

MFM - Disks issues, infos , problems

Background:

The [ST506-Interface](#) was designed in 1982 by the company Seagate for their 5 ¼-inch drive ST506 (5.4 MB) , ST412 (10.1 MB) and ST225(20.4 MB) and every well-known computer manufacturer was using this technology. The ST506-Interface is working based on the [MFM](#)(Modified Frequency Modulation) recording method:

But:

Each manufacturer did have its own implementation according to the slogan :

Be 100% incompatible with any other manufacturer

References:

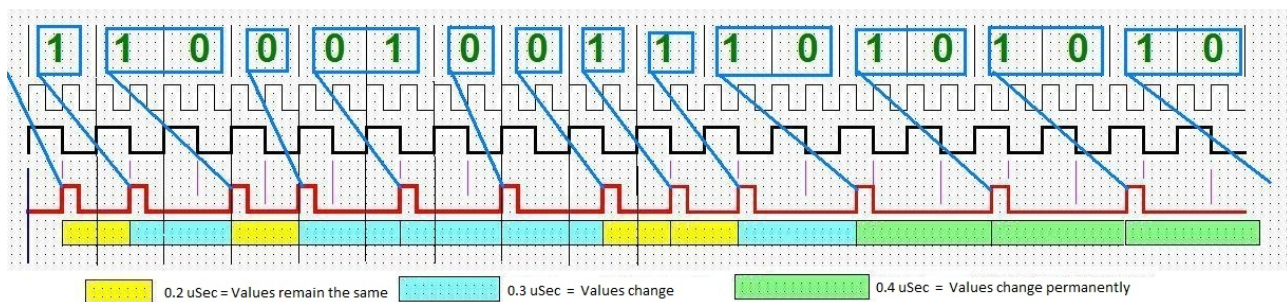
<https://github.com/pdp11gy/SoC-HPS-based-MFM-disk-emulator>

Download and unzip the file **MFM-disk_Emulator_SoC.zip**

<http://www.pdp11gy.com/>

<http://www.minuszerodegrees.net/manuals/Seagate/Seagate%20ST506%20-%20Service%20Manual%20-%20May82.pdf>

MFM timing overview



The MFM transfer bandwidth is defined as 5 MHz = 0.2 uSec. The FPGA clock is running at 80 MHz, = 0,0125 uSec. which is 16 times higher. This was necessary to prevent a chatter, primarily with the MFM Encoder, also implemented in the same way at the RL RL01 / RL02 emulator project. The entire design runs synchronously in real time based on the 80MHz clock. Since the design runs in real time, MFM decoding can be done “on the fly”. It’s a real time design, based on FPGA CyclonV

Requirements :

During development, I had chosen a method to write a well-defined pattern on the disk. This method was very helpful for the RL01 / RL02 emulator development, so I did use this method in the development of the MFM disk emulator as well.

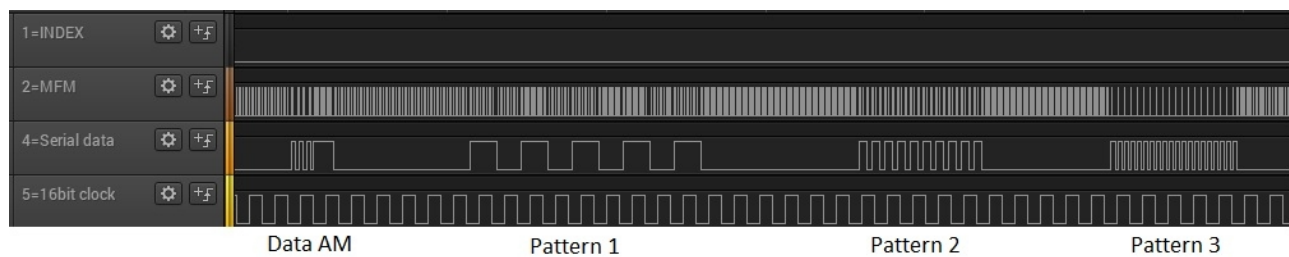
Abstract, used Pattern:

- 1) DEC 00 255 , HEX 00 FF , BIN 0000 0000 FFFF FFFF
test change from short to long cycle
- 2) DEC 51 , HEX 33 , BIN 0011 0011
test long cycle to long cycle
- 3) DEC 85 , HEX 55 , BIN 0101 0101
test verylong to verylong cycle

I used a RT-11 basic (from 1985) program as follow and copied the output file to a MFM disk.

```
5 A$="" \ B$="" \ PRINT "GENERATE TEST-PATTERN"
6 FOR I=1 TO 5 \ A$=A$+CHR$(0) \ NEXT I
11 FOR I=1 TO 5 \ A$=A$+CHR$(255)+CHR$(0) \ NEXT I           Pattern 1
18 FOR I=1 TO 5 \ A$=A$+CHR$(0) \ NEXT I
21 FOR I=1 TO 5 \ A$=A$+CHR$(51) \ NEXT I                   Pattern 2
28 FOR I=1 TO 5 \ A$=A$+CHR$(0) \ NEXT I
31 FOR I=1 TO 5 \ A$=A$+CHR$(85) \ NEXT I                   Pattern 3
38 FOR I=1 TO 5 \ A$=A$+CHR$(0) \ NEXT I
41 FOR I=1 TO 3 \ A$=A$+CHR$(73)+CHR$(146)+CHR$(36) \ NEXT I
48 FOR I=1 TO 5 \ A$=A$+CHR$(0) \ NEXT I
51 FOR I=1 TO 3 \ A$=A$+CHR$(35)+CHR$(145)+CHR$(220) \ NEXT I
58 FOR I=1 TO 5 \ A$=A$+CHR$(0) \ NEXT I
61 FOR I=1 TO 10 \ A$=A$+CHR$(128) \ NEXT I
68 FOR I=1 TO 5 \ A$=A$+CHR$(0) \ NEXT I
71 FOR I=1 TO 3 \ A$=A$+CHR$(231)+CHR$(156)+CHR$(243) \ NEXT I
78 FOR I=1 TO 5 \ A$=A$+CHR$(0) \ NEXT I
81 FOR I=1 TO 3 \ A$=A$+CHR$(99)+CHR$(140)+CHR$(241) \ NEXT I
91 FOR I=1 TO 10 \ A$=A$+CHR$(127) \ NEXT I
98 FOR I=1 TO 5 \ A$=A$+CHR$(0) \ NEXT I
510 A$=A$+CHR$(10)+CHR$(13)
520 A$=A$+" MFM-DISK EMULATOR, (C) REINHARD HEUBERGER "
525 A$=A$+CHR$(10)+CHR$(13)
540 A$=A$+" SOC/HPS VERSION WWW.PDP11GY.COM "
545 A$=A$+CHR$(10)+CHR$(13)
550 PRINT "A-STRING-LAENGE: ";LEN(A$)
610 FOR I=1 TO 19
620 A$=A$+CHR$(255)+CHR$(0)
630 NEXT I
635 A$=A$+CHR$(0)
636 PRINT "A$ STRING-LAENGE: ";LEN(A$)
640 FOR I=1 TO 125 \ B$=B$+CHR$(0) \ NEXT I
642 B$=B$+CHR$(255)+CHR$(255)+CHR$(0)+CHR$(0)+CHR$(255)+CHR$(255)
650 FOR I=132 TO 254 \ B$=B$+CHR$(0) \ NEXT I
660 B$=B$+CHR$(0)
691 PRINT "B$ STRING-LAENGE: ";LEN(B$)
699 goto 800
700 OPEN "DU0:PATT4.TXT" FOR OUTPUT AS FILE #1
720 FOR I=1 TO 5000
730 PRINT #1,A$;
731 PRINT #1,CHR$(0);
740 PRINT #1,B$;
741 PRINT #1,CHR$(0);
750 NEXT I
760 CLOSE #1
770 PRINT "DONE"
800 END
```

Of course you can also implement the program in C (see my source code), but at these time it did not exist. The following figure shows the timing from pattern 1 to 3 and Data AM

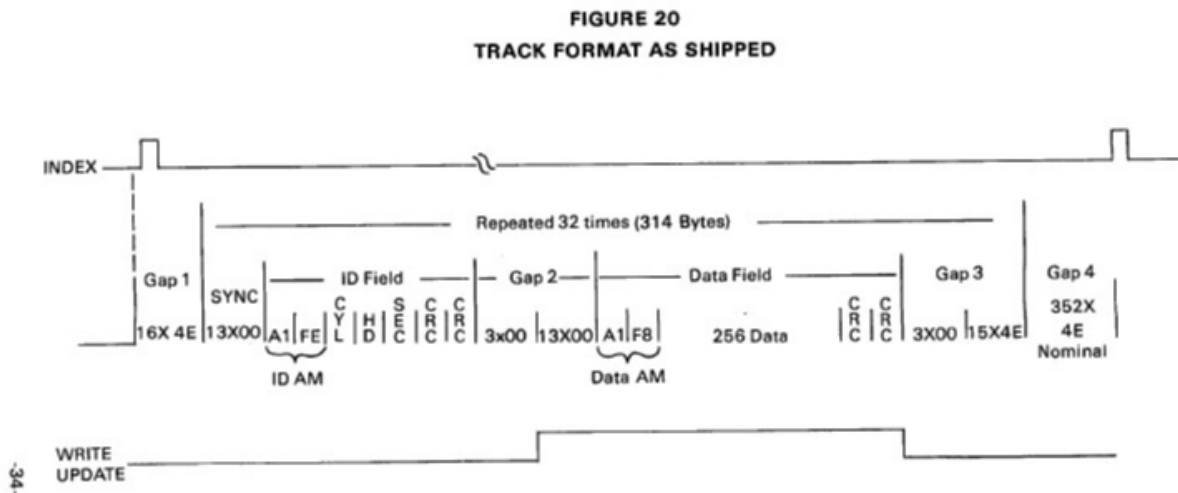


The folder software /READC/contains the program readc. This program reads a cylinder and a track with head 1 and saves the data to the SD card. Then you can view the file with a HEX editor. Here is an example where you can find the Pattern 1 to 3 again:

00009F30	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
00009F40	12	12	12	12	12	12	12	12	12	12	12	12	12	12	41	21A!
00009F50	E4	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	a\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$
00009F60	3F	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	CA	80	?yyyyyyyyyyyyyyËE
00009F70	21	F3	FF	25	2F	3F	FF	FF	FF	00	00	00	00	00	00	00	!óy%/?ýýý.....
00009F80	00	00	00	00	00	00	A5	F8	00	00	00	00	00	FF	00	FFÆø.....ÿ.ÿ
00009F90	00	FF	00	FF	00	FF	00	00	00	00	00	00	33	33	33	33	.ÿ.ÿ.ÿ.....333.
00009FA0	33	00	00	00	00	00	55	55	55	55	55	00	00	00	00	00	3.....UUUUU.....
00009FB0	49	92	24	49	92	24	49	92	24	00	00	00	00	00	23	91	I' \$I' \$I' \$.# \'
00009FC0	DC	23	91	DC	23	91	DC	00	00	00	00	00	80	80	80	80	Ü# 'Ü# 'Ü.....eeee
00009FD0	80	80	80	80	80	80	00	00	00	00	00	E7	9C	F3	E7	9C	eeeeee.....çæóçæ
00009FE0	F3	E7	9C	F3	00	00	00	00	00	63	8C	F1	63	8C	F1	63	óçæó.....cÊñcÊñc
00009FF0	8C	F1	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	00	00	00	00	CÊñ.....
0000A000	00	0A	0D	20	20	4D	46	4D	2D	44	49	53	4B	20	45	4D	... MFM-DISK EM
0000A010	55	4C	41	54	4F	52	2C	20	28	43	29	20	52	45	49	4E	ULATOR, (C) REIN
0000A020	48	41	52	44	20	48	45	55	42	45	52	47	45	52	20	20	HARD HEUBERGER
0000A030	0A	0D	20	20	20	20	20	20	53	4F	43	2F	48	50	53	20	.. SOC/HPS
0000A040	56	45	52	53	49	4F	4E	20	20	57	57	57	2E	50	44	50	VERSION WWW.PDP
0000A050	31	31	47	59	2E	43	4F	4D	20	20	20	20	20	20	0A	0D	llGY.COM ..
0000A060	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	ÿ.ÿ.ÿ.ÿ.ÿ.ÿ.ÿ.ÿ.
0000A070	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	FF	00	ÿ.ÿ.ÿ.ÿ.ÿ.ÿ.ÿ.ÿ.
0000A080	FF	00	FF	00	FF	00	00	00	00	00	00	00	00	00	00	00	ÿ.ÿ.ÿ.....
0000A090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000A0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000A0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000A0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000A0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000A0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000A0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000A100	00	00	00	00	00	FF</											

Very important is the field **data AM** (A5 F8 @ 9F86). That's the section now where open points/questions begin.

With reference to the SEAGATE ST-506 Manual, the disk format is Pre-configured as in the following picture:



- NOTES:**
1. Nominal Track Capacity = 10416 Bytes
 2. Total Data Bytes/Track = 256 x 32 = 8,192
 3. Sector interleave factor is 4. Sequential ID Fields are sector numbered 0, 8, 16, 24, 1, 9, 17, 25, 2, 10, 18, 26,...etc.
 4. Data Fields contain the bit pattern 0000 as shipped
 5. CRC Fire Code = $x^{16} + x^{12} + x^6 + 1$
 6. Bit 7 of Head Byte ID Field equals 1 in a defective sector (Cylinder 0 is error free)
 7. Bit 5 of Head Byte reserved for numbering cylinders greater than 256
 8. Bit 6 of Head Byte reserved for numbering cylinders greater than 512

Capacity

Nominal Track Capacity: = 10416 (Byte)

Total Data Bytes/Track	= 256	x 32 =	8192
SYNC = 13x00	= 13	x 32 =	416
ID AM = 2 Byte	= 2	x 32 =	64
CYL/HD/SEC = 3 Byte	= 3	x 32 =	96
Header-CRC = 2 Byte	= 2	x 32 =	64
Gap2 3 + 13 = 16Byte	= 16	x 32 =	512
Data AM = 2 Byte	= 2	x 32 =	64
Data-CRC = 2 Byte	= 2	x 32 =	64
Gap3 1of2 = 3x00	= 3	x 32 =	96
Gap3 2of2 = 15x4E	= 15	x 32 =	480
<hr/>			
Einmalig dazu:	SECTOR: 314	TRACK: 10048	CYLINDER: 40192
	Gap1 16x4E	= 16	64
	Gap4 352x4E	= 352	1408
<hr/>			
	(Byte)	10416	41664
	(Bit)	83328	333312
	(Word)	5208	20832
<hr/>			

Understanding and analysis

The interface and the corresponding signals were described in detail by the company Seagate and were widely respected. It looks quite different at data and timing format. Everything here is incompatible. Each manufacturer has guaranteed implemented his own track and data format which was generated with their own low-level format program. The following differences exist:

>> CRC algorithm is different, such as different preset value.

>> Track format: ID AM differently.

>> Track format: DATA AM differently.

>> SYNC character differently.

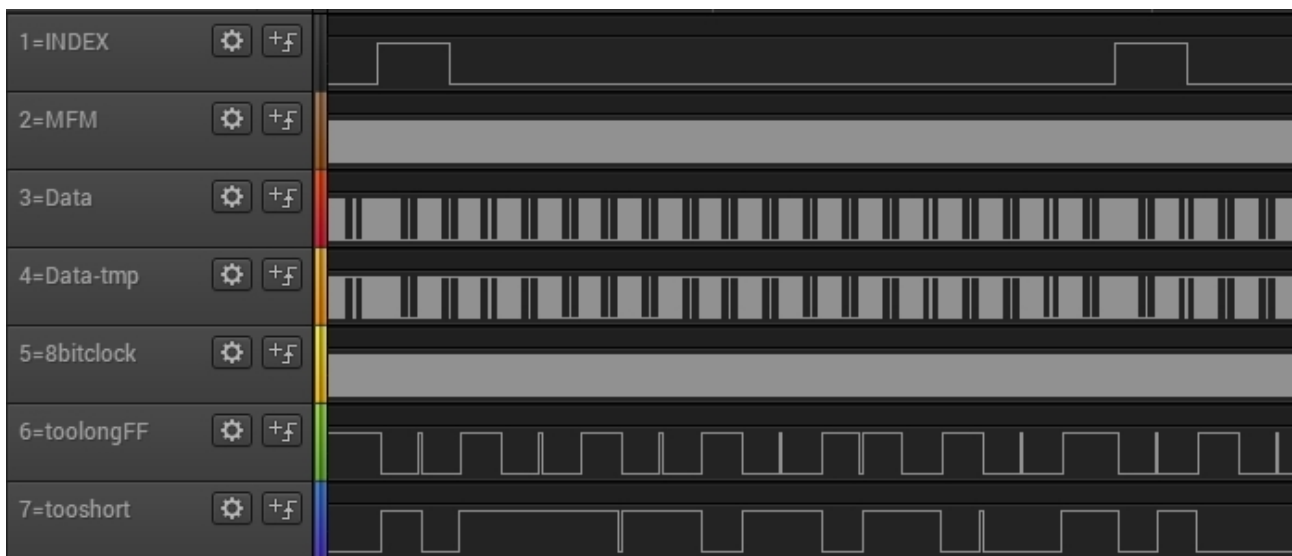
Even the same manufacturer, for example, DEC. There were different formats used. A disk, formatted with the RQDX-1 controller Disk could not be used in a RQDX-3 environment.

Problems

At the moment I am only able to work reasonably with a PDP-11/23 / RQDX-1 and RD51. The RQDX-3 is broken, my Schneider PC is broken and my ST225 disk is also broken. (I hope to get my SANYO PC up and running soon).

In a PDP-11/23 /RQDX-1 environment, I found strange things concerning the timing outside the data field. I found too short and too long MFM gaps.

Example, logic analyser:



The MFM decoder (MFM-disk_Emulator_SoC/my_Verilogs/MFM_gap_DECODER_V1_0.v) is able to detect too short and too long MFM gaps. If a wrong gap is detected, then a flipflop is switching. Usually the following times are correct :

short = 0,2 uSec

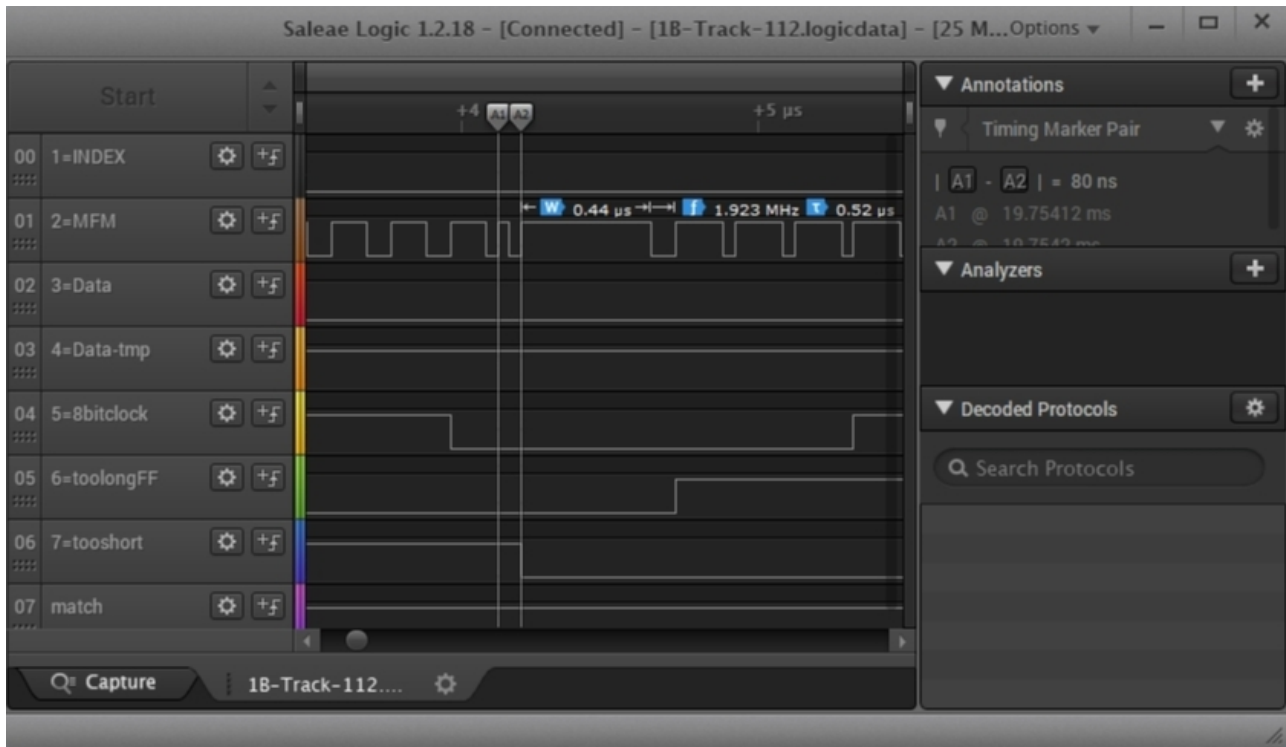
long = 0.3 uSec

verylong = 0.4 uSec

There may be small deviations but in this case too short MFM gaps with 80nSec and too long MFM gaps with 0.52 or 0.72 uSec could be found.

This symptom confuses the timing with the result that the data FlipFlop sometimes tilts uncontrolled. Thus the data are wrong and the boundaries of the byte counter are also no longer correct .

Sometimes a too long cycle comes direct after a too short cycle like in following picture:



Note : The symptom is not visible in the data field.

I don't know exactly how to handle this cycles. At the moment, I switch the data to low on a too long cycle detection. Important to know : I could not see this peculiar symptom in the PC environment (My problem: At the moment, I don't have any PC related reference hardware).

The way for a solution:

The indicator for data field is the **end** of the **data AM** (A5 F8) . **So you can find the beginning of the data in the sector.** But unfortunately each manufacturer has always used a different data AM pattern.

Solution:

The **MFM_gap_DECODER_V1_0.v** will trigger on detection of a Data AM pattern. These decoder makes real-time MFM-decoding with serial and 8 bit parallel output and will align it to byte boundary after detecting the 16 bit Data AM pattern . With these possibilities a .img file can be created in real-time to be also compatible with the SIMH project.

Handicap:

For each manufacturer, you have to analyze it individually to get the proper Data AM pattern. I can not do that alone! Any hint and help is welcome

Note: It is intended to create for each disk-type its own configuration file. This can be modified with any standard editor.

To verify the detection of a Data AM pattern, use the program soc_mfm_beta/MFM/readc.
Exmple:

```
root@socfpga:~/MFM# ./readc
```

```
***** MFM-DISK   READER   @ Soc/HPS *****
READ one Cylinder+Track and save it to SD card
DE10-Nano ST-506/412/225   Beta Version
*****
(c) Reinhard Heuberger WWW.PDP11GY.COM
```

```
>>>>> DEBUG-MODE = ON <<<<<<
>>>> Device Type = ST506 <<<<
*****
***** +Test-Mode *****
*****
```

```
Anzahl der Cylinder: 153
Drive_select #0 DRV_SLCTD = LOW
Drive_select #1 DRV_SLCTD = LOW
Drive_select #2 DRV_SLCTD = HIGH
READY =        HIGH
SEEK_cmplt = HIGH
TRACK_0 =      LOW
DRV_SLCTD = HIGH
Drive = ready
Drive is NOT @ home
Drive positioned to home
```

```
Cylinder - nummer eingeben: 112
```

```
Trigger DataAM , (4Hex, like A5F8) :A5F8
```

```
Cylinder: 112 ,Trigger DataAM: lsb : 0xA5  msb: 0xF8
```

```
***** Step to Cylinder 112  done *****
```

```
Select Head 1 ... 2 ... 3 ... 4
```

```
found: DataAM_msb 0xA5  DataAM_lsb 0xF8  @ 529  Nr.: 1  Gap: 530
found: DataAM_msb 0xA5  DataAM_lsb 0xF8  @ 1312  Nr.: 2  Gap: 783
found: DataAM_msb 0xA5  DataAM_lsb 0xF8  @ 1882  Nr.: 3  Gap: 570
found: DataAM_msb 0xA5  DataAM_lsb 0xF8  @ 2453  Nr.: 4  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8  @ 3024  Nr.: 5  Gap: 571
```

```
.
.
.
```

```
found: DataAM_msb 0xA5  DataAM_lsb 0xF8  @ 41303  Nr.: 66  Gap: 571
```

```
Save track@head1 data-image to SD-Card into file:  ST506-Track-image_112.img
```

```
Save 1 cylinder data to SD-Card into file:  ST506-cylinder_112.mfm
```

```
***** Select Head 1 and loop *****
```

```
Save 1 track@head1 data to SD-Card into file:  ST506-Track_head1_112.mfm
```

```
Press RESET/Button-1 for exit, Reconfig/Button-2 for restart
```

```
^C
```

Info: I could not see the too long/too short symptom on a disk in a Schneider PC

Any hint and help is welcome

Would be nice if someone can get Data AM pattern and disk data from another vendor.
Maybe you can also find the data in the source listing of the low level format program or
use the method with the test-pattern and a HEX edit as explained on page 2 and 3 .
If there is no other way, unfortunately the data has to be recorded with a logic analyzer.

Logic analyser connections:

8 test pins are configured from the Arduino Uno R3 Expansion Header . See DE10-Nano
user manual, chapter 3.6.3.

Arduino_IO2 PIN_AG10	Arduino IO2 = Test_1
Arduino_IO3 PIN_AG9	Arduino IO3 = Test_2
Arduino_IO4 PIN_U14	Arduino IO4 = Test_3
Arduino_IO5 PIN_U13	Arduino IO5 = Test_4
Arduino_IO6 PIN_AG8	Arduino IO6 = Test_5
Arduino_IO7 PIN_AH8	Arduino IO7 = Test_6
Arduino_IO8 PIN_AF17	Arduino IO8 = Test_7
Arduino_IO9 PIN_AE15	Arduino IO9 = Test_8

For comments and questions, please contact me.
Reinhard Heuberger
INFO@pdp11gy.com