

Drive Imaging as Part of Data Recovery

Andrei Shirobokov
Chief Technology Officer
ACE Data Recovery Engineering Inc.
<http://www.acedre.com>
ashirobokov@acedre.com
October 11th, 2006

Introduction

Imaging is the process of copying the data on a damaged hard disk drive (HDD) to an undamaged one. It is an important part of the data recovery process.

A typical data recovery process can be thought of as comprising of three stages.

1) Drive Restoration: The problem with the HDD must be diagnosed and the drive has to be repaired (if necessary). There are three main types of damage:

- *Physical/Mechanical damage:* for instance, failed heads or other physical problems. This is often fixed by having the damaged hardware replaced by a donor part.
- *Failed PCB board:* A failed PCB board must be replaced with a donor, and the contents of the failed PCB ROM usually copied to the donor using a product such as the ACE Laboratory PC3000.
- *Firmware failure:* firmware failures can be handled by a product such as the ACE Laboratory PC3000.

2) Disk Imaging: The repaired drive must have its contents read and copied to another drive. This prevents further data loss caused by working with an unstable drive during the final stage.

3) Data Retrieval: The original files that have been copied onto the image drive must be retrieved. This can involve File System Recovery (the recreation of a corrupted file system structure such as a corrupted directory structure or boot sector, due to data loss), File Verification and File Repair (if necessary).

Often drives will be presented for recovery with relatively minor physical degradation that is due to wear from normal use. The wear will have been severe enough for the drive to stop working in its native system. However, imaging software can work with slightly degraded drives, so part replacement is often not warranted. In these cases Stage I of the recovery process is skipped and Stage II (imaging) is where the process starts.

The first and last steps are well catered to by the data recovery industry. The necessary software, hardware, knowledge, and skilled labour are possessed by many data recovery companies. However, the process of image creation has been relatively neglected, and marks it as a weak link in data recovery. That often leads to the whole process failing, at least partially, and the loss of user data.

The Basic Imaging Problem

Read Instability

Disk drive imaging is not a trivial task. This stems from the fact that degraded disks and even disk drives that have been repaired after developing physical defects are often subject to a high read error rate, or read instability. By read instability we mean that multiple readings of the same data will tend to yield different results each time. This is different from the case where data was simply written in error in the first place, but is read consistently each time or the situation where, for instance, a physical scratch may simply make an entire part of the disk unreadable no matter how many read attempts are made.

There are several reasons for repaired drives having a high read instability. For instance donor parts, while nominally identical to the original failed part, differ slightly due to tolerances in the manufacturing process. The majority of modern disk drives are individually tested at the factory and have optimized adaptive parameters burned into ROM. If one or more part is replaced at a later date, the adaptive parameters are no longer optimized to the new configuration. Due to the fine-tuning of modern drives so as to obtain maximum performance and capacity, even small deviations from optimum can introduce a high rate of read errors.

In drives that have not been repaired, physical degradation due to normal use will also mean that adaptive parameters will no longer be optimum, thus leading to read instability. For example, wear in the actuator bearings will cause slight deviations in the distance the arm moves across the platter. So, the current that was once required to move the actuator to a certain track is no longer sufficient to move the arm correctly; thus, the adaptive information that was once correct is no longer optimal. Another reason for read errors is that dust infiltration into the disk drive over the course of its life may make the read process electronically noisy.

The important point to note is that HDDs that have failed will tend to have read instability no matter what the cause for failure. This will be despite any physical repair performed on them; in fact it will probably be because of it, as we saw in the preceding paragraph. In the case where no physical repair is made, a read instability noticed when the drive was in service is likely to be the reason it is being presented for data recovery. (We shall see below that HDD firmware and computer operating systems have a low tolerance for read instability; relatively low read instability can cause the HDD to hang persistently thus rendering the system unusable).

For the purposes of imaging a drive, read instability is, in principle, not a major obstacle since each sector can be read several times and statistical methods applied to obtain the most likely correct value for each byte. However, because of the way that the computer BIOS, OS, and HDD firmware are written it is impossible to obtain data during HDD read instability without specialized hardware and software.

BIOS/OS Issues

The primary reason for this difficulty is that the BIOS/OS reads each disk sector presuming the sector is uncorrupted, that is the data is not read unstable. A sector on a hard disk typically comprises a header, 512 bytes of data, and Error Correction Code (ECC) bytes (Figure 1). There are also other bytes present to facilitate read/write synchronization. For simplicity, the ECC bytes can be thought of as a checksum, which is intended for purposes of integrity checking.

Most current disk drives are ATA (Advanced Technology Attachment) compliant devices. BIOS/OS will read the drive by the standard ATA read sector command. In this case if a sector has a bad ECC checksum no data will be returned, just an error. This is despite the fact that most of the bytes in the sector are likely correct and all are actually readable, even if some of the data is corrupt. This is because BIOS/OS is unable to access data by other ATA read commands that will read data irrespective of ECC status. The use of ATA commands that ignore ECC can be thought of as reading the data byte-by-byte rather than sector-by-sector.

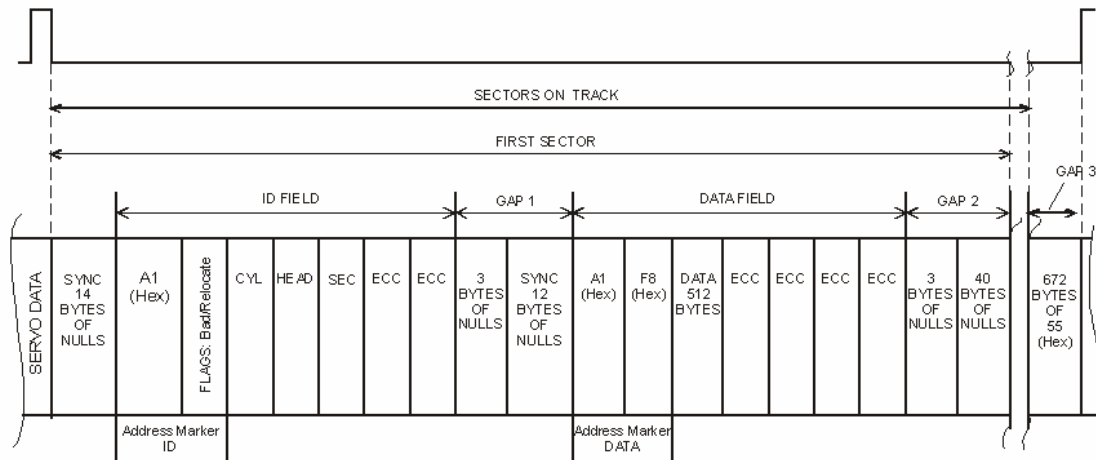


Figure 1: Example of Sector Format.

BIOS/OS reads drives in this manner because the mandate of computer architecture is to create a reliable, stable machine, which runs at the highest possible speed. The BIOS cannot allow errors to be propagated from the hard drive to RAM. These errors might be part of an executable that would then crash the machine. Hence there is a low tolerance of read instability. Thus, system software assumes the disk drive is working correctly and does not delve into drive errors except to flag whole sectors as “bad” and to avoid reading them.

Also, in many cases BIOS is unable to handle HDD errors, a situation that may lead to system crash.

Using BIOS to read the drive is also time consuming and involves intense internal processing by the HDD, leading to further wear. This wear increases the likelihood of further disk degradation during the imaging process. We shall see later that drive imaging is by its nature very demanding on the HDD, involving multiple reads on the entire disk and modifying its firmware data. Since progressive disk degradation is common (i.e. read instability increases with use), reducing the demands on the HDD is desirable.

Thus, even a relatively low level of read instability, which is common in drives in recovery, can lead to the computer not being able to read the drive at all, despite the fact that the drive is still readable by other means.

Solving Imaging Problems

Basic Characteristics Needed for Imaging Software

While the system software works well with correctly working disk drives, specialized software that bypasses the OS and BIOS is needed in the presence of read instability. This specialized software must be able to use ATA read commands that ignore ECC status. Also, the software should have the capability of reading the drive Error Register (Figure 2), which the regular BIOS/OS doesn't provide access to. This allows it to employ different algorithms for different errors. For instance, in the case of the UNC error ("Uncorrectable data: ECC error in data field, which could not be corrected"), a read command ignoring ECC status can be issued. AMNF error can in many cases be dealt with in a similar way.

7	6	5	4	3	2	1	0
BBK	UNC	0	IDNF	0	ABRT	TONF	AMNF

- Bit 0 - Data Address Mark Not Found: If during "Read Sector" command a data address mark has not been found after finding the correct ID field for the requested sector (usually a media error or read instability).
- Bit 1 - Track 0 Not Found: Track 0 was not found during drive recalibration.
- Bit 2 - Aborted Command: The requested command has been aborted due to a device status error.
- Bit 3 - Not Used (0).
- Bit 4 - ID Not Found: Required cylinder, head and sector could not be found, or ECC error in ID filed.
- Bit 5 - Not Used (0).
- Bit 6 - Uncorrectable Data: ECC error in data field that could not be corrected.
- Bit 7 - Bad Mark Block: Bad sector mark was found in the ID field of the sector or Interface CRC Error.

Figure 2: ATA Error Register.

Only in the case where the sector header has been corrupted (IDNF error) it is unlikely that the sector can be read, since the drive in this case will be unable to find it. Experience shows, however, that over 90% of all problems with sectors are due to errors in the data not in the header. This is simply because the data constitutes the largest portion of the sector. Also, the data area is constantly being rewritten, which increases read instability. (Header contents usually stay constant throughout the life of the drive). This means that over 90% of sectors that are unreadable by ordinary means are in fact still readable. Moreover, there are typically only a small number of errored bytes per problem sector, and these can often be corrected by a combination of multiple reads and statistical methods.

If a sector is read ten times and a particular byte returns the same value eight times out of ten, then that value is statistically likely to be the correct one. More sophisticated statistical techniques are also likely to be useful.

In fact ACE Data Recovery Engineering Inc has had experience with damaged drives in which every sector had a problem. This would normally leave the data totally unrecoverable, but nonetheless these HDDs were read and corrected in their entirety.

Unfortunately, most imaging tools available currently on the market uses BIOS/OS to access the drive and are therefore extremely limited in their imaging capabilities. These products will attempt to read a sector several times in the hope that one time it will read without an ECC error. But this approach will not be very successful. If for instance, there is even a one percent chance

of any byte being read incorrectly, the chances of reading all 512 bytes correctly on one particular read is low (about 0.58%). On average 170 read attempts would be necessary to yield one successful sector read. As read instability increases the chances of a successful read rapidly become very low. For instance if there were a ten percent chance of reading any byte wrong, 2.7×10^{23} read attempts on average would be needed for one successful one. If one read attempt took 1 ms, this number of reads would take 8000 billion years, or many times the age of the universe! No practical number of read attempts will give a realistic chance of success.

If, however, we can read ignoring ECC, a ten percent probability of a byte being read in error is no problem. In ten reads very few bytes will return less than 7 or 8 consistent values, making it virtually certain that this value is the correct one. Thus imaging software that bypasses BIOS has a much greater ability to deal with read instability.

Drive Pre-configuration Prior to Imaging

While it might appear that an accurate image of the data might be obtainable with the methods outlined in the previous section, other problems still remain. One of these is the fact that when HDD firmware identifies a bad sector it may remap it to a reserved area on the disk which is hidden from the user (Figure 3). This remapping is recorded in the HDD defects table (G-List). Since the bad sector could not be read, the data residing in the substitute sector in the reserved area will not be the original data. It might be null data or some other data in accordance with the vendor-specific firmware policy, or even previously remapped data in the case where the G-List was modified due to corruption. Moreover, this is done transparently to the BIOS/OS. When a request is made to the HDD to retrieve data from a sector that the HDD has identified as bad, automatic redirection to the alternate sector in the reserved area is made, without notifying the system before the error is returned. This is despite the fact that the “bad” sector is still probably readable and only contains a small number of errored bytes.

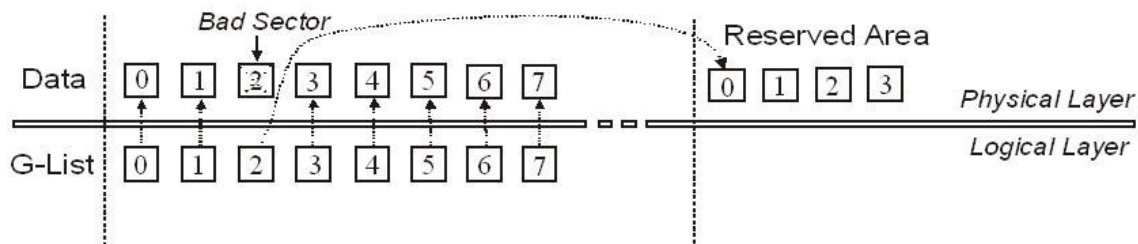


Figure 3: *G-List Remapping.*

This process performed by HDD firmware is known as sector auto-relocation. It can be turned off (and should be) before the imaging process begins. Auto-relocation on a drive with read instability not only obscures instances when non-original data is being read. It is also time consuming and increases HDD wear, possibly leading to increased read instability.

Effective imaging software should be able to turn off auto-relocation so that it can identify problem sectors for itself and take appropriate action. In this way it is certain that the original data is being read.

Unfortunately the ATA specification does not have a command to turn off auto-relocation. Therefore imaging software should utilize vendor-specific ATA commands to do this. For instance, the PC-3000 product is able to do this.

A similar problem exists regarding SMART attributes. The firmware constantly recalculates SMART attributes and this constitutes a large amount of overhead that increases imaging time and the possibility of further HDD degradation. Therefore imaging software should be able to disable SMART attribute processing.

Other drive pre-configuration issues exist, but auto-relocation and SMART attributes are among the most important that imaging software should address.

UDMA Mode and the Impact on the Imaging Process

Modern PCs are equipped with HDDs and ATA controllers that are capable of the Ultra Direct Memory Access (UDMA) mode of data transfer. Direct Memory Access (DMA) simply means the CPU is freed from the task of transfers to and from memory. UDMA can be thought of as an advanced DMA mode, and is defined as data transfer occurring both on the rising and falling edge of the clock, thus doubling the transfer speed compared to ordinary DMA. Both DMA and UDMA are in contrast to the earlier Programmed Input Output (PIO) system in which the CPU must perform the data transfer itself. Faster UDMA modes also require an 80-pin connector, instead of the 40-pin connector required for slower UDMA and DMA.

The advantages are obvious. Not only does UDMA speed up the data transfer, but the CPU is free to perform other tasks.

While modern BIOS is capable of utilizing UDMA, imaging software, which bypasses BIOS, should also be able to use this data transfer mode. Also, if the source and destination HDDs are on separate IDE channels, read and write transfers can occur simultaneously, doubling the speed of the process again! Moreover, with the CPU free, processing of the imaged data can occur on the fly.

Requirements of Imaging Algorithms

Imaging of a failed HDD involves the consideration of conflicting factors. Consider the following:

- A high number of read operations on a failed drive increases the chances of recovering all the data, and decreases the number of probable errors in that data.
- Intensive read operations increase the rate of any disk degradation that may be occurring and increase the chance of catastrophic drive failure during the imaging process, thus defeating its purpose. Moreover, the time to image a drive can be long (for example, one-two weeks) and depends on how intensive the read operations are. Customers with time-sensitive needs may prefer to reconstruct data themselves rather than wait for recovered data.

Clearly these two points suggest the idea of an imaging algorithm that maximizes the probable data recovered for a given total read activity, taking into account the rate of disk degradation and the level of probability of catastrophic drive failure. No universal algorithm for this exists however. A good imaging procedure depends on such things as the nature of the HDD problem, and the characteristics of the vendor-specific HDD firmware. Moreover, a client is often interested only in a small number of files on an HDD and is willing to sacrifice the others to

maximize the possibility of recovering those few. To meet these concerns, the judgement of the imaging tool user will come into play.

Drive imaging can consist of multiple read passes. A pass can be defined as one attempt to read the entire drive, although problem sectors might be read several times on one pass or not at all, depending on the configuration. The above concerns suggest that different algorithms should be used on each pass, or at least that different parameter values should be used.

The first pass could be configured to read only error-free sectors. There is a fair possibility that the important files can be recovered faster in this way in just one pass. Moreover this pass will not be read-intensive since only good sectors are read and the more intensive multiple reads needed to read problem sectors are avoided. This reduces the chances of degrading the disk further (including the chances of catastrophic drive failure) during the pass while having a good chance at recovering much of the data.

Second and subsequent passes can then incrementally intensify the read processes, with the knowledge that the easily-readable data have already been imaged and are safe. For instance, the second pass may attempt multiple reads of sectors with the UNC and/or AMNF error (Figure 2). Sectors with the IDNF error are a less promising case, since the header could not be read and hence the sector could not be found. However, even in this case multiple attempts at reading the header might result in a success, leading to the data being read. Success of data recovery of sectors with different errors is dependent on the drive vendor. For example, drives from some vendors have a good recovery rate with sectors with the IDNF error, while others virtually nil. Prior operator experience will come into play here, and the software should be configurable to allow different read commands and a varying number of reread attempts after encountering a specific error.

HDD firmware often has vendor-specific error-handling routines of its own which cannot be accessed directly by the system. While the imaging operator may want to minimize drive activity to speed up imaging and to prevent further degradation, firmware will increase that activity and slow down the process when faced with read instability. For this reason it is mandatory for the imaging software to implement a *sector read timeout*, which is a user-specified time before a reset command is sent to the drive to stop processing the current sector. For instance the operator might have noticed that good sectors are read in 10 ms. If this is a first pass, and the operator's policy is to skip problem sectors at this point, the read timeout value might be 20 ms. If 20 ms have elapsed and the data has not yet been read, it is clear that the sector is corrupted in one way or another and that the drive firmware has invoked its own error-handling routines. In other words, a sector read timeout can be used to identify problem sectors. If the read timeout is reached, the imaging software at this time notes the sector and sends a reset command. After the HDD cancels reading the current sector, the read process continues at the next sector.

By noting the sectors for which a timeout occurs, the software can build up a map of problem sectors. This information can be used by the imaging algorithm for use during subsequent read passes.

In all cases the following should be among the list of configurable parameters:

- sectors to be read during this pass

- type of read command to apply to a sector
- number of read attempts
- number of sectors read per block
- sector read timeout value
- drive ready timeout value
- error-handling algorithm for problem sectors

Many other parameters should also be configurable but this list gives the most critical ones. This will be discussed again later in more detail.

Imaging Hardware

As well as the software described above, it is also necessary to have specialized hardware to perform imaging in the presence of read instability. Firmware is often unstable in the presence of read instability, which may result in the HDD hanging. To resolve this issue the imaging system must have the ability to control the IDE reset line if the drive becomes unresponsive to software commands. Since the ATA specification does not state how to control the IDE reset line, this has to be implemented with a specialized hardware. There are also cases when a hanging drive will not respond even to a hardware reset, and the hardware should also have the ability to re-power the drive to facilitate a reset.

If the BIOS/OS is unable to deal with an unresponsive hard drive it will crash. This requires the user to perform a manual reboot of the system each time in order to continue the imaging process. This is another good reason for the imaging software to bypass BIOS.

Both of these reset methods must be implemented by hardware but should be under software control. They might be activated by a *drive ready timeout*. Under normal circumstances the read timeout will send a software reset command to the drive as necessary. If this fails and the drive ready timeout value is reached, the software directs the hardware either to send a hardware reset, or to re-power the drive. A software reset is the least taxing on the disk drive and the re-power method the most taxing. Therefore this sequence of software, hardware, and re-power resets is used to minimize drive activity during reset thus minimizing additional wear.

Moreover, because this can be under software control via the user-configurable timeouts, the process is faster and the need for constant user supervision is eliminated.

The application of the drive ready timeout can also help reduce the chances of drive self-destruction due to a “heads clicking” situation, which is a major danger in read-unstable drives. “Heads clicking” is essentially a firmware exception in which repeated improper head motion occurs, usually of large amplitude that leads to rapid drive self-destruction. “Heads clicking” renders the drive unresponsive and thus the drive ready timeout will be reached and the software will shut the drive down, hopefully before damage has occurred. A useful addition to an imaging tool is the ability to detect “heads clicking” directly, so it can power down the HDD immediately without waiting for a timeout, thus virtually eliminating the chances of HDD loss due to this problem.

The Disk Imager

From the above discussion it is clear that many factors need to be taken into consideration if an imaging tool is to approach the maximum possible imaging effectiveness using current technology. The Disk Imager (“DI”) successfully incorporates all the functionality to do this. As well, it incorporates additional features, which further enhance its performance. This will be discussed in this section.

Product Description

Figure 4 shows a schematic of the product system configuration. The host system can be any modern PC, or even just a PC motherboard with a keyboard and a monitor. The Disk Imager hardware module sits between the source drive and the host computer. A single IDE cable connects the host computer, the DI hardware module, and the source drive. The system is booted with the DI software from the flash memory of the hardware module. The interconnection of the DI hardware module allows it to send hardware resets and re-power the source drive without resetting the entire system.

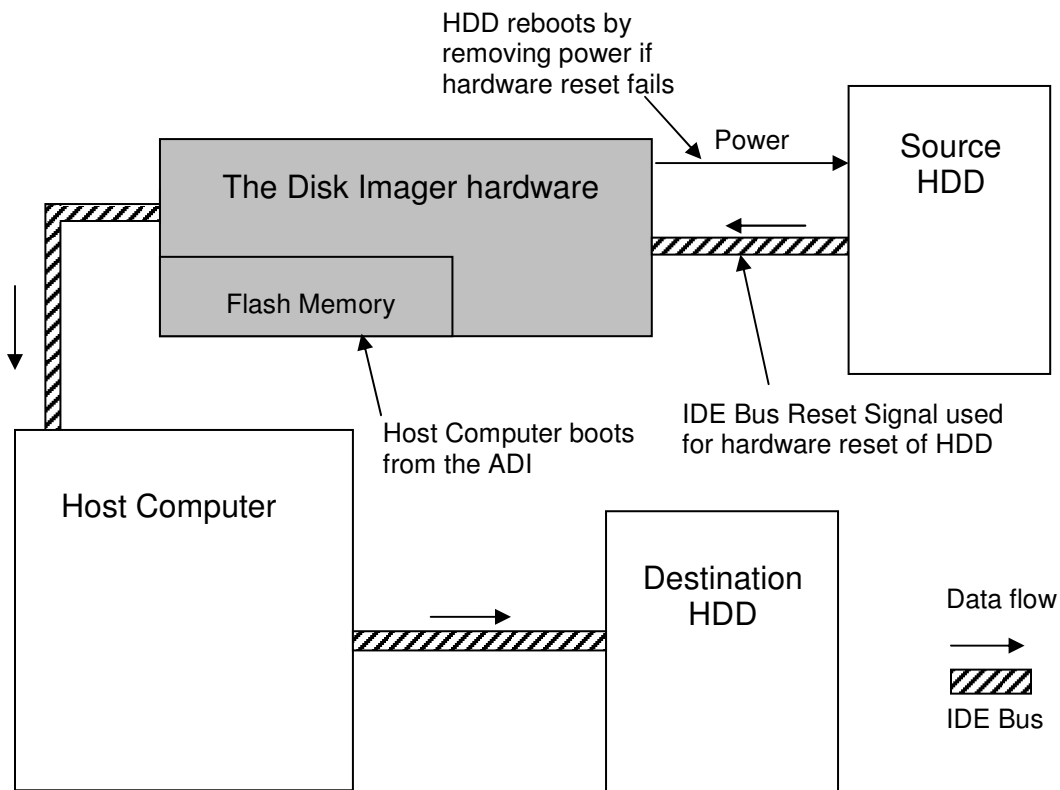


Figure 4: System architecture. A single IDE cable connects the PC, the Disk Imager hardware, and the source HDD.

The imaging software is written in assembly language bypassing BIOS/OS. This combination was chosen for its low overhead, and high performance. It also reduces the load on the HDD. The software addresses all issues discussed earlier:

- implementation of drive pre-configuration
- implementation of “sector read” and “drive ready” timeouts
- the bypassing BIOS/OS

- the utilization of UDMA mode with simultaneous (overlapped) read/write operations (in this mode a speed of imaging HDD can be close to the HDD maximum read access speed, i.e. the speed of its surface verification)
- the ability to run efficiently while unsupervised, with minimal risk of drive self-destruction
- a flexible algorithm with many configurable parameters to customize the process based on prior operator experience, and the needs of each particular job

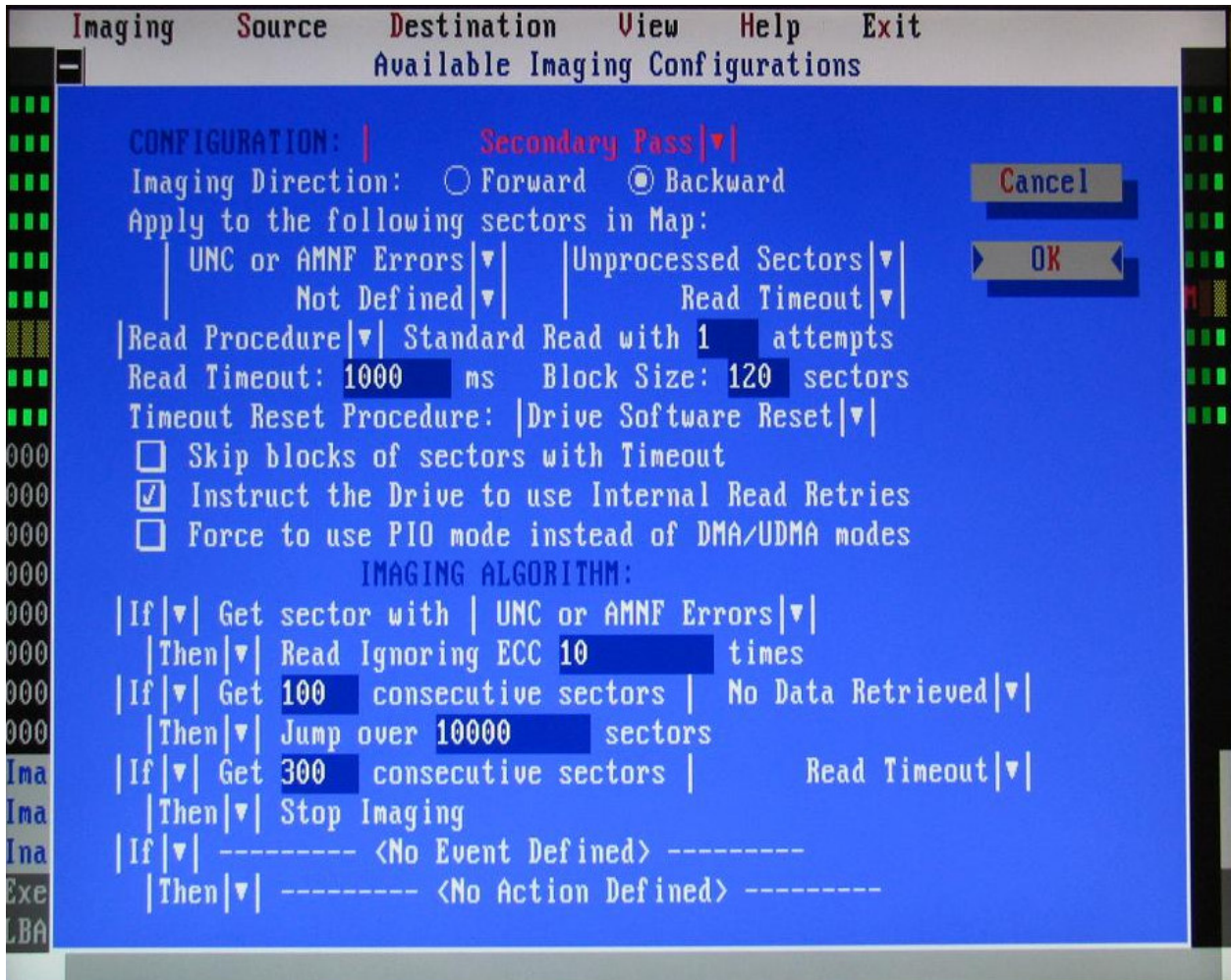


Figure 5: One of the Disk Imager dialog boxes showing configurable parameters of the imaging algorithm. The user can select various event-action scenarios.

Destination HDD used to store all information

As well as incorporating all the functionality discussed above, the product is set up to write all important information about the imaging process to the destination drive:

- configuration parameters
- map of sectors already read or attempted, along with error type, if any
- the state of the imaging process
- information about the source HDD

This method of information management has several advantages. The image map ensures that on subsequent passes, the imager will only focus its processes on sectors that were not successfully read during initial passes. The information and the map are updated dynamically so that if the imaging process is interrupted for any reason, the process can be restarted where it left off without any loss of information. Moreover, with all information in one place, the operator does not need to keep other records of a particular job including a map of all problem sectors.

This might be useful if another, higher-priority job arrives. Work on the current job can be interrupted without any extra operator work, and recommenced at any time in the future with no loss of imaged data or the associated information. Another use is after a system power failure. Once power is restored, imaging can simply recommence where it left off.

Configuration information and the image map written to the destination hard drive creates an overhead of about 2% of the data, so the destination drive should be larger than the source drive.

UDMA enables CPU to process data on the fly

Since the Disk Imager utilizes UDMA with overlapped read/write operations, a typical data transfer rate for modern drives usually reaches up to 60 Mbytes/s (about 3.6 Gbytes/min). This greatly speeds up the imaging process. Moreover, it leaves the CPU free to perform other tasks during imaging! One task performed by the DI is to display information about imaged data on the screen on the fly. Figure 6 shows a typical information page displayed during imaging.

The status of recently read sectors is shown in the top portion of the screen. This tells the operator the success rate of the read process with the current set of parameters. The operator can change the parameters as necessary based on this information.

The sector contents are shown in the middle of the screen for each sector on the fly. This lets the operator know whether real data is being imaged, or whether the data is just, for instance, an array of nulls. With other tools, the contents of the imaged data are sometimes not known until imaging is complete. The time and resources invested in imaging may be for nothing if this is the case. Recovery firms are particularly sensitive to this problem since most client arrangements are on a “no data, no fee” basis. So, data evaluation on the fly is a real advantage. With the Disk Imager, if the current sectors contain no real data, the operator can skip ahead and try another part of the disk.

Finally, while the imaging process is under way the software also identifies file system structure elements (for FAT and NTFS), and the files of known types. This information is shown at the bottom status bar (Figure 6).

Note that utilizing the CPU to provide information on the fly in this way does not slow down the imaging process since data transfer is handled by a UDMA controller and not the CPU. There is no speed trade-off for having all this information available on the fly.

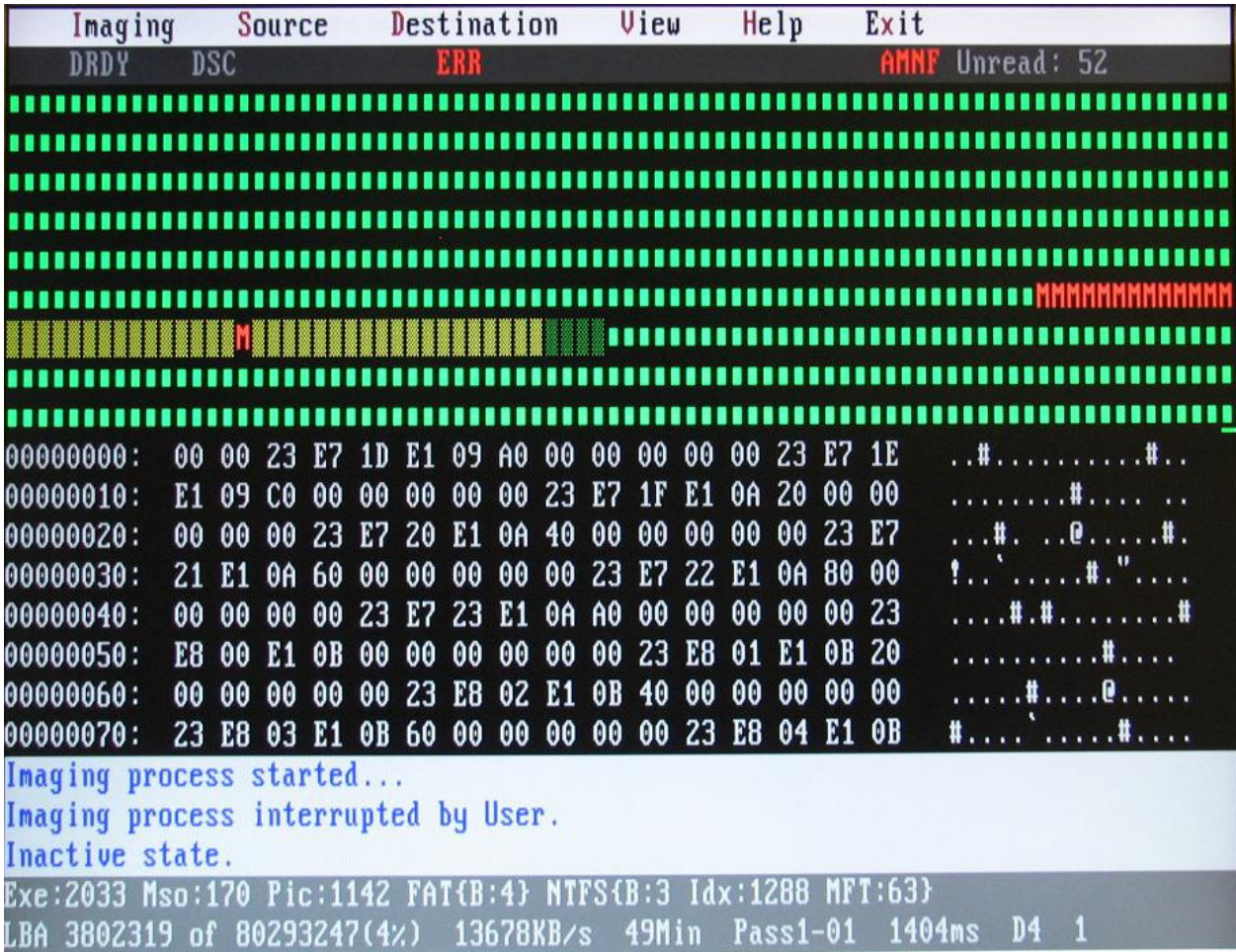


Figure 6: Typical main screen during the imaging process. The top bar shows the status of recently accessed sectors: read successfully, skipped, errors, if any. Contents of sectors are shown on the fly. The bottom status bar shows the number of files of various types and OS structure elements imaged so far: Executable files, MS Office files, Pictures, FAT/NTFS elements.

Features for Future Releases of the Disk Imager

The above features are available in the first release of the Disk Imager. In addition, ACE Data Recovery Engineering Inc. plans to add more functionality in the future releases of the DI.

Recognition of the “Heads clicking” situation

The incorporation of the drive ready timeout as discussed earlier is a good way to minimize the chances of drive self-destruction due to the “heads clicking”. However, the incorporation of a sensor to detect “heads clicking” directly would totally eliminate any chance of drive loss due to this problem. ACEDRE intends to incorporate such functionality into the DI in the near future and to add this detected event into the error-handling algorithm, so that the DI might either re-power the drive or even stop imaging process in this case. This way the imaging process can continue overnight with no operator present, with no danger of HDD destruction due to this problem.

Optimum Temperature for Drive Operation

ACEDRE is currently conducting research onto the optimum ambient temperature for drive operation. It turns out that this definitely affects read instability. The optimum environment might be either higher or lower than a normal drive operation temperature. Future releases of the Disk Imager will have an optional environmental cabinet to house the source HDD. The temperature is varied under software control and the lowest read instability found using a proprietary algorithm.

Conclusion

A successful data recovery depends on the effectiveness of the data imaging tool. As you can see, there are many factors that must be taken into account if an imaging tool is to provide the maximum performance possible with current technology. However, very few of these factors have been incorporated into the imaging tools that are currently on the market. Given the prevailing “no data, no fee” arrangement, failures in data imaging, which might not be known until the process is complete, are an economic problem for recovery firms. Moreover, the length of time taken to recover data is often critical to a client, and a firm unable to image quickly is at a disadvantage.

The Disk Imager incorporates all useful methods possible with current technology. Compared with competing products, it is much faster, more likely to return data if it indeed exists, and has a much lower risk of catastrophic failure of the drive. Unlike other products it can be run safely without operator supervision, and allows for data assessment on the fly.