PHILIPS

PHILIPS

PHILIPS INTERACTIVE MEDIA of America

Technical Note #79

Monitoring Audio Play From Memory

Charles Golvin

September 29, 1992

This technical note summarizes the methods available for monitoring audio play from memory. In particular, this note describes differences between the Green Book description and current implementation for some of these methods.

Note: Some of the information in this note pertains to the current implementation in the Sony IVO-V10 portable CD-I player, whose behavior with respect to audio played from memory will change in the next system revision. There remains some debate as to whether the current implementation is, strictly speaking, in accordance with the Green Book.

Monitoring Audio Play From Memory

Purpose

Playing audio from memory has proven to be one of the more problematic features in CD-I. Many applications require very tight synchronization with audio playback: either with video display or other responses to the user. There are a number of system calls provided in CD-RTOS that are intended to provide this synchronization capability. The purpose of this note is to examine methods based on these calls and to highlight differences between the observed performance of these calls and the likely interpretation of the Green Book.

Playing Audio

The ADPCM decoder in a CD-I player has its own internal buffer from which it decodes data. The buffer's size is not specified in the Green Book and may vary in size among players produced by different manufacturers. To achieve seamless decoding of audio, when the audio processor has completed decoding a sector's worth of data, it is imperative that another sector of data be present in the audio processor buffer to be decoded. Failure to maintain this flow results in audio disturbances. It should be emphasized that there is a non-trivial delay between the time the audio is transferred to the audio processor and the time the decoded audio is audible.

When playing audio in "direct from disc" mode, the layout of audio sectors on the disc ensures this seamless flow of data. The layout of audio sectors is dictated by the amount of time it takes the audio processor to decode one sector's worth of audio data.

When audio is playing from memory, the device driver is responsible for maintaining the flow of audio data from RAM to the audio processor buffer. This process is essentially transparent to the application process when playing a single sound map. However, in the case in which the application must make a seamless transition from one sound map to another, it becomes the application's responsibility to ensure that the audio processor receives the first sector of the new sound map before the audio processor has completed decoding the last sector of the first sound map.

Determining the Status of a Play

There are essentially three different methods provided in the Green Book for determining the status of audio played from memory: the asynchronous status block supplied to SM_Out, SM_Stat, and SM_Info. The first is essentially a Boolean operation (is the play active or completed?), and the second two provide updated information during the decoding of audio. Keep in mind that the fundamental activity surrounding the play of audio from memory is the transfer of audio data from RAM to the audio processor. Hearing decoded audio is merely the result of this.

When an **SM_Out** request is issued, the driver begins transferring audio data to the audio processor, one sector at a time. When the audio processor detects that part of its internal buffer is empty, it sends a hardware interrupt back to the driver, requesting more data. This dialog continues until the driver has no more data to transfer, at which time the driver sets the "play done" bit in the **asy_stat** field of the asynchronous play block. In addition, if the **asy_sig** field of this block is not zero, it sends the requested signal number to the application.

It is also possible to receive information from the driver while an **SM_Out** call is active. This can be achieved either by continuous calls to **SM_Stat**, or by obtaining a copy of the sound map descriptor structure associated with the current sound map and monitoring its contents while the play is active. As with the asynchronous play block, this information pertains only to <u>the transfer of audio data from RAM to the audio processor</u>. Although the first word value in the sound map status block is described as the "sector number currently being played", the current implementation on all known players updates this field with the next sector to be transferred to the audio processor's internal buffer. The same is true with respect to the **SMD_CurAddr** field in the sound map descriptor—this holds the address of the next sector to be transferred.

Implications for Applications

From the above explanation concerning the asynchronous play block, it is apparent that the timing of the system's recognition of the completion of a play (either via **asy_stat** or the desired signal) depends on the size of the audio processor buffer. For example, if the audio processor buffer is three sectors in size and a five sector sound map is played, the application will receive the play done signal after the decoding of three sectors of this sound map's audio data. Note that in the case of C mono, nearly half a second of audio remains to be heard when the application receives this signal.

This method provides a reliable timebase for playing continuous audio from memory. However, it is important to realize that the delay between the SM_Out call and the associated audio being audible may depend on the size of the audio processor buffer (*this may only be a Sony dependency*). For applications that can impose this delay initially and base all timing on that delay, there is no problem—one such example is the Guitar discs. However, in an application that wishes to play audio immediately in response to a user action (for example, a gun firing in an arcade-type game), such a delay may be unacceptable. In order to achieve this response, the application must ensure that no more than one sector of audio is queued in the audio processor buffer at any given time.

From these explanations, it is also clear that the information received via **SM_Stat** and **SM_Info** is dependent on the size of the audio processor internal buffer. These calls should only be used to provide information on the current state of play, and not as a basis for any sort of synchronization.