PHILIPS

Philips Interactive Media

Technical Note #90

Improved Seek Times with I$Seek

Cor Luijks,
Philips Interactive Media Systems, Eindhoven
Charles Golvin
Philips Interactive Media, Los Angeles

January 12, 1994

Abstract

This technical note describes the use of the function I$Seek to improve seek times.

Publication History:
No revisions.

To receive Philips Interactive Media technical notes and other publications, or for more information, please contact the person below who is designated for your area.

From Europe and the Middle East:

Hein Zegers
Developer Support
Philips Interactive Media Centre
Maastrichterstraat 63
B-3500 Hasselt
Belgium
Fax: +32 11 242168
Internet ID: hein@pimc.be

From the USA and Asia:

Lucy Lediaev
Developer Support
Philips Interactive Media
11050 Santa Monica Boulevard
Los Angeles, CA 90025
USA
Fax: +1 310 477 4953
Internet ID: lucy@aimla.com
CompuServe ID: 72056,1130

Improved Seek Times with I$Seek

Introduction

In contrast to hard disks, where the data is stored in multiple tracks, the data on a compact disc is stored in one "track." This "track" is laid down on the disc in the form of a spiral. The beginning and end of the spiral are indicated by the lead in and lead out. A side effect of having a spiral is that once the head is positioned it automatically proceeds along the "groove" until it reaches the lead-out area.

CD-RTOS provides two different mechanisms for seeking, one synchronous (**I$Seek**) and one asynschronous (**SS_Seek**). Although the asynchronous seek initially seems a better choice due to the structure of the optical disc, the synchronous seek in general provides better performance.

Optimization of Seek Operations

All functions that retrieve data from the disc (**SS_CDDA, SS_Raw, SS_Play, I$Read**) start reading from the current file position pointer. This pointer is merely a system reference to the location from which the delivery is to take place; changing the position of this pointer via **I$Seek** (or **lseek()** for C programmers) has no effect on the physical location of the read head.

If the current file position pointer does not agree with the physical location of the head, each of these read operations forces the repositioning of the head prior to fetching data from the disc. That is, each read function has an implicit seek built into its operation. This implicit seek represents a physical operation, and it can be time consuming. The Green Book mandates that the seek time may be no more than one second within a distance of 20 Megabytes, increasing linearly with distance to a maximum of three seconds across the entire disc.

Because no new data can be delivered while the head is being repositioned, seek times are a large contributor to "dead" times in titles. To optimize performance, a title developer must attempt to minimize these seek times. Seek time is minimized when the physical distance between the head's location and the desired read location is minimized. Therefore, title developers should optimize the layout of data on the disc so that the data needed to satisfy a user's requests is close to the location from which data was last read.

One simple example of this optimization concerns the startup of the application program on a CD-i disc. The player shell uses **I$Read** (via **F$Load**) to load the application module into memory, so the application program should be as close to the start of the disc as possible to minimize the duration of the implicit seek to the start of the application program on disc. Similarly, the first data required by the application program should be in the next location on the disc, etc.

Of course, interactivity means providing choices to the user so it is necessary to seek on the disc. CD-RTOS provides two mechanisms for seeking :

I$Seek	Change the file position pointer
SS_Seek	Change the file position pointer and reposition the head

Because of the implicit seek discussed above, it is evident that the sequence of **SS_Seek** and then **SS_Play** causes the head to move twice; whereas the sequence **I$Seek, SS_Play** will only cause the head to move once. Thus, in general, when an application is ready to deliver data from disc the latter sequence will be more efficient in starting the delivery of data. The main use for **SS_Seek** is in the case that the application knows well in advance that the next location from which it will read data is far from the current file position, and will continue a presentation to the user before having to read the disc. In this case the head may be moved to the correct location asynchronously, invisibly to the user, and when the time arrives to read the disc the implicit seek will be minimal. In all other cases where the application must seek immediately prior to delivering data, it is more efficient to use I$Seek and rely on the implicit seek in the read command.

Conclusion

All of the functions that retrieve data from the disc perform an implicit seek to position the head in the correct location for reading. When reading data from the disc it is more efficient to reposition the file position pointer (via **I$Seek**) and rely on this implicit seek than to explicitly position the head (using the **SS_Seek** call). The **SS_Seek** call should only be used to asynchronously reposition the head well in advance of reading data.