CwR

08 FEB 1994

PHILIPS

Philips Media Electronic Publishing

11050 Santa Monica Boulevard
Los Angeles, CA 90025 USA

FAX MESSAGE

+1.310.444.6515 (TEL)
+1.310.477.4953 (FAX)

To: Marc de Krock PIMC

From: Charles Golvin PMEP

Date: February 9, 1994

Subj.: **MonoII Player Status**

Dear Marc,

Following is a technical summary of the status of the MonoII players. It begins with some general information and then gives specific problem reports. If workarounds are known they are presented. This list should not be considered exhaustive, but rather the start of a living document that will continue to be updated.

History of Philips Players

Philips has produced three different main boards for CD-i players, by which all Philips consumer players can be classified. As all players use the 68070 CPU and all players carry the same version of the operating system, this discussion will focus on the other components that make up the system.

The first board, called "mini MMC", is the basis of the 605 authoring player. The consumer players based on this design are called "910" (NTSC) and "205" (PAL). This layout uses two separate boards, and two chips to control the CD and ADPCM decoding, the combination of which is often referred to as "CDIC/DSP." For video decoding three chips are used - one "VSD" and two "VSC"s, one per plane. The basecase planes of RAM are contiguous - the first byte of plane B follows the last byte of plane A.

The next layout, called "monoboard I", combines the two boards into one, and replaces the three separate video chips with a single chip, called "VDSC." All players based on this layout are called "220", with PAL and NTSC players distinguished by the number following the "/" in the model number. The VDSC is capable of addressing up to two MB per plane, and as a result the basecase memory planes are not contiguous in players based on this layout. The release of this player exposed many compatibility problems, nearly all of which were the fault of the applications. These issues are well documented in PIMA Technical Note #??, *Ensuring Title Compatibility Across Players.*

The most current layout, called "monoboard II", replaces the CDIC/DSP combination with a single DSP chip; of course the system software for CD and

audio decoding is different as well. There are three model numbers based on this layout: 200, 210, and 220. The 200 (NTSC) and 210 (PAL) do not have S-VHS output, front panel headphone jack and volume control, use the player shell from previous players, and carry 8kB of NVRAM. The 220 player, both PAL and NTSC, has S-VHS and headphone jack with volume control, carries 32 kB of NVRAM, and a new player shell. In addition, the total amount of contiguous memory available to the application at startup is somewhat less on these players than on previous players. However, these memory blocks are well within the Green Book limits.

Problems Encountered on Monoboard II Players

The CSD entry for the NVRAM device on monoboard II players should have a parameter ("SZ#32") indicating that the device has more than the default 8 kB. Some players do not reflect the correct size of the NVRAM device.

Workaround: deleting the CSD file and resetting the player will cause the CSD to be generated correctly. As most titles merely query the NVRAM device to see if sufficient space exists, and are not attentive to the size parameter, this should not cause any problems. However, future titles that use more memory if available will not provide the full functionality as designed.

A problem exists when ss_play, sm_out, and video interrupts are active in the system at the same time. The problem manifests itself by terminating the ss_play, and the error number in the PCB_Sig field can be either 244 or 249. The fundamental cause of the problem is not yet known. It may be the case that the 244 and 249 results are actually different problems - still under investigation.

Workaround: the proposed workaround is to simply prevent the three interrupts from occurring simultaneously. Since the soundmap play is the only event under the program control, the application can delay playing the soundmap until the next video interrupt. While this has the drawback of potentially delaying the audio response for one video field, most applications should be able to tolerate the delay (especially in contrast to the alternative). I believe one developer has experimented with delaying the soundmap play until the next sector from disc is received, but the title still has problems which may or may not be related to this problem.

sd_loop when called with an end loop value equal to smd_smsize/128 returns error $600 (illegal parameter). Previous players accepted this value, however the Green Book is not clear whether the counting for sd_loop is 0-based or 1-based.

Workaround: always count soundgroups 0-based, so that end loop values should be one less than a multiple of 18.

When a data overrun error occurs (i.e., CDFM attempts to deliver a sector to a PCL whose 'buffer full' bit is set) resulting in the play ending prematurely, the file position pointer returned by gs_pos is incorrect (2048 bytes too large). The Green Book is not clear in describing the proper behavior of gs_pos in this case, and in fact the Sony player shows the same behavior.

Workaround: because there are only two possible behaviors, an application can work around the problem by first forcing the overrun error then observing the file position pointer value. For example, start at a known location with consecutive sectors in the same channel (e.g., the start of a picture). Set up the play with a one sector PCL, circularly linked to itself. Start the play but don't clear the PCL buffer full bit after it is set. In this case the overrun error will occur when the system attempts to deliver the second sector. At this point gs_pos will either return the position of the second sector or the following position. In the second case, the application would then subtract 2048 from the current file position every time an overrun error, and in the first case no adjustment is necessary.

While the timing for the delivery of the soundmap done signal is in general consistent with previous players, the timing for single sector soundmaps is not. The known behavior in this case is that the done signal is generated when the audio begins to be decoded; on the monoII players this signal is generated near the completion of decoding. This can cause serious problems for titles whose audio timing is based on this done signal.

Workaround: none known. For titles that are mixing audio from memory with audio from disc, the data arrival from disc should be used as the basic timing mechanism rather than the done signals from the audio processor.

On some early mono II players, if sm_out is issued while ss_seek is active, the driver will abort the seek before starting the audio playback. This is clearly a mistake and in contradiction to the Green Book.

Workaround: use lseek rather than ss_seek whenever possible (cf. PIMA Technical Note #??...). If ss_seek must be used then do not execute the soundmap play if a seek is active.

Some early players have their left and right channels reversed. This was also true on early monoboard I players.

Workaround: none.

Some early players do not correctly update the file position pointer when the only sectors selected from disc are being routed to the audio processor.

Workaround: place sectors in the stream that can be selected for delivery to

memory in this case.

The application is supposed to have all the resources of the system available to it, and the player shell is supposed to use **FSChain** in starting the application program. On monoboard II players the **launcher** program that is responsible for starting the application instead uses **FSFork** and remains an active (but sleeping) process. This is for the purpose of solving the problem whereby the volume buttons on the remote control device do not control the volume of MPEG audio.

Workaround: none.

Clarification

The player's error correction behavior has two levels. The first level is the CIRC, the same error correction used for CDDA, and the second level is the error correction/detection code present in form 1 sectors. This secondary error correction mechanism will only be activated in the case that the primary error detection mechanism also indicates that an error has occurred in the data delivered. For example, if one were to create a disc image with data sectors then corrupt the data in one of those sectors prior to pressing the disc, when reading the corrupted sector no data error will be reported. The reason for this is that the system will have faithfully delivered the data on the disc, so despite the mismatch in the secondary error detection code, the primary error detection indicates no error.

Kind regards,

Charles Golvin