PHILIPS

PHILIPS INTERACTIVE MEDIA of America

# Technical Note #85

# Ensuring Title Compatibility Across Players

Charles Golvin

April 2, 1993

This technical note describes potential issues to which a title developer must attend in order to assure compatibility with all CD-I players. In particular some issues exposed by the introduction of the Philips 220 player are discussed.

# Ensuring Title Compatibility Across Players

## Overview

The CD-I standard is intended to ensure compatibility between all CD-I players and all CD-I titles. This standard defines the minimum set of attributes that a CD-I player must possess (described as "base case"); however, most development systems exceed this set in order to provide a complete development environment. Hence, there are many issues to which a title developer must attend to make sure that a title will be compatible with all players—it's not enough to simply test that the title behaves correctly in the development environment. This purpose of this note is to communicate the compatibility issues of which the author knows to title developers. Issues that have come to light by the introduction of the Philips 220 player are specifically highlighted.

Some of the issues discussed in this note are not compatibility issues per se, but rather reflect good engineering practices. Failure to follow these directions may result in difficult to locate bugs that are exposed only when the title is played on a type of player other than that on which the title was developed.

## Compatibility Issues

• *Memory configuration*

The Green Book defines the base case memory requirement as two 512 kilobyte banks of "colored" memory. More exactly, prior to the start of the title application program, each bank must contain a contiguous block of free memory of at least 480 kilobytes. While most players exceed this requirement, every title must play under the minimum condition.

In order to test for compliance with the base case memory condition, use the **cdi_basecase** floppy application available from PIMA, or, using the player shell memory setting in the **Tools** menu of the Philips 605 player, set the additional memory to **0.0**. It is important to perform this test in both NTSC and PAL, because a title's memory allocations may be different in these two cases.

• *Initialization of player settings*

This topic is covered completely in PIMA Technical Note #57.1, *Initializing a Player's Configurable Parameters*. In particular, the Philips 220 player initializes the

player's attenuation to maximum—failure to initialize the attenuation setting of the player means the user will not hear any audio output.

To test that all of a player's settings are initialized, use the floppy application **cdi_deinit** available from PIMA. Also, the initial settings of the player may be set in the **vid_regs.prf** file stored in the 605 player NVRAM. This file's settings are selected by setting the Video Settings to "Previous" in the player's **Tools** menu. Consult the 605 technical manual for more information on the use of the options available from the **Tools** menu.

• *System tick timing*

The Green Book dictates that the kernel must deliver a system tick every 10 milliseconds. However, the tolerance of the system tick is 0.2 percent; that is, the system tick frequency may be anywhere from 99.8 Hz to to 100.2 Hz. A given player's system tick remains fixed at some value in this range; it does not fluctuate within the range.

Applications may use the system clock in a variety of ways, the most common of which is the use of alarms and the counting of system ticks. Any such use of the system clock by an application must be tolerant of the full range of values that is specified in the Green Book. For example, an animation program that loads data off the disc to keep the animation going, but uses a cyclic alarm to determine the display of the next animation frame, must be sure that the new data is not eventually delivered into the active display buffer. In such instances, it is better to slave the application to a single time base, the most preferable of which is the disc itself. Consult IMS Application Note #TSA-004, "Various Time Bases in CD-I," by Alty van Luijt, dated July 23, 1992, for more information in this regard.

In order to test that a title is tolerant of the full range of system tick speeds, use the **cdi_chgtix** application available from PIMA (for developers using the Philips 18x player). This application changes the setting of the 18x player's system clock. The system clock speed of the Philips 605 player may be set in the **vid_regs.prf** file stored in the 605 player NVRAM—use the program **rdvid** in the ROM of the 605 player to modify this value, and then set the Video Settings to **Previous** in the player's **Tools** menu. This operation is optimized by using the program **cdi_chgtix.605** available from PIMA.

• *Serial devices*

It is a common debugging and diagnostic technique to use standard output for informational messages. Unfortunately, the file manager responsible for handling this type of output (called "SCF") is not a base case device. Applications that output information to standard output or standard error are not compliant with the base case; however, because most development environments provide SCF, this lack of compliance is easily overlooked. Some applications have attempted to avoid this

problem by routing messages to the device called **/nil**. Alas, this is also an SCF device and, thus, is not guaranteed to be present in every player.

The best defense against reliance on SCF is to prohibit the inclusion of those functions that require it in your application. Carefully controlling the libraries to which your application links and the use of an ANSI compiler to prevent references to unintended functions are the best insurance against this source of incompatibility. Placing all such references inside preprocessor symbols provides an easy method of compiling them out when preparing to press a disc.

Another method for dealing with this problem is to determine in software whether paths are open to standard output and/or standard error. PIMA Technical Note #72, *Passing Open Paths Between Processes*, includes sample code that describes how this determination can be made. In this case, the application performs this assessment initially, sets a global flag for whether the standard paths are present, and output is or is not written depending on the state of this flag.

This raises one other suggestion for disc building: provide a debugging version of your application software on all the discs that you press, including (if possible) a symbol table file. It is always better to have, but not need, a debugging version than the other way around.

• *Seek tolerance*

The seek performance of most players surpasses the seek tolerances in the Green Book: at most one second within 20 Megabytes, increasing linearly to, at most, three seconds across the disc. However, all applications must behave correctly on a player possessing this worst case behavior. This is, in general, an issue only for titles that rely on new data from the disc to continue a fluid audio and/or video presentation.

The best technique for testing for compliance with this requirement is to use the **Seek Delay** setting in the **Tools** menu of the 605 player. Also, some emulation systems provide a delay setting—such a setting may require some experimentation to determine the correct value to achieve the worst case performance.

• *Display device types*

The Green Book supports three types of display devices (resolutions): NTSC television (384x240), NTSC monitor (360x240), and PAL (384x280). All applications must play correctly on all display devices. Consult PIMA Technical Note #48, *Preparing Titles for the International Marketplace*, for a description of some of the options available for tackling this problem.

Since all of the development systems known to the author provide native display in both NTSC television and PAL, the only test software developed is for the more unusual case of NTSC monitor display. In order to test that an application's images

display correctly on this type of device, use the floppy application **cdi_monitor** (Philips 18x) or **cdi_monitor.605** (Philips 605) available from PIMA. This test can be somewhat confusing: the test program modifies the CSD to indicate that the display is an NTSC monitor, but the hardware continues to display in NTSC television (the test is not germane for PAL). Applications that correctly query the CSD and modify the display to account for the current setting will display images *incorrectly*, the exceptions being (1) images using a new display start address on every line and (2) run-length images whose final 24 pixels are the same color value.

• *Multiple input device awareness*

The Green Book allows for four general types of pointing devices (touch screen, absolute, relative, and maneuvering devices), and the CSD reflects the device connected at player initialization. Applications must behave correctly with all types of input devices. For applications that simply read pointer coordinates, the type of device may be irrelevant. However, applications that are optimized for a particular type of input device must check the CSD to confirm the type of device attached to the player.

In order to test the various types of input devices, the CSD entry for the pointing device (the **#define** for **DT_PTR** in the file **csd.h**) may be modified to reflect the type of device being used. Consult the Green Book, Appendix VII, pp. 17-27, for the mapping between values and device types. Also, on most Philips players, plugging the input device into port 2 and resetting the player causes the CSD to be updated based on the type of input device.

• *Video display settings*

The Green Book describes the allowed combinations of display control program opcodes, but does not define the behavior of a player when an illegal combination of opcodes is present. Problems that are rooted in this area can be very difficult to find.

One such problem has been exposed by the 220 player: the Green Book states that in DYUV mode, the resolution setting in the display parameters instruction must be eight bits per pixel. Using DYUV in four bits per pixel mode on a player results in an illegal DCP; the plane containing these settings will not appear correctly. However, the fact that the DCP is illegal means that ANY behavior by the DCP may be expected, including blanking of the display. In fact, on the 220 player, this combination in a plane entirely masked will result in both planes displaying a solid gray image.

This particular instance of the general problem is exacerbated by the fact that changing resolution in plane B requires two separate writes to the DCP (see PIMA Technical Note #62, *Living With Reality* , for more information in this regard). In order to avoid this general problem when changing resolution, first hide the plane being changed. Next, write the image coding method and display parameter

instructions in a scan synchronized fashion so that the two writes will not be interrupted by the vertical refresh. For the 220 player in particular, ensure that if the scan interrupts your process between the two DCP writes, then the illegal case that obtains will be CLUT4 coding in eight bits per pixel, not DYUV in four bits per pixel.

• *Well defined DCP*

Most applications allocate LCTs that are as many lines high as a full screen image: 240 lines in NTSC and 280 lines in PAL. However, this is, in general, not required, and there are many techniques for displaying full screen images using less than full height LCTs. All such techniques share one common feature: the last line of the LCT must be linked to an LCT so that for each display line there exists an LCT line controlling it. Failure to provide this LCT link instruction will result in the display hardware accessing memory locations that contain undefined values; this could result in video anomalies, depending on the contents of this memory.

The LCT link instruction is one that may not be written directly by an application; it may be written only via the **dc_llnk()** function. This instruction is also unique in that, if present, it is required to be in the seventh column of the LCT. This instruction may therefore be inadvertently overwritten by other LCT writes. The point of this discussion is very simple: make sure that the DCP for your application is well-defined at all times. Do not unwittingly delete LCT links that maintain the integrity of your LCT, and, when rewriting these links, be sure that the new LCT maintains the DCP's integrity.

• *"Don't care" memory requests*

Memory management in CD-I remains an area that must be attended to with great care. It is possible for the system to allocate memory at unexpected times, so it is critical that an application control its own memory allocations as carefully as possible. For this reason, applications should always use **srqcmem()** [not **malloc()**] to allocate memory, and these calls should never use the "type" parameter **0**. Such a call is interpreted by the system as a "don't care" request, and the system allocates the memory from the plane that is less used at that moment. Because the extent to which memory is balanced between the planes depends on the initial setting of memory when the player boots, such requests can lead to problems when titles are played on various manufacturer's players.

In particular, the **cdi_basecase** program provides a memory environment that is nearly perfectly balanced between the two planes of memory; however, all of the Philips consumer players with which the author is familiar provide more initial memory in plane A than in plane B.

• *Uninitialized variables, stack problems, etc.*

These are not compatibility issues per se, but rather a reminder to practice good

programming techniques. An uninitialized variable may contain a value that is quite innocuous on your development system, but may contain a value deadly to your program on another player. Similarly, accessing the memory beyond your allocated stack area may only become a problem on a player with a different initial memory allocation. For these and many other reasons, every developer should employ the best development tools available: even if your cross compiler is not ANSI compatible, most native compilers today are, so take advantage of this. Even though you may not be able to link your code, the compiler may catch some problems that the cross compiler does not. Sun developers should use **lint** for this purpose.

When developing code, it is best to make use of stack checking code and libraries that provide parameter checking. These options are easy to discard when the final disc is being prepared, but it is worth avoiding the problems their absence may prevent you from finding.

- *Play termination*

The most reliable technique for terminating a real time play is to use the system call **ss_abort**. However, the Green Book also allows that an application may terminate a real time play by setting the **PCB_Rec** field in the PCB to zero; this change will be recognized on receipt of the next selected sector. Some title developers have noticed that, on their particular development player, this method takes less time than does **ss_abort**. Alas, several players with which the author is familiar only recognize this change in the PCB on receipt of the next sector selected for delivery to RAM. This manifests itself only when this technique of play termination is used when playing audio from disc directly to the audio processor, resulting in a play that doesn't end when the application intends.

While not technically a compatibility issue, a simple technique exists to work around this problem, as follows. Prior to changing the **PCB_Rec** field, clear the bit in the **PCB_AChan** field selecting audio for delivery to the audio processor, instead directing audio sectors from that channel to RAM. If you do not have the memory to collect these sectors, make sure that the CIL for audio in the selected channel is 0. In this way, the driver will immediately recognize the change in the **PCB_Rec** field on receipt of the next selected sector.

## Summary

This note has summarized all the known sources of compatibility problems with which the author is familiar. While not a complete list, this note provides a guideline for avoiding problems in the future.

**Addendum to Technical Note #85:** ***Ensuring Title Compatibility Across Players***
(This information will be integrated into the next revision of this note.)

- *Device name independence.*

The CSD in each player gives the names of all the devices present in the system. Each player manufacturer is at liberty to name these devices as they choose, the exception being the NVRAM device which must be named "/nvr".

To date, every player manufacturer has adopted the device names that are present in the Philips players: "/vid" for the video device, "/ap" for the audio device, etc. For this reason, it is possible to hard code device names into an application and, in general, to achieve satisfactory results. However, this is not according to the Green Book, and as soon as such an application finds itself on a player that uses different device names, the calls to open devices or files within a device will fail.

The most common instance of breaking this edict regards the use of the string "/cd" in applications. This can be a remnant of using files from other devices during development , or simple ignorance. In general the name of the CD device should not be needed, because the player shell ensures that all title applications begin with the current data and execution directories set to the CD device. All references to files and directories may be made relative to this initial setting. If your application requires the name of the CD device, it must be obtained from the CSD.

Unfortunately, there is no test program to catch this error, so it is up to you to search your code for any such references.