RTMPS copy

DATA RECORDING DISK DRIVE

This invention relates in general to data recording disk drives, and in particular to sector formats for multiple track data storage media used in fixed block architecture (FBA) disk drives.

All disk drives require some means for determining the radial and circumferential position of the read/write heads over the disks so that the heads can be accurately positioned over any desired track and sector. Typically, this is accomplished by placing servo information on one or more of the disk surfaces for use by magnetic or optical heads in determining their positional orientation over the disk. In sector-servo (also known as embedded servo) disk drives, the servo information is interspersed with data on each disk surface. This approach has the advantage of providing the positioning information close to the data sectors it identifies, thereby eliminating sources of track misregistration which otherwise tend to limit track density. However, a disadvantage of the sector servo approach is that it incurs additional overhead in order to permit transitions between data regions and servo regions and to distinguish data regions from servo regions.

Much attention has been focused in recent years on reducing the overhead associated with sector servo architectures. One approach, known as the no-ID format, is disclosed in EP-A-522 750. No-ID disk drives use servo sectors in combination with a defect map to identify the data sectors and completely eliminate the use of an ID region. Using the no-ID format, each sector on a track is composed of two regions: a servo region and a data region. The servo sectors are located using a servo ID mark or address mark. ECC may be added to track ID information to provide a more robust servo pattern. Each data sector is identified by its cylinder, head and servo sector number counted from an index location. This format is the same for substantially all sectors on all tracks of the disk.

A second strategy which has been used to improve recording density in sector servo disk drives in recent years is known as zone bit recording (ZBR), as taught by Hetzler in U. S. Patent 5,210,660. In ZBR disk drives, the disk is divided into multiple zones oriented in the radial direction. Each zone is comprised of a set of tracks. Since tracks in the outer zones are longer than those in the inner zones, the tracks in the outer zones may store more data than the tracks in the

inner zones. Typically, data is stored in sectors, each of which has the same number of data bytes. In this configuration, the additional capacity in the outer zones is utilized by having a larger number of data sectors on each track in the outer zones. This results in the number of data sectors per track varying from zone to zone. In order to provide a constant servo sampling rate for all zones, a single fixed number of servo sectors is used across the entire disk. The combination of a varying number of data sectors per track and a fixed number of servo sectors per track can result in some of the data sectors being split by servo sectors. An example of a disk formatted according to US-A-5,210,660 is shown in Fig. 1, where data recording disk 101 is split into three zones -- 102, 104, and 106. Each zone is comprised of a plurality of tracks 103. Each track has a number of data sectors 105 with associated ID fields 107. Various servo sectors, designated as 108, are shown interspersed with data sectors 105 around the disk. An index location 109 is shown, where the data sectors in each zone align with a servo sector 108. A portion of a track on the disk is shown expanded at 110. Four complete data sectors are shown (130, 122, 132 and 124), each with their associated ID field (140, 141, 142 and 143). Three representative servo sectors 125, 126 and 128 are also shown. As can be seen from this example, some data sectors will be split by servo sectors, and some data sectors will not start immediately following a servo sector. For example, data sectors 122 and 124 are split by servo sectors 126 and 128, respectively, while data sectors 130 and 132 are not split by servo sectors. Data sectors 122, 132 and 124 and associated ID fields start immediately after another data sector, rather than immediately following a servo sector.

The necessity of splitting data sectors and of having some data sectors that do not start immediately following a servo sector presents complications which heretofore have prevented the use of ZBR and no-ID together in the same disk drive. For instance, in No-ID disk drives, the physical location of a data sector is derived from the address mark field, which is also used to locate the servo sector. However, this technique is dependant on a fixed, constant one-on-one relationship between the locations of the servo sectors and the data sectors, a relationship which does not exist in a ZBR-formatted disk drive.

A technique has recently been introduced which addresses part of the problem by providing electronics to generate timing pulses to mark the locations of data sectors which are not necessarily adjacent to servo

sectors and which may be split by servo sectors. The technique was introduced by AT&T in the ATT93C010 servo channel/multiprocessor chip which generates a start of data sector pulse for each data sector starting between two servo sectors. This is achieved through the use of programmable registers whose values are updated at every servo sector. Two values are required at each servo sector: the number of clocks (the length) from the prior servo sector to the start of the first complete data sector; and the number of data sectors which start before the next servo sector. Also, the system must know the number of clocks required for a full data sector, a value which is typically constant for each zone.

However, while the ATT93C010 is able to locate the start of a data sector, it cannot identify a data sector -- that is, distinguish it from other data sectors, such as by computing its data sector number. In fact, it cannot even compute a partial data sector number for use in distinguishing a sector from others on the same part of a track. As such, it is insufficient for use in a disk architecture which must both locate and identify data sectors without using an ID field.

Another possibility for locating data sectors without using ID information is to add a servo-style address mark prior to each data sector. This approach ensures that each data sector can be located independently of a servo sector. However, it suffers several drawbacks. First, it does not allow the disk drive to be reformatted with a sector size different from the original sector size since the address marks must be written by a servo writer which permanently fixes the disk format. Second, this approach can increase the complexity of the servo write process, because additional steps may be required to create the additional address marks. Third, the address marks occupy space on the disk, increasing the overhead. Fourth, the address marks require a write-to-read recovery region between data sectors, further increasing the overhead. Finally, when used with a magneto-resistive read/write head and micro jog technology, each address mark must be reliably read in a partially off-track position during write operations. This requires guard bands at the zone boundaries, since the data address marks do not line up with one another from one zone to the next. The guard bands, of course, further increase the overhead penalty associated with this method.

THIS SHEET HAS BEEN REPLACED

Accordingly, there has existed a heretofore unmet need in the art for a sector architecture which effectively combines the ZBR and no-ID formats, which sector architecture enables the data recording head to locate and identify data sectors for read and write operations without resorting to an address mark and without requiring write-to-read recovery between adjacent data sectors.

Accordingly, viewed from a first aspect the present invention provides a data recording disk for use with a fixed-block architecture disk drive having a head capable of reading positioning information, the disk being divided into a number of radially spaced tracks, at least one of the tracks being divided into a number of angular sectors including only data sectors for recording user data and servo sectors having pre-recorded head positioning information for identifying track and servo sector locations, wherein the number of servo sectors on the track is not equal to the number of data sectors on the track, and wherein information establishing the circumferential locations and identities of the data sectors is encoded within the servo sectors.

Viewed from a second aspect, the present invention provides a fixed-block architecture embedded servo disk drive comprising: a data recording disk having a number of radially spaced tracks, at least one of the tracks being divided into a number of angular sectors including only servo sectors and data sectors, wherein the number of servo sectors on the track is not equal to the number of data sectors on the track; recording head means which reads information in the servo sectors and which writes and reads user data in the data sectors; means, responsive to the information read from the servo sectors, for determining the locations of the data sectors based on their distance from the servo sectors; and means, responsive to the information read from the servo sectors, for determining the identities of the data sectors.

Viewed from a third aspect, the invention provides a method for locating and identifying a selected data sector in a fixed-block architecture embedded servo disk drive having a data recording disk with radially spaced tracks and circumferentially spaced angular sectors including data sectors and servo sectors and having a recording head which reads information in the servo sectors and which writes and reads user data in the data sectors, the method comprising the steps of: reading information from a preceding servo sector; receiving from electronic storage sector layout information; computing a data sector

This Sheet Has Been Replaced



THIS SHEET HAS BEEN REPLACED

number from the sector layout information; and computing a distance from a preceding servo sector to the start of the data sector as a function of the sector layout information.

Preferably, the locations and identities of data sectors are continuously computed as each data sector arrives at the recording head until the selected data sector is located and identified. Alternatively the location and identity of the selected data sector is computed in advance of the selected data sector reaching the recording head and wherein its arrival is signalled by a sector pulse.

It is a further preferred feature of the method that the computed distance is represented as a clock count.

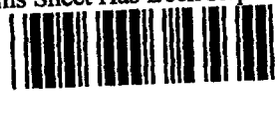
Preferably the method further comprises, after computing the data sector number, the step of adjusting the data sector number to compensate for cylinder and head skewing. The step of adjustment preferably includes modifying the data sector number by a skew value.

Viewed from a fourth aspect, the present invention provides a fixed-block architecture embedded servo disk drive comprising: a data recording disk having radially spaced tracks and circumferentially spaced angular sectors including data sectors and servo sectors; a recording head which reads information in the servo sectors and which writes and reads user data in the data sectors; servo electronics which determines, based on information read from a single servo sector, the locations and identities of at least two subsequent data sectors, and which further determines, based on information read from the same servo sector, whether that servo sector splits a data sector.

Preferably the information read from the single servo sector is used to retrieve additional information from electronic storage.

Viewed from a fifth aspect, the present invention provides a data recording disk for use with a fixed-block architecture disk drive having a head capable of reading positioning information, the disk having radially spaced tracks and circumferentially spaced angular sectors and a track architecture which divides a track into a plurality of repeating segments, each segment having a plurality of equally spaced servo sectors and a plurality of data sectors, at least one of the data sectors being split by a servo sector, the data sectors containing no data

This Sheet Has Been Replaced



THIS SHEET HAS BEEN REPLACED

identification information for use in locating the circumferential positions of the data sectors and no data identification information for use in identifying the data sector numbers of the data sectors.

It is preferred that the servo sectors include positioning information followed by servo pad fields, the data sectors include data pad fields, and the at least one split data sector includes a split pad field.

It is further preferred that the servo sectors contain information for use in locating the circumferential positions of the data sectors and information for use in identifying the data sector numbers of the data sectors.

It is yet a further preference that the at least one segment includes a stub field.

Viewed from a sixth aspect the present invention provides a fixed-block architecture embedded servo disk drive comprising: a data recording disk having radially spaced tracks and circumferentially spaced angular sectors including data sectors and servo sectors, wherein for at least one track the number of data sectors is not equal to the number of servo sectors; a recording head which reads positioning information in the servo sectors and which writes and reads user data in the data sectors; and a head positioning system wherein the data sectors are identified and located using formatter electronics in cooperation with servo electronics, wherein the formatter electronics converts a logical block address to a data sector number, and wherein the servo electronics uses the data sector number to provide a byte clock count representing the distance from a preceding servo sector to the data sector identified by the data sector number.

Preferably the servo electronics includes sector pulse logic which accesses a format table containing segment information for each core. It is further preferred that the sector pulse logic includes a data counter, a pad counter, and a sync counter which count byte clocks used to locate and identify data sectors. In operation, the sector pulse logic of the disk drive uses a servo sector number received from the servo sector counter to address the format table.

This Sheet Has Been Replaced



THIS SHEET HAS BEEN REPLACED

It is further preferred that, in operation, the sector pulse logic computes the starting location and data sector number of data sectors on the data recording disk. Preferably the sector pulse logic receives a servo modulo count for use in computing data sector locations and identities.

It is a preferred feature that the format table includes storage, for each segment subdivision of the data recording disk, and each region within a segment, containing the location and identification information for the data sectors in the segment. Preferably the format table includes storage, for each zone, which enables the servo electronics to locate and identify data sectors which are not positioned immediately following a servo sector.

It is further preferred that the servo electronics includes a data sector number counter which counts data sector pulses from index and which is preset at servo sectors. Presetting of the data sector number counter avoids latency after a zone switch and also avoids latency after return from a power saving mode.

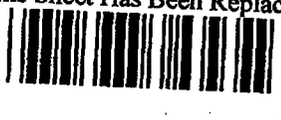
Preferably the preset value provides a skew between data sectors and servo sectors.

It is further preferred that the disk drive further includes a servo modulo counter which continuously computes segment numbers and servo sector numbers corresponding to the position of the recording head relative to the data recording disk. The inputs to the servo modulo counter preferably include stagger offsets, which preferably include offset values and timing adjustment values. It is preferred that the stagger offsets are organized into a look-up table according to head shift value.

It is further preferred that the data sector number is adjusted to compensate for cylinder and head skewing, said adjustment preferably including modifying the data sector number by a skew value.

Thus in a preferred embodiment, to be described in detail below, a data recording disk drive is provided with an FBA sector architecture which enables a data recording head to identify and locate data sectors based solely on information obtained from electronic storage and from servo sectors which need not be adjacent to the data sectors. The data

This Sheet Has Been Replaced



THIS SHEET HAS BEEN REPLACED

recording disks in the disk drive are divided radially into zones, each zone including a number of tracks. Each track contains a number of data sectors and a number of servo sectors, with the number of data sectors varying from zone-to-zone. The tracks contain servo information and data, but no data sector ID information. Format information is maintained in the electronic storage to describe the data sector layout for each zone. The tracks in each zone are circumferentially divided into segments. Included in each segment is a number of data regions separated from one another by servo sectors. All data regions in a given segment are the same number of bytes in length, but this length need not be an even multiple of data sectors. Accordingly, various regions may contain any of the end of a data sector whose start is located in the previous region, one or more complete data sectors, and the start of a data sector whose end is located in the following region. The format information provided to identify and locate data sectors, including ones whose region locations are not adjacent to servo sectors, includes an entry for each region in the segment. Each entry includes the lengths of the first and last data sections in the region, whether the sectors represented by those sections are split between two regions, the total number of data sectors in the region, and the sector number of the first data sector in the region. In addition to the information recorded for each region, data is maintained, for each zone, to locate data sectors which do not start immediately following servo sectors. This data includes the number of bytes in a complete data sector, the length of a complete data sector (measured in byte clocks), the length of a VCO synchronization field (measured in byte clocks), the length of a VCO resynchronization field following a servo sector (measured in byte clocks), the lengths of one or more data pad fields (measured in byte clocks), and the numerator and denominator of the reduced fraction of the ratio of the number of data sectors per track and the number of servo sectors per track. Portions of both the region information and the zone information may be stored in random access memory accessible to electronics in the disk drive; other portions may be stored in a set of registers also accessible to the micro processor.

Also in accordance with the invention, a method is provided to compute the starting location and sector number of any data section based on the above-described information. A set of counters is used to time the length of the various fields in the format to compute the starting location of a required data sector. The data sector number from index is

This Sheet Has Been Replaced



THIS SHEET HAS BEEN REPLACED

computed based on a relative data sector number, the servo sector number from index, and the reduced number of data sectors per track.

Further in accordance with the invention, a method and system are provided for continuously developing a servo-modulo count for use in computing segment numbers and servo sector numbers. The servo-modulo counter computes two quantities: an index value to the segment layout information for the current zone; and the data sector number for the first data sector in the current segment. The continuous availability of the servo-modulo count allows a disk drive configured in accordance with the present invention to avoid a latency penalty which would otherwise be incurred in waiting for an index mark as a basis to compute segment numbers and servo sector numbers.

Further in accordance with the invention, a method and system are provided to identify the servo sector number following a head switch, in particular for disk drives which are written with staggered servo patterns. A portion of the servo sector number is written into each servo sector, and combined with offset values obtained from electronic storage to produce a servo sector number following a head switch.

Further in accordance with the invention, a method and system are provided for developing data sector numbers from servo sector numbers where the relationship between the first data sector on a track and the first servo sector varies within a zone, and from zone to zone. This extends the benefits of the present invention to disk drives designed with track skew which minimizes latency for head and cylinder switch operations.

A preferred embodiment of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Fig. 1 is a schematic diagram illustrating a fixed block sector architecture in accordance with the prior art.

Fig. 2 is a schematic diagram illustrating a fixed block architecture disk drive in accordance with the present invention.

Fig. 3a is a schematic diagram illustrating a segment subdivision of a data recording track in accordance with the present invention.

This Sheet Has Been Replaced

Fig. 3b is a schematic diagram illustrating a track format with a stub in accordance with the present invention.

Fig. 4 is a block diagram illustrating a hardware embodiment of the servo functions of the present invention.

Fig. 5 is a block diagram illustrating the storage components used to compute data sector locations in accordance with the present invention.

Fig. 6 is a flowchart illustrating servo-modulo computations in accordance with the present invention.

Fig. 7 is a block diagram illustrating a hardware embodiment of a servo-modulo counter in accordance with the present invention.

Fig. 8 is a schematic diagram illustrating a staggered sector servo written disk drive.

Fig. 9 is a schematic diagram illustrating an LBA to PBA mapping architecture in accordance with the present invention.

Fig. 10 is a flow chart illustrating LBA to PBA conversion computations in accordance with the present invention.

Fig. 11 is a memory map table illustrating zone conversion storage components in accordance with the present invention.

Fig. 12 is a flow chart illustrating PBA to zone, cylinder, head, sector computations in accordance with the present invention.

I. Sector Architecture

Shown in Fig. 2 is a disk drive configured in accordance with the present invention. The disk drive is formatted using a fixed block architecture with sector servo and zone-bit recording. The disk drive, designated generally as 202, includes data recording disk 204, actuator arm 206, data recording transducer 208 (also called a recording head), voice coil motor 210, servo electronics 212, read/write electronics 213, interface electronics 214, formatter electronics 215, microprocessor 216 and RAM 217. Data recording disk 204 includes center of rotation 211,

and is divided for head positioning purposes into a set of radially spaced tracks, one of which is shown at 218. The tracks are grouped radially into a number of zones, three of which are shown as 251, 252 and 253. The disk contains a plurality of servo sectors 220, which extend across the tracks in a generally radial direction. Each track has a reference index 221. Within each zone, the tracks are also circumferentially divided into a number of data sectors 254. As will be discussed hereafter, the data sectors contain no sector ID fields. In accordance with the normal meaning of "fixed block architecture", all data sectors are substantially the same size, expressed in bytes of data. However, it should be noted that the present invention may easily be adapted to tolerate some variation in data sector size, such as from 512 bytes per sector to 520 bytes per sector, in the event such a configuration was desirable for a particular implementation. The number of data sectors per track varies from zone to zone, and some of the data sectors do not begin immediately following a servo sector. Further, some of the data sectors are split by servo sectors. If the disk drive has multiple heads, then the set of tracks which are at the same radius on all surfaces is referred to as a "cylinder".

Read/write electronics 213 receives signals from transducer 208, passes servo information to servo electronics 212, and passes data signals to formatter 215. Servo electronics 212 uses the servo information to produce a current at 240 which drives voice coil motor 210 to position recording transducer 208. Interface electronics 214 communicates with a host system (not shown) over interface 262, passing data and command information. Interface electronics 214 also communicates with formatter 215 over interface 264. Microprocessor 216 communicates with the various other electronics over interface 270.

In the operation of disk drive 202, interface electronics 214 receives a request for reading or writing data sectors over interface 262. Formatter electronics 215 receives a list of requested data sectors from interface electronics 214 and converts them into zone, cylinder, head and data sector numbers which uniquely identify the location of the desired data sectors. The head and cylinder information are passed to servo electronics 212, which is responsible for positioning recording head 208 over the appropriate data sector on the appropriate cylinder. If the cylinder number provided to servo electronics 212 is not the same as the track number over which recording head 208 is presently

positioned, servo electronics 212 first executes a seek operation in order to reposition recording head 208 over the appropriate cylinder.

Once servo electronics 212 has positioned recording head 208 over the appropriate cylinder, servo electronics 212 begins executing sector computations in order to locate and identify the desired data sector. As servo sectors 220 pass under recording head 208, the no-ID approach described in EP-A-522 750 is used to identify each servo sector. In brief, an index mark identifies the first servo sector, an address mark locates subsequent servo sectors, and a count of address marks uniquely identifies each servo sector. Additional information, which is described in greater detail below, is maintained in association with servo electronics 212 and formatter electronics 215 and is used to determine whether the present servo sector splits a data sector or whether a new data sector starts immediately following the present servo sector. Further information is maintained in servo electronics 212 and formatter electronics 215 which identifies the location of (or the distance to) the start of the next data sector from the present servo sector. Still further information is maintained which identifies the location of (or the distance to) any additional data sectors which begin before the next subsequent servo sector. Still further information identifies the number of the data sector from the index mark. This information is used to allow formatter electronics 215 to compare the data sector number passing under the recording head with the list of sectors received from interface electronics 214.

Shown in Fig. 3a is a detailed schematic diagram of the sector architecture for an exemplary track from a data recording disk in accordance with the present invention. A portion of a track is shown as 302, containing segment 304. Segment 304 is subdivided into a plurality of data regions, 306, 308 and 309. The data regions are separated from one another by servo sectors 310, 312, and 314. Segment 304 also includes a plurality of data sectors labelled D1 through D5. Finally, each data sector is made up of one or more data sections, labelled 320, 322, 324, 326, 328, 330 and 332. Logically, a segment is the set of servo sectors and data sectors having a unique spatial relationship between the sectors. The format for a track may then be produced by repeating the segment. A data region is the space between adjacent servo sectors. A data sector is the smallest individually addressable unit of user data, independently readable and writable. Finally, a data section

is a contiguous portion of a data sector, not interrupted by a servo sector.

The exemplary track of Fig. 3a contains a number of data sectors and a number of servo sectors, not necessarily equal. Note that servo sector 316 is not part of the segment since data sector D5 ends just prior to servo sector 316. Each data region contains a number of data sectors, some of which may be split by servo sectors. For example, region 306 contains the entire data sector D1 (section 320), and only a portion of data sector D2 (section 322). Likewise, data sector D2 is split by servo sector 312 into sections 322 and 324.

Also shown in Fig. 3a are details of the contents of servo sector 310. Write-to-read recovery and speed compensation field 342 is used to allow the read/write electronics to switch from a data writing operation to a servo reading operation, and to allow for fluctuations in the disk rotational speed. Address mark field 344 precisely identifies a specific position within the servo sector which is used as a timing reference. Position field 346 contains the actual servo information used to position the head, typically including a position error signal and other information such as a track number (track ID or TID), index value, servo sector number (or any portion thereof) and head number (or any portion thereof). Servo pad field 348 allows for the electronics to switch from reading servo to writing or reading data, as well as for disk rotational speed variations.

Also shown in Fig. 3a are details of the contents of data section 332, which contains a full data sector D5. VCO sync field 352 permits the read/write electronics to enable the voltage controlled oscillator (also known as a phase locked loop) to obtain proper phase lock for reading the data. Data and ECC field 354 contains the user data and error correction information. Data pad field 356 allows for differences in processing time for reading and writing data, as well as for flushing any encoder/decoder, and for disk rotational speed variations. It also provides sufficient time for the electronics to prepare for operating on the following servo or data sector.

Also shown in Fig. 3a is a detailed view of split data sector D2, labelled 360. Two additional fields are typically required when a data sector is split by a servo sector: split pad 364 and a VCO resync 368. Field 322 shows a portion of data sector D2 prior to servo sector 312.

Split pad field 364 allows for the electronics to interrupt the reading or writing of data in a manner well known in the art. Servo sector 312 is followed by VCO resync field 368, which is used to restore the system to allow for continuation of the read or write operation. Finally, a portion of data section D2 following servo sector 312 is shown at 324. Note that split pad field 364 may be the same number of bytes in length as data pad field 356, or it may be different. Also, VCO resync field 368 may be identical in content to VCO sync field 352, but this is not required. More capacity may be achieved by making fields 364 and 368 shorter than their counterparts 352 and 356; methods for achieving this benefit are discussed in the prior art.

For any given data recording disk drive, there is a fixed number of servo sectors per track (hereinafter designated as N) throughout the disk. Also, for each zone there is a fixed number of data sectors on each track (hereinafter designated as M). If M is not a multiple of N, then some of the data sectors will be split by servo sectors. The split portions of each data sector are denoted as sections. Further, the first section belonging to a data sector is called the primary section and any remaining sections are called secondary sections. Since all data sectors on a track have the same number of bytes, and since the servo sectors are equally spaced, there will be a limited number of unique data sections on the disk drive. The set of data sectors and servo sectors which defines one period of the unique pattern of data sections is called a segment. The number of data sections in a segment (hereinafter designated as nss) is given by:

$$nss = m + n - 1 \qquad (1)$$

where $\frac{m}{n}$ represents the reduced fraction of $\frac{M}{N}$.

Since m and n represent the numerator and denominator of the reduced fraction of the ratio of the number of data sectors per track to the number of servo sectors per track, it is apparent that there are n servo sectors and m data sectors in a segment. For exemplary segment 304 shown in Fig. 3a, n = 3, m = 5, nss = 7, N = 84, M = 140, and nst (the number of segments per track) = 28. It is to be noted that in accordance with

the no-ID sector architecture, neither the servo sectors nor the data sectors include data ID fields. Instead, the information necessary to identify data sector numbers and data sector locations is provided in servo sectors 310, 312, 314, etc. and in electronic storage accessible to the servo electronics, as will be described in greater detail below.

It should be noted that the choice of the segment configuration is flexible. For example, the entire track could be defined as a segment. In some circumstances, this may be the natural choice, such as when M and N are relatively prime, resulting in m = M and n = N. However, nothing precludes choosing $\frac{m}{n}$ to be an integer multiple of the reduced fraction. Moreover, there are cases where the above analysis on the ratio of the number of servo sectors to the number of data sectors is not the preferred choice for defining a segment. This can occur when a space is left at the end of a track, where the last data sector on the track ends substantially prior to the end of the track. Such a case is illustrated in Fig. 3b. The track is designated 380, and contains 7 servo sectors 384. There are 11 data sectors, DS1 through DS11. Note that data sector DS11 ends prior to the end of track 380. The remaining space is stub 385, which contains no user data since it is shorter than a data sector. In this example, N = 7 and M = 11, which from the above analysis would lead to a segment size of n = 7 and m = 11. However, Fig 3b illustrates a second possibility. The unique spatial relationship between the servo sectors and data sectors is achieved with n = 3 and m = 5. This choice results in two full segments, 381 and 382, and one partial segment 383. The disk drive need only know the total number of data sectors on the track to handle the partial segment. Once the data sector number has reached the maximum value, the drive will wait until the next servo sector, which resets the data sector counter to the first data sector number. It is to be noted that a track format having multiple stubs may be selected, including stubs located within tracks as well as at their ends. In any event, an advantage may in some cases be achieved in the presence of stubs by redefining the region using a smaller segment size since this in turn decreases the amount of memory required to store the format information.

II. Formatter and Servo Electronics

Fig 4. is a schematic diagram of the preferred embodiment of the servo and formatter electronics used to locate and identify data sectors

according to the present invention. Servo electronics 212 includes address mark (AM) detector and track number (TID) decoder 412, servo sector counter 414, safety logic 416, actuator position control logic 418, timing generation logic 420, sector pulse generation logic 421 and format table 422. Formatter electronics 215 includes defect map table 450, data sector identifier 454, logical block address (LBA) to physical block address (PBA) converter 456, PBA to zone, cylinder, head and data sector (ZCHS) converter 458 and control function 476.

In operation, formatter 215 receives a request for a read or write operation on a list of data sectors 264. The sectors are identified by their LBAs. The LBA list is converted to a PBA list by converter 456 using defect information 460. The PBA list 468 is converted to a list of physical ZCHS. Both of these conversions processed are discussed fully in section VI below. The cylinder and head values 466 (C and H) are passed to actuator position control logic 418 to affect a seek. Actuator position control logic 418 functions in a manner known in the art. Zone and sector values 464 (Z and S) are passed to data sector identifier 454 and to servo electronics 212. Additionally, servo electronics 212 receives servo information 266 from the read/write electronics. AM detector 412 detects the servo address mark, and signals address mark found (AMF) at 432. This signal is passed to timing logic 420, which generates the timing signals necessary for operation of servo electronics 212. AMF 432 is also passed to servo sector counter 414. In addition, AM detector 412 decodes the TID information, including cylinder (track) number, servo index, servo sector number and head number. Index signal 433 is used to reset servo sector counter 414, and the counter is incremented by AMF signal 432 at each subsequent servo. In this manner, the servo sector counter will always output the current servo sector number at 436. Safety logic 416 receives decoded TID information 430, and servo sector number 436. This logic performs various safety tests to ensure the proper operation of the servo electronics. Its functions include comparing the generated servo sector number 436 with any servo sector number information in the TID (including index) and processing any error handling information in the TID. Error information, along with the cylinder and head number, are output at 438. Actuator position control 418 compares the cylinder and head values 438 with the target values 466, and acts upon any errors. Sector pulse logic 421 uses servo sector number 436 to generate the address for format table 422, retrieving segment information 440 for the zone (described in detail below). Sector pulse logic 421 also contains three counters: a sync counter, a data

counter, and a pad counter. Each of these counters is used to count byte clocks during the various fields in the format, whereby the data sectors are located and identified. Sector pulse logic 421 is thereby able to identify both the starting location and the data sector number of the data sector about to pass under recording head 208. The current data sector number 442 is sent to formatter electronics 215, and a start of data sector pulse 444 is sent when the start of a data sector is under the head, enabling the formatter to perform functions with zero latency. Upon receipt of sector pulse 444, data sector identifier 454 compares the current data sector number 442 to the list of ZCHS values 464. If a match is found, this information is passed via 470 to control function logic 476, at which point the data sector is further processed in accordance with methods known in the prior art.

With reference to the fields described above, it is to be noted that many alternative configurations exist which would accomplish the same purpose. For instance, the LBA may be replaced with any logical identifier, while the ZCHS may be replaced with any value or combination of values identifying a unique sector number. The combination of the above electronics and microcode in microprocessor 216 is able to detect and act on any errors discovered between the target ZCHS values and the detected values. This provides the disk drive with a high degree of reliability, even in the absence of data ID fields. For example, if a detected track number does not match the target value, a seek error will be posted, and this error may be handled as is known in the art. Other errors, such as a mismatch between the servo sector counter and the TID information (index and possible servo sector number bits) will be detected by safety logic 416, for action by other parts of the drive. Such errors may be handled as data ID mis-compares, and recovered by forcing the servo sector counter to align with the index mark and repeating the operation. Finally, as will be described in greater detail below, an important input required for sector pulse logic 421 to continuously compute data sector locations is a servo-modulo count. The servo-modulo count allows for a reduction in the total memory required for the format information by taking advantage of repetitive patterns in the format.

III. Locating and Identifying Data Sectors

Shown in Fig. 5 are the components required for sector computation in accordance with the present invention. In general, there is provided

a random access memory 504 and a set of registers designated as 506, connected to an address and data bus (not shown). These may reside within servo electronics 212, in format table 422. Microprocessor 216 (or other electronics) stores information in the random access memory and in the registers, and then accesses this information in order to perform sector identification and location computations.

In particular, RAM 504 stores the information required to identify the data in each region in a given unique segment. For convenience of description the information fields are shown organized into a table format, although any appropriate data structure may be substituted. RAM 504 is addressed by the region within the segment (equivalent to the servo sector number within the segment). The fields required for each region include DS1Len 508, DS1Num 510, DS1V 512, NumFull 514, DS2Len 516, and DS2V 518. DS1Num is the number, from the start of the segment, of the first data sector following the servo. In the preferred embodiment, this field contains a 7 bit value. DS1Len is the length in bytes of the first data section in the region. In the preferred embodiment, this field contains a 10 bit value. DS2Len is the length in bytes of the last data section in the region. In the preferred embodiment, this field contains a 10 bit value. NumFull is the number of full (not split) data sectors in the region. In the preferred embodiment, this field contains a 3 bit value. DS1V is a flag, which if set, indicates that the first data sector is split, and therefore that the value DS1Len is valid. In the preferred embodiment, this field contains a 1 bit value. DS2V is a flag, which if set, indicates that the last data sector is split, and therefore that the value DS2Len is valid. In the preferred embodiment, this field contains a 1 bit value.

Since every segment within a given zone has the same number of bytes, for each zone it is only necessary to store format information for a single segment. The servo sector number within the segment (equivalent to the region number) is used to address the format information for the appropriate segment. The value DS1Num is included for performance reasons. Sector pulse logic 421 includes a data sector number counter, which starts counting data sector pulses at index. As long as the servo electronics remain active, the data sector number counter will be correct. However, when a zone switch occurs, the data sector number counter must be preset. On the other hand, it is preferable to avoid waiting for index to start counting data sectors, as this causes a latency penalty. With the DS1Num value, the counter is preset at every

servo sector, thereby avoiding additional latency. Further, this technique allows the disk drive to recover quickly from power saving modes, where much of the electronics is powered down when not performing data operations. Using the present invention, the data sector counter will be preset with the correct value at the first servo sector following the end of power saving, rather than at index.

From the definition of the sector architecture shown in Fig. 3a, only the first and last data sections in any given data region may contain partial sectors. Therefore, only the lengths of these two sections must be stored in the table; thus the inclusion of DS1Len and DS2Len. NumFull is the number of complete data sectors in the data region. It is used to prevent the generation of false data sector pulses prior to the end of a region, where the end of a data sector may be close the start of to a servo sector, by disabling the data sector number counter once NumFull data sector pulses have been generated. It is also used to control the generation of data sector pulses for the data sectors which start within the region. DS1V and DS2V are used to indicate the validity of the DS1Len and DS2Len values. This is prompted by the necessity to know if the first data section in a region is a primary section so that a read or write operation initiated at a servo sector will not be started on a secondary data section. For example, in region 308 the DS1Num value after servo sector 312 is D2 for data section 324. However, a read or write request for data sector D2 must start at data section 322, therefore the servo electronics must ensure that data section 324 is not mis-identified as the start of data sector D2. Similarly, DS2V is used to indicate that the last data section is split by a servo sector. The use of these flag values is preferred for performance reasons, since they provide for a direct decode. However, it is apparent that the limited range of values for DS1Len and DS2Len allows for the use of specific values (e.g. 0) to provide the same function.

Only the lengths of the first and last data sections in a region must be stored in the table because all other sections are necessarily full length. Data-section-split flags are required only for the first and last sections since all others must be complete. DS1V will be zero if the first section is primary (that is, not split). DS2V is likewise used to initiate the split data sector function at the end of the last section if it is split.

It is to be noted that the above-described information may be stored in RAM memory as shown in Fig. 5, and addressed by the servo sector number within a segment. However, it is also possible to organize the information by data sector number, which may in turn be used to address the RAM. In this case the specific fields must change, but they still support the functions described above. Further, any or all of the above information may also be stored in any other convenient medium, such as in registers, flash memory, or any other storage device accessible to servo electronics 212.

Aside from the above-described information which is stored for each region in a segment, additional information is required, for each zone, to enable the servo processor to locate data sectors which are not positioned immediately following a servo sector. In Fig. 5, this information is contained generally in registers 506, including Sync Long register 520, Sync Short register 522, Pad Long register 524, Pad Short register 526, and Data Full register 528. Registers 520-528 are used to load the three counters in Sector Pulse Logic 421. All three counters (sync counter, data counter, and pad counter) are down counters whose input clocks are the current data byte clock. Only one counter is enabled at a time, in a sequence determined by the track format. Additional information for other operations is contained in n register 530, which holds the reduced numbers of data sectors per track, m register 532, which holds the reduced number of servo sectors per track, NDS register 534, which holds the number of data sectors per track, and DSkew register 536, which holds the data sector skew value.

Turning in particular to registers 520-528, Sync Long register 520 contains the number of byte clocks in VCO sync field 352. Sync Short register 522 contains the number of byte clocks in VCO Resync field 368. In the preferred embodiment, this field is shorter than field 352; however, if the fields are of equal length then only Sync Long register 520 is required. Pad Long register 524 contains the number of byte clocks in Data Pad field 356 when it is followed by VCO Sync field 352. Pad Short register 526 contains the number of byte clocks in Data Pad field 356 when it is followed by a servo sector. The Pad Long and Pad Short values are different since the servo sector contains W-R and Speed field 342, which shares some common function with Data Pad 356. Data Full register 528 contains the number of byte clocks in Data and ECC field 354, which is the total number of data and ECC bytes in a data sector.

During operation of servo electronics 212 in cooperation with RAM 504 and registers 506, the sync counter is started following a servo sector. Prior to this, the sync counter is preloaded from Sync Long register 520 if the value of DS1V is 0, and from Sync Short register 522 if the value of DS1V is 1. Further, if DS1V is 0, data sector pulse 444 is generated and the data counter is preloaded with the value in Data Full register 528. Otherwise it is loaded with the value in DS1Len. When the sync counter reaches 0, the data counter is started. If the number of data sector pulses generated in the region equals NumFull, then the pad counter is loaded with the value in Pad Short register 526; otherwise it is loaded with the value in Pad Long register 524. When the data sector counter reaches 0, the pad counter is started. When the pad counter reaches zero, the end of the current data sector has been reached. If the number of data sector pulses generated in the region equals NumFull, then a servo sector follows. Otherwise a data sector pulse is generated, the data sector number counter is incremented, and the sync counter is preloaded with the value in Sync Long register 520. This process repeats until the region is completed, which is determined by the generation of NumFull data sector pulses. When the number of data sector pulses generated in the region equals NumFull and DS2V is 1, then the data counter is preloaded with the value in DS2V instead of with the value in Data Full register 528.

Three counters are used instead of one due to the types of fields being counted, to limit the number of tap points on a single counter, and to allow for each counter to be preloaded while another counter is running. It is to be noted that while a particular preferred counter arrangement has been disclosed, the above function can be implemented using many alternative counter and register arrangements whose result would remain within the spirit and scope of the present invention.

Using the above information, servo electronics 212 is able to locate the start of any data sector in a segment. Further, the sector number from index for a given data section may be determined using the equation:

$$SN[i] = m \times SGN + DS1Num + i \qquad (2)$$

where SN [i] is the data sector number from index (zero based) for the i-the data sector in the segment, and SGN is the segment number from index (zero based).

During operation of disk drive 202, read and write operations are received which require recording head 208 to be repositioned over various tracks and then to read or write various data sectors. Once recording head 208 is positioned over the appropriate track, the above equation is used to determine when the appropriate data sector is passing under the head. In particular, the equation is used to compute a current data sector number which is continuously compared with the data sector number requested for the read or write operation. If the values compare, the desired operation is performed. Since there are no ID fields, the data section lengths computed based on the data values described above identify the location of data in particular segments and regions.

In the preferred embodiment, various ones of the above-described information fields are maintained in various storage areas in order to improve operational performance. For instance, the information for the current zone may be maintained in dedicated local storage analogous to registers 520-536 in order to avoid bus arbitration. The values may be reloaded from general storage such as RAM 217 after each zone switch. The disk format determines the amount of storage required to hold the values for a particular implementation. Since known implementations require 32 or fewer regions per zone, the local storage requirements for the zone tables are 30 bits per region or 256 bytes per zone, including allowance for ECC. Thus, in the preferred embodiment, high performance may be achieved without incurring a storage access penalty.

IV. Servo-Modulo Counter

In order to use the above-described sector identification/location schema effectively, the system of the present invention must be able to determine both a segment number and a servo sector number within the segment prior to performing a read or write operation. To avoid the latency penalty created by waiting for an index mark as a basis to compute the above information, servo-modulo counter electronics are provided to generate this information continuously, even after a head or zone switch. In the preferred embodiment, this is achieved using a

hardware circuit whose inputs are m, n, servo (servo sector number) and DS1Num.

The purpose of the servo-modulo counter is to produce the quantities servo mod n and mx(servo/n). The former is the remainder of servo/n, the number of servo sector n from the start of the segment, which is used as the address for RAM-based zone table 504 described previously. The latter is the data sector number of the first data sector in the segment, hereinafter referred to as DSB, the data sector base number, which is also the first term in equation (2). Fig. 6 illustrates in flowchart form the operations required to perform these computations. At 602 the remainder value is initialized to the servo count. At 604, the quantity DSB is initialized to zero. At 606, a loop is entered which computes the value of servo mod n (remainder) by repeated subtraction until an overflow occurs. In particular, at 606 temp, a temporary variable, is assigned the value of remainder - n (of servo/n - n). At 608, the underflow test is performed (temp < 0). If an underflow is detected, the process is exited at 610. If no underflow is detected, the processing continues at 612, where the quantity m is added to DSB. Finally, at 614 the remainder is set equal to temp (the remainder minus n), and processing returns to 606. At exit point 610, both DSB (mx(servo/n)) and the remainder (servo mod n) have been computed.

Shown in Fig. 7 is a schematic diagram illustrating an electronic circuit in accordance with the preferred embodiment of the present invention. The circuit computes the values of mx(servo/n) and servo mod n in accordance with the process shown in the flowchart of Fig. 6 and described above. As in the flowchart, servo mod n is computed by repeated subtraction which is halted by an underflow operation. The circuit includes control logic 702, subtracter 704, adder 706, registers 708 and 710, MUX 712, adder 714 and data sector number counter 716. In operation of the servo modulo counter circuit, control logic 702 receives start signal 720 to begin a computation, and produces done signal 744 when complete. Register 708 stores the result of the repetitive subtractions and at the end of computations contains the correct value of servo mod n (the remainder). Subtracter 704 receives input 738 from register 708 and input 726 (the value n) from register 530 (shown and discussed previously with reference to Fig. 5). The input to register 708 is from MUX 712. The MUX allows either the numerator, servo 728, or the results of the previous subtraction to load register 708. The

register load signal (not shown) comes from control logic 702, and is asserted once per subtraction operation. Underflow signal 742 is passed to control logic 702. If an underflow is detected, the register load signal is stopped, and the remainder value will be stable on 740. Control logic 702 also controls the output from MUX 712 to register 708 via select line 732. MUX 712 is configured such that the first subtraction uses current servo count 728 while subsequent subtractions use the intermediate results. Further, register 708 is located prior to subtracter 704 so that it contains the results of the prior subtraction when the loop exits, since the exit condition is an underflow.

The value of mx(servo/n) is computed by repeated addition of the quantity servo mod m for each subtraction operation performed in the modulo block. Adder 706 adds input 730 (the value m) from register 532 (shown and discussed previously with reference to Fig. 5) and the result of the previous addition 746. The intermediate values of the addition 748 are stored in register 710 at the output of adder 706. The register load signal (not shown) comes from control logic 702, and is asserted once per addition operation. Control logic 702 synchronizes the addition and subtraction operations through the register load signals, thereby eliminating the need for a separate multiplier or an accumulator to compute DSB (mx(servo/n)). When the operation is complete, DSB value 746 produced by the circuit may be used in accordance with equation 2 to generate the current data sector number. Specifically, result DSB 746 is passed to adder 714, along with DS1Num 754, addressed by remainder 740. These values are added and output at 750 as the data sector number from index of the first data sector in the segment. Data sector number counter 716 is preloaded with adder value 750 upon receipt of a load signal from the control logic (not shown). Data sector number counter 716 increments upon receiving data sector pulses 756, and outputs the current sector number at 752.

To avoid a latency penalty which would otherwise be created upon head or track switches due to the time required to settle on the new track, it is well known in the art to use cylinder and head skewing between disk surfaces and tracks of a data recording disk drive.

In accordance with the present invention, a technique is provided for implementing this skewing by adjusting the value in data sector counter 716 to account for the skew. The raw (unskewed) current data sector number 752 is adjusted by subtracting the value in Dskew Register

536 from the current data sector number 752. This subtraction is performed modulo the value in NDS Register 534. The result is the current skewed data sector number which may be provided as input 442 to Data Sector Identifier 454. Finally, it should be noted that while this skewing technique has been described with reference to data sectors, it may also be used equally effectively to implement skewing on the basis of servo sector number in a manner analogous to that described above.

V. Staggered Sector Servo

The use of a hardware-based servo modulo counter in accordance with the preferred embodiment insures that no latency is added for head or zone switches. For a zone switch, the servo sector counter maintains count of the servo location. Once servo and data modulo values of m and n are changed, the section length and data sector number will be correct. Similarly, for a head switch, once the servo count is correct, the remaining values follow.

For a disk drive which implements a staggered sector servo approach, the servo counter value must be set properly following a head switch in order to account for the staggering. Fig. 8 illustrates, in cross-sectional view, a staggered sector servo disk drive. The disk drive is generally designated 802, and includes spindle 804 and disks 806, 808 and 810. The locations of the servo sectors are shown schematically as black rectangles in two groups 820 and 822. The servo sectors are not vertically aligned, rather they are arranged diagonal to permit the servo writing of all surfaces of the disk drive in a single pass write per track. After using one head to write a servo sector on one surface, the next head (on the next surface) is activated and a servo sector is written there, allowing multiple surfaces to be written in one revolution, decreasing the servo write time and cost.

To write a staggered sector servo pattern in accordance with a preferred feature of the present invention, the servo sector counter must be synchronized with the servo sector numbers on the surface being switched to. This function may be accomplished by writing a subset of the servo sector number (possibly the entire sector number) into the TID information in position field 346. Alternatively, servo sector counter 414 may be updated based on a lookup table containing the stagger offsets. In the former case, the servo sector number read from the servo

sector is decoded by AM detector 412 and passed to safety logic 416. Safety logic 416 uses this value to update servo sector counter 414.

For example, the entire servo sector number may be encoded in the TID. Following a head switch, safety logic 416 uses the read value to preload servo sector counter 414, ensuring synchronization. In the lookup table case, a table of offsets is kept which is used to increment or decrement the value in servo sector counter 414. An exemplary table is shown at 840. The table comprises 3 columns, head shift 842, servo sector counter offset 844 and servo timing adjustment 846. Upon commencement of a head switch, the head shift value (plus being down, minus being up) is used to look up the servo counter offset and timing adjustment. Offset values 844 are used to increment or decrement servo sector counter 414. Timing adjustment values are used by timing logic 420 to adjust for the new servo sector positions. For convenience, the values in column 846 are listed as fractions of the servo-to-servo spacing. Thus, for example, if the current head is on the lower surface of disk 806, and a head switch to the upper surface of disk 810 is desired, the head switch value would be +3. From lookup table 840 the servo sector count increment would be 0, and the servo sector timing adjustment would be 1/2 of the servo-to-servo spacing. The table construct shown in Fig. 8 also allows for more general offsets, such as skewing the index from surface to surface. This would result in a unique value for each head shift value. Further, the table may be used in conjunction with encoding the servo sector number in the TID to add a further degree of reliability to the system. Of course, this table may be stored in RAM or any other appropriate medium.

VI. Converting from LBA to ZCHS

As discussed previously, in order to find a requested data sector on a disk a received logical block address (LBA) must be converted into a zone, cylinder, head, sector (ZCHS) value. In general, this involves first converting the LBA, which is the user identifier for the data sector, into a physical block address (PBA) which is a mapping of the LBA into the physical space of the disk drive. The aforementioned application EP-A-522 750 teaches the use of a defect map and a basic process for performing the LBA to PBA mapping. However, since the LBAs for neighbouring defects share most of their high order bits, much of the information stored in the defect map is redundant, serving only to hamper performance and unnecessarily occupy RAM. The present invention includes

a particular map which removes the high order bits of each LBA to minimize the bits required in representing each defect. The map takes account of skip sectors allocated on the disk, either due to defects or sparing, with use of a minimum amount of RAM and in a fashion that increases the performance of the disk drive.

The map is implemented as a pair of tables known as a virtual track (VT) table -- which contains entries representing the mostly redundant high order bits of the LBA -- and a virtual sector (VS) table -- which contains entries representing only the low order bits of the LBA --. The output of the VT/VS table access, the PBA, is then located in a zone conversion table in order to develop coefficients for use in computing the cylinder, head, and sector at which to perform the desired operation on the requested data sector. Thus, two conversions are used, the first of which removes skip sectors and the second of which provides the appropriate ZCHS value. The invention reduces the storage required for conversion to either two or one bytes per entry plus an offset based on the drive capacity and choice of one or two byte entry. The invention further reduces the magnitude of the search required to locate an LBA and thus the performance impact of the searching process.

Shown in Fig. 9 is the LBA to PBA mapping architecture which forms the basis for the LBA to PBA portion of the conversion process. In particular, the figure shows the LBA represented in binary form at 904, virtual track table 906, virtual sector table 908, and defect spare table 910. As shown at 904, the LBA as received from the disk drive interface is first subdivided into two sections. The full length of the LBA in bits, shown as A, includes the number of bits in the high order portion required to hold the virtual track number, designated as B, as well as the number of bits required to hold the virtual sector number, designated as C.

A virtual track is defined as a contiguous set of data sectors which have exactly $2^{(A-B)}$ good data sectors. A virtual track may contain both good sectors and skip sectors or alternatively it may contain only good sectors. All virtual tracks are contiguous with one another starting from a defined virtual track of zero at the beginning of the disk drive and running through the entire disk drive. Sub spaces of the disk drive may also be mapped separately in the same manner. The virtual track number may be obtained from the LBA either by simply tapping the high order bits of the LBA or by logically shifting the LBA to the right

by the quantity A-B shifts. A virtual sector is defined as a good sector contained within a virtual track. There are 2^{A-B} different virtual sector numbers available. Thus, all virtual sectors within a virtual track are numbered consecutively starting at the first good sector in the track and ending at the last sector in the track.

Given a subdivision of an LBA into a virtual track number and a virtual sector number, the virtual track number is referenced into virtual track table 906 in order to obtain an index point into virtual sector table 908. The index point is used as a starting point in VS table 908 at which a sequential search is commenced, which search continues until a virtual sector number which is higher than the searched-for virtual sector number is located in virtual sector table 908 or the index exceeds the entry for the next virtual track. Once the appropriate entry is found, the PBA is computed as the received LBA plus an index into VS table 908 corresponding to the final search entry.

VT table 906 contains an entry corresponding to every virtual track in the disk drive. Each entry is at least B bits wide and contains a pointer into VS table 908. The arrangement of the table thus gives each entry two meanings. First, the entry value identifies the total number of skip data sectors prior to the virtual track represented by the entry. Second, this same value represents the correct entry point into the virtual sector table at which the skip sectors are listed for the virtual track number represented by the entry. In operation, VT table 906 is entered by indexing into it according to the value of B, the virtual track number, obtained from entry 904. The value found at the determined index point is then added to the start of VS table 908 to locate the first skip sector associated with the applicable virtual track.

Unlike VT table 906, VS table 908 does not contain an entry for every virtual sector on the disk drive. In contrast, VS table 908 need only contain entries for skip sectors (defective sectors and/or spare first sectors). Each entry in VS table 908 corresponds to the good virtual sector following skip sector. As discussed above, VS table 908 is entered at the index value established through use of VT table 906. Starting at that index value, a sequential search is conducted against the value of the C bits received from the low order of LBA value 904 until a virtual sector number is located which is greater than the value obtained from the low order C bits of 904. This greater-than virtual sector number establishes the number of skips which must be passed in

order to arrive at the correct virtual sector number. An offset value which is equal to the offset into virtual sector table 908 of the first entry which is greater than the sought-after virtual sector number is added to the LBA to arrive at the PBA value unless the next virtual track starts where the index is, in which case this offset/index is added to the PBA value.

DS table 910 contains an entry for each entry in VS table 908. The purpose of DS table 910 is to distinguish the virtual sector entries in VS table 908 as either corresponding to defective data sectors, also known as bad blocks, or to spare data sectors, also known as spare blocks. In the preferred embodiment, DS table 910 comprises a single bit entry corresponding to each entry of VS table 908, wherein the polarity of the bit in each entry is used to distinguish between bad blocks and spare blocks. It is to be noted that an advantage of DS table 910 is that this table does not need to be resident in RAM storage except during reassignment operations. Reassignment operations occur when grown defects develop during use of the disk drive, at which time spare sectors are reassigned as active sectors, and the defective active sectors are reassigned as skip sectors. Part of the reassignment operation involves changing bit values in DS table 910, in addition to various values in VT table 906 and VS table 908. Of course, in order to change values in DS table 910, this table must be resident in RAM accessible to the microprocessor. At all other times, DS table 910 need not be resident or readily accessible to the disk drive electronics, because there is no requirement that the electronics determine whether a sector identified in VS table 908 is bad or spare; rather it is only necessary for the electronics to determine that the sector is a skip sector.

Shown in Fig. 10 is a flowchart illustrating the LBA to PBA conversion computations. The conversion computations perform three basic functions. First, they set the bounds on the search to be conducted through VS table 908. This is accomplished by examining the virtual track number entry identified in the LBA along with the succeeding virtual track number entry in VT table 906 to establish starting and ending offsets in the virtual sector table. Second, once the bounds of the search are set, the conversion process sequentially searches through VS numbers in the VS table until its position in the table corresponds to the starting point of the next virtual track (that is, the upperbound of the search) or a VS table entry greater than the C bits is found. Third, as a criterion of the search, each virtual sector number in virtual

sector table 908 must be greater than or equal to its predecessor unless a virtual track boundary is crossed, at which point searching is ended.

With particular reference to Fig. 10, the variable D is taken to represent the quantity A-B, where A is equal to the number of bits in the LBA while B is equal to the number of bits in the virtual track number. Thus, D corresponds to the number of right-hand shifts of the LBA required to obtain the virtual track number. The variable E represents the quantity $2^c - 1$, where c is equal to the number of bits in the virtual sector number. Thus, E simply provides an AND mask for use in determining the virtual sector number. It is to be noted that the values of D and E are established according to system parameters. For instance, since most modern microprocessors handle bits in multiples of 8, it may be desirable to establish D and E so that all values are provided in multiples of 8 bits. Finally, p refers to the present offset in the virtual sector table.

The flow chart of Fig. 10 is entered at 1004, where VTN is set equal to the LBA shifted right by a quantity of D bits. Next, at 1006, the virtual sector number is set equal to the LBA anded with E. Thus, at this point the logical block address has been separated into its virtual track number and virtual sector number components. Next, at step 1008, the present offset into the virtual sector table is set equal to the value found in the virtual track table at the virtual track number offset. This establishes a starting point for further searching in the virtual sector table. Next, at step 1010, the value of E is set to the next virtual track number (the upper bound of the search to be executed on the virtual sector table).

At 1012, a loop is entered in which a test is performed to determine whether the end of the virtual track has been reached. It is to be noted that if this case occurs on the first iteration through the loop, the result would indicate that the present track contains no defective sectors. If the end of the track has not been reached, at 1014 a second test is performed in which it is determined whether the virtual sector being searched for still exceeds the virtual sector entry being tested in the virtual sector table. If the result of the test is positive, this indicates that the sought after index has not yet been located. In this case step 1016 is executed to increment the present offset into the virtual sector table to the next virtual sector table entry. After this, control is returned to the beginning of the loop at

1012. If the result of the virtual sector number comparison conducted at step 1014 is negative, this indicates that the sought-after virtual sector index has been located in the virtual sector table. The effect on the flow control is the same as if the end of the track is detected at step 1012, and results in entry to step 1018. At 1018, the search is finalized by setting the PBA equal to the LBA added to the value of the present offset in the virtual sector table. Finally, the process is exited at 1020.

An important advantage is achieved through the use of overlap between certain portions of the virtual track number and the virtual sector number. This overlap feature can be seen in the definition of the LBA shown at element 904 in Fig. 9 with reference to the B bits and C bits defining the virtual track number and the virtual sector number, respectively. When such an overlap is present, a portion of the virtual sector number corresponds to the least significant bits of the virtual track number. This overlap is intentionally provided in order to improve the mapping efficiency from the LBA to the PBA. The overlapping bits between the virtual track number and the virtual sector number give the virtual sectors the ability to distinguish between adjacent and nearby virtual tracks without having to refer back to the virtual track table. The amount of overlap between the virtual track number and the virtual sector number establishes the quantity of virtual tracks that may be distinguished from one another using a single virtual sector number. Since the entries in the virtual sector table occur in increasing numerical order, so long as there is at least one skip sector represented in the virtual sector table within the quantity of virtual tracks that can be distinguished using a single virtual sector number, the virtual sector table alone can be used to detect all changes in the virtual track number and indeed can be further used to count the virtual track number associated with the various entries in the virtual sector table. Thus, the search through the virtual sector table may be performed without having to revisit the virtual track table each time a virtual track boundary is crossed.

A second advantage of encoding some of the least significant bits of the virtual track numbers into the virtual sector numbers becomes apparent when computing absolute distances between skips across track boundaries. For two skips which are separated by one or more track crossings, so long as the distance between them is within the resolution of the overlap bits provided in the virtual sector, the virtual sector

value for one skip sector can be subtracted from the other to establish an absolute number of sectors between the skips. Thus, absolute distance computations may be performed across track boundaries also without reference to the virtual track table.

Once the physical block address has been computed, a second conversion takes place in which the PBA is further converted to zone, cylinder, head, and sector location. Fig. 11 illustrates the zone conversion storage components required to initiate the conversion process in a zone bit recording disk drive. The table is generally designated as 1104 and is preferably stored in RAM, but may alternatively be stored on any other convenient media. Zone table 1104 includes three columns, zone PBA column 1110, zone cylinder column 1112, and zone sector column 1113. Zone table 1104 includes an entry for each zone in the disk drive. The first column of each entry, zone PBA 1110, identifies the starting PBA of that zone, and is used to determine which zone a desired PBA is located in. Thus, the top entry in the table would identify the first PBA in zone 1 of the disk drive, etc. The sought-after PBA number is compared with each zone PBA entry until the zone containing the PBA is located. The corresponding entries in zone cylinder column 1112 and zone sector column 1113 are then used to perform the conversion computations which will be described below.

Fig. 12 illustrates in flowchart form the steps which are preformed, once the zone conversion storage components have been located, to convert from PBA to ZCHS. At step 1204, the zone table is searched for the sought-after PBA number to retrieve the zone PBA, zone cylinder, and zone sector entries as described with reference to Fig. 11. At step 1206, zone PBA is subtracted from PBA to determine how many physical blocks (sectors) into the zone the desired PBA resides. A PBA offset into the zone is returned from this operation. At 1208, the PBA offset is divided by the number of sectors per track. The mod, or remainder of this operation is designated as the starting sector number of the data transfer. At step 1210, the quotient of the same division process is taken as the track offset from the start of the zone at which the data transfer is to begin. At 1212, the absolute cylinder number which starts the data transfer is computed by dividing the track offset by the number of heads in the disk drive to produce a cylinder offset which is then added to the starting cylinder of the zone. Finally, at 1214 the remainder, or mod of the same division provided in 1212 is taken as the selected head for the start of the data transfer. At this point the

process exits at 1216, having computed the zone, cylinder, head, and sector at which to begin the data transfer.

One beneficial result of decoupling the LBA to ZCHS conversion process into two stages is a two step mapping process which significantly reduces the number of spares that must be allocated across the disk drive to handle grown defects occurring in the field. The two step LBA to ZCHS process in turn allows the allocation of skip sectors to be broken into a two step process. First, surface analysis testing (SAT) is performed across the entire disk drive to determine the factory defect locations before the disk drive is shipped from the factory. The defect locations are mapped into a virtual sector table by inserting skip sectors in numerical order. Also, a virtual track table is created and updated as required by incrementing each virtual track entry following that track in which the error is detected. Second, after SAT is complete, spare locations are sprinkled evenly across the disk at distances consistent with the probability of a grown defect. It is to be noted that the probability of a grown defect, and thus the density of spares, may vary as a function of position on the disk. The result is that sparing is reduced to minimal levels.

Given the above scheme for sparing, two alternative embodiments for reassignment of sectors are presented. Both reassignment techniques are intended to map a spare sector into the logical address space of the disk drive as well as to map the defective sector into the virtual sector table and, if necessary, the virtual track table, in order to establish the defective sector as a skip sector.

For either technique the re-assignment process begins by identifying the closest available spare sector (to be used as the reassignment sector). This involves accessing the DS table and searching both forward and backward from the entry closest to the defect until a spare entry is found. The locations of the forward and backward entries are subtracted from the defect location, and the lowest absolute value result is used to select the re-assigned sector. Once the re-assignment sector has been identified, one of the two re-assignment techniques is applied. Turning to the first, the original tables are left unmodified, and a separate list of re-assigned sectors is maintained. This separate list contains sectors chosen from the sprinkle list which have been used as re-assigned sectors. The list is in the form of a map which includes a first column representing grown defective ZCHS values

and a second corresponding column representing new ZCHS values. The list itself may be organized either by LBA number or ZCHS number. If it is organized by LBA number, it is checked before translation from LBA to PBA. In this case, the list entries are interpreted to indicate which LBAs are mapped out and what new LBAs they are mapped to. If the list is organized according to ZCHS values, then it is referenced after translation from LBA to PBA and from PBA to ZCHS, and its entries are interpreted to indicate which ZCHS values are mapped out and what new ZCHSs they are mapped to. In any event, when the defective reassigned sector is encountered during a data transfer operation, a second seek is executed to the assigned spare location and the data is read or written there. However, as discussed above, since the spares are sprinkled throughout the disk drive and the drive has selected for use the sprinkled spare nearest the defective re-assigned sector, the length of the seek is minimized and performance is thereby improved.

The second re-assignment technique effectively replaces the grown defective spare with the selected re-assign spare in the virtual track and virtual sector tables. The virtual sector table is modified by inserting a new skip sector at the grown defect and then shifting all values between the grown defect and the reassigned sector. If the closest spare is across a track boundary from the defect, the virtual track table is also updated to indicate the pointer changes. Thus, by simply shifting LBAs in the virtual sector table between the defect and the reassigned sector, the spare takes the logical place of the defect and no performance degradation is experienced thereafter.

It is to be noted that the skip sector conversion schema has the additional advantageous property of operating in the presence of skewing without further modification. Where skewing is in effect, prior computations which compute sector number are considered to be with reference to unskewed sectors. Conversion to skewed sector then occurs outside the conversion process, as was described previously.

One modification to the above-presented embodiment, which may be useful in cases where simplified mapping is desired, is to use a single table conversion process which maps directly from LBA to ZCHS. The table entries still require three or four bytes as in the prior art, however the new table construction is based on LBAs adjusted by table location. Thus, the table contains LBAs which directly follow a given skip. The advantage of this approach is that the table look-up process requires

only a simple single level search, either sequential or binary. The RAM requirement is fairly small so long as the number of skips in the system is kept small. However, for large modern disk drives the storage requirement expands rapidly as the number of defects increases, thereby overwhelming the utility of this approach.

It is also to be noted that the skip sector mapping technique disclosed in this section is useful beyond the realm of no-ID disk drives. In fact, this technique is considered applicable to any disk drive in which benefit may be obtained from knowing conclusively at the beginning of every seek operation exactly what the final target zone, cylinder, head, and sector are. Thus, in all disk drives, including those using ID fields and those using no ID information, the disclosed LBA to ZCHS mapping avoids the requirement to slip heads in order to handle spares and defects. Moreover, the system leads to easy implementation of just-in-time seek profiles, where knowledge of the exact seek destination is used to plan the seek trajectory so that the head settles on track just before the desired sector arrives at the head. Finally, the system allows for implementation of seek reordering including latency minimization, since the exact seek destination is known before seeking.

Most importantly, it should be noted that while the above description has focused on partitioning the LBA into two portions, the concept may easily be extended to three or more partitions. For example, a virtual cylinder partition may be added which contains neighbouring virtual tracks. This becomes advantageous as the number of virtual tracks becomes large, since it reduces the amount of storage required for the virtual track table. Thus, in general the mapping schema presented above should be taken to encompass n-level hierarchical storage of skip sectors.

VII. Conclusion

A further advantage of the preferred embodiment hardware design lies in its built-in power management features. Since the servo-modulo count block actually computes the modulos, the system requires only a single servo sector time for initialization. The field length counters in sector pulse logic 421 must operate only during read and write modes, and may be powered down otherwise. Thus, in a low power implementation of the present invention, much of servo electronics 212 and formatter

electronics 215 is powered down between read and write operations. Typically, timing logic 420 is active during this time, and generates signals to activate AM detector 412 and other electronics at each servo sector. Since the servo sector counter is active, a read or write operation may commence following the next servo sector. If even greater power savings is desired, an additional stage may be added in which the servo sector counter and associated electronics are powered down. In this mode, the system must wait for an index to initialize itself if the full servo sector number is not encoded within the TID. If it is encoded, the system need only wait for the next servo sector.

While the invention has been described with reference to data sectors which are shorter than the data regions separating servo sectors, it is to be noted that the invention is equally applicable to the opposite case -- that is, a sector size or servo spacing selected such that the data sector length exceeds the data region length. This may occur in an implementation which uses closely-spaced servo sectors for performance or other reasons. It may also occur where the data sector size is chosen to be relatively large for use in applications dealing with long blocks of information, such as multimedia presentations and scientific data. The result of the data sector size exceeding the data region size is that a single data sector may span two or more servo sectors. This is handled seamlessly according to the invention as disclosed above by simply programming the register and RAM values accordingly.

While the preferred embodiment hardware design uses a servo modulo counter to reduce memory requirements, it is apparent that the segment information may be expanded to store the information for an entire track. In this case, there is no need for the servo modulo counter, since RAM 504 will be addressed directly by servo sector number. Such a design uses more RAM than the preferred embodiment, but will have simplified hardware. The choice of designs depends on factors such as circuit cost for the two implementations. Further, the data in RAM 504 may be organized to allow addressing by values other than the servo sector number. Specifically, the data sector number may be used to address the table.

Further, it is to be noted that in some implementations it may be judged beneficial for error handling purposes to include some portion of the LBA in each data sector. This allows for LBA verification during

read operations, but is of no use during write operations since the sought-after sector is being overwritten rather than read. Accordingly, LBA encoding in data sectors is not to be considered as analogous to ID information encoding in data sectors. Moreover, it is intended that the removal or exclusion of ID information as discussed above encompasses an implementation which removes the ID from data sectors information, but includes the LBA.

It is also apparent that the locations of the various functions shown in the electronics may be altered, and that software may be substituted for some of the hardware shown and described. It is further apparent that while the preferred embodiment has been described in the context of a zone recording format, the invention may be applied to other complex formats where the number of data sectors on a track is not equal to the number of servo sectors on a track, so long as the format details are known to the disk drive. Moreover, the terms register and RAM should be viewed as interchangeable, and the specific locations of the registers and RAM as unimportant. For example, registers 506 and RAM 504 may be located within RAM 217 instead of format table 422. Finally, it should be noted that while the values and tables described above are stored in RAM and registers during operation of the disk drive, they must be stored in non-volatile storage when the drive is not operating. Any non-volatile storage may be used, but it is preferred to store the information on the disk drive itself. The use of modifiable storage (i.e., the disk drive) allows for the disk drive to alter its own format without need for external intervention.

While the invention has been particularly described and illustrated with reference to a preferred embodiment, it will be understood by those skilled in the art that changes in the description and illustrations may be made with respect to form and detail without departing from the scope of the invention. Accordingly, the present invention is to be considered as encompassing all modifications and variations coming within the scope defined by the following claims.

* C L M S *

THIS SHEET HAS BEEN REPLACED

CLAIMS

1. A data recording disk for use with a fixed-block architecture disk drive having a head capable of reading positioning information, the disk being divided into a number of radially spaced tracks (218), at least one of the tracks being divided into a number of angular sectors including only data sectors (254) for recording user data and servo sectors (220) having pre-recorded head positioning information for identifying track and servo sector locations, wherein the number of servo sectors on the track is not equal to the number of data sectors on the track, and wherein information establishing the circumferential locations and identities of the data sectors is encoded within the servo sectors.

2. A data recording disk as claimed in claim 1, wherein the established identity of a data sector includes a data sector number.

3. A data recording disk as claimed in claim 1 or claim 2, wherein the established location of a data sector includes a distance from a preceding servo sector.

4. A data recording disk as claimed in any preceding claim, wherein the information encoded within the servo sectors includes servo sector number information.

5. A data recording disk as claimed in claim 4, wherein the servo sector number information contains a subset of the full servo sector number.

6. A data recording disk as claimed in any preceding claim, wherein the information encoded within the servo sectors includes an address mark.

7. A data recording disk as claimed in claim 6, wherein additional information establishing the locations and identities of the data sectors is encoded in an index mark identifying a fixed location on the track.

8. A data recording disk as claimed in any preceding claim, wherein at least some of the data sectors are split by at least some of the servo sectors into primary and secondary data sections, the primary data sections preceding the dividing servo sectors and having their circumferential positions determined from the circumferentially preceding

This Sheet Has Been Replaced

THIS SHEET HAS BEEN REPLACED

servo sectors, the secondary data sections following the dividing servo sectors and having their circumferential positions determined from the dividing servo sectors.

9. A data recording disk as claimed in any preceding claim, wherein at least some of the data sectors are immediately preceded in the circumferential direction by other data sectors and have their circumferential positions determined from circumferentially preceding servo sectors.

10. A fixed-block architecture embedded servo disk drive comprising:

a data recording disk (204) having a number of radially spaced tracks, at least one of the tracks being divided into a number of angular sectors including only servo sectors and data sectors, wherein the number of servo sectors on the track is not equal to the number of data sectors on the track;

recording head means (208) which reads information in the servo sectors and which writes and reads user data in the data sectors;

means, responsive to the information read from the servo sectors, for determining the locations of the data sectors based on their distance from the servo sectors; and

means, responsive to the information read from the servo sectors, for determining the identities of the data sectors.

11. A disk drive as claimed in claim 10, wherein the data sectors are identified by data sector number.

12. A disk drive as claimed in claim 10 or claim 11, the track further including an index mark identifying a fixed location on the track, and wherein the means for locating and the means for identifying the data sectors are responsive to the index mark.

13. A disk drive as claimed in any of claims 10 to 12, wherein the information read from the servo sectors includes servo sector number information.

14. A disk drive as claimed in claim 13, wherein the servo

This Sheet Has Been Replaced

* C L M S *

THIS SHEET HAS BEEN REPLACED

sector number information contains a subset of the full servo sector number.

15. A disk drive as claimed in any of claims 10 to 14, wherein the determining means further includes:

electronic storage containing sector layout information, and servo electronics for selecting portions of the sector layout information from the electronic storage based on the information read from the servo sectors, computing the distances from the servo sectors to the data sectors as a function of the selected sector layout information; and

computing the identities of the data sectors as a function of the selected sector layout information.

16. A disk drive as claimed in claim 15, wherein the sector layout information includes segment layouts.

17. A disk drive as claimed in claim 16, wherein the segment layouts include data section lengths.

18. A disk drive as claimed in claim 17, wherein the data section lengths are represented as clock counts.

19. A disk drive as claimed in any of claims 16 to 18, wherein the segment layouts include data sector identifiers.

20. A disk drive as claimed in any of claims 16 to 19, wherein the electronic storage containing the segment layouts is random access memory or read only memory.

21. A disk drive as claimed in any of claims 15 to 20, wherein the sector layout information further includes zone layouts which enable the servo electronics to locate and identify data sectors which are not positioned immediately following a servo sector.

22. A disk drive as claimed in claim 21, wherein the zone layouts include synchronization field lengths, pad field lengths, data sector lengths, number of servo sectors per track, and number of data sectors per track.

This Sheet Has Been Replaced

* C L M S *

THIS SHEET HAS BEEN REPLACED

23. A disk drive as claimed in claim 22, wherein the synchronization field lengths, pad field lengths, and data sector lengths are represented as clock counts.

24. A disk drive as claimed in claims 21 to 23, wherein the electronic storage containing the zone layouts is dedicated register storage.

25. A disk drive as claimed in any of claims 15 to 24, wherein the determining means computes a data sector number using a servo sector number and the sector layout information.

26. A disk drive as claimed in any of claims 10 to 25, wherein the determined data sector identities are adjusted to compensate for cylinder and head skewing.

This Sheet Has Been Replaced