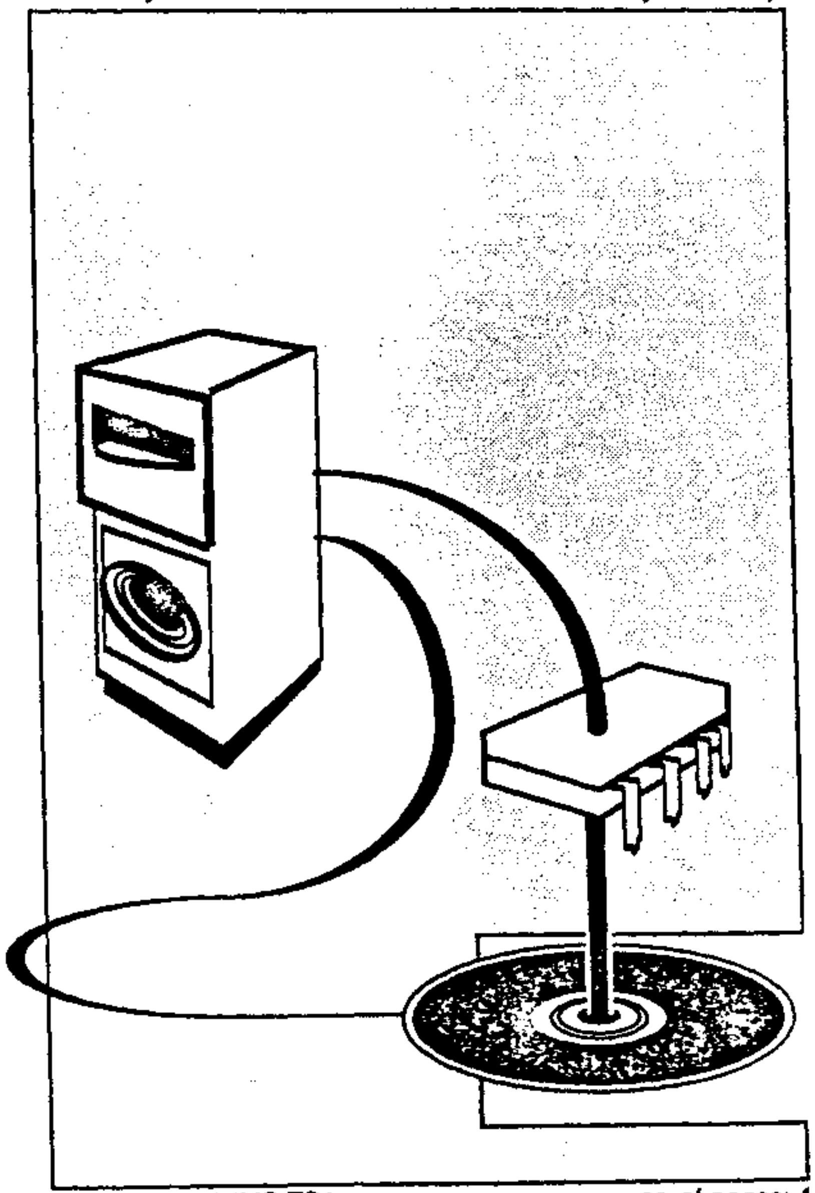
TN 92

TSA APPLICATION NOTES NR. TSA-001 05-04-1992

Audio Considerations in CD-I

In CD-I there are two mechanisms for audio playback, direct and through soundmaps. Each of them and especially the combination have some caveats. This note describes what trade-offs play a role in this area.

Written by *Alty van Luijt*



nr. of pages: 4

1. Introduction.

In CD-I there are two basic mechanisms for audio playback. The most straight-forward is the play of a Real Time File containing interleaved audio sectors directly from disc. The other method is by using Soundmaps. The audio is then stored in memory and fed to the audio processor under control of the CPU. In the early stages of a title design an analysis needs to be made as to what methodology is best suited for a particular result, because the choices have consequences for synchronization, timing, memory usage etc.

2. Direct audio play.

In this mode the audio data doesn't really enter the system. The only control the CPU has consists of Starting, Stopping, Pausing and Continuing of the play, and in the case of ADPCM, selecting which one of the Audio channels is actually fed to the audio processor. Once a play is kicked off, no intervention is required and the actual data transfer happens fully autonomously. As a matter of fact, the transfer doesn't even use the CPU bus, so in that respect there is no system load caused by the playback of audio in this way.

For synchronization purposes the CPU can interrogate the File Position Pointer or can use Triggers in the file that is playing (in the case of ADPCM playback; this option is not available when playing back CD-DA files). Both alternatives have a small caveat: In the case of the File Position Pointer, you have to be aware that for ADPCM audio playback this pointer is only updated when a selected sector (so one that passes the Mask) enters the system. In the case of Triggers, any Trigger (also from a non-selected Channel) will give a signal to the CPU, so that in a complex file layout it can become difficult to determine which one is causing the signal. Also, the Triggers are built into the disc image itself, so that modifications require you to go through a whole loop of rebuilding the disc image. Monitoring the File Position Pointer is done within the application itself, so in case of a modification only this part needs to be adapted. This leads to a much faster iteration cycle. For all of these reasons the File Position Pointer methodology is preferred; now we have to find a method that assures that the Pointer is kept current under all conditions.

The recommended solution is to set up the PCB masks in such a way, that every sector passes the Chann_Mask. In this case you are assured that the File Position Pointer gets updated for every sector. You then use the AChann_Mask to select

the audio channel you want to hear and the Channel Index List (CIL) mechanism to prevent the data from all unwanted sectors from entering the system. Now the File Position Pointer will always be current, so that you can use it for synchronization. In the case of CD-DA the system always keeps the File Position Pointer up to date, so this method works for ADPCM as well as for CD-DA. The only price you pay is a few percent of CPU load for
handling the File Position Pointer for every incoming sector, but when accurate synchronization is desired that is a small price to pay.

3. Soundmap play.

Soundmaps are fed from memory to the Audio Processor by the CPU. This means that there is maximum control over the Audio. The data could be constantly held in memory, like in the case of a "confirmation beep" when a user presses a button, or alternatively it could be brought into the system from the disc continuously and fed to the Audio Processor continuously. This latter might seem to be an unnecessary detour compared to direct Audio play, but it has the benefit of better control. Also in direct mode a mute circuitry is activated upon a channel switch to prevent switching pops and clicks. This makes switching of direct play very clean for the case that you switch between totally different channels. Switching between two different language tracks is a good example. By contrast, the Soundmap method is used in musical applications that require seamless switching between channels. The one caveat in the use of Soundmaps is that the Green Book is not very specific on timing details, and that player variation may occur for that reason. This can be explained by the fact that the SM_Out call actually causes a data transfer from memory to a local buffer that is associated with the Audio Processor. The effective size of this buffer varies between manufacturers. The status of the Soundmap Play as indicated by the Status Block pertains to the transfer from memory to buffer and not to the audible result itself. So the fact that you get a Soundmap_Done signal does not mean the user doesn't hear audio anymore. The amount of audio that is still "in the pipeline" depends on the effective buffer size of that particular player and of course on the quality level of the audio used. Recommendation/Charlie

4. Combining Real Time Audio with Soundmaps.

The Combination of the two mechanisms is also possible. This is useful when a confirmation beep has to be given while audio is playing from .he disc. The general rule is that the Soundmap will get priority in this case. The Real Time Play continues normally, only the data is not turned into audio for the duration of the Soundmap. In this simple example of a single Soundmap temporarily "overruling" the Real Time Play there is no complication. However when more complex arrangements are required the hybrid use of Real Time Play and multiple Soundmaps can lead to timing problems that are hard to understand; they are often the cause of bugs that are extremely hard to analyze and potentially can delay a project by many weeks. Therefore a strong recommendation is given that you avoid this problem area altogether and use Soundmaps exclusively instead. Let me give you an example of such a complex arrangement. Suppose a kid's title uses "Talking Hotspots", because it is intended for kids that cannot yet read. Suppose audio is playing from disc and a hotspot is entered. Nothing wrong so far: The Soundmap audio will get priority over the disc audio and the intended effect is achieved. However if the first hotspot is left quickly and a second one is entered, the first Soundmap play has to be terminated and the second one started, while the audio play is still going on in the background. In this case there are driver complications (a waiting time associated with the Audio chip buffer management) that can cause the CPU to sleep unexpectedly. Although it is possible to avoid this problem by careful timing of the second SM_Out, this is not a trivial exercise. It is better to avoid this situation altogether. In the example as described above the complication goes away by routing the Real Time Audio through memory. So instead of a direct play, a Soundmap based indirect play simplifies the situation drastically. The price for this is some memory overhead for the buffers that are required to hold the audio temporarily and some CPU overhead for doing the transfers. However, when you design this in from the beginning, this can usually be accomodated.

5. Disc building considerations.

One other aspect has to be mentioned in the choice between Real Time Audio and Soundmaps: disc filling efficiency. In the case of Real Time Audio the sectors containing the audio information are deposited in a regular pattern on the disc. The distance between neighboring audio sectors is determined by the audio quality level. The remaining sectors can be filled with Video, Data or other audio channels. An example will show that the most simplistic arrangement will lead to very inefficient disc usage. Suppose you have a linear story with a bilingual soundtrack in C-Mono, and you have a new video still coming in every 5 seconds. This leads to a sector pattern where 2 out of every 16 sectors are filled with the Audio information and every 5 seconds there is a burst of less then a second where the video information (roughly 100K) completely fills the gaps between the audio sectors. This arrangement leads to a disc that is mostly empty. It is of course possible to interleave more than one sequence and when this is carefully planned, disc packing efficiency can go up to 80 or even 90 %. However, the strict interleave requirements impose rather severe design restrictions (e.g: in every sequence the images cannot come in faster than a preset rate of say one image per 3 seconds). If your application has a requirement for long sequences and you can afford the memory overhead a totally different solution can be selected by the use of Soundmaps and operation of the drive in a start-stop mode. The use of Soundmaps eliminates the need for a strict interleave pattern. You only have to ensure that the queue feeding the audio processor never gets empty, and you have achieved continuous audio again. The biggest factor in this mode of operation is the size of the buffer that is required. Since the Green Book guarantees only a one second access time to data in the proximity of the current disc position, you need to buffer at least the amount of data that is required to mask that one second. In this mode it is possible to achieve close to100 % packing density on the disc.

6. Summary and conclusion.

CD-I has very flexible audio facilities. Direct playing from Real Time files has the lowest overhead associated with it. Reliable synchronization to the Real Time audio can be achieved by monitoring the File Position Pointer. Real Time Audio can be combined with simple Soundmap play without complications. For more complex audio functionality it is recommended to use Soundmaps exclusively. Soundmap based audio play also opens up the possibility for operation of the drive in start-stop mode for applications that require maximum disc packing density. A careful analysis in the early stage of a project will allow you to make the correct choice of the audio architecture for any given title.