp     A, endpoint_stalled
016B B1 74 [05]      jnz    remote_wakeup
016D 19 01 [04]      mov     A, 01h
016F 31 1A [05]      mov     [endpoint_stall], A
0171 92 72 [10]      call   no_data_control
0173 3F   [08]      ret
0174                remote_wakeup:
0174 92 72 [10]      call   no_data_control
0176 3F   [08]      ret

```

```

0177          clear_feat:
0177 1A BA [05]      mov     A, [endp0_dmabuff2]
0179 16 00 [05]      cmp     A, endpoint_stalled
017B B1 81 [05]      jnz    not_endp_clear
017D 19 00 [04]      mov     A, 00h
017F 31 1A [05]      mov     [endpoint_stall], A
0181          not_endp_clear:
0181 92 72 [10]      call   no_data_control
0183 3F   [08]      ret

0184          get_interf:
0184 19 82 [04]      mov     A, 82h
0186 31 18 [05]      mov     [data_start], A
0188 1A BE [05]      mov     A, [endp0_dmabuff6]
018A 31 19 [05]      mov     [data_count], A
018C 92 0D [10]     call   control_read
018E 3F   [08]      ret

018F          set_interf:
018F 92 72 [10]     call   no_data_control
0191 3F   [08]      ret

0192          get_config:
0192 19 87 [04]      mov     A, 87h
0194 02 1B [06]     add     A, [CONFIG_NUM]
0196 31 18 [05]      mov     [data_start], A
0198 1A BE [05]      mov     A, [endp0_dmabuff6]
019A 31 19 [05]      mov     [data_count], A
019C 92 0D [10]     call   control_read
019E 3F   [08]      ret

019F          set_config:
019F 1A BA [05]      mov     A, [endp0_dmabuff2]
01A1 31 1B [05]      mov     [CONFIG_NUM], A
01A3 92 72 [10]     call   no_data_control
01A5 3F   [08]      ret

01A6          get_stat:
01A6 1A BF [05]      mov     A, [endp0_dmabuff7]
01A8 16 00 [05]      cmp     A, device_status
01AA A1 DE [05]      jz     dev_status

01AC 19 03 [04]     mov     A, stall          ; send a stall to indicate that the requested
01AE 2A 12 [05]     iowr   end0_mode         ; function is not supported
01B0 3F   [08]     ret

01B1          endp_status:
01B1 1A 1A [05]     mov     A, [endpoint_stall]
01B3 16 00 [05]     cmp     A, endpoint_stalled
01B5 B1 C2 [05]     jnz    endp_stalled_status
01B7 19 83 [04]     mov     A, 83h
01B9 31 18 [05]     mov     [data_start], A
01BB 1A BE [05]     mov     A, [endp0_dmabuff6]
01BD 31 19 [05]     mov     [data_count], A
01BF 92 0D [10]     call   control_read
01C1 3F   [08]     ret

01C2          endp_stalled_status:
01C2 19 85 [04]     mov     A, 85h
01C4 31 18 [05]     mov     [data_start], A
01C6 1A BE [05]     mov     A, [endp0_dmabuff6]
01C8 31 19 [05]     mov     [data_count], A
01CA 92 0D [10]     call   control_read
01CC 3F   [08]     ret

01CD          get_desc:
01CD 1A BB [05]     mov     A, [endp0_dmabuff3]
01CF 16 01 [05]     cmp     A, device
01D1 A1 E9 [05]     jz     device_desc

01D3 1A BB [05]     mov     A, [endp0_dmabuff3]
01D5 16 02 [05]     cmp     A, configuration
01D7 A1 F4 [05]     jz     config_desc

01D9 19 03 [04]     mov     A, stall          ; send a stall to indicate that the requested
01DB 2A 12 [05]     iowr   end0_mode         ; function is not supported
01DD 3F   [08]     ret

```

```

;*****stage three..determine descriptor type (descriptor)

01DE          dev_status:
01DE 19 7F [04]    mov     A, 7Fh
01E0 31 18 [05]    mov     [data_start], A
01E2 1A BE [05]    mov     A, [endp0_dmabuff6]
01E4 31 19 [05]    mov     [data_count], A
01E6 92 0D [10]   call    control_read
01E8 3F    [08]    ret

01E9          device_desc:
01E9 19 00 [04]    mov     A, 00h
01EB 31 18 [05]    mov     [data_start], A
01ED 1A BE [05]    mov     A, [endp0_dmabuff6]
01EF 31 19 [05]    mov     [data_count], A
01F1 92 0D [10]   call    control_read
01F3 3F    [08]    ret

01F4          config_desc:
01F4 19 12 [04]    mov     A, 12h
01F6 31 18 [05]    mov     [data_start], A
01F8 1A BE [05]    mov     A, [endp0_dmabuff6]
01FA 16 6D [05]    cmp     A, 6Dh
01FC C2 02 [05]    jc     config_desc_data_length
01FE 20    [04]    NOP
01FF 1F    [04]    XPAGE
0200 19 6D [04]    mov     A, 6Dh
0202          config_desc_data_length:
0202 31 19 [05]    mov     [data_count], A
0204 92 0D [10]   call    control_read

;          ;***** get ready for endpoint 1!!!! *****

0206 29 14 [05]    iord   end1_mode
0208 19 05 [04]    mov     A, out_iso          ; isochronous out
020A 2A 14 [05]    iowr   end1_mode

020C 3F    [08]    ret

;*****USB library main routines*****

;*****
;
;   function: Control_read
;   Purpose:  Performs the control read operation
;             as defined by the USB specification
;   SETUP-IN-IN-IN...OUT
;
;   data_start: must be set to the descriptors info
;               as an offset from the beginning of the
;               control_read_table
;   data_count: must be set to the size of the
;               descriptor
;*****

020D          control_read:
020D 1C 00 [04]    mov     X, 00h
020F 19 00 [04]    mov     A, 00h
0211 31 17 [05]    mov     [endp0_data_toggle], A
0213          cr_wr:
0213 19 01 [04]    mov     A, nak              ;clear PID bits, leave in nak mode
0215 2A 12 [05]    iowr   end0_mode
0217 29 12 [05]    iord   end0_mode
0219 16 01 [05]    cmp     A, nak
021B B2 13 [05]    jnz    cr_wr

021D          control_read_data_stage:
021D 1C 00 [04]    mov     X, 00h
021F 19 00 [04]    mov     A, 00h
0221 31 12 [05]    mov     [loop_counter], A
0223          crds_wr:
0223 19 0E [04]    mov     A, con_rd_nak      ;clear PID bits, leave in nak mode
0225 2A 12 [05]    iowr   end0_mode
0227 29 12 [05]    iord   end0_mode
0229 16 0E [05]    cmp     A, con_rd_nak

```

```

022B B2 23 [05]      jnz      crds_wr
022D 1A 19 [05]      mov      A, [data_count]
022F 16 00 [05]      cmp      A, 00h
0231 A2 70 [05]      jz       control_read_status_stage

0233                dma_load_loop:                ; loop to load data into the data buffer
0233 1A 18 [05]      mov      A, [data_start]
0235 F2 85 [14]      index   control_read_table
0237 32 B8 [06]      mov      [X + endp0_dmabuff0], A ; load dma buffer
0239 23 18 [07]      inc      [data_start]
023B 22 [04]         inc      X
023C 23 12 [07]      inc      [loop_counter]
023E 27 19 [07]      dec      [data_count]                ; exit if descriptor
0240 A2 48 [05]      jz       dma_load_done                ; is done
0242 1A 12 [05]      mov      A, [loop_counter]            ; or 8 bytes sent
0244 16 08 [05]      cmp      A, 08h
0246 B2 33 [05]      jnz      dma_load_loop

0248                dma_load_done:
0248 29 11 [05]      iord     end0_count
024A 1A 17 [05]      mov      A, [endp0_data_toggle]
024C 13 80 [04]      xor      A, 80h
024E 31 17 [05]      mov      [endp0_data_toggle], A
0250 0E 12 [06]      or       A, [loop_counter]
0252 2A 11 [05]      iowr     end0_count

0254                cr_wr_ack:
0254 19 0F [04]      mov      A, con_rd_ack
0256 2A 12 [05]      iowr     end0_mode
0258 29 12 [05]      iord     end0_mode
025A 10 0F [04]      and      A, 0Fh
025C 16 0F [05]      cmp      A, con_rd_ack
025E B2 54 [05]      jnz      cr_wr_ack

0260                wait_control_read:
0260 29 12 [05]      iord     end0_mode                ; wait for the data to be
0262 10 01 [04]      and      A, 01h                ; transfered
0264 A2 1D [05]      jz       control_read_data_stage
0266 29 12 [05]      iord     end0_mode
0268 10 A0 [04]      and      A, A0h                ; check if out/setup was sent by host
026A B2 70 [05]      jnz      control_read_status_stage
026C 82 60 [05]      jmp      wait_control_read

026E 82 1D [05]      jmp      control_read_data_stage

0270                control_read_status_stage:        ; OUT at end of data transfer
0270 3F [08]         ret

;*****
;
;      function: control_write
;      purpose:  performs the control write operation
;                as defined by the USB specification
;      SETUP-OUT-OUT-OUT...IN
;*****

0271                control_write:
; not implemented with a speaker, but may be needed for
; other devices

0271 3F [08]         ret

;*****
;
;      function: no_data_control
;      purpose:  performs the no-data control operation
;                as defined by the USB specification
;      SETUP-IN
;*****

0272                no_data_control:
0272 29 21 [05]      iord     21h
0274 19 06 [04]      mov      A, con_wr_stall ; setup for status stage IN
0276 2A 12 [05]      iowr     end0_mode
0278 29 12 [05]      iord     end0_mode

```

```

027A 16 06 [05]      cmp      A,con_wr_stall
027C B2 72 [05]      jnz     no_data_control

027E      wait_nodata_sent:
027E 29 12 [05]      iord    end0_mode          ; wait for the IN to be
0280 10 40 [04]      and     A, 40h             ; transfered
0282 A2 7E [05]      jz      wait_nodata_sent

0284 3F [08]        ret

;*****
;                               rom lookup tables
;*****

0285      control_read_table:
0285      device_desc_table:
0285 12 [00]         db      12h          ; size of descriptor (18 bytes)
0286 01 [00]         db      01h          ; descriptor type (DEVICE descriptor)
0287 01 [00]         db      01h, 01h     ; USB spec release (ver 1.1)
0288 01 [00]
0289 00 [00]         db      00h          ; class code (each interface specifies class information)
028A 00 [00]         db      00h          ; device sub-class (must be set to 0 because class code is 0)
028B 00 [00]         db      00h          ; device protocol (no class specific protocol)
028C 08 [00]         db      08h          ; maximum packet size (8 bytes)
028D B4 [00]         db      B4h, 04h     ; vendor ID (note Microsoft vendor ID)
028E 04 [00]
028F FF [00]         db      FFh, 0Fh     ; product ID (Microsoft USB mouse product ID)
0290 0F [00]
0291 FF [00]         db      FFh, 00h     ; device release number
0292 00 [00]
0293 00 [00]         db      00h          ; index of manufacturer string (not supported)
0294 00 [00]         db      00h          ; index of product string (not supported)
0295 00 [00]         db      00h          ; index of serial number string (not supported)
0296 01 [00]         db      01h          ; number of configurations (1)
0297      config_desc_table:
0297 09 [00]         db      09h          ; length of descriptor (9 bytes)
0298 02 [00]         db      02h          ; descriptor type (CONFIGURATION)
0299 6D [00]         db      6Dh, 00h     ; total length of descriptor (109 bytes)
029A 00 [00]
029B 02 [00]         db      02h          ; number of interfaces to configure (2)
029C 01 [00]         db      01h          ; configuration value (1)
029D 00 [00]         db      00h          ; configuration string index (not supported)
029E 80 [00]         db      80h          ; configuration attributes (bus powered...or will be in future)
029F 32 [00]         db      32h          ; maximum power (100mA)

; *Standard AC Interface Desc.*
02A0 09 [00]         db      09h          ; length of descriptor (9 bytes)
02A1 04 [00]         db      04h          ; descriptor type (INTERFACE)
02A2 00 [00]         db      00h          ; interface number (0)
02A3 00 [00]         db      00h          ; alternate setting (0)
02A4 00 [00]         db      00h          ; number of endpoints (0)
02A5 01 [00]         db      01h          ; interface class (AUDIO)
02A6 01 [00]         db      01h          ; interface sub-class (AUDIOCONTROL)
02A7 00 [00]         db      00h          ; interface protocol (unused)
02A8 00 [00]         db      00h          ; interface string index (unused)

; *Class-specific AC Interface Desc*
02A9 09 [00]         db      09h          ; length of descriptor (9 bytes)
02AA 24 [00]         db      24h          ; descriptor type (CS_INTERFACE)
02AB 01 [00]         db      01h          ; descriptor subtype (HEADER)
02AC 00 [00]         db      00h, 01h     ; Audio Device Class spec release (ver 1.0)
02AD 01 [00]
02AE 27 [00]         db      27h, 00h     ; total size of class specific descriptors
02AF 00 [00]
02B0 01 [00]         db      01h          ; number of AudioStreaming interfaces (1)
02B1 01 [00]         db      01h          ; AudioStreaming interface # in the interface collection (1)

; *Input Terminal Desc*
02B2 0C [00]         db      0Ch          ; length of descriptor (12 bytes)
02B3 24 [00]         db      24h          ; descriptor type (CS_INTERFACE)
02B4 02 [00]         db      02h          ; descriptor subtype (INPUT_TERMINAL)
02B5 01 [00]         db      01h          ; ID of this Input Terminal (01)
02B6 01 [00]         db      01h, 01h     ; terminal type (USB streaming)
02B7 01 [00]
02B8 00 [00]         db      00h          ; terminal association (no association)
02B9 01 [00]         db      01h          ; number of logical output channels (1)
02BA 00 [00]         db      00h, 00h     ; channel configuration (mono)
02BB 00 [00]
02BC 00 [00]         db      00h          ; channel names string (unused)
02BD 00 [00]         db      00h          ; terminal description string (unused)

; *Output Terminal Desc*

```

```

02BE 09 [00] db 09h ; length of descriptor (9 bytes)
02BF 24 [00] db 24h ; descriptor type (CS_INTERFACE)
02C0 03 [00] db 03h ; descriptor subtype (OUTPUT_TERMINAL)
02C1 03 [00] db 03h ; terminal identification (03)
02C2 01 [00] db 01h, 03h ; terminal type (Speaker)
02C3 03 [00]
02C4 00 [00] db 00h ; terminal association (unused)
02C5 02 [00] db 02h ; ID of the source terminal (02)
02C6 00 [00] db 00h ; index of Output Terminal string descriptor (unused)
; *Feature Unit Desc*
02C7 09 [00] db 09h ; length of descriptor (9 bytes)
02C8 24 [00] db 24h ; descriptor type (CS_INTERFACE)
02C9 03 [00] db 03h ; descriptor subtype (OUTPUT_TERMINAL)
02CA 02 [00] db 02h ; terminal identification (02)
02CB 01 [00] db 01h ; ID of the source terminal (01)
02CC 02 [00] db 02h ; control size (2 bytes)
02CD 03 [00] db 03h ; master channel controls (volume, mute)
02CE 00 [00] db 00h ; logical channel 1 controls (none)
02CF 00 [00] db 00h ; string descriptor (unused)
; *Standard AS Interface Desc Alternate Setting 0*
02D0 09 [00] db 09h ; length of descriptor (9 bytes)
02D1 04 [00] db 04h ; descriptor type (INTERFACE)
02D2 01 [00] db 01h ; interface number (01)
02D3 00 [00] db 00h ; index of this alternate setting (00)
02D4 00 [00] db 00h ; number of endpoints used (0)
02D5 01 [00] db 01h ; interface class (AUDIO)
02D6 02 [00] db 02h ; interface sub-class (AUDIOSTREAMING)
02D7 00 [00] db 00h ; interface protocol (not used)
02D8 00 [00] db 00h ; index of interface string descriptor (unused)
; *Standard AS Interface Desc Alternate Setting 1*
02D9 09 [00] db 09h ; length of descriptor (9 bytes)
02DA 04 [00] db 04h ; descriptor type (INTERFACE)
02DB 01 [00] db 01h ; interface number (01)
02DC 01 [00] db 01h ; index of this alternate setting (01)
02DD 01 [00] db 01h ; number of endpoints used (1)
02DE 01 [00] db 01h ; interface class (AUDIO)
02DF 02 [00] db 02h ; interface sub-class (AUDIOSTREAMING)
02E0 00 [00] db 00h ; interface protocol (not used)
02E1 00 [00] db 00h ; index of interface string descriptor (unused)
; Class-specific AS Interface Desc
02E2 07 [00] db 07h ; length of descriptor (7 bytes)
02E3 24 [00] db 24h ; descriptor type (CS_INTERFACE)
02E4 01 [00] db 01h ; descriptor subtype (AS_GENERAL)
02E5 01 [00] db 01h ; terminal connected to (01)
02E6 01 [00] db 01h ; interface delay
02E7 02 [00] db 02h, 00h ; Audio Data Format (PCM8 Format)
02E8 00 [00]
; Format Type Desc
02E9 0B [00] db 0Bh ; length of descriptor (11 bytes)
02EA 24 [00] db 24h ; descriptor type (CS_INTERFACE)
02EB 02 [00] db 02h ; descriptor subtype (FORMAT_TYPE)
02EC 01 [00] db 01h ; format type (FORMAT_TYPE_I)
02ED 01 [00] db 01h ; number of physical channels (1)
02EE 01 [00] db 01h ; # of bytes in an audio subframe (1)
02EF 08 [00] db 08h ; bits per sample (8)
02F0 01 [00] db 01h ; # of frequency supported (1)
02F1 80 [00] db 80h, 3Eh, 00h ; 16000hz
02F2 3E [00]
02F3 00 [00]
; Standard AS Isochronous Audio Data Endpoint Desc
02F4 09 [00] db 09h ; length of descriptor (9 bytes)
02F5 05 [00] db 05h ; descriptor type (ENDPOINT)
02F6 01 [00] db 01h ; OUT Endpoint 1
02F7 09 [00] db 09h ; transfer type (ADAPTIVE, ISOCRONOUS)
02F8 10 [00] db 10h, 00h ; maximum packet size (16 bytes per packet)
02F9 00 [00]
02FA 01 [00] db 01h ; interval for polling endpoint (must be set to 1ms)
02FB 00 [00] db 00h ; refresh (unused)
02FC 00 [00] db 00h ; endpoint used to communicate synchronization info (unused)
; Class-specific AS Isochronous Audio Data Endpoint Desc
02FD 07 [00] db 07h ; length of descriptor (7 bytes)
02FE 25 [00] db 25h ; descriptor type (CS_ENDPOINT)
02FF 01 [00] db 01h ; descriptor subtype (EP_GENERAL)
0300 01 [00] db 01h ; attributes (sampling frequency)
0301 00 [00] db 00h ; lock delay units (unused)
0302 00 [00] db 00h, 00h ; lock delay (unused)
0303 00 [00]
0304 get_dev_status_table:
0304 00 [00] db 00h, 00h ; bus powered, no remote wakeup

```

```
0305 00 [00]
0306      get_config_table:
0306 01 [00]      db      01h          ; configuration 1
0307      get_interface_table:
0307 00 [00]      db      00h          ; interface 0
0308      get_endp_status_table:
0308 00 [00]      db      00h, 00h      ; endpoint not stalled
0309 00 [00]
030A      get_endp_stalled_status_table:
030A 01 [00]      db      01h, 00h      ; endpoint stalled
030B 00 [00]
030C      Another_get_config:
030C 00 [00]      db      00h          ; new configuration value
030D 01 [00]      db      01h          ; after set config
```

```
Checksum = ADEF
Warnings = 0
Errors   = 0
```

```
Product: 64013, CPU Family=B, RAM=256 bytes, ROM=8160 bytes
```