PHILIPS

PHILIPS INTERACTIVE MEDIA of America

Technical Note #69

The Technical Status of the Philips 910 CD-I Player

Alty van Luijt

September 20, 1991

In the current state of CD-I players, many of the older problems have been resolved, and a very workable situation has been achieved. However, a number of issues remain in which the actual available implementations deviate from Green Book definitions, where Green Book definitions are somewhat ambiguous, or where timing aspects play a role. This notes describes some of the issues that need to be taken into account and suggests workarounds whenever possible. The notion of a workaround implies that the suggested solution is always upward compatible. That is, discs that utilize these workarounds will work well on future players, even after the player bugs have been fixed or timing constraints have been relaxed.

TABLE OF CONTENTS

1. Introduction

CD-I players have gone through an evolution of various versions of hardware and software leading to the players that are part of the U.S. consumer launch. There is a fundamental philosophical difference between earlier updates and the current one: this is the model for the launch. Although future players will be subject to further improvements, this model represents the minimum common denominator for the installed base. Therefore, the workarounds that apply to this model will have to be kept in mind for all current and future discs.

Keep in mind that PIMA still enforces backward compatibility to the 180 1.1E players as well, not only because there is an installed base of those players, but also because some of the key components that were used in those players are likely to show up again in players that will be produced by the various Japanese manufacturers. Please consult PIMA Technical Note #62,* which describes problems and suggested workarounds for those players. The hardware-related aspects of these problems (CLUT restrictions, etc.) are still especially relevant.

* van Luijt, *Living with Reality: Remaining Bugs in CD-RTOS 1.1*

2. Cosmetic problems.

Some of the remaining problems are merely cosmetic. The esthetics of the imagery could be affected or the quality of the audio might not be optimal. Some of these problems are due to the hardware and have no known workarounds. Others can be avoided by following a proper procedure. Since the effects of flaws in this area are not considered show stoppers, observation of the workarounds is encouraged, but not at all cost. PIMA does not consider these workarounds mandatory. They are subject to the normal cost/benefit trade-offs.

2.1. Video. The player image is off-center by roughly 12 pixels.

This is a timing issue in the chip that is used to generate the video which causes the image to be off-center to the right by about 12 pixels; there is a slight variation depending on whether the connection to the monitor is made by RGB or by CVBS, but it is roughly 12 pixels. Future versions might have correct timing and/or manufacturers might want to compensate for this by delaying the sync signal so that the active video portion of the lines is properly centered.

Workaround: Reduce the safety area by 12 pixels on the right hand side. Instead of using the center 320 pixels, restrict the area for critical information down to 308 pixels. offset to the left.

Warning: Do not try to compensate for the shift in software by shifting your whole image to the left. Once manufacturers fix the hardware problem, your disc would exhibit imagery shifted to the left.

2.2 Display control program for two planes not in sync.

The driver, when updating the pointers to where the DCPs start in each of the video planes, is not scan synchronized so that it can always avoid the two DCPs from becoming out of sync. If you execute a new DC_Exec or DC_Flink, the driver update for both planes is not atomic. If the display logic fetches these start values between the two "write" operations by the driver, the result is that the new plane A is shown in combination with the old plane B. This gives a uncontrolled situation during one display field, after which it restores itself. The most annoying effect is when the properties of plane B are controlled from the DCP associated with Plane A. As an example, if the image coding method for plane B is controlled by DCP(A) and the image changes from DYUV to CLUT, a nonsense picture is displayed momentarily.

Workaround: See PIMA Technical Note # 63, which treats this problem in detail.*

* van Luijt, *Display Synchronization in CD-I*

2.3 Audio. Mute circuitry active at beginning and end of sounds.

This affects both sound maps and audio played directly from disc. The player contains "soft mute" circuitry that has a time constant associated with it. This suppresses audio at the very beginning and end of a sound.

Workaround: Make sure that the first and last 50 msec of a sound map or file do not contain vital information. If necessary, pad the audio with silence or other appropriate sound.

2.4 Audio. Seamless channel switching by SS_CChan impossible.

The functionality that is offered by modifying the AChannel mask in the PCB or SS_CChan is fine for switching channels that are unrelated (e.g., language choices). However, it is not possible to achieve seamless channel switching this way because of the activation of mute circuitry during the switch.

Workaround: For applications requiring seamless channel switching, audio has to be routed through memory, filling sound maps from the disc, and the switch can be performed by feeding the proper sound map to the audio processor by SM_Out.

2.5 Audio. Pops and clicks

Sometimes audio pops and clicks can be heard at the beginning and end of a sound map or real-time audio play.

Advice: When editing audio files, make sure they start and end on a zero crossing. Also, fading in and fading out may be of help. We expect that players will gradually improve in this respect.

2.6 Reset status and eject.

The 910 player closes the CD tray upon reset. Reset also occurs when the application exits back to the player shell. That means that if your application actually wants to eject the disc upon exit, dependent on the timing between SS_Eject and the exit to the player shell, either nothing happens (the reset overrules the eject before it actually starts to happen) or the disc starts to be ejected, only to be drawn in again when the reset hits.

Workaround: We have flagged this to the product management that is responsible for the player and are hoping for a modification after a certain number of players have been produced. The modification we are requesting is to leave the tray in the position in which it was left by the application. However, even if we are successful in obtaining that modification, there still will be players out in the market that close the tray upon reset.

If the functionality of ejecting the disc is vital to your application, your best bet is to put up one final dialog screen that says: "Please click here if you have taken out the <application_name> disc" or something along those lines and only then exit to the player shell.

2.7 CDi-Ready discs.

The 910 player will treat CDi-Ready or other hybrid discs as CD-I discs. There is no functionality in the player shell to treat them as regular audio discs. Thus, features like FTS are not available for CDi-Ready discs.

3. Functional problems and restrictions.

This category contains problems that could affect the functionality of an application, rather than cosmetic aspects. Some of these problems are hardware related and some are system software bugs or timing peculiarities. Application of a workaround is mandatory for PIMA titles, because the resulting defects are considered to be unacceptable for consumer quality titles.

3.1 Timing of SM_Out after ADPCM from disc.

This is not really a bug in the traditional sense of the word. It is more a functional restriction due to an implementation timing issue. Specifically, when switching from direct ADPCM to sound map play, time is required to modify the buffers that are associated with the ADPCM circuitry. This is handled by the driver; the driver starts the SM process, sets an internal flag that an action is being processed, sets an alarm so that it will be notified after a time, and returns to the user. For an initial SM_Out, there is no problem with doing it this way: The SM_Out call does its thing fully asynchronously and, after a time, when the last sector of the sound map is transferred to the audio processor, an SM_Done is generated.

A complication appears when the driver is activated with a second SM_Out within the time slot before the first action is completed. In this case, the driver finds the flag set to specify that a previous action is still in progress. It waits until that previous action is completed (the alarm goes off, the flag is reset). In this period, the driver sleeps; thus, in a single-tasking architecture it occupies the CPU. Only then does the second SM_Out really start; in its turn it sets the alarm again and returns to the user. The sleep in the driver could amount to a loss of the CPU for up to 60 milliseconds, provided no new commands are issued to the CD/AP driver. In a critical timing situation, this could lead to missing incoming sectors (in case your PCL buffers are not large enough to hold the sectors that you cannot process) or a hiccup in video transitions, blitting, or animation.

Workaround: Since this situation occurs only in the combination ADPCM from disc and multiple sound map play, two classes of applications should be distinguished. The first are the simple ones that play audio from disc and perhaps use sound maps for prompts or confirming beeps and other sound effects. These will never see a problem in this area, and it is perfectly safe for them to play the audio from disc. Most current applications fall into this category. The other class is where there is a more complicated audio functionality. Let me use the Book Engine discs as an example. In the game sections, these discs have questions that play directly from disc, audio prompts that a user can activate by moving into a talking hotspot, sound maps for the incorrect answers, a play for the correct answer, and a background music bank that starts playing to provide background music or ambient sound when the user is inactive. In such a complex arrangement, it is possible that while audio is playing directly from disc (question or music bank), the user may hit multiple talking hotspots in rapid succession. This causes *the condition described above. In these more complicated cases, there are two ways to handle the situation. The first option is careful design and control of the timing behavior of the application (which is the solution that the Book Engine discs have actually implemented). The alternative for new designs is to avoid this situation altogether by never playing ADPCM directly from the disc, but to reroute the audio that comes off the disc into memory and to use sound maps for audio output. This gives the application full control over audio output. So, if you can afford the memory and overhead required, this is the recommended strategy for more complicated audio titles.*

3.2. Interrupting a Play

A Play really has two different phases. One phase is the seek period between the start of the play (the application requesting an SS_Play and the asynchronous call successfully returning) while the head is being repositioned, until the moment at which the first selected sector has actually come in. The second phase is the actual play itself, where a regular stream of sectors is delivered to the system. The definitions for Abort, Pause and Continue are such, that these functions make sense only for the second part of the play while sectors are coming in. The complication is that SS_Play is an asynchronous system call that returns fairly quickly to the application—much faster than the mechanical repositioning of the optical head. Thus, Play returns while the actual drive is still seeking. When a Pause (or Abort) request hits in this interval, the Pause (or Abort) takes longer to return, because it will complete the seek phase, and if that seek for some reason takes longer than expected, a time out causes a DevBusy error to be returned.

Workaround: *Make sure that Pause or Abort requests are not in a time critical part of your code if there is a chance that they might be issued in this period of implied Seek.*

3.3. DCP instruction execution timing for line being linked to.

The model of the DCP, as described in the Green Book, is not totally identical to the actual implementation in the underlying hardware. The basic difference is when the LCT for the first line that the FCT is linked to is executed. In reality, the DCP instructions contained in this line are executed immediately after the last instruction of the FCT. Thus, with different length FCTs in plane A and plane B, the LCTs for the first line can be executed at different times, and even an instruction at the tail end of a longer FCT(A) could overrule a different instruction in the first LCT line of plane B.

Workaround: *Balance the FCTs by adding padding to the shortest FCT. Alternatively, pay attention to the content of the DCPs so that the conflict never arises.*

3.4. Currently selected sector timings.

The Green Book refers quite often to "next selected sector coming off the disc." This is a criterion for many disc-related function updates, such as changing the file position pointer, modifying the channel mask, etc. In the 180 players, these updates took place for each sector of file number zero, regardless of whether it was selected or not. Since file zero was the default that the disc builders created at the time, this is the situation developers were accustomed to on the 180 players. The difference for the 910 player is that it follows the Green Book more literally; these updates are now restricted to incoming sectors that actually pass the channel mask. This reduces the system load; however, it may cause different timing behavior, so applications have to be aware of this.

Suggestion: *If you still want to recreate the same timing characteristics that you have been used to on the 180, one of the ways is to set the channel mask so that all sectors are selected, and use the CIL mechanism to do the selection between wanted and unwanted channels. The price you pay is a slightly higher system load (back to the 180 level) because of the additional interrupts that now fire for every sector. However, for some applications the tighter control of the timing that you get by making every sector a selected sector might be worth it. So, whether you want to apply this technique or not depends on your needs.*

4. Incomplete definitions.

Despite the best efforts to make the 1.1 release of the Green Book fully current, some issues have slipped through because it has not been possible to reach agreement on the exact phrasing in a timely fashion. The items mentioned in this part of the note constitute the author's best interpretation of the most likely future developments in this area. As such, compatibility cannot be guaranteed on such a loose and informal basis. At best, these issues can grow into a *de-facto* standard that ultimately might be absorbed into a future release of the Green Book.

4.1 Sound map done signal timing vaguely defined.

In the current version of the Green Book there is no description of exactly when the "Sound map done" signal is generated. This creates a window of uncertainty for the application; the duration of this window is equal to the time it takes to play back the current sector of the sound map. This makes the generation of contiguous audio from sound maps questionable.

Interpretation: *The signal is sent at the moment the transfer of the last sector of the sound map to the actual audio processor is completed. Due to the local buffering in the audio processor, this gives a time window equivalent to the play back time of one sector of audio for the application to react with a new SM_Out to achieve contiguous audio operation.*

4.2 SM_Stat information varies in different player architectures.

Interpretation: *The status information that is obtained from this function call pertains to the next sector to be transferred from memory to the local audio processor buffer. However, player architectures vary in that some effectively have a single sector audio processor buffer, and others (including the Philips reference player) have a two sector buffer. So, for true portability, the SM_Stat information cannot be used for synchronization purposes between audio and video presentation. These have to be timer driven. The only practical use of the SM_Stat information is for buffer regulation purposes in an SM_Out environment.*

4.3 Front panel controls unspecified.

One serious omission in the current Green Book is that the controls that are accessible to the user, both on the front panel and on a remote control, are not available to the application in any standardized way. This is especially important for discs with linear controls (like music titles). In the annex, we are publishing the way in which the Philips 910 and 200 series of players implement this feature in the hope that wide distribution of this knowledge might cause *de-facto* standardization in this area.

Recommendation: *Use this functionality as a feature only in parallel to screen-based functionality. In other words, make sure that your application runs even if these functions are not available on a base case player.*

Note: In the 910 player, only the keys that are available on its own remote and local keyboard are recognized. Thus, if you use a 180 remote and press any of the additional keys (e.g., the numerics), nothing happens.

4.4 Error codes poorly defined.

Although some specific errors are listed for most function calls, there is an underlying assumption that programmers in CD-I should respect "good programming practices for OS-9."

Recommendation: *Always look for the return status of your calls. Other than very specific errors (like PCB errors, which are well defined), don't rely on the individual error codes that are being returned. Instead, define a generic error handling strategy that is appropriate for the situation. So, in other words, exploit the fact that an error is returned, and do not rely on an exact error number. A typical strategy for all disc-related functions (Pause, Continue, etc.) could be to retry a number of times until a time out of perhaps 3 seconds, and then go on and pretend that the error has never occurred. (Some errors indicate that you are trying to perform a function that is already current: for example, if you Pause on a Play that was already in Pause.)*