HBSound Programmers Document

This document describes the specific locations of the address space for the flags, control and data io for the HBSound DAC based sound card for the HB6809. This is intended to be rudimentary and non-specific as the interface can be installed in any one of the 4 slots on the computer.

Reading the flags latch:

The flags latch is located on the base address for the slot, so for instance if the card is loaded into slot 0x9000 (the space the demo code expects); when you read this address space it should return a 2 bit value:

- 01 – register is empty
- 00 – register is emptying with data loaded
- 10 – register is full (decimal / HEX value is 2)

This value is latched on the rising edge of the system clock and represents the state of the clock at that moment. Since the addressing is only partially decoded for the interface; you can technically read from any address in the full range and it will return the value of the flags inside the interface.

Setting the control register / Understanding clock generation:

The DAC gets its sample clock on the falling edge of the system clock when the 8 bit counter hits specific addresses which output a zero on the LSB of the ROMS Q outputs. This signal also resets the counter latches on the rising edge of the next clock cycle. The resulting low value from the EEPROM is OR'd with the system clock so that the data is only shifted out on the falling edge of the clock. This prevents the flags register for HBSound from updating when the flags are in a unknown state.

This clock can be set by referencing the following (assuming a 1843200 Hz System clock):

Sample Rt	Dec Count	HEX Count	BIN Count	OPCODE	HEX Address For Bit
32000	56.6	38	0011 1000	00	38
16000	114.2	72	0111 0010	01	172
8000	229.4	E5	1110 0101	10	2E5
7200	255	FF	1111 1111	11	3FF

The resulting OPCODE must be set in the control register on the device to change the clock frequency. This can be accomplished by writing a value to the base address of the device + 1. So in our example program this address is 0x9001. Furthermore you can control the DAC clock output by setting a '1' in the 3rd least significant bit.

For instance setting the clock to the lowest speed, and halting the clock would be done by sending a 111 to the interface (DEC 7), sending 011 (DEC 3) would start the clock again.

Sending sample data to the interface:

Sample data can be sent to the interface by writing a byte to the base address. The control logic and the flexible register will maintain that data in a frame of 16 bytes. You can determine the frame is full using a “write and check” technique. This is described in the example program with the following C subroutine:

```
/* Fill up the buffer till flags indicate it's full.
   *flags & *sample are pointers set by #DEFINE statements
   The pointer data tells the compiler where the flags and
   sample data should go in memory.                       */

void fillBuffer (unsigned char maxFrequencyCount) {
    while (*flags != 2) {       // while not full
        if (maxFrequencyCount == 0)  {  // handles control code
            frequencyState = 0;
            goto CNTCODE;
        }

        frequencyCount++;

         /* switches between high and low when 50% of the period is
            reached. */
        if (frequencyCount == maxFrequencyCount) {
            frequencyState = !frequencyState;
            frequencyCount = 0;
        }
        CNTCODE:
        *sample = frequencyState;
    }
}
```

The above example describes a square wave generation technique but the data being sent could be any value that the DAC can understand. Unlike the flags latch, the logic controlling the shift in signal for the FIFO requires that the data be written to the base address of the device, as base address+1 is the control latch and the value of the LSB on the address bus is used to determine where the data is going.

Signals other than address / data bus and unselected persistence:

There are three signals on the interface which have persistence inside the device while it is not selected. These are the E clock, the system reset and the R//W line. The remaining signals are in tri-state when the device is not selected due to a pair of bus transceivers which tie the device to the bus. The R//W line selects the direction of the bus transceivers, but is also buffered for use on the inside of the device when selected. The E system clock is used to increment the 8 bit counter for the clock generation, and in the clock synchronization circuit which is used to generate the shift out signal for the FIFO. The reset line resets all the internal registers and is inverted to create a active high reset line for the FIFO.