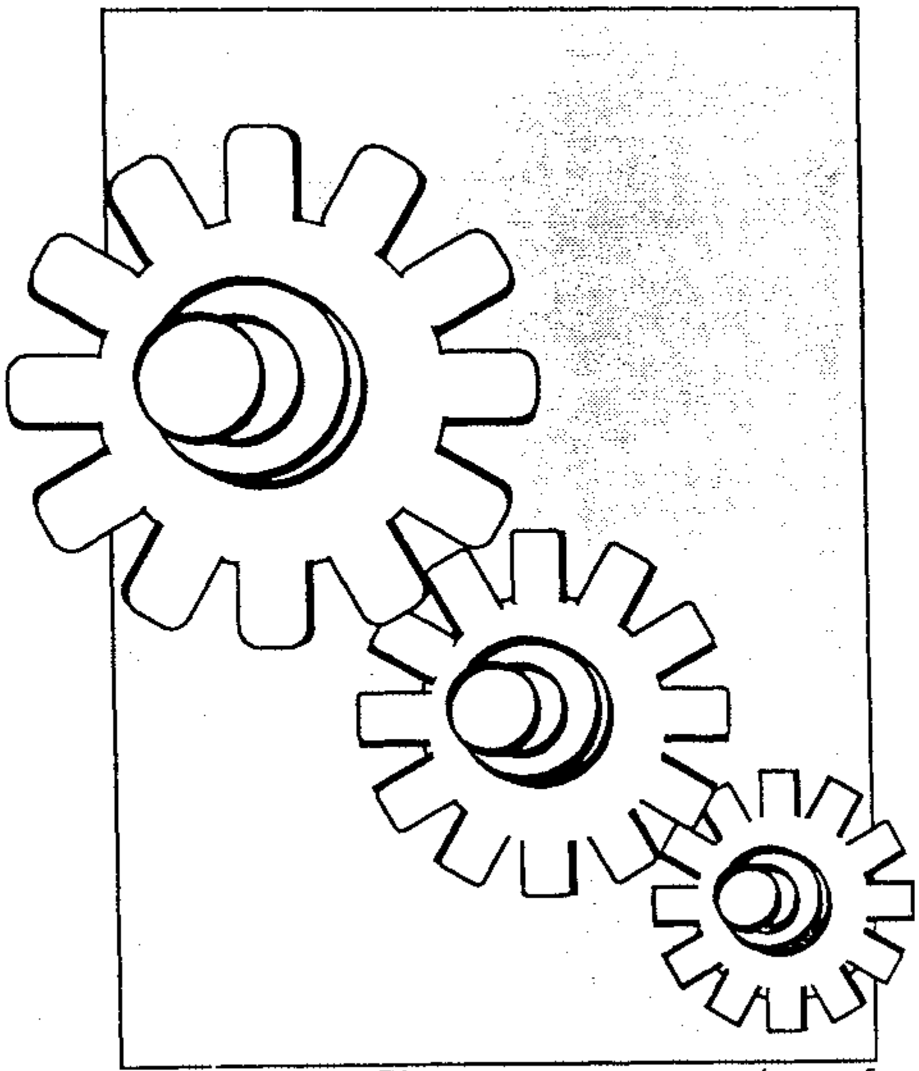
TN 94

TSA APPLICATION NOTES NR. TSA-003 05-03-1992

Various time bases in CD-I

The CD-I player has three different timebases: the disc rate, the video field rate and the timer or system tick. This note describes the properties of these timebases and the pitfalls that an application has to avoid to achieve proper operation across a variety of players.

Written by *Alty van Luijt*



copyright 1992 IMS TSA nr. of pages: 5

1. Introduction.

Timing is a very important aspect of CD-I title architecture, especially during sequences. Several timebases are running simultaneously in the CD-I player and these are asynchronous and not locked to each other. The data comes in from the disc at a rate of 75 sectors per second, the video fields run on their own time base of 50 or 60 fields per second and a system tick is available at a nominal timing of 10 mseconds. Unfortunately there is a tolerance associated with the video and the system tick timing, so that a reliable timebase for an application is not a trivial thing to accomplish.

2. The Disc rate.

The only absolute timebases that are tightly coupled in the CD-I system are the data rate of the disc and the processing rate by the ADPCM decoder/ Audio Processor. There is no slack between these two, so that for Audio playing directly from the disc there is never a problem of not enough data being supplied to the Audio Processor or buffer under- or overflow. The same is true for Soundmap timings. Also these are absolutely accurate in relation to the disc rate. The nominal rate is 75 sectors per second and the Green Book does not describe a tolerance; in reality the tolerance will be in the order of typical crystal oscillator tolerance, which is always better than 10 to the power -6. This is the most stable time base CD-I has.

3. The Video rate.

Of course the first obvious point to notice about the Video field rate is that there are two different possibilities: 50 Hz. and 60 Hz. The application can check the CSD of the player to find which one of the two is the actual one. It can then adapt itself to the proper timing. One point to note is that this time base is not absolute. The Green Book allows timings that slightly deviate from the nominal 50 or 60 Hz. In this regard it is worth mentionning that also in broadcast NTSC the actual field frequency is slightly lower than 60 Hz. So because the timings in the Green Book are only recommendations, differences between manufacturers can exist. There is no worst case figure given in the Green Book, so it is not wise to use this timebase as the basis for an application. On the other hand, smooth video effects can best be achieved by locking the effect itself to the field rate, so a method has to be found to decouple the whole video effect duration from the main timing of the application. More on that later.

4. The system tick

The Green Book says that the system tick is 10 msec. +/- 0.2 %. This seems quite a large tolerance for crystal controlled systems like CD-I. However it does allow manufacturers to generate the 10 msec. by dividing an already existing clock in the system by a fixed divider, rather than taking a separate crystal for it. This makes the systems simpler from a radiation and interference point of view. So the result is that the 10 msec. has this tolerance. However a specific player will be on a specific point in the tolerance range and it will sit at that point with great stability. The clock in other words does not fluctuate. It is merely set at a point that deviates slightly from the nominal value. This clock is used by the system internally to do task switching in a multitasking mode. Also updating of pointer registers etc. happens typically at a multiple of the system tick.

5. An example of a complication.

The different clocks in the system can lead to problems. A simple illustration of such a complication would be an animation title for which the animation was created at 10 frames per second. Let us assume the playback software gets the animation data from the disc into a memory buffer at a rate of ten frames per second and puts it on the screen based on the system tick timer. Every ten ticks the next frame gets displayed. Suppose the animation is run length coded and has a maximum frame size of 10 Kbytes. The application has allocated a 60 K buffer and starts to display the animation once the buffer is half full. The designer thought that this buffer would be adequate to absorb potential timing variations. Now suppose however that the software plays back on a player that is worst case in regard to the system clock.
Now it is easy to see that the influx of data into the buffer is at a different rate from the outflux, so that it is only a matter of time before the buffer over- or underflows. In this particular example the net difference between ingoing and outgoing data is 0.2% of 100 KBytes per second, or 200 bytes per second. So in (30K/200=) 150 seconds a problem appears in the buffer management. A brute force solution is increasing the buffer size till the longest piece of animation on a disc will still just be correct, but it will be clear that this is not a general or optimal solution.

6. What strategies are possible ?

From the problem description it is clear that you will have to select one of the time bases to be the dominant one in your application and that you will have to take precautions that the other ones are properly handled. Which one should you take as the dominant one ? That depends on the nature of the application. A video game that has only disc accesses to load a new playing field, and only occasional soundmap play for sound effects might well be synced to the video, in order to get the smoothest possible sprite movement. Since there is no conflict with disc access or audio play, this strategy works. However in the more general case that there is either continuous play from disc, or continuous audio play, there is really only one choice: the disc data rate, and coupled to it, the audio processor rate. The reason for this is that the application cannot influence this process at all. The only remote possibility for influencing the disc data rate would be to operate the disc in Start-Stop mode. However, the one second access time that the Green Book specifies for the next access means that this is not a suitable method for "fine tuning" the disc data rate. However, given a buffer large enough to bridge a one second gap it is a possibility. In the case of continuous audio play a fine tuning is impossible. Every action the application could potentially take would be noticable to the user, so this is ultimately the deciding factor: the disc/audio rate is the master clock. Of course it is perfectly valid that within one application there are dynamic transitions from one time base dominance to another, depending on the state of the application. An example would be to use the disc time base during audiovisual sequences and the system clock for blinking etc. during menu selections. Also keep in mind that use of different time bases only leads to problems if there is a real conflict, like in the case of buffer management. Most activities in a CD-I application can be completely asynchronous. As an example looking at the pointer position every 30 milliseconds based on the system tick can be perfectly combined with data flowing from the disc controlled by the disc rate and color cycling the cursor based on the video field rate. All of these issues are totally independent of each other.

7. Video effects.

One of the examples where the result of a video field based timing is usually not a problem is in the area of video effects. Normally it is not a problem that if you have programmed a wipe to last e.g. 120 fields, that in a 60 Hz. system this transition will last 2 seconds, while in 50 Hz. systems it will last 2.4 seconds. Although it is possible to check the CSD and adapt the application to compensate for this, this is not necessarily optimal: The video effect is typically smoothest if it is coupled to the field rate. So a wipe of two or three pixels per field (=192 or 128 fields) will be smoother than a wipe that has a non-integer relationship with the field rate: in that latter case it will periodically seem to "Hiccup". As long as the main timebase of your application is coming from the disc or the timer and the visual effects are isolated, you can set this up very easily. It becomes a differnt matter if you have one long visual effect during the duration of your presentation. A good example of this would be lyrics that scroll vertically for the duration of a song. While it is possible to space the lines of lyrics differently for 50 and 60 Hz. playback, a video-only timebase will not compensate for potential time base differences between disc rate and video rate. So without precaution audio and video could gradually loose sync during the longer songs. The only remedy for this would be to check periodically for the relationship between the scrolling lyrics and the File Position Pointer, and slow down or accelerate the scroll for a single field in order to absorb potential differences.

8. How to set up your system.

Let's go back to the animation example and see how we could make the disc data rate the dominant time base. There are four possibilities to get information from the disc/audio time base: Either by buffer full signals in the PCL, by monitoring the file position pointer, by Trigger/EOR signals and by SM_Done signals in the case that soundmaps are used. Buffer full is not very suitable for run length data, because there is no fixed size per frame. In theory one could preload a table with all of the sizes of the individual frames and modify the buffers dynamically to match the incoming frame sizes, but this is perhaps not the most practical option. The File Position Pointer is a very real possibility. You have to take into account that this pointer only gets updated when a selected sector enters the system. This should not be a problem in this particular example, and it can be forced in the general case. Trigger/EOR signals could be sprinkled throughout the disc image and be mapped to the sector that is closest to a 100 msec. multiple. From a runtime perspective this would be easy to handle. However building a disc image this way is not easily achieved without undue manual intervention. SM_Done signals could be suitable, but using them requires routing the audio through memory with its associated memory and CPU overhead. Also the granularity of Soundmaps is fairly course, especially at lower audio quality levels.

All in all, the File Position Pointer method would seem easier. Having decided that, how should this pointer be interrogated ? Unfortunately there is no possibility to have it signal the application when a certain position has been reached, so the only option is to check the value periodically. One of the options for this would be to use the video event: it can be programmed to signal back after a number of fields, so that no unnecessary polling is taking place. In the case of a 50 Hz system, you program it to send you a signal after 4 fields; in the case of 60 Hz., after 5 fields. Then you check the File Position Pointer and check whether it has reached a position 100 mseconds after the previous update. If not, arm the Video event for the next field and repeat the exercise. This method ensures that also the display timing is based on the disc time base. It is easy to see that a similar strategy could be created on the basis of the system timer. Of course these are not the only methods or even necessarily the best ones. One could devise different solutions based on the same principle of disc rate dominance. One could base the display on the system tick (every 10 ticks) and periodically (for instance every 5 seconds) poll the File Position Pointer and adjust the time till the next frame update to 9, 10 or 11 ticks, depending on where the File Position was. This is probably even easier to implement. Which exact solution is chosen is not so important, as long as the mechanism prevents the buffer problem.

One could even bypass the File Position Pointer altogether by monitoring the degree of filling of the buffer and adapting the readout speed accordingly (when between 1/3 and 2/3 full, use nominal readout; when less than 1/3 full, use slower readout; when fuller than 2/3, use faster readout). Again, the suggestions in these examples are just that: examples. You can design your own solution to accomodate the needs of your particular application.

9. Summary and conclusion.

The CD-I system has three different timebases. These are fully asynchronous and two of them have a tolerance. In most cases this is not a problem, since the effects that are based on them are typically independent as well. There is the potential for complications when for instance two of the timebases are used respectively for filling and emptying a buffer. Without precautions, over time a buffer management problem will develop. This note has illustrated that by relatively simple means these problems can be avoided, as long as you are aware of the underlying mechanisms.