THE DISK DRIVE AS AN AUDIO RECORDER

Francis Rumsey

University of Surrey, U.K.

Introduction

The following tutorial paper describes how a block-structured mass storage device, such as a Winchester disk drive, may be made to function as part of a digital audio recording and editing system: the so-called 'tapeless recorder'. It outlines the principles of random access sound file storage and buffering, together with a discussion of digital audio requirements and the historical precedent for disk recording.

Background

There is something of a historical precedent for the storage of sound on disks. Edison and Berliner both brought sound reproduction to the masses, back in the early part of this century, in the form of random access storage media (a cylinder in the case of the former, and a disc in the case of the latter). It was not until wartime (1940's) that AC-biased tape recording became a viable proposition, although some experimental work had been going on prior to this, and the advent of tape recording brought with it two distinct advantages over disc recording: firstly the possibility for higher sound quality, and secondly the possibility for editing.

Editing and high sound quality have remained the prime advantages of tape over disc until recent years, when digital audio has meant that sound might be stored in numerical form, as binary data, on media other than tape. Sound, once in digital form, appears as any other data might appear to a computer system, although there are some constraints on sound data from the operators point-of-view, such as the need for unbroken recording and replay. Computers often deal with data in a time-segmented, block-structured, or burst format, whereas audio engineers like their music replayed without the short gaps that such processing might imply! Provided that such hurdles may be overcome, audio data may be stored on virtually any mass storage medium which is capable of holding computer data, also provided that the performance of such a store is adequate for the real-time replay and recording of one or more channels of sound data (see below). It should be stressed that, provided any errors are corrected, the sound quality of digital audio is almost entirely independent of the medium on which it is stored.

Disk drives and solid state RAM (random access memory) form the basis of data storage in today's computers, and they have the advantages of fast access to stored data, especially when compared with tape which has very slow access to stored material, due to the need for serial spooling to distant material. Sound quality now being independent of the storage medium, it is possible to concentrate on the operational advantages of random access disk storage over serial access tape storage (see Fig. 1).

Digital Audio Requirements

The storage of sound onto computer disks is only possible if that sound is digitised. In other words it must be subject to sampling and quantisation, as in any digital tape recorder, a process which assigns binary values of a certain word length to sample pulses which have been modulated by the analogue audio waveform (the process of *pulse code modulation*, or PCM). In professional audio systems which use conventional linear PCM (that is without any data compression or non-linear compression), 16-bit sampling has become accepted as the norm, since this is the resolution of the Compact Disc, although greater resolution in professional systems may be desirable in some circumstances.

The sampling rate dictates the number of 16-bit words produced by a PCM system per channel per second, and this must be at least twice the highest audio frequency. In professional systems, a sampling rate of 48kHz is the norm, although 44.1kHz is used for CD mastering, and 32kHz in some broadcast systems. The higher the sampling frequency the higher the maximum audio frequency which may be successfully reconstituted, and 48kHz allows for an audio bandwidth in excess of 20kHz. In a digital recording system using a sampling rate of 48kHz with 16-bit

resolution, 768,000 bits of data are produced per second. This is equivalent to 94 kbytes (a byte is eight bits, and a kbyte is 1024 bytes). Compared with the amount of data normally produced by a computer system in applications such as word processing, audio recording produces a high data rate per second.

In order to store one minute's worth of monophonic audio at the above rates, 5.5 Mbytes (Megabytes) of storage space would be required. (A typical 3.5in floppy disk holds just under 1 Mbyte). If more bits per sample or a higher sampling rate are required (in order to achieve greater recorded resolution) the result will be a greater storage requirement per second, and for this reason it might be suggested that systems should operate at rates which are sufficient, but not excessive, unless storage space is unlimited. Techniques such as oversampling may be used to improve sound quality without increasing storage requirements.

Digital audio samples, since they are time-discrete, may be processed and stored either contiguously or non-contiguously, provided that they are re-assembled into their original order at a fixed sampling rate before they are converted back into the analogue domain. This attribute makes digital audio ideal for storage on block-structured media such as disks, provided that buffering is employed at the inputs and outputs to such a system in order to smooth the transfer of data to and from the disk (see below). Editing of sounds may be accomplished in the digital domain by the joining of one recording to another in RAM, using the buffer to provide a continuous output, and the fast access time of the disk drive makes it possible to locate recordings made at different times in a fraction of a second. Multi-channel recording may be accomplished by dividing the store between the channels, such that a store which might hold one hour of mono audio could be used to store half an hour of stereo or fifteen minutes of 4-track, provided that the transfer rate of the store was adequate for the purpose (see below).

Principles of Sound File Storage on Disks

The Sound File

It is important to grasp the concept of the 'sound file' as an individual sound recording of any length. With tape there is the distinct possibility that parts of the reel will have been recorded at different times, and thus there will be sections of the tape, perhaps separated by leader tape, which represent distinctly separate periods of time: they may be 'tracks' for an album, 'takes' of a recording session, or short individual sounds such as sound effects. This is the closest that tape recording gets to the concept of the 'sound file': that is a unit of recorded audio, of fixed size once it has been recorded, although the size of the unit may be anything which fits into the available space.

In the tapeless system, one must grasp the concept of a 'sound store' in which no one part has any specific time relationship to any other part: no one section can be said to be 'before' another or 'after' another. In this store may be kept a number of 'sound files' of different lengths, and containing different material. It is possible that one file might be a ten minute music track whilst another might be a one second sound effect. Essentially, one may keep as many sound files in the store as will fit in the space available. Anyone familiar with computers will already have a concept of storage space on media such as floppy disks, and will know that there is a limit to the number of word processor files that may be stored on a disk of fixed size. Indeed we are dealing with a similar approach to the word processor in tapeless audio, except that instead of text files we are now concerned with sound files, but as far as the storage device is concerned there is really not much difference between one and the other.

Sound File Storage on Disk

Just as in any well-organised filing system it is necessary to be able to find where items have been stored so that they can be retrieved again. In a computerised filing system, which is really what the tapeless audio system is, a 'directory' is used as an index to the store. The directory contains entries specifying what has been stored, the size of each file, and also its location. By referring to the directory the system can retrieve previously stored files by looking in the appropriate place. Every time a new file is stored, an entry must be made by the system into the directory specifying the location of the file. In the case of large files, the directory entry may refer to an 'index' or look-up-table in which is stored a list of locations which must be read sequentially to re-assemble a file.

Disk storage often differs from the storage of paper documents and also from the storage of sound files on tape, in that a file may well be broken up and stored in a number of different locations. This need not worry the operator of such a system because he will never usually be confronted with the problem of finding the pieces of a sound file, this job being done by the system, using the directory as a reference. Within the directory or its sub-indexes, the locations of all the pieces of the sound file will be registered, and when the particular file is requested by the user the system will re-assemble the pieces by retrieving them in sequence from the various storage locations.

Buffering

Buffering is the key to the process of ensuring that time-continuous data may be broken up, and that broken-up data may be made continuous again. A buffer is a short term store which holds a small portion of audio data at any one time, and it is normally made up of RAM (random access memory). It may be imagined as analogous to a bucket of water with holes in it, because this neatly demonstrates the processes involved. A bucket with a small hole in the bottom may be filled intermittently, but the water will tend to come out of the hole continuously. Provided that the average flow rate of water entering the bucket is the same as the average rate at which it flows out of the hole, then the bucket will remain at the same state of fullness. If water flows out of the hole faster than it is coming into the bucket then the bucket will eventually become empty. Furthermore, the water could enter the bucket continuously and could be let out of the hole in bursts, the important point being that the reservoir of water makes possible the translation from continuous to burst flow or vice-versa (see Fig. 2).

In digital audio terms, the bucket is the RAM and the water is the audio data. Disk media require that audio is broken into blocks for storage, and this is achieved by filling the RAM to a given point from the continuous audio input, and then reading out of the RAM in bursts to the storage medium. On replay the RAM is filled in bursts from the store, and read out at a constant rate (the sampling rate) to give an audio output. Provided that the average rate of data flow into and out of the buffer is the same, the buffer will remain at a steady level of fullness. If data is read out faster than it is written in then eventually the reservoir will run dry, and the result will be either no audio at the output (on replay) or nothing written to the store (while recording). In order to preserve the original order of samples, the buffer must operate in the FIFO (first-in-first-out) mode.

The analogy could be taken further, as it may be appreciated that there might be more than one hole in the bucket (more channel outputs), larger holes in the bucket (higher data rates) or a tap with low water pressure (a slow storage device). It should be pointed out that the bucket analogy does not hold water if examined too closely, as water will flow faster out of the holes in the bucket the fuller the bucket is: this does not hold true for memory buffers in tapeless recording systems!

The buffering of digital audio has a number of other useful possibilities. Firstly, it can be used to ensure that any short-term timing irregularities in the data coming from the storage device will be ironed out and will not be allowed to affect audio quality. Data written into memory from the store, even if it has timing jitter, can be read out from the store at a constant steady rate, under control of an accurate crystal clock. The only penalty of buffering (which may not be noticeable, depending on the mode of operation) is that it introduces a small delay between the input to and the output from the buffer, the extent of which depends on the delay between the writing of samples to the RAM and the reading of them out again. The maximum delay is limited by the size of the buffer, as with a small buffer there will come a point where the memory is filled and must be emptied to some extent before any new samples can be written in.

Secondly, the buffer may be used for synchronisation purposes. If audio data being read from a storage device is required to be synchronised with an external reference such as timecode, then the rate at which data is read out of the buffer can be finely adjusted to ensure that lock is maintained. Thirdly, buffering is vital to the process of editing in tapeless recording. Since sections of audio which may be stored in a variety of different locations are to be joined together, there may well be short breaks between the acquisition of one section and the acquisition of another from the store. It will be the job of a buffer to smooth out the effect of the discontinuity, and to ensure a continuous output at the join.

In a simple disk-based system, whilst recording, continuous audio data from the system's A/D convertors, or from a digital input, will be written into an area of the buffer, and from here it will be written to the appropriate parts of the store as and when the store is ready, making up a sound file whose details and locations are written into the directory (see above). On replay, a file may be read from the store into the buffer, and from there in continuous form to the D/A convertor or digital output (see Fig. 3).

It is worth introducing at this stage the notion that there might be more than one audio input to or output from the buffer (in other words, more than one audio channel is being handled) in which case certain parts of the buffer would be reserved for individual channels, acting as 'sub-buffers'. As far as the user is concerned these might as well be separate entities, although they are likely to be sub-divisions of a large block of RAM, with the system software keeping track of which blocks of addresses correspond to which channels.

A Generalised Tapeless Recording System

Fig. 4 shows a generalised block diagram of a tapeless recording system. The system consists of a user interface, a central processor (CPU) to handle the overall control of operations, a digital signal processor (DSP) to handle the real-time processing of audio data, a block of solid-state RAM (random access memory) to act as a temporary store for audio data, a direct memory access (DMA) controller, and a storage device which is likely to be a disk drive with its associated controller. There are also audio interfaces, both analogue and digital, connected to the system via further memory to act as a buffer (see below), as well as interfaces for timing information such as SMPTE/EBU timecode which communicate with the timing controller.

In the operation of the system, audio data is transferred to and from the store to the RAM, usually using a technique known as direct memory access, whereby data may be transferred directly from the store to the RAM or vice-versa without having to pass via the central processor. This assists in speeding up transfer. Data is transferred between RAM and the audio interfaces via the buffers, under control of the central processor which takes commands from the user interface. During editing, and any other audio processing operations such as fading or mixing, data is written from the store to the digital signal processor, via RAM, which in turn passes it to the buffered audio outputs. It may be appreciated that the RAM and the buffers need not necessarily be separate entities, as the buffers will often just be areas of RAM set aside for the purpose of ensuring continuous audio input and output in real-time.

Performance Requirements of Storage Devices

The system software, under the user's direction, will regularly be requesting the output of sound files from the disk in normal operation. It may also be organising the input of sound files to the disk. *Access time* is that time taken between the system requesting a file from the disk and the first byte of that file being accessed by the disk controller. It may also refer to the average time taken for the storage device to jump between one block of a file and another block which may be located in a different place, as the file may not be stored in contiguous blocks (see below). In a disk drive, access time is governed largely by the speed at which the read/write heads can move accurately from one place to another, and to a smaller extent it is governed by the physical size of the disk and the speed at which it rotates. If the head is told by the controller to go to a particular location then there will be two delays to consider: one while the head moves radially across the disk to the required position (seek latency), and one while the disk rotates until the desired block passes under the head (rotational latency). Fig. 5 illustrates this concept.

Transfer rate is the speed at which data can be transferred to and from the disk once the relevant location has been found. It will be measured in bits (or more usually Megabits) per second. Transfer rate, in conjunction with access time, limits the number of audio channels which can successfully be recorded into or replayed from the disk simultaneously, and these two factors also limit the freedom with which long crossfades and other operational features may be implemented. Going back to the bucket/buffer analogy, the transfer rate corresponds roughly to the rate at which water comes out of the tap into the bucket, and the access time corresponds to the gaps between times when the tap is turned on. If it is assumed for the moment that a hole exists in the bucket out of which water can flow at a constant rate (it would not in the real world, but let us assume that there is a valve in the hole which keeps the flow rate constant), then water will continue to flow out of the hole while there is water in the bucket. The water flowing out of the hole corresponds to the audio output of a tapeless

recording system. The bucket will become empty only if a) the bucket is filled too slowly while the tap is turned on (too low a flow rate), b) if the gaps between fills are too long, or c) if the rate of flow out of the hole is too great.

Returning to the real world, it has already been stated that at a sampling rate of 48kHz, using 16-bits, the data rate for one channel amounts to 768,000 bits per second, or around three-quarters of a Megabit per second. Thus it might be assumed that a disk with an instantaneous transfer rate of 0.75 Mbit per second would be able to handle the replay of one audio channel's data satisfactorily. If the store is made up of solid state RAM which has a negligible access time (of the order of tens or hundreds of nano seconds) then a transfer rate of 0.75 Mbit/s, in terms of the speed in bits per second at which the system can transfer data from the RAM to the convertors, might well be adequate. With a disk drive, the access time (of the order of milliseconds) will severely limit the average transfer rate to the buffer. Although the instantaneous transfer rate from the disk to the buffer may be high, the gaps between transfers as the drive searches for new blocks of data will reduce the effective rate. This is not to say that fast transfer rates are not desirable, because it is the combination of access time and transfer rate that go to make up the *effective* transfer rate. What is needed is fast transfer rates *and* fast access times.

To take an example, assume a disk drive with an average access time of 30 milliseconds and a transfer rate of 10 Mbit/s. If the access time had been near zero then the transfer rate of 10 Mbit/s would have allowed the audio data for some thirteen channels to be transferred at the rates given above, but the effective transfer rate in real operation will bring this number down to perhaps six channels or less for safe, reliable operation in a wide variety of operational circumstances. It may be clear by now that the optimisation of efficiency of data transfer to and from a storage device will depend on keeping the number of accesses to a minimum for any given file, and this requires careful optimisation of the size and position of the audio data blocks.

Fragmentation of Disk Stores

As files are stored by the system they will be assigned to blocks on the disk, and the blocks' locations will be entered in the index table for that file. If the file is subsequently required by the system it will read each of the appropriate blocks in sequence to re-assemble the file. Since the size of a block on a disk drive is typically 512 bytes, most files will span a number of blocks.

It may be that the store has already been used many times and that old files have been erased to make room for new ones. Unless a new file is exactly the same size as the space left behind by an old one the new file will be required to use a variety of locations, some of which belonged to the old file and some of which did not. If every new file were to use only brand new storage space which had never been used before the store would soon become full. The more a store is used, the more 'fragmented' it will become: that is, the more little spaces there will be dotted around which are no longer being used, as the file to which they related has ceased to exist. These 'fragments' will be used up in the storage of new files, although it will mean that newer files will possibly be more fragmented than old ones (see Fig. 6).

Fragmentation may not be a bad thing for tapeless sound recording, as has been proven in a number of cases by researchers looking into the optimum division of sound files and their associated blocks for storage on disk drives. It may in fact be desirable to fragment sound files for multi-channel operations where there is no fixed relationship in time between the sound files to be sent to different outputs. The whole approach to the design of a successful tapeless sound storage system is a perpetual trade-off between efficiency, number of output channels, size of memory buffer, length of cross-fade allowed, and so on. Fragmentation is good for efficiency in some cases, yet bad for it in others, to some extent depending on the application, and in any case it cannot usually be avoided!

The Allocation Unit

Typically, a data block (that is the smallest addressable unit in the store) may contain 512 bytes of information, although some magneto-optical disks use 1024 byte blocks. This is very small in relation to the size of a digital audio file of even moderate length, and if a file were to be split up into blocks of 512 bytes spread all over the disk then transfer efficiency would be impossibly reduced due to the large number of accesses required to different parts of the disk.

For this reason a minimum allocation unit (AU) is usually defined, which is a contiguous sequence of blocks that are always used together, in order to improve efficiency. It might be that an AU would contain 8 kbytes of audio data, which in the case of 512 byte blocks would correspond to sixteen blocks. The size of the AU must be small enough to engender efficient use of the disk space in cases of fragmentation, and large enough to engender efficient data transfer.

Physical placement of AUs corresponding to the same sound file on the disk surface or surfaces must be as carefully arranged as possible to allow the shortest access time between them, but effort spent in ensuring this is often negated when a store becomes badly fragmented or nearly full as there is then no option but to store data where there is space available. These factors depend considerably on the performance of the disk drive, whether it is able to transfer all the data on one track in one revolution, and what its access time is. The problem of AU placement becomes further complicated in the case of multi-channel operation, and when sound files for different channels may be accessed with virtually any time relationship between them.

Multi-Channel Storage, Transfer and Buffering

Allocation of Storage Locations

In a tapeless system which caters for a number of inputs and outputs one has to consider how many channels may be recorded at the same time and also how many may be replayed simultaneously. It is clear from what has been said already that the maximum number of channels simultaneously to be recorded or replayed will be limited by the effective transfer rates to and from the store. In some cases a number of storage devices will be used in a multi-channel system, with each device being allocated to the handling of only a small number of channels. In this way the transfer rates of each are kept within reasonable limits and performance at all times is assured.

Allocation for True Random Access

One of the principle features of the tapeless system is its ability to replay files to any output channel at any time, because any pre-conceptions about fixed time relationships between files no longer hold true. The idea that any file may be required at any time by the system, and routed to any output, is one reason why the designer does not need to bother quite so much about the physical placement of files and AU's in relation to each other, and why fragmentation of a disk drive is not really as much of a problem as it might have appeared.

If files which had once been close in time were to be placed physically close on the storage device this would favour a particular timing relationship between them, as it would give an access time advantage in the case where these files were required together, but a disadvantage if one of them was needed at the same time as one which was physically far away. Because of this it might be suggested that any storage strategy which places any one file physically closer to another than it is to the rest of the files in the store will favour a particular time relationship between those files. If there is the possibility for true random access to any file at any time then the only viable strategy is for files to be randomly distributed throughout the store, in which case no one relationship is favoured over any other.

Allocation for Serial Access

The technique of pseudo-random storage described above maximises transfer efficiency in the case of true random access, but in the case of a system acting more like a tape recorder, in which the tracks are played sequentially and fixed in their time relationship, it may be more sensible to adopt a strategy in which audio data is written to disk in a physically sequential order, such that a serial time relationship is established. If such a situation is likely, then possibly the random storage strategy would result in lower efficiency.

REFERENCES

Abbott, C. (1984) Efficient editing of digital sound on disk. Journal of the Audio Engineering Society, **32.6**, June, pp 394-402.

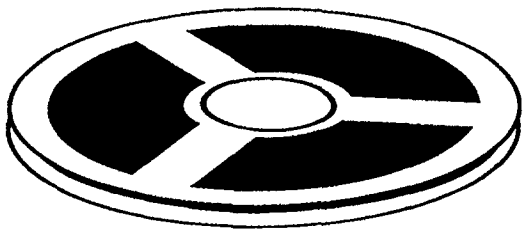
Ingenretsen, R.B. and Stockham, T.G. (1984) Random access editing of digital audio. Journal of the Audio Engineering Society, **32.3**, September, pp 114-121.

McNally, G.W., Gaskell, P.S., and Stirling, A.J., (1985) Digital Audio Editing. Presented at the 77th AES Convention, Hamburg, March. Preprint. Audio Engineering Society.

Rumsey, F. (1990) Tapeless Sound Recording. Focal Press, London and Boston

Watkinson, J. (1988) The Art of Digital Audio. Focal Press, London and Boston.

TAPE



High storage capacity

Slow access to material

Editing involves physical cutting or electronic copying

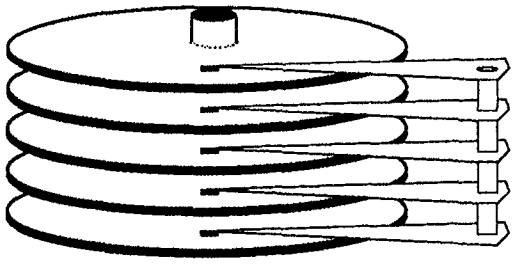
Reasonable cost

Removable

Highly suitable for multitrack recording

Psychological advantage

DISK



Medium-high storage capacity

Fast access to material

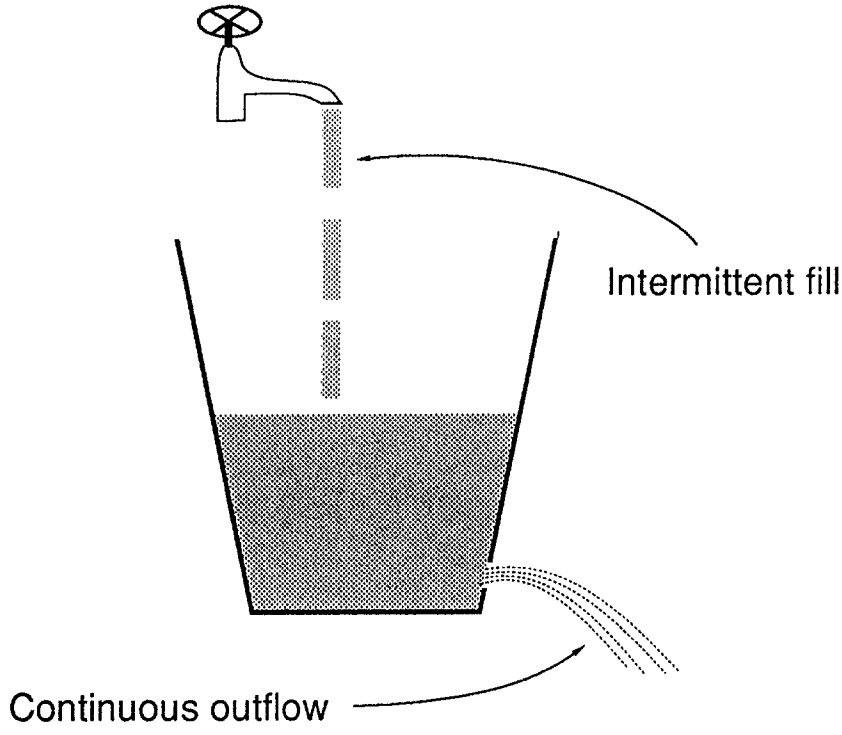
Editing involves no cutting or copying

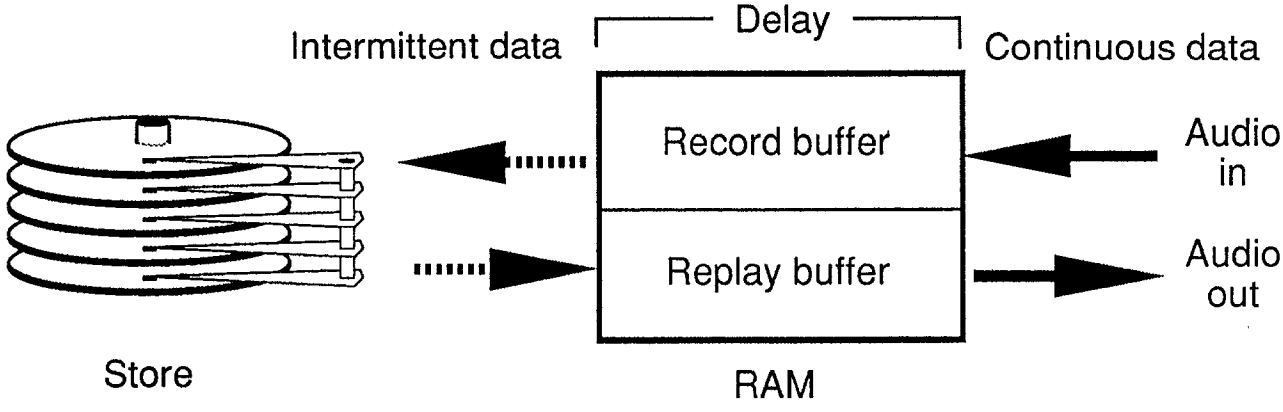
Cost depends on drive type

Some types are removable

Only moderately suitable for multitrack recording

Figure 1





Figures 2 and 3

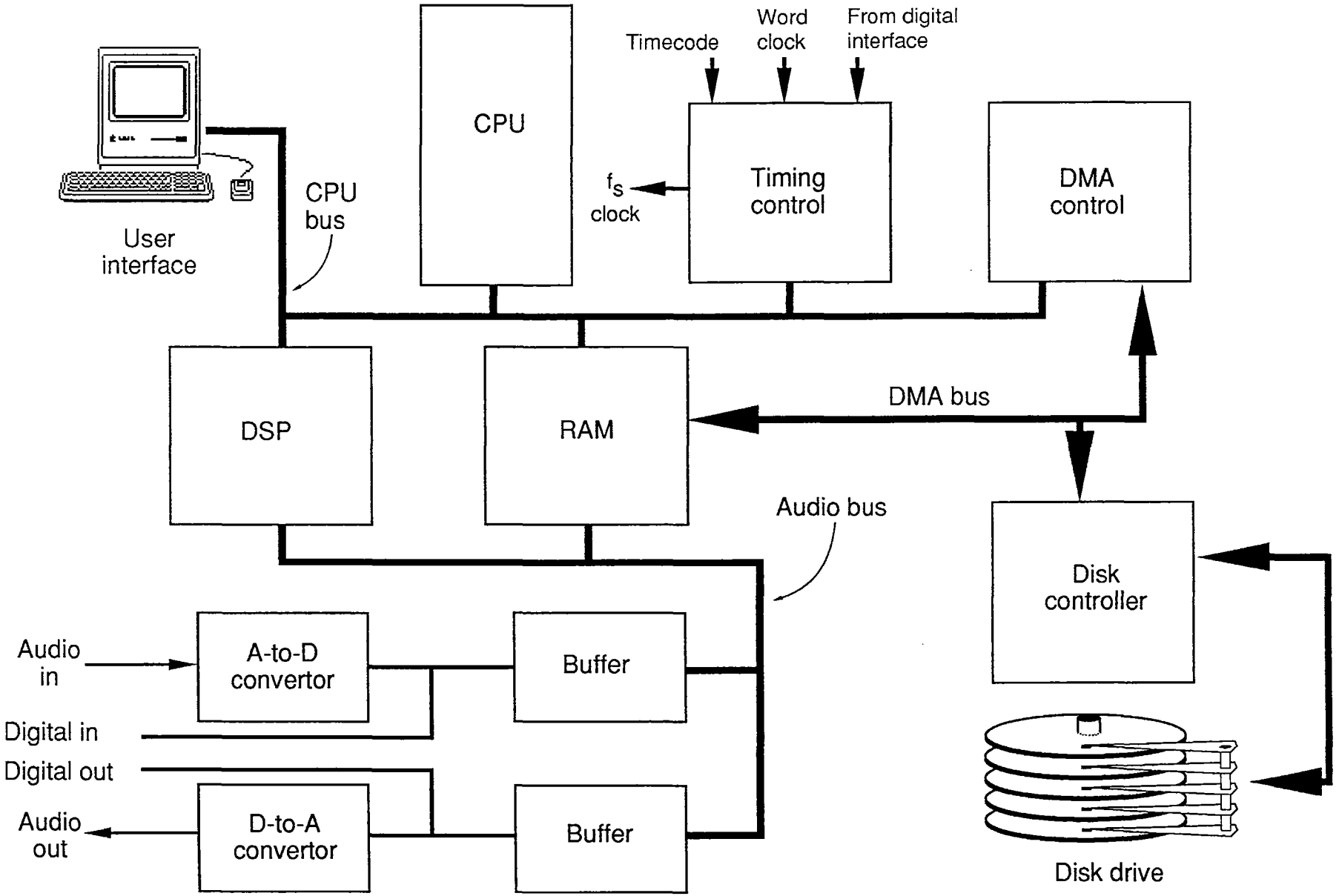

Timecode
Word clock
From digital interface
CPU
f_s clock
Timing control
DMA control
User interface
CPU bus
DSP
RAM
DMA bus
Audio bus
Disk controller
Audio in
A-to-D convertor
Buffer
Digital in
Digital out
Audio out
D-to-A convertor
Buffer
Disk drive

Figure 4

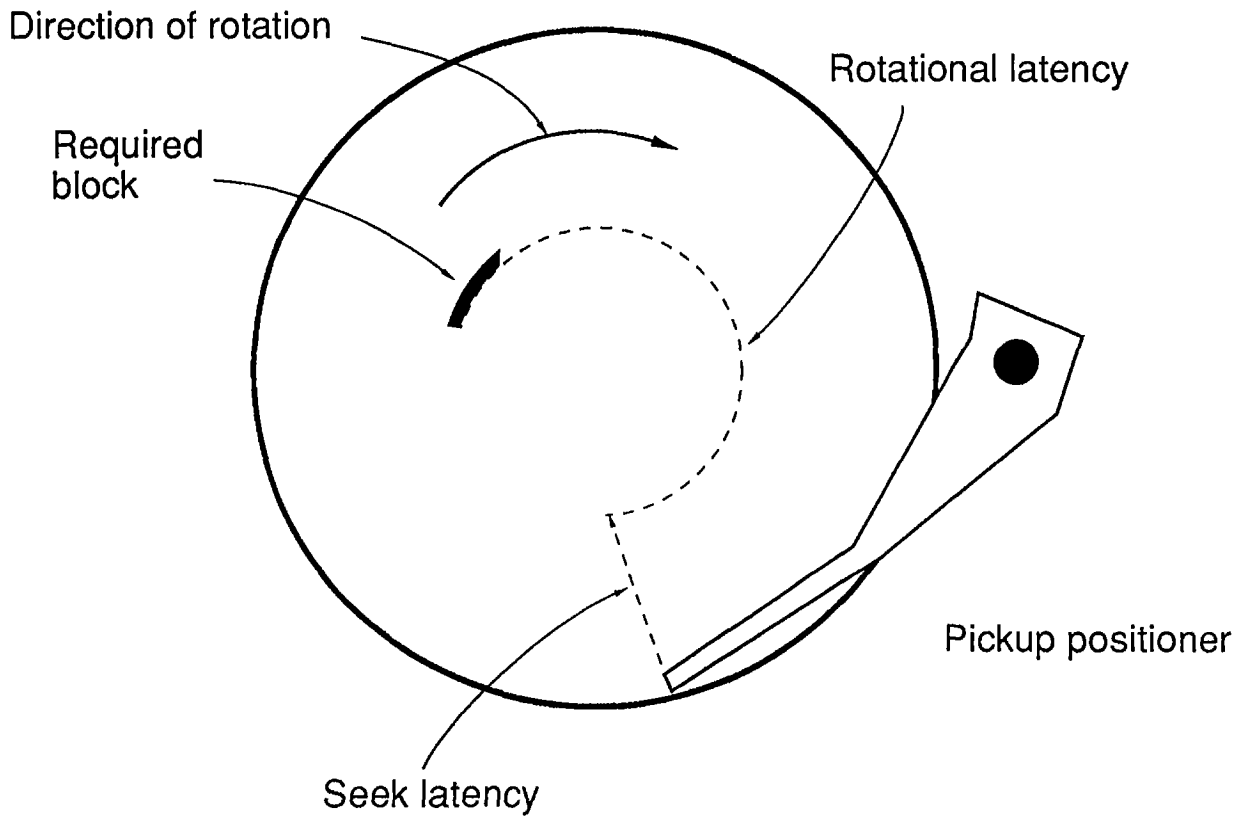

Direction of rotation
Rotational latency
Required
block
Pickup positioner
Seek latency

Figure 5

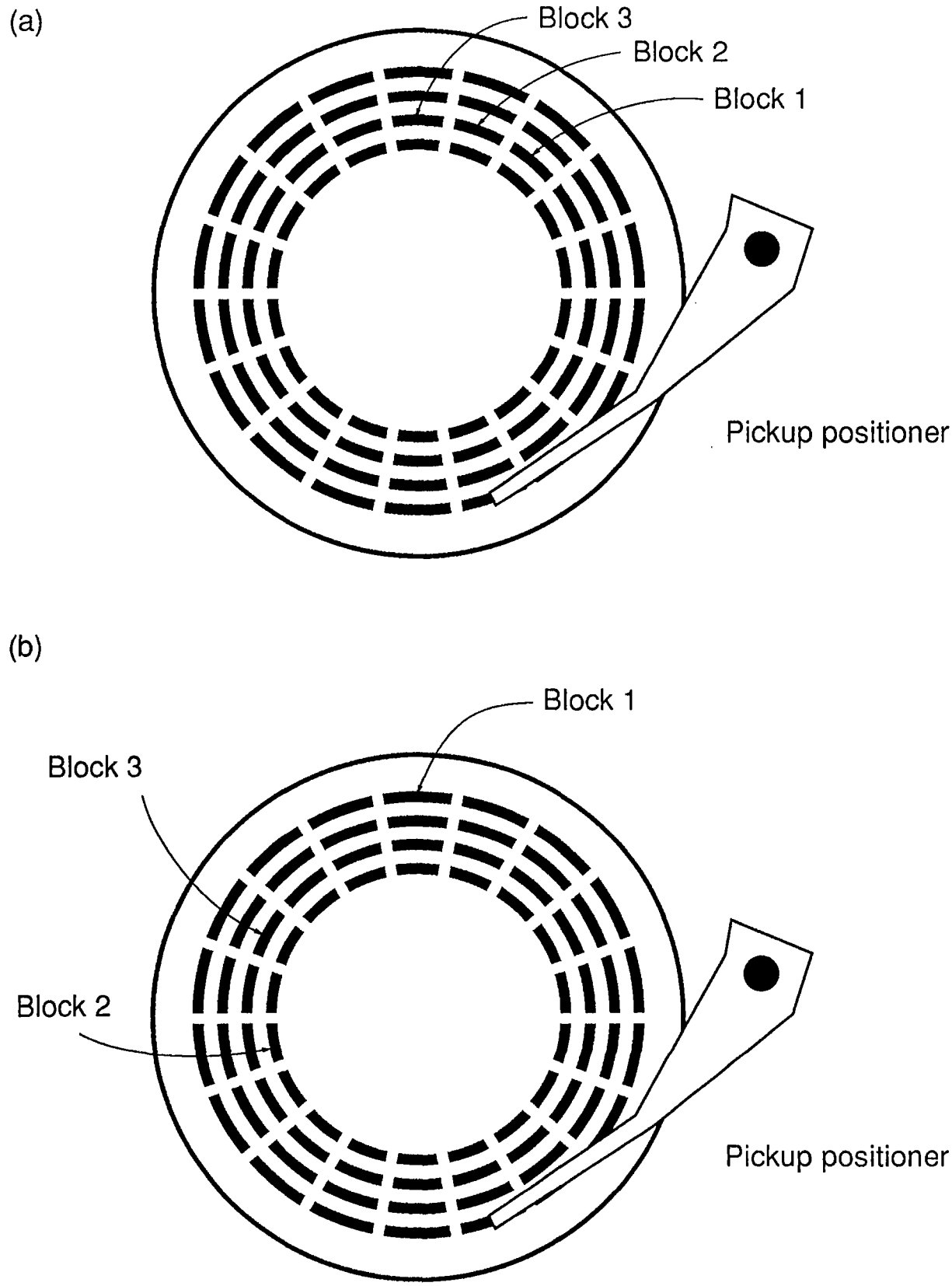

(a)
Block 3
Block 2
Block 1
Pickup positioner
(b)
Block 1
Block 3
Block 2
Pickup positioner

Figure 6