So werden die Bits auf der Festplatte gespeichert:

Um die einzelnen Bits auf den Platten speichern zu können, müssen diese in magnetische Flussrichtungswechsel umgesetzt werden. Dies bedeutet nichts anderes, als dass die Magnetpartikel der Oberfläche eine andere Polung für eine »0« einnehmen als für eine »1«.

BPI und FCI: Die Aufzeichnungsdichte

- Aufzeichnungsdichte wird in BPI (Bits per Inch = Bits pro Zoll) angegeben (40000 BPI und mehr)
- Flux Changes per Inch (FCI) (=Flusswechsel pro Zoll), gibt an wie oft die Ausrichtung der Magnetpartikel pro Zoll geändert werden kann

Aufzeichnungsverfahren

Die 0- und 1-Informationen werden auf verschiedene Codierungsarten in verschieden gerichtete Magnetpartikel umgesetzt. Die Ausrichtung links/rechts könnte dabei beispielsweise dem magnetischen Nord-/Südpol entsprechen. Der Schreib-/Lesekopf besitzt dazu eine Spule. Wird diese Spule von Strom durchflossen, so bildet sich ein je nach Stromrichtung anders ausgerichtetes Magnetfeld, welches die Plattenoberfläche entsprechend magnetisiert. Da auch Festplatten nie hundertprozentig gleich laufen, müssen zusätzlich zu den Daten sogenannte Taktinformationen gespeichert werden. Diese sichern, dass immer mit gleicher Geschwindigkeit gelesen und geschrieben wird, indem etwaige Abweichungen durch eine Änderung der Drehzahl korrigiert werden (man stelle sich nur das Chaos vor, wenn jeder Sektor unterschiedlich lang wäre). Werden die Taktinformationen, mit den zu speichernden Daten verknüpft, auf jeder Plattenoberfläche gespeichert, so spricht man von einem »Embedded Sector Servo«, dass heißt, die Informationen für den Gleichlauf der Festplatte sind in die Datensektoren eingebettet. Man kann jedoch auch eine Oberfläche des Plattenstapels ausschließlich für diese Taktinformationen verwenden, während alle anderen nur für Daten genutzt werden. Erstere Methode ist eleganter und wird auch meist vorgezogen.

Würden sich ständig 0- und 1-Bits abwechseln, so könnte man auf einen separaten Takt verzichten und beispielsweise die 1 zur Synchronisation verwenden. Probleme bereiten dabei aber längere Folgen von gleichen Bits (zum Beispiel 000 oder 111111). Diese sogenannten Läufe (englisch Runs) stellen besonders hohe Anforderungen an den Gleichlauf. Die beschriebenen Codierungsarten verwenden hier unterschiedliche Methoden, um nicht aus dem Takt zu geraten.

Das Aufzeichnungsverfahren "RLL-2,7"

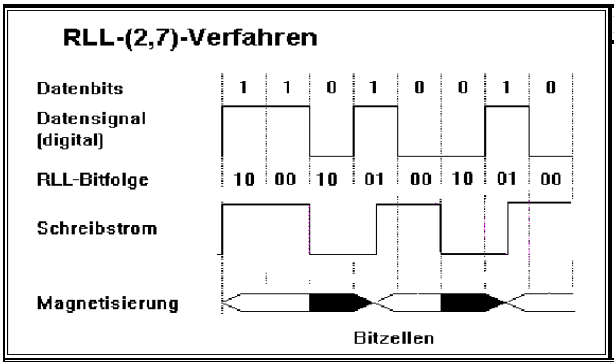

Bild 3: Ein Beispiel für die Magnetisierung der Plattenoberfläche beim RLL-2,7 Verfahren.

Bit	Kontext	RLL-Code
1	1	10 00
1	0	01 00
0	10	10 0100
0	11	00 1000
0	00	00 0100
0	011	00 001000
0	010	00 100100

Bild 4: Datenbits werden im Zusammenhang mit den darauf folgenden Bits zum RLL-Code

Ziel des RLL-Verfahrens ist es, die Länge der »0«-Läufe zu begrenzen. RLL steht für Run-Length-Limited, das bedeutet Lauflängenbegrenzung. Beim hier betrachteten RLL-2,7-Verfahren liegen zwischen zwei 1-Bits mindestens zwei und höchstens sieben 0-Bits. Neuartig an diesem Verfahren war, dass erstmals auch die auf das zu kodierende Datenbit folgenden Bits, der sogenannte Kontext, beim eigentlichen Codierungsverfahren berücksichtigt wird. Die Abb. 4 zeigt, wie aus den Datenbits (in Abhängigkeit von dem Kontext) die RLL-Bitfolge entsteht.

Die in Bild 3 dargestellte Bitfolge wird demnach in drei Schritten übersetzt. Aus 11 wird im RLL-Code 1000, und aus den beiden folgenden 010 wird jeweils 100100. Die Datenbits 11010010 sind also zur RLL-Bitfolge 1000100100100100 geworden; aus ursprünglich 8 Bit werden nun 16. Auffällig sind allerdings die recht zahlreich vertretenen 0-Bits. Der Schreibstrom (also die Magnetisierung) wechselt vor jedem 1-Bit der RLL-Folge. Es entstehen vier Magnetisierungszellen. Zunächst erscheint es paradox, dass RLL Platz sparen soll, sind doch aus 8 auf einmal 16 Bit geworden. Noch einmal soll wiederholt werden, dass die Flusswechsel in der Magnetschicht einen bestimmten Abstand nicht unterschreiten können. Nun folgt der eigentliche Trick: Da beim RLL-(2,7)-Verfahren auf ein 1-Bit wenigstens zwei 0-Bits folgen lässt sich die Aufzeichnungsdichte dadurch steigern, dass man die Bitfolge »001« auf dem kleinsten Stück aufzeichnet. Geht man also von Datenbits aus, so kann man 1,5 dieser Datenbits auf dem minimalen Stück unterbringen, statt wie sonst nur 1 Datenbit. Somit erklärt sich der 50prozentige Platzgewinn von RLL-(2,7) gegenüber älteren Codierungsverfahren.

Michael Schwab
IAV2, März 2002

Dateisysteme

Ein Dateisystem hat im wesentlichen drei Aufgaben:

1) belegten und freien Speicher zu registrieren
2) Verzeichnisse und Dateinamen zu verwalten und
3) festzuhalten, wo die unterschiedlichen Teile einer Datei physikalisch auf dem Datenträger gespeichert sind.

FAT16

- DOS, Windows 3.x, Windows 95 aber auch Windows NT und OS/2
- Verwendung einer Dateizuordnungstabelle (file allocation table - FAT) und Cluster
- Cluster, kleinste Datenspeichereinheit; bestehen aus mehreren Festplattensektoren
- in der FAT wird aufgezeichnet, welche Cluster belegt und welche nicht belegt sind und wo die Dateien gespeichert wurden
- von dieser Tabelle wird ein Duplikat angefertigt, um die darin enthaltenen Daten zu schützen
- FAT-Dateisystem unterstützt maximal 65.536 Cluster, Maximale Größe eines FAT-Datenträgers ist 2 GB
- Enthält ein Stammverzeichnis, in dem eine maximale Anzahl von Verzeichniseinträgen zulässig ist
    i. muss sich an einer bestimmten Position auf dem Datenträger befinden
    ii. Im Betriebssystemen durch den Rückwärts-Schrägstrich (\) dargestellt

Dateiname 8 Byte	Erweiterung 3Byte	Attribute 1Byte
Reserviert 10 Byte	Uhrzeit 2 Byte	Datum 2 Byte
Startcluster 2 Byte	Dateilänge 4 Byte	= 32 Byte

FAT32

- von den neueren Windows 95-Versionen (Version 4.00.950B oder höher)
- DOS, Windows 3.1, Windows NT und die Originalversion von Windows 95 nicht
- 32-Bit-Einträgen in der Dateizuordnungstabelle, im Gegensatz zum FAT-Dateisystem (16-Bit-Einträgen)
- FAT32 kann folglich wesentlich größere Datenträger unterstützen (bis zu 2 Terabyte)
- kleinere Cluster als im FAT-Dateisystem (z.B. 4-KB-Cluster für bis zu 8 GB große Datenträger)
- enthält ein beliebig großes Stammverzeichnis, das an jeder Position auf dem Datenträger abgelegt werden kann

NTFS

- “New Technology File System” (NTFS) wird nur vom Windows NT-Betriebssystem erkannt
- nicht weniger als 400 MB, da ein beträchtlicher Teil des Speichers für die Systemstrukturverwendet wird
- Kernstück“ die Master File Table” (MFT), vom wichtigsten Teil der MFT mehrere Kopien
- NTFS speichert in Clustern, Clustergröße unabhängig von der Größe des Datenträgers (min. 512 Byte)
- Hotfixing, d.h. fehlerhafte Sektoren automatisch ausfindig machen und kennzeichnen damit sie nicht verwendet werden

HPFS

- High Performance File System (HPFS), bevorzugtes Dateisystem unter OS/2, älteren Vers. von Windows NT
- sortiert das Verzeichnis nach Dateinamen
- verwendet eine effizientere Struktur zur Organisation des Verzeichnisses, deshalb häufig schneller als FAT
- ordnet Daten Sektoren zu
- welche Sektoren genutzt wurden und welche nicht, organisiert HPFS einen Datenträger in 8-MB-Bänder mit 2-KB-Zuordnungs-Bitmaps zwischen den Bändern

Ext2

- ausschließlich unter Linux
- Unterstützt Dateinamen mit bis zu 255 Zeichen, echte Unterscheidung zwischen groß- und kleingeschriebenen Dateinamen ("haus.txt" und "HAUS.TXT" sind zwei verschiedene Dateien!) und defragmentiert sich selbst.
- Legt alle Daten in Blöcken mit vordefinierter Größe ab, ebenso wird blockweiße ausgelesen
- Einteilung der Daten in Blockgruppen

Michael Schwab
IAV2, März 2002