NOTE 2

Recommended buffering practices for AudioScience WAV drivers

1. Introduction
This application note discusses the ramifications of various buffering methodologies pertaining to the playback and recording of digital audio using the AudioScience WAV driver and Microsoft's Multimedia API.

The structure of the AudioScience driver software and adapter hardware is briefly outlined. Typical playback operation is described and, finally, particular buffer sizes recommended. Although the treatment here primarily addresses the playback case, the same arguments hold for recording. In the following discussion the term "on-board" buffer refers to buffer memory on the digital audio adapter.

2. Driver Structure
AudioScience digital audio adapters have a minimum of 128kBytes of on-board buffering per device and at least 400kBytes is typical. Obviously, this memory partitioning differs from that promoted by SoundBlaster and other compatible digital audio solutions. The AudioScience WAV driver "hides" hardware buffering details from the user application and attempts to fully utilize the on-board audio buffers to minimize the possibility of audio drop out.

During playback the AudioScience WAV driver fills the on-board device buffers in an controlled manner. If all 0.5MBytes of a device's on-board buffer were to be filled at once, there would be a significant peak in both buffer handling and disk accesses required that may come at the expense of correct playback on one of the other devices. The AudioScience driver therefore trickle fills the on-board device buffers at a faster rate than the playback consumption rate. After a few seconds the device's on-board buffer becomes completely full (subject to conditions outlined in 3, below).

3. Playback Buffering Options
The AudioScience WAV driver supports two distinct, user selectable, playback buffering mechanisms. The two settings allow the user to select whether the buffers returned by the MM_WODM_DONE message have been completely played, or have been transferred to the adapter's hardware buffers.

The default mechanism is defined by:
Hardware Buffering=off
in the ASIWAV.INI (driver 3.01 and earlier), ASIDRV.INI (driver 3.02 to 4.00), or ASIDRV.TXT (driver 4.02.01 and later) initialization file (in C:\Windows). In this configuration the adapter's hardware buffers hold an image of information contained in the user application's audio buffers. The maximum adapter buffering utilized is limited to the amount of buffering provided by user audio buffers. Audio buffers are return to the application only after they have been played.

Maximum use of the adapter's on-board buffers is enabled by setting:
Hardware Buffering=on
in the ASIWAV.INI (driver 3.01 and earlier), ASIDRV.INI (driver 3.02 to 4.00), or ASIDRV.TXT (driver 4.02.01 and later) initialization file (in C:\Windows). This allows audio buffers to be returned to the application before playback has been completed. The last buffer is only returned after it has been completely played.

In summary, Hardware Buffering settings:
on - buffers being returned will get several seconds ahead of audio being played.
off - buffers are returned only after they are played.

4. Buffering Recommendations

Based on the information provide in section 2., it is obvious that the AudioScience WAV driver will work best if large “chunks” of data can be handled at once. This has the following advantages:

- It allows the driver to take advantage of the fact that the large on-board device buffers can be written (or read) in a single operation.
- The Windows messaging sub-system overhead is reduced because many small buffer write/done messages are replaced by fewer write/done messages.

AudioScience recommends large buffers in the size range of 16k to 64k[1]. The number of buffers should be some number greater than 3.

There are no known disadvantage to using large size buffers. AudioScience uses a mechanism on the adapter to count output samples played, so the returned value from a call to waveOutGetPos() is independent of the buffersize.

[1] Buffer sizes should be an exact multiple of 4 bytes. Other sizes will not work correctly due to the 32bit PCI interface.