```
/* Defines for Soundblaster and Soundblaster Pro IO address */
#define LEFT_FM_STATUS      0x00    /* Pro only */
#define LEFT_FM_ADDRESS     0x00    /* Pro only */
#define LEFT_FM_DATA        0x01    /* Pro only */
#define RIGHT_FM_STATUS     0x02    /* Pro only */
#define RIGHT_FM_ADDRESS    0x02    /* Pro only */
#define RIGHT_FM_DATA       0x03    /* Pro only */
#define MIXER_ADDRESS       0x04    /* Pro only */
#define MIXER_DATA          0x05    /* Pro only */
#define DSP_RESET           0x06
#define FM_STATUS           0x08
#define FM_ADDRESS          0x08
#define FM_DATA             0x09
#define DSP_READ_DATA       0x0A
#define DSP_WRITE_DATA      0x0C
#define DSP_WRITE_STATUS    0x0C
#define DSP_DATA_AVAIL      0x0E
#define CD_ROM_DATA         0x10    /* Pro only */
#define CD_ROM_STATUS       0x11    /* Pro only */
#define CD_ROM_RESET        0x12    /* Pro only */
#define CD_ROM_ENABLE       0x13    /* Pro only */

#define ADLIB_FM_STATUS     0x388
#define ADLIB_FM_ADDRESS    0x388
#define ADLIB_FM_DATA       0x389

/* Types of Soundblaster Cards */
#define SB15  1
#define SBPro 2
#define SB20  3

/* DSP Commands */
#define DIRECT_8_BIT_DAC    0x10
#define DMA_8_BIT_DAC          0x14
#define DMA_2_BIT_DAC          0x16
#define DMA_2_BIT_REF_DAC   0x17
#define DIRECT_ADC      0x20
#define DMA_ADC         0x24
#define MIDI_READ_POLL         0x30
#define MIDI_READ_IRQ          0x31
#define MIDI_WRITE_POLL     0x38
#define TIME_CONSTANT          0x40
#define DMA_4_BIT_DAC          0x74
#define DMA_4_BIT_REF_DAC   0x75
#define DMA_26_BIT_DAC         0x76
#define DMA_26_BIT_REF_DAC  0x77
#define HALT_DMA        0xD0
#define CONTINUE_DMA           0xD4
#define SPEAKER_ON      0xD1
#define SPEAKER_OFF     0xD3
#define DSP_ID                 0xE0
#define DSP_VER         0xE1
#define MDAC1           0x61
#define MDAC2           0x62
#define MDAC3           0x63
#define MDAC4           0x64
#define MDAC5           0x65
#define MDAC6           0x66
#define MDAC7           0x67

/* High-speed parameters */
#define MAX_LO_REC 12048
#define MAX_LO_PLAY 22222
#define SET_HS_SIZE 0x48
#define HS_DAC 0x91
#define HS_ADC 0x99


Programming the Sound Blaster ADC/DAC:

2x6h    DSP Reset Port                      Write Only
2xAh    DSP Read Data Port                  Read Only
2xCh    DSP Write Data or Command           Write
2xCh    DSP Write Buffer Status (bit 7)     Read
2xEh    DSP Data Available Status Bit 7)    Read Only
```

x = 1,2,3,4,5,6 for the Sound Blaster <= 1.5
x = 1,2,3,4,5,6 for the Sound Blaster Micro Channel Version
x = 2,4 for the Sound Blaster 2.0
x = 2,4 for the Sound Blaster Pro

The DSP: ----------

Due to the different mode and the hardware performance of different audio cards, the maximum sampling rate is different.

Input (ADC):

Sound Blaster (<=1.5)	Mono/Normal mode:	8-bit only	4KHz to 13KHz
Sound Blaster (2.0)	Mono/Normal mode:	8-bit only	4KHz to 13KHz
	Mono/High Speed mode:	8-bit only	13KHz to 15KHz
Sound Blaster Pro	Mono/Normal mode:	8-bit only	4KHz to 23KHz
	Mono/High Speed mode:	8-bit only	23KHz to 44.1KHz

Output (DAC):

Sound Blaster (<=1.5)	Mono/Normal mode:	8-bit only	4KHz to 23KHz
Sound Blaster (2.0)	Mono/Normal mode:	8-bit only	4KHz to 23KHz
	Mono/High Speed mode:	8-bit only	23KHz to 44.1KHz
Sound Blaster Pro	Mono/Normal mode:	8-bit only	4KHz to 23KHz
	Mono/High Speed mode:	8-bit only	23KHz to 44.1KHz

Some DSP Commands:

10h Direct mode 8-bit DAC (single byte data transfer)
14h DMA mode 8-bit DAC
20h Direct mode 8-bit ADC (single byte data transfer)
24h DMA mode 8-bit ADC
40h Set Time Constant
48h Set Block Size
91h High Speed DMA mode 8-bit DAC
99h High Speed DMA mode 8-bit ADC
D1h Turn on Speaker
D3h Turn off Speaker
D0h Halt DMA in progress
D4h Continue DMA
E1h Get DSP version (read major ver then minor)

To reset the DSP:

1. Write a 01h to port 2x6h
2. Wait for 3 microseconds
3. Write a 00h to port 2x6h
4. Read port 2xAh until a 0AAh is read (see below for how to read from 2xAh)

If there is no 0AAh after about 100 reads, abort and declare that there is no Sound Blaster present (or error)

To write to the DSP (all writes to 2xCh MUST follow this procedure)

1. Read 2xCh until bit 7 is clear
2. Write to 2xCh

To read from the DSP (all reads from 2xAh MUST follow this procedure)

1. Read 2xEh until bit 7 is set
2. Read from 2xAh

Interrupts:

In DMA DAC and DMA ADC modes, a single interrupt will occur after the block of data has been read/written. To clear the interrupt, read 2xEh once (as well as clearing the PIC).

Ignoring Interrupts:

The interrupt can be ignored if you poll the DMAC (DMA Controller). Once the DMAC reports a count of 0FFFFh the transfer is finished, read 2xEh once and you are finished. You can also poll the DMA status register and wait for the Terminal Count reached bit to be set (I found that the first read after starting the transfer had the bit set, but not the second).

Note to VESA/Local Bus Video users:

These video cards use DMA channel 1 which is the DMA channel used by the Sound Blaster. Any video accesses will screw up the output of the Sound Blaster.

Calculating the Time Constant:

Normal Speed:

```
Time Constant = 256 - (1,000,000 / sampling rate)
              = 256 - (1,000,000 / 8,000 )
              = 131
```

High Speed:

```
Time Constant = (MSByte of) 65536 - (256,000,000 / sampling rate)
              = (MSByte of) 65536 - (256,000,000 / 44,100)
              = (MSByte of) 59731
              = (MSByte of) 0E953h
              = 0E9h
```

Direct mode DAC:

1. Write a D1h to 2xCh
2. Write a 10h to 2xCh
3. Write the 8-bit data sample to 2xCh
4. Wait for the correct timing (must do your own timing)
   Repeat steps 2-4 until end of data
5. Write a D3h to 2xCh

Normal speed DMA mode DAC:

1. Write a D1h to 2xCh
2. Setup Interrupt service routine
3. Write a 40h to 2xCh
4. Write Time Constant to 2xCh
5. Program the DMAC (DMA Controller)
6. Write 14h to 2xCh
7. Write the LSByte of Data Length - 1
8. Write the MSByte of Data Length - 1
9. Service Interrupt (may need to repeat steps 5-7 in the ISR)
10. Restore original Interrupt Service Routine
11. Write a D3h to 2xCh

Commands can be written to the DSP while waiting for the interrupt

High speed DMA mode DAC:

1. Write a D1h to 2xCh
2. Setup Interrupt service routine
3. Write a 40h to 2xCh
4. Write Time Constant to 2xCh
5. Program the DMAC (DMA Controller)
6. Write 48h to 2xCh
7. Write the LSByte of Data Length - 1
8. Write the MSByte of Data Length - 1
9. Write 91h to 2xCh
10. Service Interrupt (may need to repeat steps 5-7 in the ISR)
11. Restore original Interrupt Service Routine
12. Write a D3h to 2xCh

Commands CANNOT be written to the DSP while waiting for the interrupt
Resetting the DSP is the procedure used to halt DMA in progress

Direct mode ADC:

1. Write a 20h to 2xCh
2. Read the 8-bit data sample from 2xAh
3. Wait for the correct timing (must do your own timing)
   Repeat steps 1-3 until finished

Normal speed DMA mode ADC:

1. Setup Interrupt service routine
2. Write a 40h to 2xCh
3. Write Time Constant to 2xCh
4. Program the DMAC (DMA Controller)
5. Write 24h to 2xCh
6. Write the LSByte of Data Length - 1
7. Write the MSByte of Data Length - 1
8. Service Interrupt (may need to repeat steps 5-7 in the ISR)
9. Restore original Interrupt Service Routine

Commands can be written to the DSP while waiting for the interrupt

High speed DMA mode ADC:

1. Setup Interrupt service routine
2. Write a 40h to 2xCh
3. Write Time Constant to 2xCh
4. Program the DMAC (DMA Controller)
5. Write 48h to 2xCh
6. Write the LSByte of Data Length - 1
7. Write the MSByte of Data Length - 1
8. Write 99h to 2xCh
9. Service Interrupt (may need to repeat steps 5-7 in the ISR)
10. Restore original Interrupt Service Routine

Commands CANNOT be written to the DSP while waiting for the interrupt
Resetting the DSP is the procedure used to halt DMA in progress

The Mixer: ----------

The mixer can only be found on the Sound Blaster Pro.

It mixes the following audio sources:
-digitized voice
-FM synthesized music
-CD-audio
-line-in
-microphone input
-PC speaker output

It allows software to control volume on:
-digitized voice
-FM synthesized music
-CD-audio
-line-in
-microphone mixing
-overall master volume

The mixer setting can be done with 2 I/O ports, 2x4h and 2x5h.
x=2,4 for the Sound Blaster Pro

2x4h is the address port (write only), 2x5h is the data port (read/write)

The programming sequence is as follows:

1. Write the address of the mixers register to 2x4h
2. Read/Write the mixers register value from/to 2x5h

Mixer registers:

Register	D7	D6	D5	D4	D3	D2	D1	D0
00h	Data Reset							
02h	Reserved							
04h	Voice Volume L				Voice Volume R			
06h	Reserved							
08h	Reserved							
0Ah	x	x	x	x	x	MIC Mixing		
0Ch	x	x	In Filter			ADC		x
0Eh	x	x	DNFI	x	x	x	VSTC	x
20h	Reserved							
22h	Master Volume L				Master Volume R			
24h	Reserved							
26h	FM Volume L				FM Volume R			
28h	CD Volume L				CD Volume R			
2Ah	Reserved							
2Ch	Reserved							
2Eh	Line Volume L				Line Volume R			

x=don't care
Reserved=preserve original value

Register Descriptions:

Reset Register (00h):
  You can write any 8-bit value to this register to reset the mixer

Voice Volume Register (04h):
  d[7..4] voice volume left
  d[3..0] voice volume right
  The default level is 9.

Microphone Mixing Register (0Ah):
  The default level is 0.

Input Setting Register (0Ch):
  ADC - input source:
  d[2] d[1]
    0   0   Microphone (default)
    0   1   CD audio
    1   0   Microphone
    1   1   Line-in
  In Filter - filter select:
  d[5] d[4] d[3]
    0   x   0   Low Filter (default)
    0   x   1   High Filter
    1   x   x   No Filter

Output Setting Register (0Eh):
  This register specifies the voice output features.
  DNFI 0 output filter is ON (default)
        1 output filter is bypassed
  VSTC 0 Mono voice mode (default)
        1 stereo voice mode

Master Volume Register (22h):
  d[7..4] master volume left
  d[3..0] master volume right
  The default level is 9.

FM Volume Register (26h):
  d[7..4] fm volume left
  d[3..0] fm volume right
  The default level is 9.

CD Volume Register (28h):
  d[7..4] cd volume left
  d[3..0] cd volume right
  The default level is 0

Line-In Volume Register (2Eh):
  d[7..4] line-in volume left
  d[3..0] line-in volume right
  The default level is 0

Rebooting:

   40:72h holds 1234h for warm boot, and 4321h (or != 1234h) for cold boot
   jmp to F000:FFF0h

Speaker:
   Port 61h read (ISA, EISA)
   Bit [7] = Parity check (parity error)
   Bit [6] = Channel check (ISA parity error)
   Bit [5] = Timer 2 output
   Bit [4] = Toggles with each refresh request
   Bit [3] = Channel check enable (enable ISA parity check)
   Bit [2] = Parity check enable (enable parity check)
   Bit [1] = Speaker data enable (Timer 2 output enable)
   Bit [0] = Speaker timer enable (Timer 2 gate enable)

   Port 61h write (ISA, EISA)
   Bit [7-4] = Reserved
   Bit [3] = Channel check enable (enable ISA parity check)
   Bit [2] = Parity check enable (enable parity check)
   Bit [1] = Speaker data enable (Timer 2 output enable)
   Bit [0] = Speaker timer enable (Timer 2 gate enable)

   Port 61h write (XT only)
   Bit [7] = 1 Clear keyboard
   Bit [6] = 0 Hold keyboard clock low
   Bit [5] = 0 I/O check enable
   Bit [4] = 0 RAM parity check enable
   Bit [3] = 0 Read low switches
   Bit [2] = Reserved
   Bit [1] = 1 Speaker data enable (Timer 2 output enable)
   Bit [0] = 1 Speaker timer enable (Timer 2 gate enable)

In Windows/DOS Developer's Journal December 1991, Robert Bybee wrote an article entitled 'Writing For The PC Speaker'. The program SPEAKER.EXE is a butchered version of his program.

-The physics:

   The PC can write individual levels to the speaker directly (a 1 or a 0). It is also true that the PC can write these bits much faster than the speaker can respond to them. At these speeds, the speaker becomes an integrator.

-1-Bit DACs:

   The latest craze in CD Players is 1-Bit dacs. What is it? Well...a 1-bit DAC really is not a DAC at all. The sample is dithered (I will use that term from here on out, even if it is wrong) with a bit pattern whose integral (assume NRZL square wave) matches the level of the sample. This n-length bit pattern is sent one bit at a time into an integrator at a bit rate of n (pattern length) times the sample rate. From the integrator to the pre-amp to the amp, etc...

For example a 3-bit sample would require 8 discrete levels, which would require a 7-bit dither pattern:

dither		input
1111111	<--	111
1110111	<--	110
1101101	<--	101
1010101	<--	100
0101010	<--	011
0010010	<--	010
0001000	<--	001
0000000	<--	000

-Good quality PC Speaker sound:

Why not use the PC Speaker as an integrator for a 1-bit DAC? Turns out to give pretty good quality sound. The example program does not attempt to find out how fast your particular machine is. You must adjust the bit pattern length until it sounds right, or figure it out the machine speed yourself. With my 486DX2-66 I was running so fast that the speaker was overwhelmed and the volume was very low. Adding a few NOPs to the end of the ophigh[] and oplow[] strings should slow it down some. This will give fewer bits per second which will give

better volume, but worse quality.

8254 Programmable Interval Timer (PIT)

```
   Counter 0:  System timer, Always on, 1.193MHz, IRQ0
   Counter 1:  Refresh request, Always on, 1.193MHz, Request refresh
   Counter 2:  Speaker frequency, enable speaker, 1.193MHz, Speaker signal

   Port 40h  Read/Write    Counter 0
   Port 41h  Read/Write    Counter 1
   Port 42h  Read/Write    Counter 2
   Port 43h  Write/Only    Control register
       Bits [7..6]     Select counter
           00          Counter 0
           01          Counter 1
           10          Counter 2
           11          Read back command

       Bits [5..4]     Operation
           00          Latch counter (bits[3..1] are don't care)
           01          Read/Write LSB only
           10          Read/Write MSB only
           11          Read/Write LSB, then MSB

       Bits [3..1]     Operating mode selection
           000         Mode 0:  interrupt on terminal count
           001         Mode 1:  hardware triggered one-shot
           x10         Mode 2:  rate generator
           x11         Mode 3:  square-wave generator
           100         Mode 4:  software triggered strobe
           101         Mode 5:  hardware triggered strobe
           xxx         For latch counter operation

       Bit [0]         Binary or BCD count down format
           0           Binary (16-bit) count down
           1           BCD count down (four-decades)
           x           For latch counter operation
   Port 43h  Write/Only    Control Register (read-back command)
       Bits [7..6]
           11          Read back command

       Bits [5..4]
           0x          latches the state of the CE in COL and COH
           x0          latches status of selected counters into the
                           status register.

       Bits [3..0]
           1xx0        Selects counter 2
           x1x0        Selects counter 1
           xx10        Selects counter 0

   If status is latched, status is read first through counter read/write
   register.
   If count is latched, the count is read back through counter read/write
   retister, one or two reads depending on how programmed.

   Status Byte Latched
       Bit [7]
           0           OUT signal 0 (low)
           1           OUT signal 1 (high)
       Bit [6]
           0           Counter loaded from the counter input registers,
                           count can be read.
           1           Write to the congtrol register or the counter, but
                           the new value has not been loaded into CE.
       Bits [5..4]
           00          Reserved
           01          Read/Write LSByte
           10          Read/Write MSByte
           11          Read/Write LSByte then MSByte
       Bits [3..1]
           000         Mode 0
           001         Mode 1
           010         Mode 2
           011         Mode 3
           100         Mode 4
           101         Mode 5
```

Bit [0]

0	Binary (16-bit) count down
1	BCD count down (four decades)

1.19318MHz = 4.77273MHz/4. Typically Counter 0 is set for 0000h (max count) 54.925ms. There are 1573040 tics per day.

This is an example code segment that shows how to speed up the timer resolution. I had no problem with this piece of code on a 486DX2-66, on slower machines you may need to slow it down.

```
---- cut ----
.model tiny
.code
org 100h
entry:
    jmp over
old08vec dd ?
keeptime dw 0000h
isr08h:
    inc word ptr cs:[keeptime]
    ; at this point, you have a sped up interrupt
    ; be careful not overload the processor
    cmp cs:[_counter],0FFFFh
    jz notenabled
notenabled:
    cmp word ptr cs:[keeptime],1000h
    jz handoff
    push ax
    mov al,20h
    out 20h,al
    pop ax
    iret
handoff:
    mov word ptr cs:[keeptime],0000h
    jmp dword ptr cs:[old08vec]
over:
    mov ax,3508h
    int 21h
    mov word ptr cs:[old08vec],bx
    mov word ptr cs:[old08vec+2],es
    mov ax,2508h
    mov dx,offset isr08h
    ;tiny no ds
    int 21h
    mov al,10h
    out 40h,al
    mov al,00h
    out 40h,al
    mov ah,00h
    int 16h
    xor al,al
    out 40h,al
    out 40h,al
    mov ax,2508h
    mov dx,word ptr cs:[old08vec]
    mov ds,word ptr cs:[old08vec+2]
    int 21h
    mov ax,4C00h
    int 21h

end entry
---- cut ----
```

The speaker timer (counter 2) can also be used for high resolution timing for periods less than 55ms total, using this timer avoids the potential harm to the system like Counter 0 or 1. One implementation might be:

Enable timer 2 gate, disable timer 2 output:
out(61h,01h)
Set Counter 2 for interrupt on terminal count:
out(43h,B0h)
Set MSB to start count and halt counting:
out(42h,FFh)
Set LSB to start count and start counting:
out(42h,FFh)
Perform task to be timed (with no speaker access)
Latch count:
out(43h,C0h)
Read count:
LSByte=in(42h)

```
MSByte=in(42h)
```

8259 Programmable Interrupt Controllers (PICs)

```
   Port 20h    write a 20h to clear interrupts (during isr)
               can write a 60h+irq, but that is not necessary if
               the PIC is configured for modes that preserve the fully
               nested structure (which is how you will find it normally)
   Port 21h    read/write, enables and disables irqs 0-7,
               0 = enabled, 1 = disabled
               Bit [7]  irq 7
               Bit [6]  irq 6
               Bit [5]  irq 5
               Bit [4]  irq 4
               Bit [3]  irq 3
               Bit [2]  irq 2
               Bit [1]  irq 1
               Bit [0]  irq 0

   Port A0h    write a 20h to clear interrupts (during isr)
               can write 60h+irq-8, see above (Port 20h)
   Port A1h    read/write, enables and disables irqs 8-15,
               0 = enable, 1 = disable
               Bit [7]  irq 15
               Bit [6]  irq 14
               Bit [5]  irq 13
               Bit [4]  irq 12
               Bit [3]  irq 11
               Bit [2]  irq 10
               Bit [1]  irq  9
               Bit [0]  irq  8
```

8237 DMA Controller (DMAC)

```
   Port 0000h  DMA Channel 0, Memory address register (r/w)
               16-bit, LSByte first then MSByte
   Port 0001h  DMA Channel 0, Transfer count register (r/w)
               16-bit, LSByte first then MSByte
   Port 0002h  DMA Channel 1, Memory address register (r/w)
   Port 0003h  DMA Channel 1, Transfer count register (r/w)
   Port 0004h  DMA Channel 2, Memory address register (r/w)
   Port 0005h  DMA Channel 2, Transfer count register (r/w)
   Port 0006h  DMA Channel 3, Memory address register (r/w)
   Port 0007h  DMA Channel 3, Transfer count register (r/w)
   Port 0008h  DMA Status register (r)
               bit [7] = 1 Channel 3 request
               bit [6] = 1 Channel 2 request
               bit [5] = 1 Channel 1 request
               bit [4] = 1 Channel 0 request
               bit [3] = 1 Terminal count on channel 3
               bit [2] = 1 Terminal count on channel 2
               bit [1] = 1 Terminal count on channel 1
               bit [0] = 1 Terminal count on channel 0
   Port 000Ah  DMA Channel 0-3 Mask register (r/w)
               bits [7..3] = 0 reserved
               bit [2] = 0 Clear mask bit
                         1 Set mask bit
               bits [1..0] = 00 Select channel 0
                             01 Select channel 1
                             10 Select channel 2
                             11 Select channel 3
   Port 000Bh  DMA Channel 0-3 Mode register (r/w)
               bits [7..6] = 00 Demand mode
                             01 Single mode
                             10 Block mode
                             11 Cascade mode
               bit [5] = 0 Address increment select
               bit [4] = 0 Autoinitialized
                         1 Non-autoinitialized
               bits [3..2] = 00 Verify operation
                             01 Write operation
                             10 Read operation
                             11 Reserved
               bits [1..0] = 00 Select channel 0
                             01 Select channel 1
                             10 Select channel 2
                             11 Select channel 3
   Port 000Ch  DMA Clear Byte Pointer (w)
   Port 000Dh  DMA Master Clear (w)
   Port 000Eh  DMA Clear Mask register (w)
   Port 000Fh  DMA DMA Channel 0-3 Write Mask register (w)
               bits [7..4] = 0 reserved
               bit [3] = 0 Unmask channel 3 mask bit
                         1 Set channel 3 mask bit
                         0 Unmask channel 2 mask bit
                         1 Set channel 2 mask bit
                         0 Unmask channel 1 mask bit
                         1 Set channel 1 mask bit
                         0 Unmask channel 0 mask bit
                         1 Set channel 0 mask bit
   Port 0019h  DMA Scratch register (r/w)
   Port 0081h  DMA channel 2, page table address register (r/w)
   Port 0082h  DMA channel 3, page table address register (r/w)
   Port 0083h  DMA channel 1, page table address register (r/w)
   Port 0087h  DMA channel 0, page table address register (r/w)
```

Notes:

First off, DMA can only access the lower meg (some channels can access 16M on ATs) which translates to a 20-bit address. Secondly, DMA cannot read/write across a physical page boundary (address with lower 16-bits = 0). This also means that DMA cannot transfer more than 64K (actually 65535 bytes) bytes at a time. To guarantee no boundary conflicts, many programmers allocate 128KBytes, and find the first page boundary within that memory space. Here is an example in Borland 'C' to get 65000 bytes:

```
---- cut ----
unsigned char *data;
unsigned char *aligned;
unsigned long aligned_physical;
...
    unsigned long physical;
    data=farmalloc(131000L);
    if(data==NULL)
    {
        printf("Memory Allocation Error\n");
        exit(1);
    }
    physical=((unsigned long)FP_OFF(data))+
        (((unsigned long)FP_SEG(data))<<4);
    aligned_physical=physical+0x0FFFFL;
    aligned_physical&=0xF0000L;
    aligned=MK_FP((aligned_physical>>4)&0xFFFF,0);
---- cut ----
```

to have a device read len bytes from the above memory (*aligned) (DMA channel 1):

```
---- cut ----
len--;
outportb(0x0A,0x05);
outportb(0x0C,0x00);
outportb(0x0B,0x49);
outportb(0x02,0);
outportb(0x02,0);
outportb(0x83,(aligned_physical>>16)&15);
outportb(0x03,(unsigned char)(len&0xFF));
outportb(0x03,(unsigned char)((len>>8)&0xFF));
outportb(0x0A,0x01);
---- cut ----
```

to have a device write len bytes from the above memory (*aligned) (DMA channel 1):

```
---- cut ----
len--;
outportb(0x0A,0x05);
outportb(0x0C,0x00);
outportb(0x0B,0x45);
outportb(0x02,0);
outportb(0x02,0);
outportb(0x83,(aligned_physical>>16)&15);
outportb(0x03,(unsigned char)(len&0xFF));
outportb(0x03,(unsigned char)((len>>8)&0xFF));
outportb(0x0A,0x01);
---- cut ----
```

Programming PC DMA Controller (8237)
Josh Cohen
CSC 390 Special Topics
H. Barada

The process, described in detail later, for programming DMA transfers is somewhat complex.

General Steps:

Set up a buffer to read from or write to.
Disable the channel you wish to set up.
Set the Page Register to the page which you want to transfer.
Clear the Byte Pointer.
Set Base Address register to the address of the beginning of the buffer.
Set the Base Word Count register to number of words you wish to transfer.
Set the DMA transfer Mode.
Enable DMA channel to start transfer.

Details:

Registers of the 8237 DMA controller
The DMA controller has several channels. Channel 1 is best for user transfers as some others are used by the system.

Address	Description
00Ah	Channel mask (enable/disable)
083h	Page Register
00Ch	Clear Byte pointer flipflop
002h	Base Address 16 bit register
003h	Base Word count 16 bit register
00Bh	Transfer mode register

To set up DMA controller

Set up a buffer to read from or write to.
Since the DMA controller does everything in 64k blocks, it is best to try to align the buffer to the segment edge, I.e. Segment A offset 0. This can be difficult depending on the language which we use.

Disable the channel you wish to set up.
If we use channel 1 we send 05h to address 00A (page register)

Set the Page Register 00A to the page which you want to transfer.
The pages are 64k blocks in main memory. We must setup multiple transfers if we wish to transfer more than 64k since the controller is only able to do 64k at a time.
The pages are 00h to FFh starting with 0000-FFFF. Notice that these are the same pages that we call segments. When we specify addresses we are only talking about the offsets from the segment, not the entire address.

Clear the Byte Pointer.
The clear byte pointer register is a flip flop. Writing to the register clears it, the actual data which we write is ignored.
Write any value to 00Ch

Set Base Address register to the address of the beginning of the buffer.
To do this we write the offset from the segment edge of the starting address of our buffer, this will be 0 if we have aligned the buffer to the segment edge.
Write this offset to 002h

Set the Base Word Count register to number of words you wish to transfer.
This register gets the number of words which we wish to transfer.
Usually this will be bytes.
Write this number to 003h

Set the DMA transfer Mode.
This register controls the mode of transfer, it will vary upon the device which we want to transfer to. In our case, using the ThunderBoard Sound Card the possible modes are:
49h Non-continuous playback
59h Continuous playback
45h Non-continous record
55h Continuous record
By playback we mean transfer data from memory out to the card
By record we mean transfer data from the card to memory.
Notice that we are sending out or receiving data from the bus. The computer really does not care or know where the data is going or coming from. We must also set up the device we are transferring with. In this case it is the ThunderBoard. See the next document on how to do that.

Enable DMA channel to start transfer.
Now that we have correctly set up the DMA controller for the transfer, we can enble the channel. However, we must set up the card to receive or send the data first. Usually, in practice, the card is set up before the DMA controller.

Programming the AdLib/Sound Blaster FM Music Chips

Version 2.0 (24 Feb 1992)

jlee@smylex.uucp

Warranty and Copyright Policy

Overview

Two of the most popular sound cards for the IBM-PC, the AdLib and the Sound Blaster, suffer from a real dearth of clear documentation for programmers. AdLib Inc. and Creative Labs, Inc. both sell developers' kits for their sound cards, but these are expensive, and (in the case of the Sound Blaster developers' kit) can be extremely cryptic.

This document is intended to provide programmers with a FREE source of information about the programming of these sound cards.

The information contained in this document is a combination of information found in the Sound Blaster Software Developer's Kit, and that learned by painful experience. Some of the information may not be valid for AdLib cards; if this is so, I apologize in advance.

Please note that numbers will be given in hexadecimal, unless otherwise indicated. If a number is written out longhand (sixteen instead of 16) it is in decimal.

| Changes from Version 1 of the file will be indicated by the use of change
| bars in the left-hand margin.

Chapter One - Sound Card I/O

The sound card is programmed by sending data to its internal registers via its two I/O ports:

```
0388 (hex) - Address/Status port  (R/W)
0389 (hex) - Data port            (W/O)
```

| The Sound Blaster Pro is capable of stereo FM music, which is accessed
| in exactly the same manner. Ports 0220 and 0221 (hex) are the address/
| data ports for the left speaker, and ports 0222 and 0223 (hex) are the
| ports for the right speaker. Ports 0388 and 0389 (hex) will cause both
| speakers to output sound.

The sound card possesses an array of two hundred forty-four registers; to write to a particular register, send the register number (01-F5) to the address port, and the desired value to the data port.

After writing to the register port, you must wait twelve cycles before sending the data; after writing the data, eighty-four cycles must elapse before any other sound card operation may be performed.

| The AdLib manual gives the wait times in microseconds: three point three
| (3.3) microseconds for the address, and twenty-three (23) microseconds
| for the data.
|
| The most accurate method of producing the delay is to read the register
| port six times after writing to the register port, and read the register
| port thirty-five times after writing to the data port.

The sound card registers are write-only.

The address port also functions as a sound card status byte. To retrieve the sound card's status, simply read port 388. The status byte has the following structure:

7	6	5	4	3	2	1	0
both tmrs	tmr 1	tmr 2	unused				

Bit 7 - set if either timer has expired.
6 - set if timer 1 has expired.
5 - set if timer 2 has expired.

Chapter Two - The Registers

The following table shows the function of each register in the sound card. Registers will be explained in detail after the table. Registers not listed are unused.

Address	Function
01	Test LSI / Enable waveform control
02	Timer 1 data
03	Timer 2 data
04	Timer control flags
08	Speech synthesis mode / Keyboard split note select
20..35	Amp Mod / Vibrato / EG type / Key Scaling / Multiple
40..55	Key scaling level / Operator output level
60..75	Attack Rate / Decay Rate
80..95	Sustain Level / Release Rate
A0..A8	Frequency (low 8 bits)
B0..B8	Key On / Octave / Frequency (high 2 bits)
BD	AM depth / Vibrato depth / Rhythm control
C0..C8	Feedback strength / Connection type
E0..F5	Wave Select

The groupings of twenty-two registers (20-35, 40-55, etc.) have an odd order due to the use of two operators for each FM voice. The following table shows the offsets within each group of registers for each operator.

Channel	1	2	3	4	5	6	7	8	9
Operator 1	00	01	02	08	09	0A	10	11	12
Operator 2	03	04	05	0B	0C	0D	13	14	15

Thus, the addresses of the attack/decay bytes for channel 3 are 62 for the first operator, and 65 for the second. (The address of the second operator is always the address of the first operator plus three).

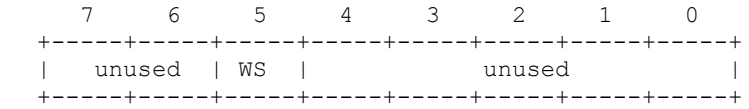
To further illustrate the relationship, the addresses needed to control channel 5 are:

29 - Operator 1 AM/VIB/EG/KSR/Multiplier
2C - Operator 2 AM/VIB/EG/KSR/Multiplier
49 - Operator 1 KSL/Output Level
4C - Operator 2 KSL/Output Level
69 - Operator 1 Attack/Decay
6C - Operator 2 Attack/Decay
89 - Operator 1 Sustain/Release
8C - Operator 2 Sustain/Release
A4 - Frequency (low 8 bits)
B4 - Key On/Octave/Frequency (high 2 bits)
C4 - Feedback/Connection Type

E9 - Operator 1 Waveform
EC - Operator 2 Waveform

Explanations of Registers

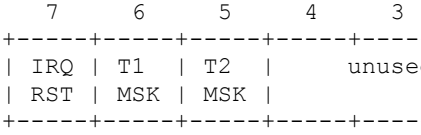
Byte 01 - This byte is normally used to test the LSI device. All bits should normally be zero. Bit 5, if enabled, allows the FM chips to control the waveform of each operator.



Byte 02 - Timer 1 Data. If Timer 1 is enabled, the value in this register will be incremented until it overflows. Upon overflow, the sound card will signal a TIMER interrupt (INT 08) and set bits 7 and 6 in its status byte. The value for this timer is incremented every eighty (80) microseconds.

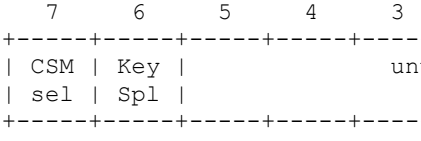
Byte 03 - Timer 2 Data. If Timer 2 is enabled, the value in this register will be incremented until it overflows. Upon overflow, the sound card will signal a TIMER interrupt (INT 08) and set bits 7 and 5 in its status byte. The value for this timer is incremented every three hundred twenty (320) microseconds.

Byte 04 - Timer Control Byte



bit 7 - Resets the flags for timers 1 & 2. If set, all other bits are ignored.
bit 6 - Masks Timer 1. If set, bit 0 is ignored.
bit 5 - Masks Timer 2. If set, bit 1 is ignored.
bit 1 - When clear, Timer 2 does not operate. When set, the value from byte 03 is loaded into Timer 2, and incrementation begins.
bit 0 - When clear, Timer 1 does not operate. When set, the value from byte 02 is loaded into Timer 1, and incrementation begins.

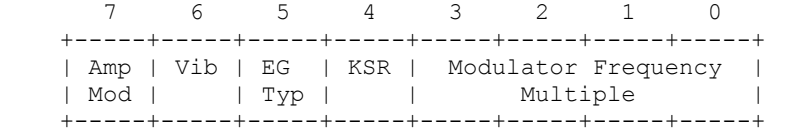
Byte 08 - CSM Mode / Keyboard Split.



bit 7 - When set, selects composite sine-wave speech synthesis mode (all KEY-ON bits must be clear). When clear, selects FM music mode.

bit 6 - Selects the keyboard split point (in conjunction with the F-Number data). The documentation in the Sound Blaster manual is utterly incomprehensible on this; I can't reproduce it without violating their copyright.

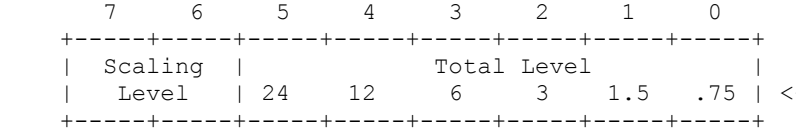
Bytes 20-35 - Amplitude Modulation / Vibrato / Envelope Generator Type / Keyboard Scaling Rate / Modulator Frequency Multiple



bit 7 - Apply amplitude modulation when set; AM depth is controlled by the AM-Depth flag in address BD.
bit 6 - Apply vibrato when set; vibrato depth is controlled by the Vib-Depth flag in address BD.
bit 5 - When set, the sustain level of the voice is maintained until released; when clear, the sound begins to decay immediately after hitting the SUSTAIN phase.
bit 4 - Keyboard scaling rate. This is another incomprehensible bit in the Sound Blaster manual. From experience, if this bit is set, the sound's envelope is foreshortened as it rises in pitch.
bits 3-0 - These bits indicate which harmonic the operator will produce sound (or modulation) in relation to the voice's specified frequency:

0 - one octave below
1 - at the voice's specified frequency
2 - one octave above
3 - an octave and a fifth above
4 - two octaves above
5 - two octaves and a major third above
6 - two octaves and a fifth above
7 - two octaves and a minor seventh above
8 - three octaves above
9 - three octaves and a major second above
A - three octaves and a major third above
B - " " " " " " "
C - three octaves and a fifth above
D - " " " " " "
E - three octaves and a major seventh above
F - " " " " " " "

Bytes 40-55 - Level Key Scaling / Total Level

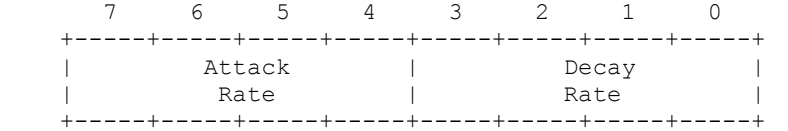


bits 7-6 - causes output levels to decrease as the frequency rises:

00 - no change
10 - 1.5 dB/8ve
01 - 3 dB/8ve
11 - 6 dB/8ve

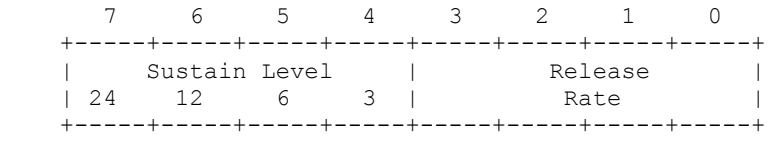
bits 5-0 - controls the total output level of the operator. all bits CLEAR is loudest; all bits SET is the softest. Don't ask me why.

Bytes 60-75 - Attack Rate / Decay Rate



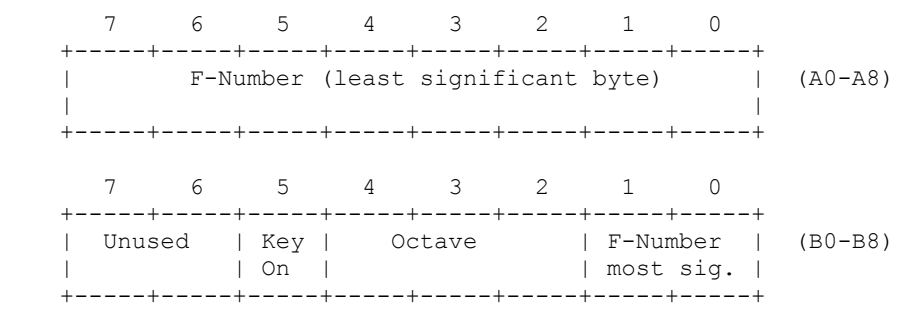
bits 7-4 - Attack rate. 0 is the slowest, F is the fastest.
bits 3-0 - Decay rate. 0 is the slowest, F is the fastest.

Bytes 80-95 - Sustain Level / Release Rate



bits 7-4 - Sustain Level. 0 is the loudest, F is the softest.
bits 3-0 - Release Rate. 0 is the slowest, F is the fastest.

Bytes A0-B8 - Octave / F-Number / Key-On

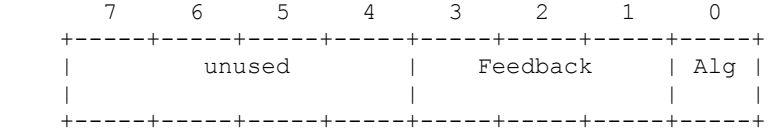


bit 5 - Channel is voiced when set, silent when clear.
bits 4-2 - Octave (0-7). 0 is lowest, 7 is highest.
bits 1-0 - Most significant bits of F-number.

In octave 4, the F-number values for the chromatic scale and their corresponding frequencies would be:

F Number	Frequency	Note
16B	277.2	C#
181	293.7	D
198	311.1	D#
1B0	329.6	E
1CA	349.2	F
1E5	370.0	F#
202	392.0	G
220	415.3	G#
241	440.0	A
263	466.2	A#
287	493.9	B
2AE	523.3	C

Bytes C0-C8 - Feedback / Algorithm



bits 3-1 - Feedback strength. If all three bits are set to zero, no feedback is present. With values 1-7, operator 1 will send a portion of its output back into itself. 1 is the least amount of feedback, 7 is the most.
bit 0 - If set to 0, operator 1 modulates operator 2. In this case, operator 2 is the only one producing sound. If set to 1, both operators produce sound directly. Complex sounds are more easily created if the algorithm is set to 0.

Byte BD - Amplitude Modulation Depth / Vibrato Depth / Rhythm

7	6	5	4	3	2	1	0
AM Dep	Vib Dep	Rhy Ena	BD	SD	TOM	Top Cym	HH

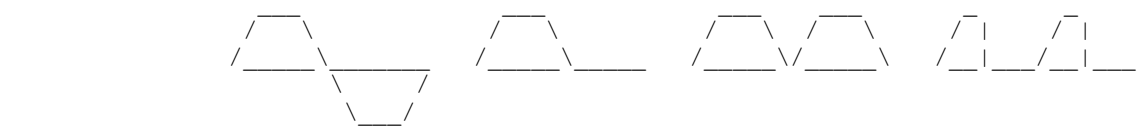
bit 7 - Set: AM depth is 4.8dB
Clear: AM depth is 1 dB
bit 6 - Set: Vibrato depth is 14 cent
Clear: Vibrato depth is 7 cent
bit 5 - Set: Rhythm enabled (6 melodic voices)
Clear: Rhythm disabled (9 melodic voices)
bit 4 - Bass drum on/off
bit 3 - Snare drum on/off
bit 2 - Tom tom on/off
bit 1 - Cymbal on/off
bit 0 - Hi Hat on/off

Note: KEY-ON registers for channels 06, 07, and 08 must be OFF in order to use the rhythm section. Other parameters such as attack/decay/sustain/release must also be set appropriately.

Bytes E0-F5 - Waveform Select

7	6	5	4	3	2	1	0
unused						Waveform Select	

bits 1-0 - When bit 5 of address 01 is set, the output waveform will be distorted according to the waveform indicated by these two bits. I'll try to diagram them here, but this medium is fairly restrictive.



00 01 10 11

| Detecting a Sound Card
|
| According to the AdLib manual, the 'official' method of checking for a sound card is as follows:
|
| 1) Reset both timers by writing 60h to register 4.
| 2) Enable the interrupts by writing 80h to register 4. NOTE: this must be a separate step from number 1.
| 3) Read the status register (port 388h). Store the result.
| 4) Write FFh to register 2 (Timer 1).
| 5) Start timer 1 by writing 21h to register 4.
| 6) Delay for at least 80 microseconds.
| 7) Read the status register (port 388h). Store the result.
| 8) Reset both timers and interrupts (see steps 1 and 2).
| 9) Test the stored results of steps 3 and 7 by ANDing them with E0h. The result of step 3 should be 00h, and the result of step 7 should be C0h. If both are correct, an AdLib-compatible board is installed in the computer.
|
|

Making a Sound

Many people have asked me, upon reading this document, what the proper register values should be to make a simple sound. Well, here they are.

First, clear out all of the registers by setting all of them to zero. This is the quick-and-dirty method of resetting the sound card, but it works. Note that if you wish to use different waveforms, you must then turn on bit 5 of register 1. (This reset need be done only once, at the start of the program, and optionally when the program exits, just to make sure that your program doesn't leave any notes on when it exits.)

Now, set the following registers to the indicated value:

REGISTER	VALUE	DESCRIPTION
20	01	Set the modulator's multiple to 1
40	10	Set the modulator's level to about 40 dB
60	F0	Modulator attack: quick; decay: long
80	77	Modulator sustain: medium; release: medium
A0	98	Set voice frequency's LSB (it'll be a D#)
23	01	Set the carrier's multiple to 1
43	00	Set the carrier to maximum volume (about 47 dB)
63	F0	Carrier attack: quick; decay: long
83	77	Carrier sustain: medium; release: medium
B0	31	Turn the voice on; set the octave and freq MSB

To turn the voice off, set register B0h to 11h (or, in fact, any value which leaves bit 5 clear). It's generally preferable, of course, to induce a delay before doing so.

Acknowledgements

Thanks are due to the following people:

Ezra M. Dreisbach (ed10+@andrew.cmu.edu), for providing the information about the recommended port write delay from the AdLib manual, and the 'official' method of detecting an AdLib-compatible sound card.

Nathan Isaac Laredo (gt7080a@prism.gatech.edu), for providing the port numbers for stereo sound on the Sound Blaster Pro.