Introduction into Copy Protection

Contents

Preface

The following discussion of copy protection issues is almost completely based on my own experiences with copy protected PC software and personal communications with my colleagues. Therefore any statement made herein can't be considered as the holy truth (or any truth at all). This document is NOT the manual on development of copy protection schemes, but a brief introduction into "breaking" them. Anyone using this document or examples supplied with it for the development of copy protection software or hardware will violate license agreement.

No legal aspects of copy protection will be covered here. Nevertheless, you can use this data from this manuscript only according to law. If there is no law on computer software (as in Soviet Union), you should always regard moral considerations. I can't be held responsible for any improper use of this document and code supplied with it.

You will see that protection examples are somewhat scarce in number and are not always up to date. (E.g., parallel port locks, currently the most popular protection device, are not covered at all). Taking into account geographic location of Novosibirsk, this can hardly be my fault.

Any work of this sort is always based on support of other enthusiasts, because no individual can afford to purchase all (or any, because they are sold for the hard currency) of new copy protected programs. So I wish to express my warmest gratitude for all people who supplied me with protected PC software.

Finally, I beg your pardon for my far from good english. I will welcome any suggestions on this document organization and style.

Serge Pachkovsky

Novosibirsk
06 June 1991

1. Copy protection based on key disk

This type (or rather types) of copy protection are of the same age as PC itself. The years of development and sophisticated floppy disk hardware give rise to numerous protection methods. Although with recently appear of parallel port locks, key disk technique seems to be old-fashioned, it still has at least two obvious qualities. First, key disk can be simultaneously used as distribution disk and second, this type of protection is very cheap (but nevertheless hard to tamper with). So, key disks still can be used for wide distributed personal use software. On my account, in the SU key disk protection will dominate for a while.

In order to understand floppy disk controller (FDC) protection tricks, one should be aware of basic FDC data and operation. Let us now consider them. (Note: Sections 1.1 and 1.2 are mostly based on Intel documents "An intelligent Data Base System Using the 8272", "8272 Singe/Double Density Floppy Disk Controller Data Sheet", "8272A Single/Double Density Floppy Disk Controller".)

1.1 MFM floppy disk data recording

Modified Frequency Modulation (MFM) floppy disk format was introduced in the IBM System 34 and is often referred to as "double density recording". Term "single density recording" corresponds to straightforward IBM 3740 Frequency Modulation (FM) format, which used 4 æs to record one bit of data. Original MFM recorded one bit in 2 æs cell, but for the IBM PC mini-floppies was used 4 æs cell. Therefore, unformatted size of one track become 6.1 Kb. So called "high density" PC disks merely implement 2 æs data bit cell of original MFM specification.

1.1.1 Data bits separation

Data recording in FM format was simple: beginning of each bit cell was specified by so called clock bit, and actual data was recorded at center of each cell (data bit) (See fig 1.1.1a). Such technique allows simple data bits separation, but wastes frequency range twice as needed to save actual data. Nevertheless, complete removal of clock bits will make recorded data containing large series of zero bits undecipherable because of random variations in floppy rotation speed and controller oscillator.

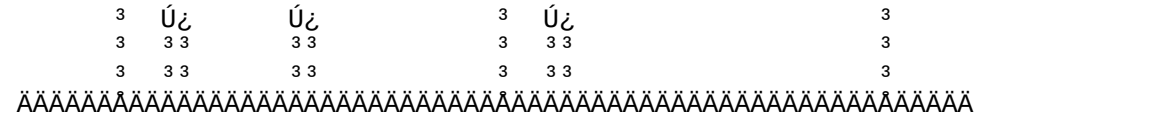


Fig 1.1.1a FM data recording.

To deal with such deficiency, most clock bits were removed in MFM by the following rule: Clock bit should be written at the edge of bit cell if no data bit was written in the previous cell and no data bit will be written in the current bit cell. (See fig 1.1.1b). Such encoding makes data bit separation a more difficult task, but nearly removes clock bits overhead.

Fig 1.1.1b MFM data recording.

One can easily see, that meaning of both FM and MFM encoding depends upon initial position of bit cell. For example, if we will place bit cell as shown by dashed lines on fig 1.1.1b, sequence '1100' will become '0010'. To provide unambiguous data bits decoding each data field on track is supplied with synchronization field.

1.1.2 Synchronization field and ID byte

MFM synchronization fields consist of 96 zero bits (i.e., cells with clock bit and no data bits) followed by 3 bytes of A1h (10100001b). Zero bits permits to align correctly data cell, and A1s to identify beginning of actual data bytes. Although 12 zero bytes are written by FDC during format (this value can't be changed by software), only 1 byte (8 bits) is needed actually to synchronize bit cell. Other 11 are for "just in case".

FM sync fields are simple - they consist of 48 zero bits only. (Again, FDC needs 8 bits and other are for safety).

Different data (both user and auxiliary data) fields can be distinguished by the single byte immediately following sync field. These bytes can't be mixed with user data even if the later contains the exact byte sequence sync/ID, because these bytes (and only these bytes) do not use standard clock bit conventions. Unfortunately, I have information on corresponding clock bits for FM encoding only. ID bytes seem to apply to MFM as well. (Note what data fields are recorded MSB-first).

ID byte	Clock	Field description
FC	D7	Index address mark
FE	C7	Sector ID address mark
FB	C7	Sector data
F8	C7	Deleted data
FE	C7	IBM bad track ID

Although in Intel documentation sync field is always referred to as part of preceding gap, we will consider it to be part of following data field.

1.1.3 Overall track format

The reference point for all floppy operations is physical index mark, which is generated by diskette index hole. Entire floppy track format starting with physical index mark can be described as follows:

- Physical index mark
- Preindex gap (GAP 5)
- Index address mark (IAM)
- Post index gap (GAP 1)

For n from 1 to N-1, where N is number of sectors on track:

- Sector n ID
- Post ID gap (GAP 2)
- Sector n data
- Post data gap (GAP 3)

For the last data sector on track:

- Sector n ID
- Post ID gap (GAP 2)
- Sector n data

- Final gap (GAP 4)

Index address mark (which is not used for any purpose by 8272A) has somewhat different sync field: instead of A1h it uses C2h (11000010b), which are followed by FCh (11111100b) ID byte.

Sector ID field contains FEh followed by one byte values C,H,R,N, where C stands for cylinder number, H for head number, R for sector number and N for sector size code. These bytes (including FEh) are validated by 16-bit CRC, which follows. Size of user data in the following data field can be calculated as 128 * 2^N, i.e., N=0 specifies data size of 128 bytes, N=1 - 256 bytes, N=2 - 512 bytes, and so on. C=H=R=N=FFh specifies IBM bad track.

Data field contains FBh followed by 128 * 2^N bytes of user data and two bytes of CRC. Both in sector ID and in data field CRC is calculated using polynomial x^16 + x^12 + x^5 + 1 with initial value of FFh (as always, MSB first).

1.2 Intel 8272A floppy disk controller

Rumors say, what original PC FDC was implemented on Intel 8272 chip. Of course, it could be any pin compatible - 8272A, NEC æPD765, and so on. I had never seen such a PC, so I just can't tell. Nevertheless, in order to maintain register level compatibility (good feature!) with first PCs, all more recent controllers are nearly the same for programmer.

Commands are performed by 8272A in three subsequent phases: command phase, execution phase and result phase. During command phase CPU instructs 8272A what to do. During execution phase, FDC performs requested action. All user data transfers (if any) will be done during execution phase. Execution phase is followed by result phase, when FDC returns status data.

While FDC data requests during command and result phases could be infinitely delayed (data will be stored in internal 8272A registers), all FDC requests during execution time should be satisfied immediately, or FDC will generate data overrun error and terminate operation. More strictly speaking, data request could not be delayed for more then time of transfer of 8 data bits. Therefore, on 360Kb disk drive, which operates at 250K (1K here = 1000) bits per second (KBPS), FDC will transmit byte of data each 32 æs or 31250 bytes per second.

Although 8272A itself can operate in either DMA or poling mode, only relatively fast CPU is capable of transmitting data at such rate. Running FDC in poling mode on double density disks requires at least 6 MHz 80286, while AT high density floppy will eat up entire 10 MHz 80286.

1.2.1 8272A registers

8272A chip interacts with CPU trough two registers: Main Status Register (MSR) and Data Register (DR). MSR is read only. Either DR could be read or written is determined by RQM bit of MSR. Meaning of bits in MSR could be seen on fig 1.2.1a.

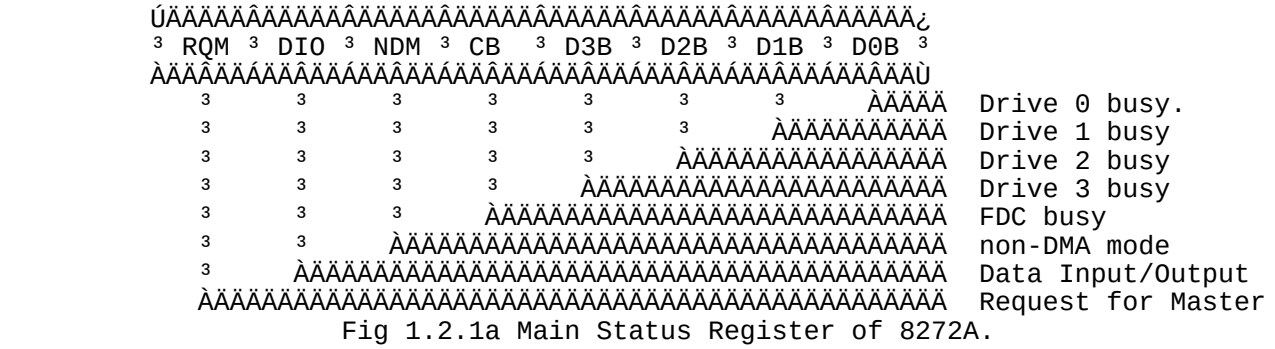


Fig 1.2.1a Main Status Register of 8272A.

Bit DxB will be set to 1 then corresponding drive performs seek or recalibrate command. CB bit will be set then FDC performing read or write operation. NDM is set when FDC is in execution phase and DMA operations disabled (So it indicates what more data should be fed to or read from DR). DIO will be one then CPU should read data from DR and zero if CPU should feed data to DR. RQM = 1 indicates that DR is ready to transfer next byte. Direction of transfer depends upon DIO value.

Internally, 8272A stores current cylinder number for each of disk drives connected to it. Because floppy drive can't distinguish one cylinder from another (except cylinder 0), this is needed to perform seek operations, which are translated by FDC to drive step pulses. These registers are cleared by FDC reset (but floppy disk drive will not return to track 0 on reset).

Other important internal registers contain step rate time, head unload time and head load time (see Specify command).

1.2.2 8272A commands summary

8272A commands can be divided into three groups: user data transfer, floppy control and diagnostic aids. User data transfer includes read data, read deleted data, write data, write deleted data and three scan commands: scan for equal, scan for lower or equal and scan for higher or equal. Floppy control includes recalibrate, seek, sense drive status, sense interrupt status and format a track commands. Read sector ID and read track commands could be considered as diagnostic. All commands with unrecognized first byte will be treated as invalid command.

8272A commands can be easily identified by low nibble of first command byte - see table 1.2.2 (although some of such commands will be invalid).

Table 1.2.2. 8272A opcodes.

x1	Scan equal
x2	Read a track
x3	Specify
x4	Sense drive status
x5	Write data
x6	Read data
x7	Recalibrate
x8	Sense interrupt status
x9	Write deleted data or Scan lower or equal
xA	Read ID
xC	Read deleted data

xD Format a track or Scan higher or equal
xF Seek

Generic sequence to execute 8272A command consists of the following steps:

0. If DMA will be used in operation, program channel 2 of 8237A for the single byte transfer mode. (Note: Terminal Count (TC) signal from DMA will cause immediate completion of FDC operations).

1. For each byte in command, wait until RQM bit = 1, when check DIO: value of 0 indicates what FDC ready to accept command, 1 says what either your command was not recognized by FDC (and next read from DR will return 80h) or that you had already fed all data in FDC (and your command is invalid, too).

If your command does not have execution and result phase (Specify), you stop here.

2. If you are running FDC in poling mode and command transfers data during execution phase (read, write, format), you wait here while NDM bit is zero. When while NDM is not zero, for each byte read or written you wait for the RQM bit to be set and then writes next byte to (or reads from) DR.

If you are using DMA-mode transfer (or no transfer at all), you simply goes to step 3.

3. End of execution phase is indicated by IRQ 6 (int 0eh). You can either enable 8272A interrupts and detect this in your interrupt routine or you can constantly poll 8259A interrupt request register (IRR).

If command does not have execution phase (Sense drive status), you go to step 4.

4. In the result phase you read command status from FDC by poling RQM byte (be sure that DIO = 1 here). 8272A can return up to 3 status bytes (which are shown below) along with other data, which will change from command to command.

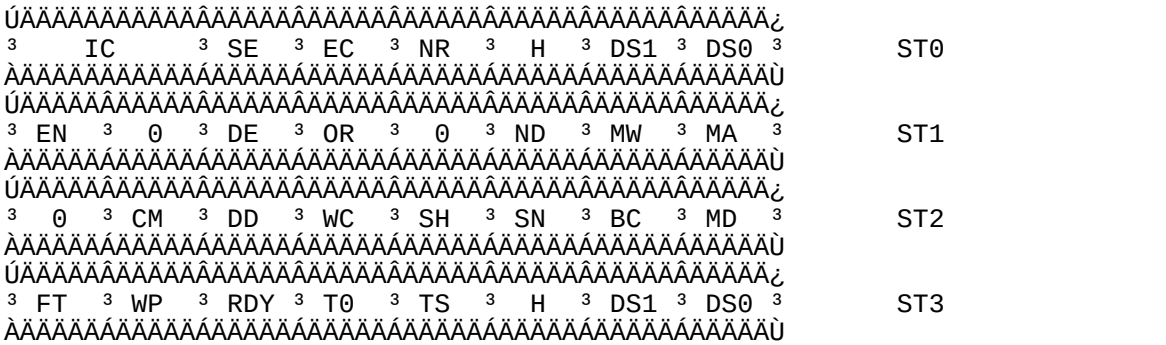


Fig 1.2.2 Status bytes 0-3 of 8272A

ST0.

IC - Interrupt code:
00 - Normal termination of command.
01 - Abnormal termination of command (Operation still can be Ok, other condition codes should be tested).
10 - Invalid command.
11 - Abnormal termination (disk ready signal changed during execution).
SE - Seek operation ended.
EC - Equipment check error (fault signal received from drive or cylinder 0 was not reached after 77 pulses during Recalibrate).
NR - Not ready (Drive becomes not ready during read or write or side 1 was requested for single-sided disk).
H - Head address.
DS0, DS1 - Drive address.

ST1.

EN - End of track error (FDC attempt to access a sector beyond the final sector of track). This flag will be set to 1 (and therefore IC will be equal to 01) if FDC had read sector specified in EOT command parameter and TC signal is low, so every read operation in poling mode will end with EN error.
DE - Data error. Either sector AM of sector data CRC is invalid.
OR - Overrun error. 8272A had not received CPU or DMA services within the specified time interval (32 æs on 360Kb drive), so data was lost.
ND - Sector not found. Specified sector was not found during 2 disk revolutions (i.e., two index pulses were detected since the start of operation). For multi-sector transfers, 2-revolutions time-out applies for each sector separately.
NW - Write protect error. Write protect signal was detected during write or format operation.
MA - Missing address mark. Either sector AM or data AM was not found.

ST2.

CM - Control mark. Deleted data AM detected during read data command or data AM during read deleted data command and SK bit was not set.
DD - Data error. Sector data CRC invalid. DE will be set, too.
WC - Cylinder address error. Cylinder address on track does not match specified cylinder address.
SH - Scan hit. Scan command conditions were satisfied.
SN - Scan not satisfied.
BC - Bad track error. As WC, but cylinder address from track is FFh.
MD - Missing data AM error. MA will be set, too.

ST3.

FT - Fault. Disk drive indicated fault.
WP - Write protect.
RDY - Drive ready signal. On PC and AT floppy subsystems this will always be set, regardless whether drive is ready (or installed at all).
T0 - Track 0 signal.
TS - Two-sided. On PC and AT floppy subsystems this will be 0 always.

1.2.3 8272A commands description

Here I'll list all 8272A commands in the order in which they appear in table "8272A command set" of Intel 8272A manual. Of course, this description is not sufficient to implement floppy disk driver. See Intel manual for complete information or Appendix A for the simple example.

Read Data
ÄÄÄÄÄÄÄÄÄ

Command:	MT	MFM	SK	0	0	1	1	0
	0	0	0	0	0	HDS	DS1	DS0
	C/H/R/N/EOT/GPL/DTL							
Execution:	Data fields on track been read.							
Result:	ST0/ST1/ST2/C/H/R/N							

MT (MultiTrack) instructs to continue read operation on track 1 of the same cylinder.
SK causes FDC to skip deleted data fields.
HDS indicates head number used in operation.
DS1,DS0 constitute a drive number.
C/N/R/N sector ID of starting sector

EOT number of sectors on track
GPL inter-sector gap length
DTL should be FFh for MFM.

Read Sector ID
ÄÄÄÄÄÄÄÄÄÄÄÄÄ

Command:

0	MFM	0	0	1	0	1	0
0	0	0	0	0	HDS	DS1	DS0
C/H/R/N/EOT/GPL/DTL							

Execution: First correct sector ID been read. (First signifies "first encountered starting from current head position" and not "first after physical address mark")

Result: ST0/ST1/ST2/C/H/R/N

Read Deleted Data
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

Command:

MT	MFM	SK	0	1	1	0	0
0	0	0	0	0	HDS	DS1	DS0
C/H/R/N/EOT/GPL/DTL							

Execution: Deleted data fields on track been read.
Result: ST0/ST1/ST2/C/H/R/N

SK instructs FDC to skip data fields.

Read a Track
ÄÄÄÄÄÄÄÄÄÄÄÄ

Command:

0	MFM	SK	0	0	0	1	0
0	0	0	0	0	HDS	DS1	DS0
C/H/R/N/EOT/GPL/DTL							

Execution: Data fields on track been read.
Result: ST0/ST1/ST2/C/H/R/N

This command will read data fields regardless of C/H/R/N values stored in sector IDs. Data field that lacks valid IDs can be read by this command. Although Intel documentation specifies that Read Track command stops then no data fields at all found on track or sector count reaches EOT value, this command will terminate then no data address mark was found after sector ID address mark with bad CRC.

Read a track command will accept ANY N value, so it can read inter-sector gap (or all track, if N is large enough) along with sector data. Sectors will be read in order of appearance under r/w head, i.e., if track was formatted with 8 512-byte sectors (interleave 1:1), and you start read track command with R = 1, N = 3, EOT = 4, sectors 1, 3, 5 and 7 will be read.

Write Data
ÄÄÄÄÄÄÄÄÄÄ

Command:

MT	MFM	0	0	0	1	0	1
0	0	0	0	0	HDS	DS1	DS0
C/H/R/N/EOT/GPL/DTL							

Execution: Data fields on track been written.
Result: ST0/ST1/ST2/C/H/R/N

Write Deleted Data

ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

Command:	MT	MFM	0	0	1	0	0	1
	0	0	0	0	0	HDS	DS1	DS0
	C/H/R/N/EOT/GPL/DTL							
Execution:	Deleted data fields on track been written.							
Result:	ST0/ST1/ST2/C/H/R/N							

Format a Track

ÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

Command:	MT	MFM	0	0	1	1	0	1
	0	0	0	0	0	HDS	DS1	DS0
	N/SC/GPL/D							
Execution:	Command formats track with SC sectors of size N. Sectors are filled with byte D. C/H/R/N values for each written sector ID should be supplied by user (as in write data command).							
Result:	ST0/ST1/ST2/C/H/R/N							

Note: GPL = 0 will be understood as 100h.

Scan Equal

ÄÄÄÄÄÄÄÄÄÄ

Command:	MT	MFM	SK	1	0	0	0	1
	0	0	0	0	0	HDS	DS1	DS0
	C/H/R/N/EOT/GPL/STP							
Execution:	Data fields on track been read and compared byte by byte with data supplied by CPU or DMA. If scan condition not satisfied, scan continues with sector R+STP.							
Result:	ST0/ST1/ST2/C/H/R/N							

Scan Lower or Equal

ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

Command:	MT	MFM	SK	1	1	0	0	1
	0	0	0	0	0	HDS	DS1	DS0
	C/H/R/N/EOT/GPL/STP							
Execution:	Data fields on track been read and compared byte by byte with data supplied by CPU or DMA. If scan condition not satisfied, scan continues with sector R+STP.							
Result:	ST0/ST1/ST2/C/H/R/N							

Scan High or Equal

ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

Command:	MT	MFM	SK	1	1	1	0	1
	0	0	0	0	0	HDS	DS1	DS0
	C/H/R/N/EOT/GPL/STP							
Execution:	Data fields on track been read and compared byte by byte with data supplied by CPU or DMA. If scan condition not satisfied, scan continues with sector R+STP.							
Result:	ST0/ST1/ST2/C/H/R/N							

Recalibrate
ÄÄÄÄÄÄÄÄÄÄÄ

Command:

0	0	0	0	0	1	1	1
0	0	0	0	0	0	DS1	DS0

Execution: Recalibrates specified drive.

All 4 drives can be recalibrated simultaneously. 8272A can perform read or write operation on any other drive. So sense interrupt status command should be used to distinguish interrupt caused by completion of recalibrate from operation completion interrupt.

FDC will not permit any command on recalibrated drive before sense interrupt status issued for seek completed interrupt.

Sense Interrupt Status
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

Command:

0	0	0	0	1	0	0	0

Result: ST0/C (Note: then IC = 10 (invalid), C will not be returned)

8272A will generate interrupt request if any of the following events occurs:

a) Result phase begins for Read/Write/Format/Scan
b) Ready signal from one of drives changed.
c) Seek/Recalibrate completed.
d) Non-DMA data transfer required.

Interrupt service routine can easily distinguish this cases:

```
if NDM = 1 then it's data transfer request
else
if CB  = 1 then it's result phase beginning
else
if SE  = 0 then it's ready signal change
else            it's seek or recalibrate end.
```

Specify
ÄÄÄÄÄÄÄ

Command:

0	0	0	0	0	0	1	1
SRT				HUT			
HLT							ND

SRT Step rate interval. Sets the minimal delay between two drive step pulses. (1-16 ms, SRT = 0Fh corresponds to 1 ms). Older PC drives supported 3 or 4 ms step rate, some never ones can cope with 2 or even 1 ms step rate.

HUT Head unload time. Delay between completion of read/write operation and lifting head. (16-240 ms, HUT = 0 is 16 ms). This is usually set to 240 ms.

HLT Head load time. Delay between head load command and start of read/write operation. (2-254 ms, HLT=1 is 2 ms). This is usually set to 2 ms.

ND 0: DMA mode. 1: non-DMA mode.

All timers inside 8272A are actually synchronized to WR CLK signal, so all values shown here are correct for WR CLK 500 KHz or data transfer rate 500 KBPS. Lowering frequency of WR CLK will lengthen all internally counted delays.

Sense Drive Status
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

```
    Command:              0     0     0     0     0     1     0     0
                         ÀÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÙ
                            0     0     0     0     0    HDS   DS1   DS0
                         ÀÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÙ
    Result:               ST3
```

Seek
ÄÄÄÄ

```
    Command:              0     0     0     0     1     1     1     1
                         ÀÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÙ
                            0     0     0     0     0    HDS   DS1   DS0
                         ÀÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÁÄÄÄÄÄÙ
                         C
```

    See Recalibrate.

1.2.4 AT floppy disk controller

Original PC and XT floppy disk controllers were capable of running four floppies. AT FDC, although register compatible, can run only 2, so one can install 2 FDCs into AT. Most clone BIOSes does not support secondary FDC, and I never had IBM AT to check wether IBM BIOS does. First FDC occupies address range 3F0-3F7, secondary FDC is at 370-377. All information here on primary FDC also can be applied to secondary by moving references to this address range. 8272A (or it's large scale integrated substitution) is mapped to ports 3F4 (MSR) and 3F5 (DR).

As you have already seen, 8272A itself can't control disk drive motor status, so register 3F2 (digital output register) provides such services. (see fig 1.2.4a).

```
    ÚÄÄÄÄÂÄÄÄÄÂÄÄÄÄÂÄÄÄÄÂÄÄÄÄÂÄÄÄÄÂÄÄÄÄÂÄÄÄÄÂÄÄÄÄÂÄÄÄÄÂÄÄÄÄÂÄÄÄÄ¿
    ³ D  ³ C  ³ B  ³ A  ³ IE  ³ EC  ³ DS1 ³ DS0 ³
    ÀÄÄÄÁÄÄÄÄÁÄÄÄÄÁÄÄÄÄÁÄÄÄÄÁÄÄÄÄÁÄÄÄÄÁÄÄÄÄÁÄÄÄÄÁÄÄÄÄÁÄÄÄÄÁÄÄÄÄÙ
       ÀÄÄÄÄÄÄÄÄÄÁÄÄÄÄÄÄÄÄÄÙ     ³     ³     ÀÄÄÁÄÄÙ
                 ³               ³     ³        ÀÄÄÄÄÄÄÄÄ  Drive select
                 ³               ³     ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ  Controller enable
                 ³               ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ  Interrupts enable
                 ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ  Motor enable
```

Fig 1.2.4a Digital output register (3F2h).

Bits DS1 and DS0 of digital output register overrides DS bits specified in 8272A command, so one can change disk drive on the fly during command execution. Although this is dangerous technique, many copy protection tricks are based on this feature.

Setting EC bit to zero resets controller, so any operation (e.g., format track or write data) can be stopped on any time. (Although special care should be applied to perform abrupt termination without oscillations).

IE bit was intended for use in multiply-controller systems. Because all FDCs use the same IRQ 6 and DMA channel 2, inactive controller can (or cannot - no one can say) drive this line low, thus forbidding other controllers from obtaining system services.

Drive motor enable bits is another source of protection tricks (See 1.4.2).

Second additional register is at 3F7 (diskette control register). (This register does not exist on old XTs, but can often be found on recent clones). To my best knowledge, only two lower bits are meaningful for write and single upper bit for read.

3F7 write - select transfer rate:

    00 - 500 KBS (MFM)   01 - 300 KBS (MFM)
    10 - 250 KBS (MFM)   11 - 125 KBS (FM)

Thirst three values (00 thru 10) was mentioned in IBM manuals. Although last value is undocumented, it works Ok on any AT FDC I'd ever encountered.

(This is significant transfer rate - it permits to read/write SD disks in 360 Kb drive). Values 00 thru 10 are intended for use with MFM floppies, so they have pulse length 1/4th of cycle length. 125 KBS clock have pulse width 1/8th of cycle length.

Because AT FDC have no 175 KBS FM clock, single density disks can't be written in high density (360 RPM) drive. Due to high error resistance of FM data format, single density disks still can be red in 360 RPM drive. FM floppies can be both red and written in 360Kb (300 RPM) drive.

3F7 read - drive change status: bit 7 set (test al, 80h) means that diskette was changed. Media changed indicator will be reset after first seek to cylinder other then zero.

Obvious use for the 3F7 register in the copy protection world concerns adjusting of bit cell position. Because FDC switches to new transfer speed almost instantaneously, switching transfer speed at calculated moment will shift bit cell by small fraction of it's length. This technique permits to read any section of track, provided that at least one valid data field exists and position of section of interest is approximately known.

1.2.5 "Normal" PC floppy formats

First standard PC floppies were designated for use with single- and double-sided 40 tracks mini-floppies. Following Intel specification, they had 8 512-bytes sectors per track, thus using 4096 bytes (65%) of 6250 bytes of unformatted track capacity. Sectors on track were ordered sequentially (interleave 1:1), as in all other standard formats. More recent 9 and 10 sector/track floppies use 74% and 82% of total space respectively. Theoretical limit for formatted DD track capacity is something about 89.2% (5575 bytes or 10.89 sectors/track).

Such increase in floppy capacity was made possible by high stability of revolution time in modern floppies. Although deviation of 2.5% from nominal revolution time is permitted by IBM, I'd never seen floppy drive with revolution time outside 0.2% region. Such stability permits to decrease intersector gap length.

For example, if possible deviation in rotation speed is ñ2.5%, one should reserve inter-sector gap sufficient to hold any resulting sector length. Suppose that floppy been formatted on drive with slowest possible rotation speed, and will be updated of drive with fastest rotation speed. In such a case, one should reserve at least (512+62) * (2*2.5%) bytes for intersector gap (62 is minimal sector header size for MFM format). This will result in minimal gap of 29 bytes, so even 10 sectors/track format is Ok for such deviations.

In order to maintain compatibility (at least partial) with 40 tracks formats on 80 tracks HD drive, they are simulated by skipping all odd tracks. Therefore 360K disks written in 360K drive always can re read in HD drive, but 360K floppy written in HD drive can be incompatible with 360K drives.

Because key floppies are almost invariably implemented on 360K drives, we will leave all high density formats alone.

1.3 Time independent protection tricks

Here we will consider floppy protection marks, which can be verified without use of devices others then FDC itself. This tricks should work on any system equipped with compatible FDC.

Note: Anytime when I will mention bit-copier, it will concern the latest version i'd seen. Anyway, they are not the latest versions exists. I have copies (illegal copies for the causes mentioned in preface) of Copy II - PC release 7.10, CopyWrite September 1988 edition and TeleDisk v 2.11.

Note: Statement "could not be reproduced" applied hereafter to one or other protection mark indicates not physical limits of PC hardware, but limits of my knowledge of it. I will be grateful to anyone who will expand these limits.

1.3.1 Extra or missing sectors

Because first MS-DOS floppies had a lot of unused space on track, it was an obvious idea to store additional sector(s) on track along with standard MS-DOS sectors. Even far better packed 10 sectors/track format will accommodate single 256-bytes sector. Other place to locate such additional sectors is track 41, which is not used by DOS, but can be accessed in most drives. On 80 track drives, extra sectors could be hidden on odd tracks.

Now, this mark alone can't be considered as good copy protection, because extra sectors can be easily found by read sector ID command. Any bit-copier on market (CopyWrite, CopyIIPC, TeleDisk, etc.) will do it. (Although sectors on odd tracks will not be found by some programs). For the nice variations of this scheme, see sections 1.3.4-1.3.8.

1.3.2 Weak data bits

Other old good protection trick is data, which will appear different in subsequent read operations (weak data). Weak data can be caused by either data falling into digital "uncertain area", or by long series of zero data and missing clock bits. In first case, decision of FDC will be driven by random noise. In later case, random variations in drive rotation speed will move bit cell out of synchronization (Such data could not be written without modification of FDC hardware, so we will not discuss them).

One good ability to generate weak data is to place them atop of surface defect. Such weak data will not disappear after write sector data command. Unfortunately (hi!), manufacturer surface defects are now rare to occur, so one should make defects manually. I'd seen a lot of such techniques, ranging from disk scratching with a rusty nail to careful evaporation of surface layer by 1 kVA infrared laser. Nevertheless, it's hardly a software issue.

Weak data bits could be created by software, too. First approach is to manipulate drive select/deselect bits in digital output register (3F2h). For example, if you wish to create weak byte on drive A:, you should start write operation, wait until desired byte will be transferred to disk drive (not to FDC!) and repeat sending to digital register values 1Dh (select drive 1) and 1Ch (select drive 0) while byte being transferred. Such operation will modulate all data (including clock bits) written to disk with square wave, moving them into uncertain region.

Second technique requires almost the same operation with diskette control register (3F7h). Data rate switching will misplace data and clock bits and deform them in shape, also moving into uncertain area.

Bit copier encountering weak data faces an interesting dilemma: whether weak bits are consequence of unrecognized surface defect, and operation should be repeated until original data will be recovered or is it a protection mark, which should be reproduced in all it's glory?

Of all bit-copiers i'd encountered, CopyWrite only was able to cope with weak data, although it converts single weak byte into something of 10 to 12 bytes in length, so original mark still can be easily distinguished from copy. Wherefore, weak data is not such a bad thing for cheap protection and will obviously pop-up in homemade SU software.

1.3.3 Data in gap

One can easily store small amount of data in gap after sector data field (GAP 3). Provided that sector is not overwritten, mark will reside in gap completely safe. Upper margin for number of data bytes, which will fit into GAP 3 given by GPL value of track format command. Verification of such mark is simple (at least for the first sector on track) - read a track command with N one large then actual value in sector ID will load gap data into memory.

Storing data in gap requires more sophisticated procedure. Assume

someone wishes to store 10 bytes of data into gap of first 512-bytes sector on track 0 head 0. He should first format track 0, specifying length code 3 (1024 bytes) for the first sector ID but length code 2 for the format parameters. Then he should start write operation on his dummy 1024-bytes sector, but stop it after transfer of 526 bytes (512-byte sector data + 2-byte CRC + 10-byte gap data + 4 byte safety margin). Then he should start format track operation with length code 2 (both format parameters and sector ID) and stop it somewhere inside GAP 2 (sector ID already written, but data field still not affected). Both format and write operations can be stopped by either controller reset (sending 0 to digital output register, 3F2h) or by changing selected drive (see 1.3.2).

Unfortunately, it can sometimes be difficult to differentiate between "empty" gap and gap containing protection data. Older floppy drives have had a relatively large write signal attenuation times, so gaps written on such drives are filled with random garbage, which can be misinterpreted as binary protection data. Protection scheme designer can enhance these difficulties, using weak bits (1.3.2) inside gap protection data.

Among mentioned bit-copiers only CopyWrite was able to detect data in gap. Single curious exception is first sector of track 0, head 0, which is deliberately ignored by CopyWrite as place for gap data.

1.3.4 Sectors with no data address mark

Sectors without data address mark will generate missing data AM error (MA bit in ST0 and MD bit it ST2 will be set) during data read and write operations. Read sector ID command will terminate normally on such sector.

Storing sector with no data AM requires simple format operation, which should be stopped after writing sector ID AM but before writing data AM (as in 1.3.3). Special care should be taken to eliminate early existed data AM, either by using electromagnetically erased disks or by preliminary format at different data transfer rate. Alternative way (which will not work for first sector on track) is: First, format track with GPL value selected to place data field where the data AM sync field should start on target disk. Second, reformat track with desired GPL value and stop before sector AM is written.

1.3.5 Sectors with no sector ID address mark

Sectors without sector ID AM can be written in the following fashion: Format track, storing length code N+1 for the sector before one of interest. Then read content of this dummy sector and write it back, stopping operation then sector ID AM already overwritten but data AM still Ok. Now you had got sector with no ID AM. Such sector will NOT raise any exceptions in any FDC operations. It can't be, generally speaking, read by any command without special care taken, so it almost does not exist.

No one of tested bit-copiers was able to recognize such sectors (but any hardware bit-copier should be able to do it), so missing ID AM can be considered as good protection. Nevertheless, verification difficulties make appearance of such mark in "live" protection scheme highly unlikely. (But see 1.3.10).

1.3.6 Sectors with bad sector ID address mark

This mark differs from 1.3.5 only by extent of ID AM corruption. Here, FDC is still able to recognize ID AM, but CRC on it verifies incorrectly. This mark can't be detected by read sector ID command, but will set DE bit in ST1 and clear DD bit in ST2 during execution of read sector command, so with values C/H/R/N been known, verification presents no problems. Sector with bad ID AM will nevertheless appear in read a track command. Then exact C/H/R/N values are not known, they still can be obtained by the following procedure: Knowing exact position of data field from time measurements, one can deduce approximate position of ID AM and read it, using technique of adjusting bit cell (See 1.2.4).

Valuable for copy protection designer modification of this mark has invalid CRC of sector ID and no data AM. Such sector will cause termination of read a track command, thus preventing detection of "normal" marks of such type placed after it by bit-copier.

In order to write bad ID AM, one should stop format operation while sector ID AM CRC being written. (Let me again note, that FDC has internal buffer for something about 3 bytes, so controller starts write CRC not then last ID byte been fed to DR, but some time after that). CopyWrite will recognize sector with bad ID AM and copy it.

1.3.7 Data field passing over index address mark

Protection marks of this type had appeared when someone asked: That FDC will do if format a track command will specify total data length on track to be slightly more then track can hold? Consider the following results obtained by formatting DD disk in HD drive with 13h 256-bytes sectors and different GAP 3 values (Sector offsets measured from physical index hole). Note that revolution time of drive used is 166.52 ms.

Table 1.3.7

GAP 3 value	Sector 1 offset	Sector 13h offset	
	Start	Start	End
01h	3.849	156.981	165.46
08h	3.844	160.334	2.29
10h	missing	164.180	6.14
14h	missing	missing	
18h	missing	1.519	10.00

One can see that there is small area at the beginning of track, which is not used by sector data (actually, its used by index address mark). This area can be overwritten by last sector on track (as on second line of table 1.3.7), but if last sector data overlaps with beginning of track too much, sector 1 will be erased (line 3). If the start of data field of last sector will pass over index hole (line 4), controller will write GAP 4 until next occurrence if index hole, thus overwriting all existing sectors on track. If the start of sector ID AM passes over index hole, too (line 5), this sector will be preserved, overwriting all previously written sectors.

Sector passing over IAM can cause significant problems for bit-copier unaware of their existence, because many protection marks are generated by repeated format operation, which will destroy data in over-IAM sector. No one of tested bit-copiers was able to reproduce such sectors (i.e., retain both sector data and sector position), so this mark being extensively used in SU copy protection schemes.

1.3.8 Multirate tracks

As we have seen in previous section, FDC ignores IAM at the beginning of track, so sectors can appear at any position inside track. It's therefore an obvious idea to have different parts of track written at different data transfer rates. (This is best done not by switching transfer rate during format operation, but by two subsequent formats at different rates). For example, one can have 9 sectors of 512 bytes on track 0 at 300 KBPS (HD drive) and single 512-bytes sectors at 500 KBPS. (Because this is an outermost floppy track, using high transfer rate on DD disks will not push data safety too far).

Although different data transfer rates are entirely in AT domain, similar mark can be generated with FM and MFM data formats (i.e., 9 MFM sectors and 1 FM one) on PC and XT. No one of tested bit-copiers was able to reproduce multirate track, so it's a good trick for a while.

Nevertheless, I'd not seen floppy protection based on such mark still.

1.3.9 Data access over gap

Now we'd come to a most popular (and possibly the best) floppy protection trick, data access over the gaps. Because all sector ID and data fields are synchronized separately, two bit cells in two subsequent fields can be shifted arbitrary. This shifts are driven by random variations in WR CLK and drive rotation speed, so they can't be controlled. (Of course, one can imagine ANALOG copying device, what will be able to do it. I'd never heard about such birds.) Any read track operation with sufficiently large length code (say 6) will give "track footprint". Number of different "footprints" can be roughly estimated in the following fashion: Assume that each joint of separately synchronized fields can have 2 different states (obviously, it's underestimation). On the track of an ordinary DD floppy there are 19 such joints (9 ID AMs, 9 data AMs, 1 IAM), so number of different footprints for each track is at least 2^19 ÷ 500,000. To make life more interesting for analog copying devices, one can further enlarge length code (say to 7). This will not increase number of footprints, but will include track more then once. Because even best device should start and stop operation somewhere, it will partially corrupt data. This mark sets single restriction on use of protected disk: key track can't be overwritten.

Other, less interesting variation of this idea, is to read next sector ID AM over the GAP 3 and check whether it changed. Such check forbids write access to a single sector only. Obviously, protection designer does not restricted to obtaining "footprints" with read a track command (and thus to chip-level access to hardware). It can be done simply by adding dummy sector with length code 6 or 7 at end of track, and job will be done by BIOS.

Neither one of these checks can be reproduced on standard PC equipment. Nevertheless, many programs using second variation can be outsmarted by enlarging GAP 3 after sector of interest and storing sector ID AM, as it appears for over-gap read, into this gap. If program does not check exact sectors position on track, it will accept such "copy" as a key disk.

1.3.10 Crazy ideas

All previously discussed protection tricks had permitted non-destructive verification. Now let me imagine protection mark, which can't be checked or detected by bit-copier without preliminary write access to key disk. Remember sectors with no ID AM (1.3.5). What will we get if no other sectors exists on track of interest? (At least, at the same transfer rate). This sector can't be read until format a track operation, which will supply ID AM. Any attempt to step inside such program with debugger will destroy protection mark, because precious timing of stopping format operation will be affected by debugging. This simple example shows that there are protection marks, which can't be baited out key disk without destroying part of it content or without inside knowledge of protection check.

1.4 Timer-based protection tricks

All PCs are equipped with a relatively good timer chip, Intel 8253 or its functional equivalent. Operating at 1,193,180 Hz, it permits to measure time intervals with 840 ns resolution (i.e., it can measure execution time of SINGLE div instruction on 16 MHz 386, which can take up to 2.4 æs). Such timer is more then adequate for floppy timing measurements (Transmission of single data byte on fastest possible rate (500 KBS) will take 16 æs), so precision of such measurements will be limited not by timer resolution, but by random rotation speed variations.

1.4.1 Sector ordering (interleave) checks

Measuring completion time of subsequently issued read track commands, one can obtain exact position of each sector on track. In terms of data bits, you can obtain sector position exact at least to one bit. Because FDC's format a track command controls sector position with byte resolution, it can't reproduce such exact sector ordering. So, sectors position alone can serve as protection mark. Nevertheless, this check will be too sensitive to CPU speed and drive rotation stability and will often reject original key disk. Therefore, sector position is ordinary used as supplement to other protection mark(s) (See 1.3.9).

1.4.2 Data transfer rate measurements

FDC data transfer rate selected by diskette control register (3F7h) is only the initial, or center, frequency used by FDC for data bits decoding. Special analog circuit, called phase-locked loop (PLL), adjusts it in order to track rate, at which actual data bits arrive. PLL should tolerate at least 4% deviation from central frequency (permitted deviation in drives rotation speed is ñ2%). In practice, PLL will cope with 10% deviation on MFM floppies and almost 100% on FM ones.

Rate of data bits arrival is determined by both angular density of floppy data and drive rotation speed, so to determine the disk characteristic (data angular density) one should measure both single sector transfer time and revolution time. Maximum accuracy of such measurement can be easily estimated: Single 512-bytes sector will be transferred at (central) frequency of 500 KBS during 8.2 ms, timer resolution of 0.84 æs gives relative accuracy of 0.01%. Relative error in revolution time determination is less at least by an order of magnitude. Giving a 10 factor for a safety margin, 0.1% seems to be a reasonable estimation. So, all PC floppy drives will fall in 40 (2*2%/0.1%) different groups, and floppies written on drive from one group can be easily distinguished from floppies written on others.

Unfortunately (Hi!), most modern drives will fall into ñ0.2% region and thus into 4 groups instead of 40, almost eliminating all "protection". Some FDCs (e.g., my old 1986 WD HDC/FDC) had permitted simple trick with digital output register (3F2h): output 0Ch to 3F2h (stop drive A: motor), wait a while (10 ms), output 1Ch to 3F2h (start drive A: motor) and immediately perform write operation. Drive rotation speed was slightly less when nominal for about 20 ms, permitting to write single sector. My new 1990 IDE HDC/FDC, however, waits until rotation speed reaches nominal, delaying write operation.

Interesting variations of this mark can be achieved by minor modifications of PC hardware (See 1.5.3).

1.5 Protection based on special hardware

I am not specialist on copy protection hardware, so discussion here will be almost invariably based on rumors and speculations.

1.5.1 Modified MFM formats

Intel 8272A FDC chip has no software control over GAP1, GAP2 and GAP5 length, but will still accept floppies with these gaps differ from standard values and has ability to measure actual gap length. For example, GAP2 (post ID gap) can be measured either by read a track command with length code large then actual sector length or by measuring difference in completion time of read sector ID and read data instructions. Floppies with different gap values could be created for PC on other computer systems, which have control over these parameters. (I'd heard that some DEC systems do.)

1.5.2 Misplaced data tracks

Some floppy controllers/drives can have more strict control over read/write head placement then PC does. (I was assured that required hardware modifications are not too large.) This can be originally done not for copy protection purposes, but in order to provide ability to read floppies written on badly adjusted drives. (Again, I'd heard that ICL FDC has such ability.) Therefore, key disk for such system can be prepared with non-standard track positions (and software will verify this). Such disk could not be verified on other computer system, so this technique could have only a limited application.

1.5.3 Non-standard transfer rates

As we have seen in 1.4.2 FDC will permit significant variations in angular data density on track (and has ability to measure them). Minor modifications of FDC or disk drive give ability to control manually center frequency of WR CLK oscillator or drive rotation speed and thus give ability to write data on slightly non-standard (but still acceptable for most FDCs) data transfer rate.

Some hardware manufacturers (e.g., of scientific equipment with built-in microcomputers) do this to "cuff" customers. (I'd seen 360 RPM 3¬ inch drives, which were therefore incompatible with "normal" 300 RPM 3¬ drives.)

1.6 Protection schemes examples

Inside knowledge of floppy protection schemes shown here based upon my personal experience with protected programs (unless other specified), so it can't be considered to be complete and can't cover all aspects and all versions of protection software discussed. All protected programs discussed here were examined with Copy Unprotector Toolkit (C.U.T.).

1.6.1 IBM Filing Assistant

This is really simple and straightforward protection dated 1986. It has no any practical significance and shown here only to demonstrate development of floppy protection art during last years. This "protected" software checks the presence of sector 8Fh length code 2 on track 39d side 1 of key floppy. No other checks are made. Because first versions of CopyWrite I'd seen were issued in 1985 (and were able to copy such marks), only hopes for dumbness of "these russian apes" can stand behind such protection.

1.6.2 SuperLok

Superlok disk I was able to acquire was dated 10 oct 86. Protected disk had contained three tracks with marked "abnormality" (all at head 0). The first one was track 5, which had unusual interleave and two sectors with short data, so sector IDs constitute the following sequence (all values are in hex):

05/00/01/02, 05/00/06/02, 05/00/8A/03, 05/00/02/02, 05/00/07/02,
05/00/65/03, 05/00/03/02, 05/00/08/02, 05/00/04/02, 05/00/09/02,
05/00/05/02

As it turned out, SuperLok does not checks this track at all, so it constitutes the sort of decoy for interested explorer.

Second protection track on disk was track 12d. It contained sector with hidden address mark (see 1.3.6) with ID 7B/46/05/00, which SuperLok had verified with read data command. Other protection check on this track, was done by read a track command with length code 6 (single 8192-bytes sector) and 3 (two 4096-bytes sectors). 16-bit checksum of bytes was calculated in both cases and compared with stored value (see 1.3.9).

Third protection track contained four special sectors. First of them had hidden sector AM and no data AM, preventing detection of others. Three

sectors had hidden sector AM: 21/47/05/00, A5/86/81/04 and EB/76/EE/04. After verification of these, SuperLok performed read a track with codes 6 and 3, again calculating 16-bits checksums.

So, SuperLok presents good copy protection. No one of tested bit-copiers was able to copy such disk "literally" without modification of SuperLok data areas (and thus, internal knowledge of program). Probably, such "literal" copy is merely impossible (see 1.3.9).

Later, I had a chance to make a quick glance on SuperLok disk dated 1990. Hidden sector AM checks were removed (?), Read a track access to protection mark was substituted (?) by additional sector with length code 6 at the very end of track. All (?) floppy access was done by BIOS.

1.6.3 Cops CopyLock II

COPS CopyLock II bases protection checks on timing measurements. It uses additional sector 00/00/6A/01 on track 0 head 0 to store variable protection data. (COPS CopyLock permits "upgrade" protected software releases without key disk upgrade, so it can't store any more then serial number in program body). Second additional sector on track 0 (00/00/F6/02), which had partially overwritten IAM, was used to verify sector 1 ID AM over gap (see 1.3.9). In order to prevent protection bypassing ala 1.3.9, CopyLock verifies position of all (?) sectors on track by read sector ID command timing.

Tracks 1 to 6 are very similar: they have two additional sectors at end of track: XX/00/14/01, which is not used (?) and XX/00/13/02, which overwrites IAM and is verified by the same manner as 00/00/F6/02. Tracks 7 to 9 have single additional sector (XX/00/14/01) each. These sectors are not used (?). On floppy I'd examined there was surface defect at track 32 head 1, but CopyLock had not verified its existence, so I didn't think it's another protection mark.

This is again example of good protection scheme, although achieved by different means then 1.6.2. Nevertheless, with aid of C.U.T. I was able to recover unprotected version of program of interest (It was Paradisk from JV ParaGraph) during about 2 hours. I didn't think that development of CopyLock took less. Although I have copy of COPS CopyLock III, I still hadn't time to explore it, so it is to appear. . .

1.6.4 PC Shield

This (homemade) disk protection scheme, devised by Alex Simkin, got a great advertisement in Soviet mass media. I had an ability to examine half a dozen versions of this product and thus monitor gradual development of the program.

First versions of PC Shield had used simple mark of sector, overwriting IAM on track 0 solely. This fooled almost all bit-copiers, which had detected this mark as short data (Half of sector is after index hole, isn't it?) and thus destroyed it. Nevertheless, this mark still can be easily written by ordinary BIOS calls (not speaking of new versions of bit-copiers), and this protection was cracked.

Next step taken was to access first sector's ID AM over gap (COPS influence?) Originally, length code 2 (512 bytes) was selected for this additional sector. This mark was cracked by technique described in 1.3.9 (again, nothing more then BIOS required). And now, the final step - length code of additional sector was increased to 6 (SuperLok influence?), making it difficult (but not impossible, still) to reproduce this mark by BIOS. Obviously, it's not an obstacle for a good bit-copier, and C.U.T. will mimic such mark easily.

Further development of this protection scheme can go into two directions: inclusion of sector position checks (as in 1.6.3) or full track over-gap read (as in 1.6.2). Second direction, although requires reservation of entire track for protection purposes, seems to be more probable, because such development does not require chip-level access to FDC (all versions of PC Shield known to me use solely BIOS level floppy

access routines).

2. Hard Disk based protection

Floppy-based keys can provide good level of security, but they are tiresome for everyday use, waste precious disk drive and can be easily damaged by improper handling. Therefore, most copy protected packages can be installed on hard disk, using either hard disk protection marks (this chapter) or motherboard marks (chapter 3). Due to existence of lot of incompatible physical hard disk interfaces, hard disk marks are usually "least common denominator" of various possibilities and are simpler to tamper with.

2.1 Chip-level protection

This level of hard disk access is highly unlikely to encounter, so we will briefly discuss AT WDC (Winchester Disk Controller), with which I have little personal experience. Primary AT WDC could be accessed at addresses 1F0-1F7 and 3F6 (This port is in address space of primary AT FDC, and good cause for hardware clashes in systems with FDC and WDC implemented on different add-on boards). Secondary AT WDC occupies addresses 170-177 and 376. AT WDC use IRQ 14 (int 76h with MS-DOS) and do NOT use DMA. Structure of AT WDC looks very alike to Intel 82062 chip operating in extended mode with 4 bytes ECC (Error Correction Code) appended to each sector. (Unfortunately, I have only Advance Information Sheet on 82062, so the description below can't be precious or complete).

1F0 - Data register, used to read/write sector buffer (512 bytes). Although word access can be used for sector data, only byte one is acceptable for ECC bytes read/write.

1F1 - read: Error flags register

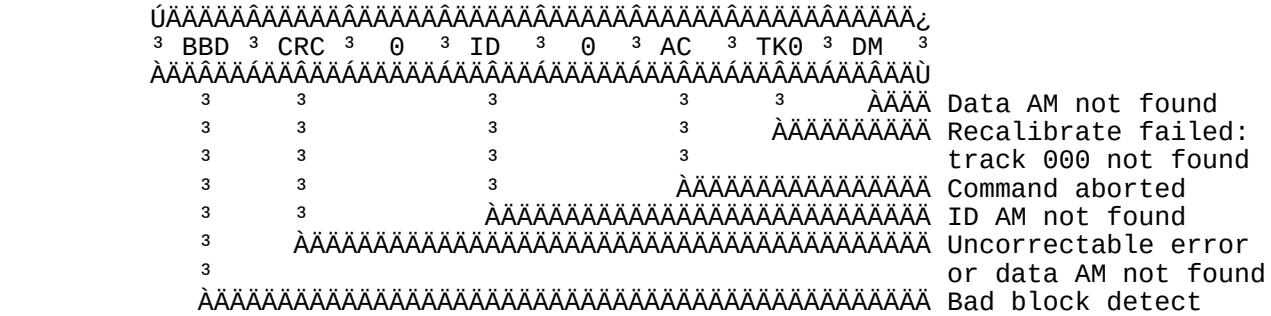


Fig 2.1a Error flags register (1F1)

1F1 - write: Start reduce write current cylinder / 4, 0FFh disables RWC feature.

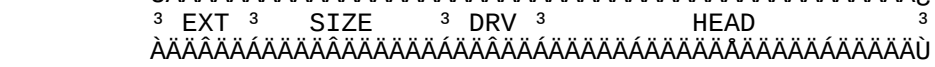
1F2 - sector count register. Used to specify number of sectors to transfer in multi-sector operations - 1, i.e., value of 1 means 2 sectors. WDC will accept more then one track worth of sectors, updating head and cylinder number correspondingly. During format a track operation, specifies number of sectors on track (0FFh is 255 sectors).

1F3 - sector number register. During format a track operation, specifies value of GAP 1 and GAP 3 (see description of MFM in 1.1.3) minus 3 bytes.

1F4 - 8 low bits of cylinder number. (Tech Help! 4.0 is wrong here)

1F5 - 2 high bits of cylinder number (bits 0-1 are used). Some never WDCs will accept more then 2 bits in this register, thus supporting hard disks with more then 1023 cylinders, but most BIOSes strip these high bits anyway.

1F6 - Sector/Drive/Head select.



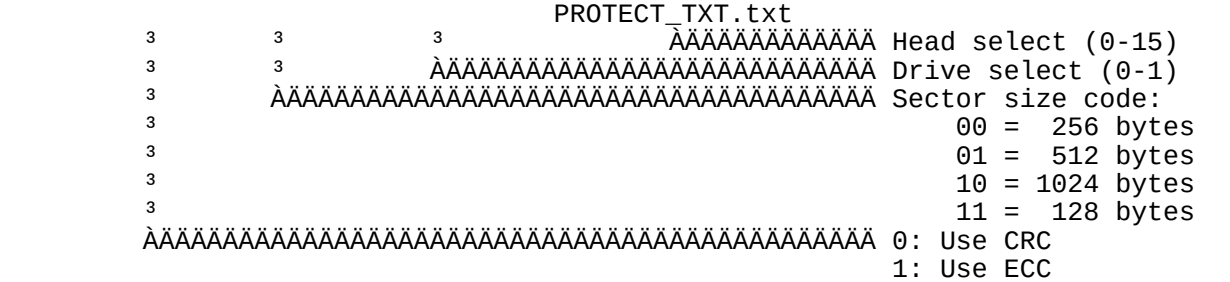


Fig 2.1b SDH (Sector/Drive/Head) register (1F6).

In original 82062 specification, 3 bits were reserved for head select field and 2 bits for drive select, but these signals were processed by external circuits, so it really does not matter.

1F7 - read: Status register

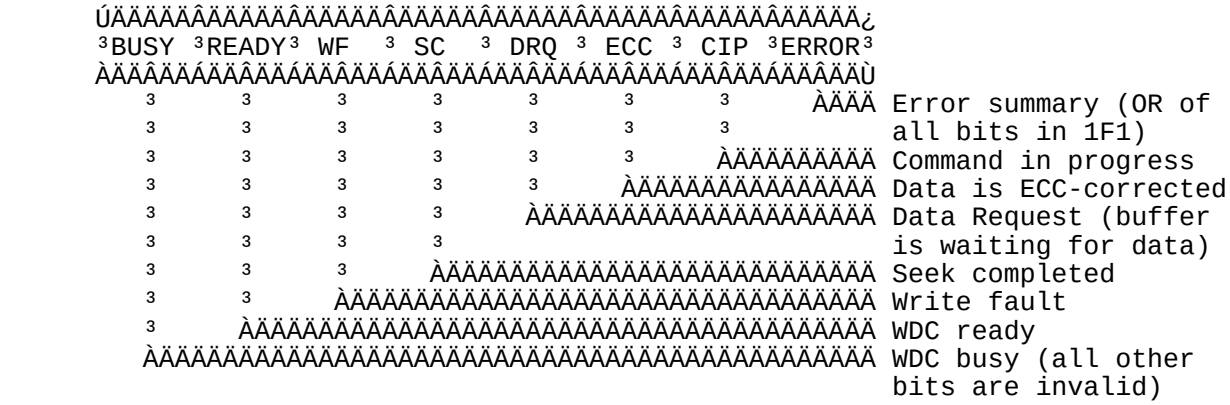


Fig 2.1c Status register (1F7)

1F7 - write: Command register. Command code is written here. Instruction set of AT FDC (which is superset of 82062), includes the following commands:

0	0	0	1	R3 - R0				Restore to track 0
0	1	1	1	R3 - R0				Seek
0	0	1	0	I	M	E	T	Read Sector
0	0	1	1	0	M	E	T	Write Sector
0	1	0	0	0	0	0	0	Scan ID
0	1	0	1	0	0	0	0	Write Format
1	0	0	1	0	0	0	0	Diagnose
1	0	0	1	0	0	0	1	Set drive parameters

R3-R0 determines drive step rate, which is 0.035+0.5*R ms for 5 MHz clock rate. (My experiments with AT WDC indicate that this value is ignored by

controller, which positions r/w head at maximum possible speed using Seek Completed hardware interface line).

I = 0 requests interrupt on DRQ goes active, as well as after completion of command, I = 1 disables interrupt on DRQ active.

M = 1 specifies multi-sector transfer, number of sectors is in sector count register.

E = 1 disables data correction and passes 4 bytes of ECC along with sectors data.

T = 0 enables transfer retries, i.e., if specified sector was not found after 6 revolutions, WDC performs automatic scan ID command, updates internal cylinder number, performs seek, if necessary, and repeats operation. T = 1 causes error if specified sector was not found after 2 revolutions.

3F6 - write: options register. Write 02h to take WDC IRQ 14 off system bus, 00h to resume normal operations. 04h resets controller.

WDC format command scheme is far less flexible then 8272 one, and permits to select manually sector number only. Although sector ID AM contains cylinder, head and size values as well, they are derived from WDC command registers and can't be explicitly specified. Because all WDC commands (except Scan ID command) have implied seek feature, these fields are difficult to modify.

Features of AT WDC, being obviously advantageous for normal computer operations, shrinks copy protection possibilities to extra/missing sector (see 1.3.1) and sector ordering (see 1.4.1) checks. ECC data correction permits to implement the third hard disk protection trick, data bit changed in specific position, which will be hidden by ECC.

All these marks can be both generated and verified using BIOS-level disk access, and so, chip-level access to WDC have no significant advantages that can justify lack of portability.

2.2 BIOS-level protection

This is obviously the basic level for hard disk based protection. Sufficient security from TSR software monitoring BIOS int 13h calls can be reached in this case by tracing int 13h till hard disk BIOS entry point (see example in Appendix B). Other reason for such elimination of watchers is possible necessity for accurate operation timing in interleave determination.

2.2.1 Interleave changes

This technique is very alike to the same floppy method described in 1.4.1. Unlike floppies, where sectors are as a rule arranged sequentially (interleave 1:1), hard disks sectors are often preciously arranged by low-level formatters to provide fastest possible data transfer rate. It is therefore more difficult to see interleave mark on hard disk then on floppy.

2.2.2 Changed sector numbers

Again, this is a twin of 1.3.1. Nevertheless, hard disks are ordinary tight packed with data sectors, so to add sector with non-standard number, one should remove one of data sectors. This situation can be easily spotted.

2.2.3 Unused disk areas

At the BIOS level, there are two unused areas on almost any hard disk: at the very beginning and at the very end of drive. First sector of each hard drive (at least, not SCSI or ESDI one) is occupied by partition table, while all other sectors on cylinder 0 track 0 are not used in IBM and Microsoft partitioning schemes. Still, this area can be occupied by proprietary partitioning software. For example, DiskManager (dmdrvr.bin) uses sector 0/0/8 for extended partition table, Olivetty MS-DOS starts first DOS partition at 0/0/2, etc.

Second unused area is the user diagnostic cylinder, which is located at the next to last disk cylinder on AT machines and on last cylinder on PS/2 ones (But they are still compatible, aren't they?) Anything written here has very little chances to last long, because any low-level disk test program has right to treat this track as it pleases. (Norton DiskTreet and Gibbson Research's SpinRite determine optimal interleave here, I believe).

2.3 DOS-level protection

BIOS-level protection on hard disk does not provide too much alternative, while DOS has convenient (and relatively portable) ways of accessing DOS partitions (int 25h/26h interface, ExtendedOpen (6Ch) in DOS 4.0+), which could be used for copy protection.

Structure of DOS filesystem is of common knowledge, so I will only briefly mention most significant parts of it.

Sector 0 is boot sector. It contains code that loads operation system and table describing partition characteristics, such as number of sectors, number of sectors per cluster, number of FATs, number of entries in root directory.

File allocation table (FAT), which follows, contains 12 or 16 bit value for each allocation unit (cluster), indicating number of the following cluster in chain. Usually, more then one (two) FATs are present on disk, thus providing data security. (Nevertheless, DOS does not treats FAT copies separately, but merely stores first FAT in additional areas duplicating any error occurred in 1st copy in all FATs, so security is really imaginatory).

Root directory (following last copy of FAT) contains 32-byte table (called directory entry) for each file in directory. Directory entry contains name of file, size and modification date of file and starting cluster number. Other directories use the same allocation methods as ordinary files.

2.3.1 Dependence on cluster number

Standard DOS tools don't permit control over cluster by cluster location of files, so this information can be used to encode program image and/or data. Starting cluster number of file can be obtained by CP/M-style call 11h (FindFirst via FCB). (CP/M calls can be hidden from most watchers by performing far call to location 0:0C0h with function code in CL instead of AL). Finding out numbers of other clusters requires browsing thru FAT (See example in Appendix C).

2.3.2 Unused (reserved) disk areas

Really, there is only one such area at offset 0Ch of directory entry (10 bytes long). Unfortunately, this area is often used by "DOS-compatible" operating systems. For example, Digital Research DOS uses this field to store file password, PC-MOS/386 (Software Links) stores here file owner ID, access rights and creation date/time.

Another reserved area, which can exist on disk, is the remainder of last sector in FAT 1, which is preserved by DOS (and even copied to all other copies of FAT).

2.3.3 Unused (allocation unit rounded) disk areas

Because DOS allocates disk space in clusters (which contain 2^N sectors), while file size is measured in bytes, most files have unused (and ordinarily invisible on file system level) tail, which can be used for protection purposes. Curious bug in DOS, which permits seeks (DOS function 42h) beyond end-of-file, makes access to this file tail easy even for high-level languages (See Appendix D).

3. Motherboard and system BIOS based protection

Other part of PC, which, along with floppy and hard disk, is always

available for copy protection, is motherboard. Although motherboards are now manufactured in many thousands series, almost each of them has (or can acquire) individual qualities.

3.1 Data-based tricks

Each motherboard has it's own BIOS. So it can be used for copy protection, although we can expect this primarily from hardware manufactures and not from independent software developers. (So the stuffing system BIOS with a lot of cryptic undocumented functions and tables, which is a dear occupation of big blue, is a sort of copy protection).

Other data area available on motherboards, is non-volatile CMOS memory, which as a rule has a lot (half a dozen bytes) unused space in it. Some chip sets (e.g., from C&T) can map part of CMOS memory into system ROM address space, thus creating appearance of "software reprogramming system ROM BIOS".

3.2 Time-based tricks

More interesting protection marks are ones based on internal timing of system board. Three basic subsystems are available for such measurements: CPU, memory subsystem, I/O subsystem (See appendix E). Differences can be actually surprisingly large (See table 3.2).

System	CPU	Memory	I/O
25 MHz 80386 (A)	35972	24576	47292
25 MHz 80386 (B)	35972	24576	49154
20 MHz 80386	44958	30112	59990
12 MHz 80286	3544	41018	46646

Table 3.2 Motherboard marks of four different machines.

Values shown in table 3.2 are an average for multiply runs of program from Appendix E. Actual values differ from mean ones by about ñ4. Systems (A) and (B) have had two sequential manufacturer's numbers, but still can be easily distinguished by I/O mark (Port 0Ch, 1st DMA controller was used). It should be also noted, what I/O mark is very sensitive to CPU mode of operation (real vs. protected), while CPU and memory marks are almost equal in both modes.

Glossary

82062	Intel WDC
8237A	Intel DMA controller
8253	Intel timer chip
8259A	Intel interrupt controller
8272A	Intel FDC
BC	IBM bad track error, bit in ST2
BIOS	Basic Input/Output System
CB	Controller busy, bit in MSR
Cluster	Allocation unit in FAT filesystem, See 2.3
CM	Control mark, bit in ST2
CPU	Central Processing Unit, e.g., 8086, 80286, . . .
CRC	Cyclic Redundancy Code
DxB	Drive x busy, bit in MSR
DD	Data error, bit in ST2
DE	Data error, bit in ST1
DIO	Data Input/Output, bit in MSR
DMA	Direct Memory Access

Double Density	See MFM
DR	Data Register of 8272A. See 1.2.1
DTL	Data transfer length, 8272A command parameter. See 1.2.3
EC	Equipment check, bit in ST0
ECC	Error correction code
EN	End of track error, bit in ST1
EOT	End of track, 8272A command parameter. See 1.2.3
FAT	File allocation table. See 2.3
FDC	Floppy disk controller. See 1.1
FM	Frequency modulation. See 1.1
FT	Fault, bit in ST3.
GPL	Gap length, 8272A command parameter. See 1.2.3
HDC	Hard Disk Controller
HLT	Head load time, 8272A command parameter. See 1.2.3
HUT	Head unload time, 8272A command parameter. See 1.2.3
IAM	Index address mark. See 1.1.3
IC	Interrupt code, 2 bits in ST0
IRR	Interrupt request register of 8259A.
IRQ	Interrupt request signal
JV	Joint Venture, "Sovmestnoe predprijatie"
Kb	Kilobyte, 1024 bytes
KBPS	Thousands of bits per second
LSB	Least significant bit
MA	Missing address mark, bit in ST1
MD	Missing data address mark, bit in ST2
MFM	Modified frequency modulation. See 1.1
MSB	Most significant bit
MSR	Main Status Register of 8272A. See 1.2.1
ND	Sector no found, bit in ST1 or non-DMA, 8272A command parameter. See 1.2.3
NDM	non-DMA transfer, bit in MSR
NR	Not ready, bit in ST0
NW	Write protect error, bit in ST1
OR	Overrun error, bit in ST1
PLL	Phase locked loop, circuit in FDC. See 1.4.2
RDY	Ready, bit in ST3
RPM	Revolutions per minute
RQM	Request for master, bit in MSR
RWC	Reduce write current, technique used in MFM recording
SC	Sector count, 8272A command parameter. See 1.2.3
SE	Seek ended, bit in ST0
SH	Scan Hit, bit in ST2
Single Density	See FM
SN	Scan not satisfied, bit in ST2
SRT	Step rate interval, 8272A command parameter. See 1.2.3
ST0, ST1, ST2, ST3	Status bytes 0-3 of 8272A. See 1.2.2
STP	Step value, 8272A command parameter. See 1.2.3
T0	Track 0, bit in ST3
TC	Terminal count - signal raised by 8237A when channel transfer count reaches zero
TS	Two sided, bit in ST3
WC	Cylinder address error, bit in ST2
WDC	Winchester disk controller
WP	Write protect, bit in ST3

Appendix A. Simple 8272A program

Program sample text is in test_fdc.c. It requires Turbo C 2.0 and Turbo Assembler (any version) to compile. Minimum hardware required to run this sample is 8 MHz/0 WS 80286 machine equipped with at least one floppy drive, floppy drive to run test on is specified by program argument (0 for A:, 1 for

B:). This test can be used on secondary FDC as well, provided that FDC_BASE in source file is changed to 0x370. 'T' command (Analyse a track) will work incorrectly with 300 RPM drives (360K, 720K, 1.44M) until REVOLUTION_TIME in program source will not be chnged to 200L*2*1193 (but when it will work incorrectly with 1.2M drives).

Appendix B. Simple HDC exerciser program

Program sample text is in file hd_scan.c, which again requires TC 2.0 and TASM to compile. It accepts BIOS drive ID (80 for 1st drive, 81 for second). If this test hangs after printing the message:
"Cyls = XXX, Heads = XXX, Sectors = XXX",
all disk caching programs should be removed from memory prior to running this test.

Appendix C. How to find out cluster number

Program text is in file cluster.c, again TC 2.0 and TASM are required to compile it. Fully qualified file name (i.e., including drive and directory, even for file in current directory) should be specified as parameter. Although this sample was tested under MS DOS 3.20, 3.30, 4.00 and 5.00 (beta), it may not work under other MS DOS versions or compatible systems, because it relies on structure of undocumented DiskInfoBlock.

Appendix D. Accessing file tail

Program text is in file tail.c. Although any ANSI compiler should accept it, this sample relies on undocumented behavior of seek() function and may not work with compilers other then TC 2.0.

Appendix E. How to distinguish motherboards

Program text is in file sysboard.c, TC 2.0 and TASM required to compile. Current value of INSTRUCTIONS parameter (16384) will cause time counter to overflow on anything less then 20 MHz 80386.