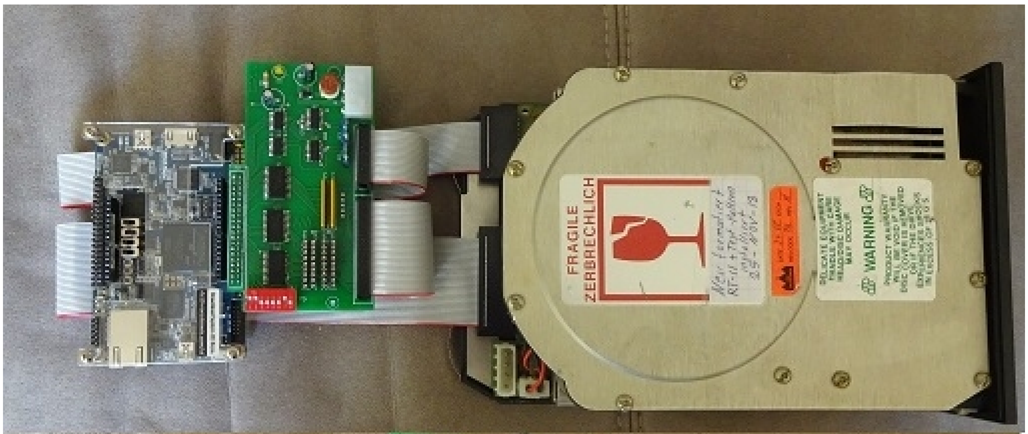
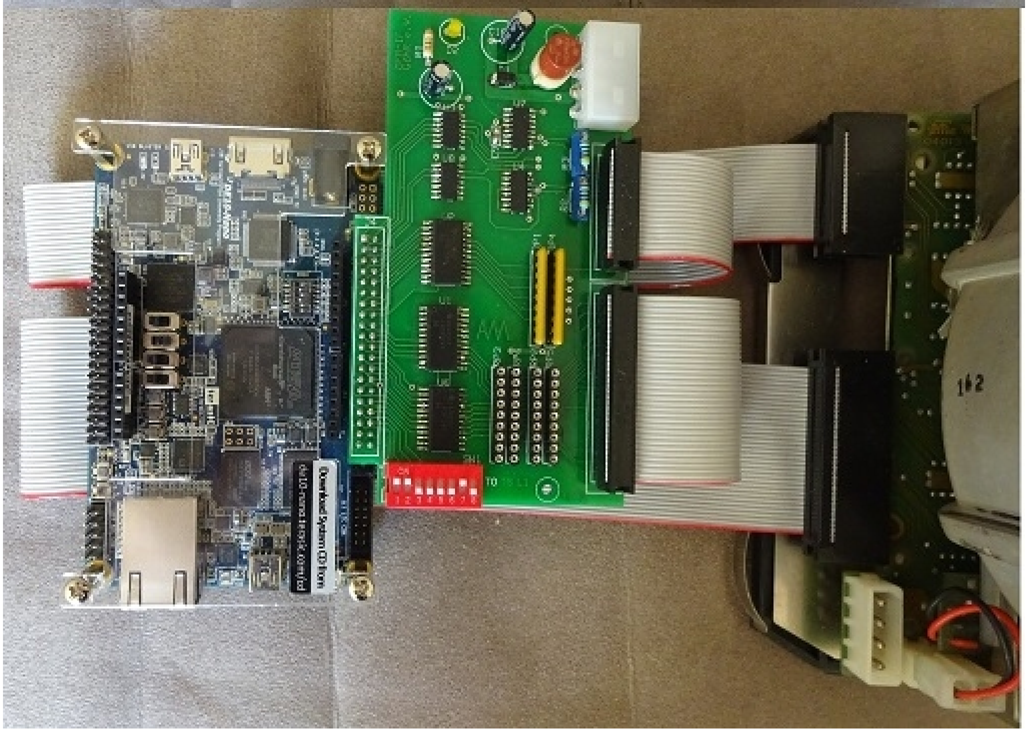
MFM DISK-DRIVE EMULATOR + READER

User Manual for the DE10-Nano board

Version 1.0





MFM-disk emulator interface connected to DE10-Nano board
SoC/HPS environment: Cyclone V FPGA + ARM Cortex-A9 CPU.

Read and analyze ST506 based MFM disk drives
Cloning MFM disk drives
Emulate MFM disk drives
Realtime MFM decoding to support .DSK data format
Open FPGA-SoC-Linux environment
SoC/HPS based disk emulator

CONTENTS

Chapter 1

General

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.

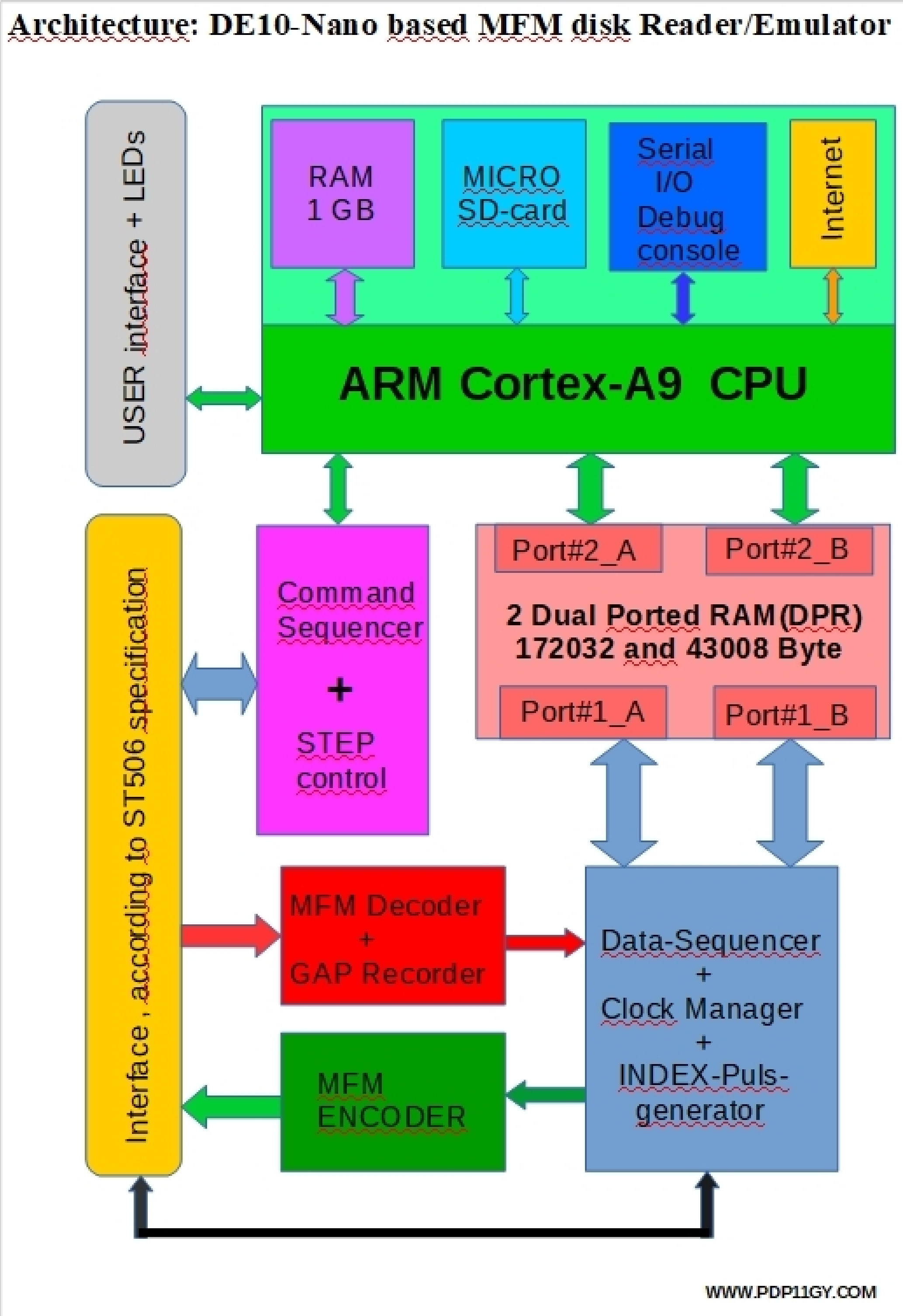
Implementation, 2019

My design is based on a SoC/FPGA environment as the DE10-Nano board. Currently, the ST506, ST412 and ST225 disks are supported . I decided to use the DE10-Nano Development Board, configurated with an Altera Cyclone® V SoC FPGA , Typ: 5CSEBA2317 and with integrated ARM Cortex-A9 CPU . In my design, the MFM data are completely decoded in Real -Time, effectively "on the fly" and stored on a Micro-SD card. Therefore a simple HEX-Editor for example can be used for viewing the received data.

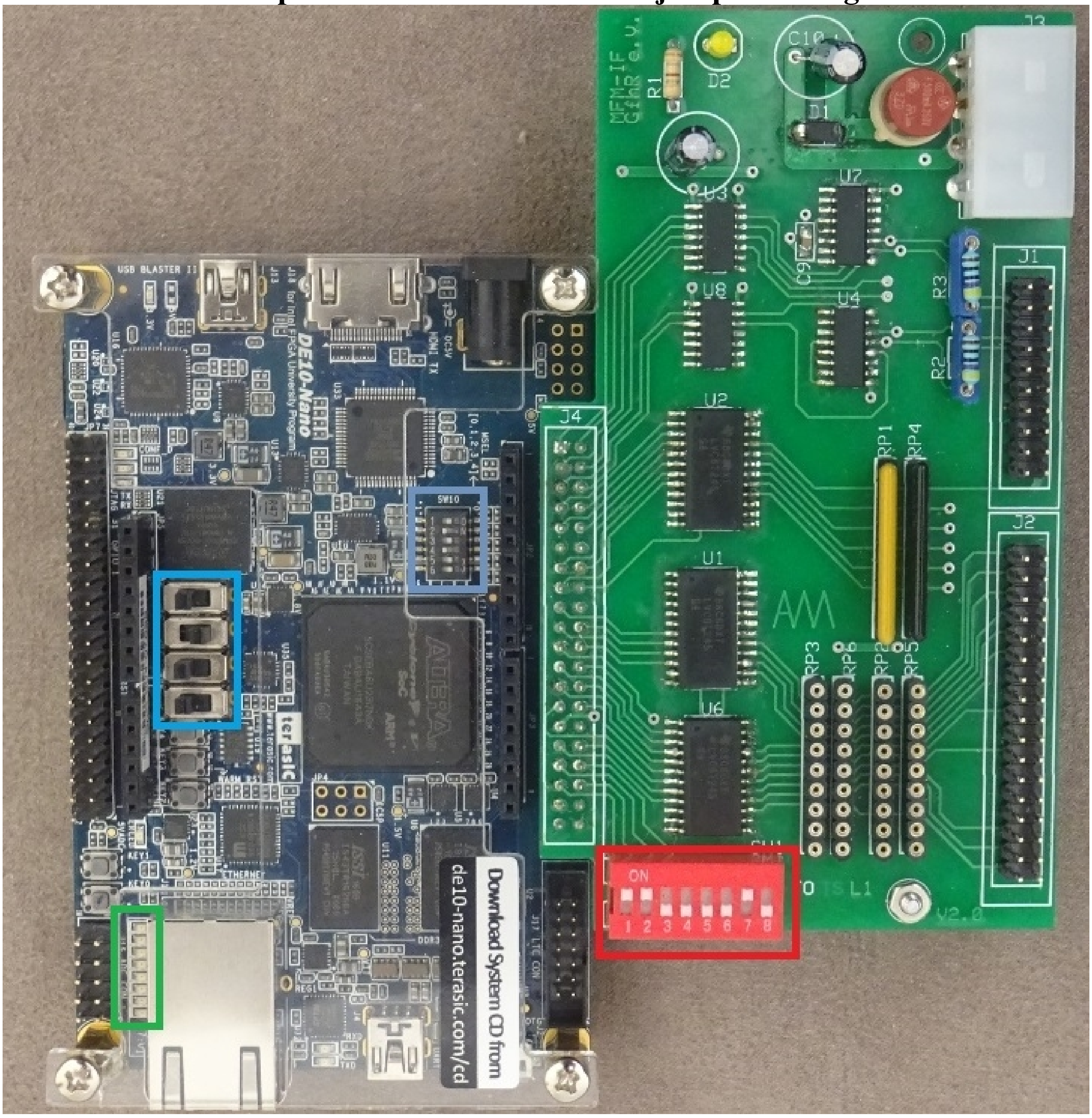
Features

- Synchronous design based on 80Mhz clock
- **Real Time** MFM DEcoding / ENcoding
- Output files:
  .mfm = decoded MFM data including Header/Servo/CRC information
  .dsk = extracted and decoded raw user data, also usefull for SIMH.
  .**gap** = MFM gap data, necessary in emulator mode.
- Currently support of ST506/412/225 disk drive with a capacity of 6.38/12.76/20.0MB and can be expanded at any time.
- The disk rotation speed is determined automatically and the additional disk parameters can be easily edited with a text editor.
- Open FPGA-SoC-Linux environment.
- Standard programming environment: C/C++ under Linux
- **No** extra tools required.

Next Page: Block diagram architecture overview


Architecture: DE10-Nano based MFM disk Reader/Emulator
USER interface + LEDs
RAM
1 GB
MICRO
SD-card
Serial
I/O
Debug
console
Internet
ARM Cortex-A9 CPU
Interface , according to ST506 specification
Command
Sequencer
+
STEP
control
Port#2_A
Port#2_B
2 Dual Ported RAM (DPR)
172032 and 43008 Byte
Port#1_A
Port#1_B
MFM Decoder
+
GAP Recorder
Data-Sequencer
+
Clock Manager
+
INDEX-Puls-
generator
MFM
ENCODER
WWW.PDP11GY.COM

Chapter 2 : Interface board + jumper settings



DE10-Nano: The four slide switches (page 26, User_manual):
switch 0 : ON=Clone-Mode OFF=EMULATOR Mode
switch 1 : Type of ENCODER output: ON = mfm output,
OFF= gap recorded output (recommended).
Button 2 and 3 : Reconfigure and Reset/Restart
De0-Nano-SoC DIP switch (SW10) configuration, see page 12 @ User_manuel

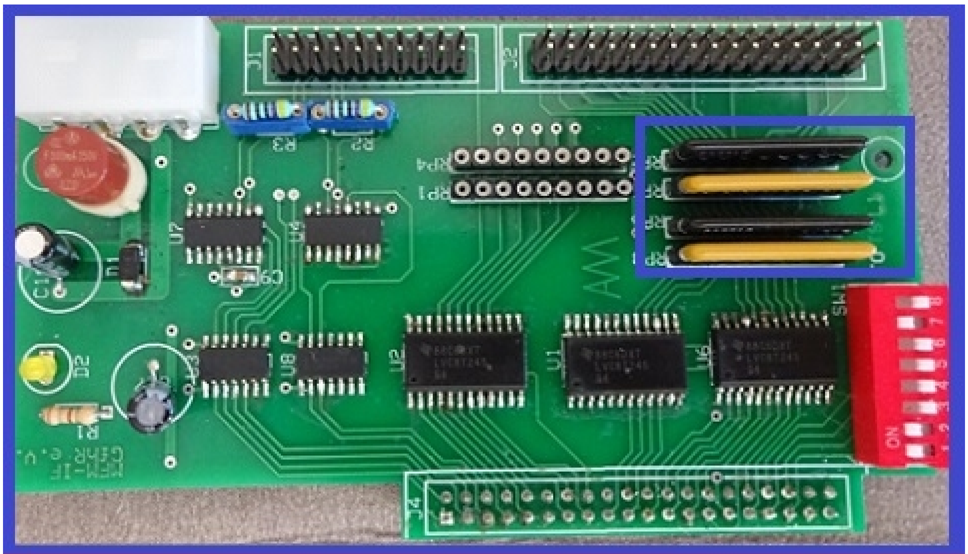
LED's : 0 = **heartbeat LED** (schould be blinking)
1 = CLONE Mode, 2 = CLONE-Mode STEP
3 = Interface enable 4 = Index-Pulse
5 = EMULATOR-Mode : Write
6 = EMULATOR-Mode : STEP
7 = EMULATOR Mode

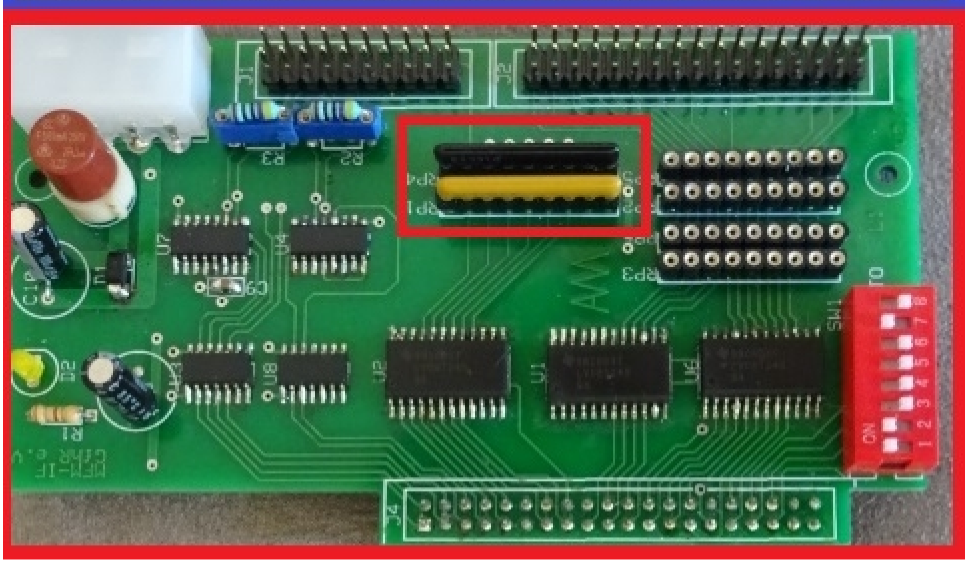
Interface-board: 8 switches :

Switch 1: ON: LED Debug info OFF=Pattern
Switch 2 : Debug Mode ON/OFF
Switch 3-4: Unit number_
Switch 5-8 : drive typ , 16 possibilities (0 to F)
0-0-0-0 = disk_drive #0 (ST506)
0-0-0-1 = disk_drive #1 (ST412)
0-0-1-0 = disk_drive #2 (ST 225)
until: 1-1-1-1 = disk_drive #15 (= F)

Interface board: Operating modes

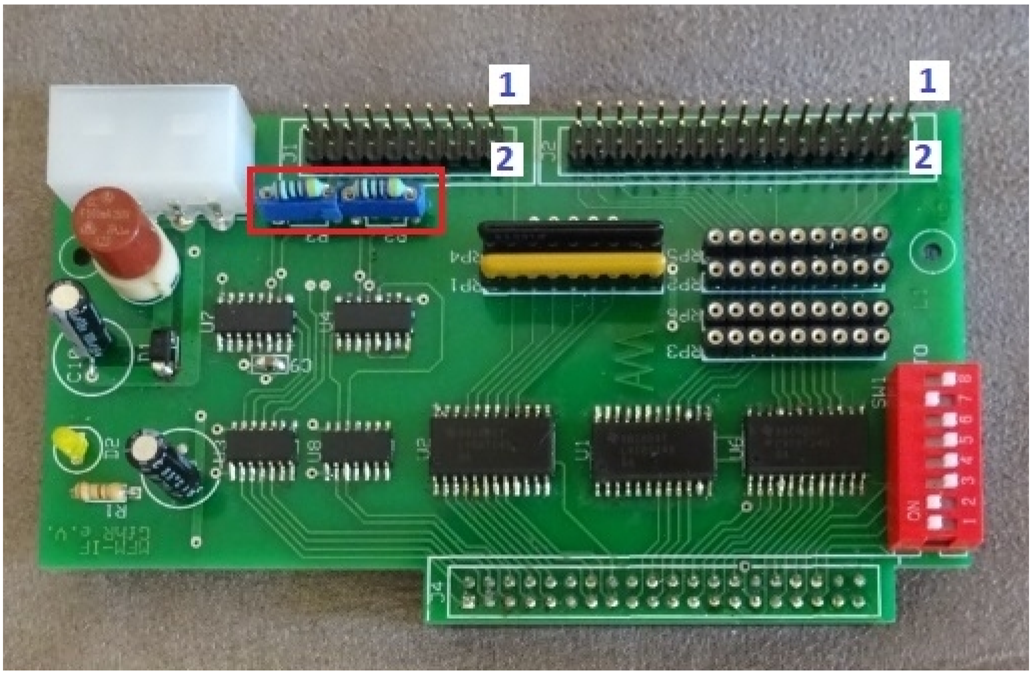
Emulator mode : Resistor network: RP2, RP3, RP5, RP6 installed
Read/Clone mode : Resistor network: RP1, RP4 installed





RP1, RP2, RP3 = 220 Ohm
RP4, RP5, RP6 = **330** Ohm

Pin assignment and differential OP receiver termination

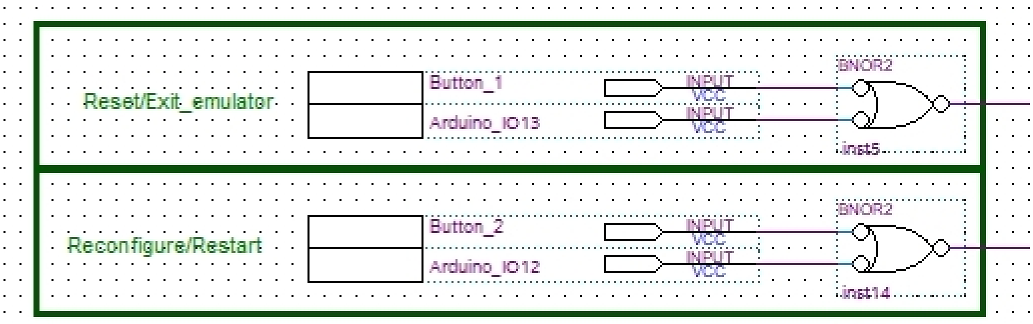


The resistors R2 and R3 are pluggable and defined by default with 100 ohms. However, this is also dependent on the used disk drive.

2.2 Reset/Reconfig buttons

Unfortunately, the reset and reconfig buttons 1 and 2 on the DE10 Nano board are very small and difficult to reach. Now it is possible to control the reset/reconfig function alternatively via 2 external button. These buttons must be connected to the Arduino connector as follows:

Arduino_IO13 = AH12 (Button 1) = reset/exit
Arduino_IO12 = AH11 (Button 2) = reconfig/restart
See also DE10 User Guide 3.6.3 Arduino Uno R3 Expansion Header , page 30



Chapter 3

Get started

Requirement : Up and running FPGA-SoC_Linux on a SoC/HPS board, like the DE10-Nano
Reference : DE10-Nano_User_manual.pdf
Further information on my homepage, pdp11gy.com and on de10-nano.terasic.com/cd

It's recommended to download and install the Unix kernel
de10_nano_linux_console
Details in the manual Getting Started Guide @ de10-nano.terasic.com/cd

Quick Start:

The firmware can be loaded in **3** different ways.

1) In the current version now works "**Load FPGA from Linux**". To load the firmware, another software is used, see
https://github.com/nhasbun/de10nano_fpga_linux_config
This software was taken over unchanged, only the Makefile was modified and the executable file is called loadrbf. As a pure user, I recommend this method because there is no additional software required like Quartus.

Here are the steps to load the firmware and start the MFM emulator:
- Suppose you are in Folder MFM `root@socfpga:~/MFM`
- First, copy the file "soc_mfm_v1_0.zip" to the DE0-Nano-SoC board, for example, using scp or winscp. Unpack the zip file and navigate to folder soc_mfm_beta.

unzip soc_mfm_v1_0.zip
cd soc_mfm_v1_0
cd MFM
chmod 777 *

The loadrbf program is using the filename fpga_config_file.rbf but the RL emulator is using the file RL_EMULATOR_SoC.rbf . Use a link to get the issue fixed as follow:

ln -s ../FW/MFM_EMULATOR_SoC.rbf fpga_config_file.rbf

That's all !

Directory listing next page:

```
root@socfpga:~/soc_mfm_v1_0/MFM# ls -l
total 176
-rwxrwxrwx 1 root root  9216 Oct  5 10:15 default.dsk
-rwxrwxrwx 1 root root 43008 Oct  5 10:15 default.gap
-rwxrwxrwx 1 root root 10752 Oct  5 10:15 default.mfm
-rwxrwxrwx 1 root root    15 Oct  5 10:15 disk_speed_0.inf
-rwxrwxrwx 1 root root    15 Oct  5 10:15 disk_speed_1.inf
-rwxrwxrwx 1 root root    15 Oct  5 10:15 disk_speed_2.inf
-rwxrwxrwx 1 root root   184 Oct  5 10:15 diskinfo_0.inf
-rwxrwxrwx 1 root root   180 Oct  5 10:15 diskinfo_1.inf
-rwxrwxrwx 1 root root   169 Oct  5 10:15 diskinfo_2.inf
-rwxrwxrwx 1 root root    15 Oct  5 10:15 diskspeed.inf
lrwxrwxrwx 1 root root    26 Oct  5 10:17 fpga_config_file.rbf ->
../FW/MFM_EMULATOR_SoC.rbf
-rwxrwxrwx 1 root root 13795 Oct  5 10:15 loadrbf
-rwxrwxrwx 1 root root 32232 Oct  5 10:15 mfmemulator
-rwxrwxrwx 1 root root 31355 Oct  5 10:15 readc
root@socfpga:~/soc_mfm_v1_0/MFM#
```

Now, you can start the firmware loader **loadrbf**

root@socfpga:~/socv2_2/RL# **./loadrbf**

A) loadrbf program output:

```
root@socfpga:~/soc_mfm_v1_0/MFM# ./loadrbf
****************************************************
MSEL Pin Config..... 0xa
FPGA State.......... Powered Off
cfgwdth Register.... 0x1
cdratio Register.... 0x0
axicfgen Register... 0x0
Nconfig pull reg.... 0x0
CONF DONE........... 0x0
Ctrl.en?............ 0x0
*****************************************************
Turning FPGA Off.
Setting cdratio with 0x3.
Turning FPGA On.
Loading rbf file.
EOF reached.
*****************************************************
MSEL Pin Config..... 0xa
FPGA State.......... User Phase
cfgwdth Register.... 0x1
cdratio Register.... 0x3
axicfgen Register... 0x0
Nconfig pull reg.... 0x0
CONF DONE........... 0x0
Ctrl.en?............ 0x0
*****************************************************
root@socfpga:~/soc_mfm_v1_0/MFM#
```

Now, the heartbeat LED on the interface board should be blinking

Load FPGA from Linux is my recommendation because there are no other tools or programs needed. Please note, requirement is to uses the Linux Console (kernel 4.5), version 1.3. This version must be downloaded from de10-nano.terasic.com/cd. By default, the Linux LXDE desktop (kernel 4.5) is installed on the micro SD. Unfortunately, the firmware loader can not be started manually with this kernel, since the kernel probably uses the hardware in an other way after starting the desktop (status OKT. 2019). If you want to use this kernel you have to do it relatively complicated with the bootloader. Alternatively, there are 2 more methods, but you need additional Software/tools, the QUARTUS programmer.

2) Load .sof file(NOT permanent)

- De0-Nano-SoC DIP switch (SW10) to default configuration, see page 12 @ User_manual
- unzip the file "`soc_mfm_beta.zip`"
- Start Quartus Lite Version 16.1
- Make sure, your USB connection to the DE10-Nano is working.
- Follow the instruction in the DE10-Nano_User_manual at page 15 and load the **MFM_EMULATOR_SoC.sof** file.
- After download , the heartbeat LED schould be blinking.

3) Permanent (EPCS): Required: Quartus Lite Version 16.1

- De0-Nano-SoC DIP switch (SW10) to EPCS configuration, see page 12 @ User_manual
- unzip the file "`soc_mfm_beta.zip`"
- Start Quartus Lite Version 16.1
- Make sure, your USB connection to the DE10-Nano is working.
- Follow the instruction in the DE10-Nano_User_manual at page 112 and flash the DE10-Nano board with the fil **MFM_EMULATOR_SoC.jic** from folder /flash.
- After repowering the DE10-Nano board, the heartbeat LED schould be blinking.

Folders:

FW: Contains the MFM_EMULATOR_SoC.jic file for flashing the FW into the EPCS and the MFM_EMULATOR_SoC.rbf for loading the FW in the FPGA.
The .cof file are configuration files if you want to convert the .sof file to .jic or .rbf by yourself.

MFM: Contains the binary runable MFM-emulator file: **mfmemulator** and the runable **readc** program which reads one track and/or cylinder.

In the Linux world you can now do smart things, like:
alias mfm='./loadrbf;sleep 2;./mfmemulator'

Chapter 4 Software

The MFM disk emulator project consists of 2 programs, the read and test program **readc** and the disk reader/emulator **mfmemulator.** Both programs are in the folder MFM located.

The readc program:

It is recommended to start this program **first**. To do this, follow these steps:

- Install the resistor network RP1 and RP4
- Set slide switch 0 to OFF, = read mode
- Connect the external disk
- Select the type of disk via Switch 5-8 (default is ST412)
- Power and start the program: **./readc**

The program first checks the connected disk and finds the disk unit number.
Then it checks if the disc is ready and in home position. After that, the rotation speed of the disk is determined by means of the index frequency and will be displayed and stored in the file diskspeed.inf. Then, the cylinder number is queried. In the example cylinder 110 is used. **Now comes the most important point:**
You must enter the 16 bit Hex **DataAM pattern**. The hex pattern A5F8 is suggested. Details can be found in the document **MFM_debug.pdf** or in the appendix of this manual. Assuming the disk has 4 tracks with 18 sectors per track and the DataAM pattern is correct, the program will find the pattern 72 times in total.
All cylinder data are stored in 3 files, see example below.
The file-extension has the following meaning:

.mfm = decoded MFM data including Header/Servo/CRC information
.dsk = extracted and decoded raw user data, also usefull for SIMH.
.gap = MFM gap data, necessary in emulator mode.

If the pattern is not found often enough or not at all, you can continue working in the track/head mode. The DataAM pattern can be reset again and the selected track is searched again. Details in the following example:
Note: It is very important to find the right DataAM pattern otherwise the data can not be read correctly because the MFM decoder can not synchronize.

Example: **readc program output:**

```
****** MFM-DISK read + test  @ Soc/HPS *******
READ one Cylinder/Track and save it to SD card
   DE10-Nano ST-506/412/225  Version  V1.0
*********************|*************************
            (c) WWW.PDP11GY.COM

     >>>>>> DEBUG-MODE = ON <<<<<<
     >>>> Device Type = ST412 <<<<

  Anzahl der Cylinder: 306
```

```
      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:  110

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

   Cylinder: 110 ,Trigger DataAM: lsb : 0xA5  msb: 0xF8

      ************ Step to Cylinder 110  done ***********

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

      Save MFM-gaps    data into file: ST412_gap-data@cylinder_110.gap
      Save RAW-image   data into file: ST412_raw-data@cylinder_110.dsk
      Save MFM-decoded data into file: ST412_mfm-data@cylinder_110.mfm

 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 58  Nr.: 1  Gap: 58
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 629  Nr.: 2  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 1200  Nr.: 3  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 1771  Nr.: 4  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 2342  Nr.: 5  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 2913  Nr.: 6  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 3484  Nr.: 7  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 4055  Nr.: 8  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 4626  Nr.: 9  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 5197  Nr.: 10  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 5768  Nr.: 11  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 6339  Nr.: 12  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 6910  Nr.: 13  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 7481  Nr.: 14  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 8052  Nr.: 15  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 8623  Nr.: 16  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 9194  Nr.: 17  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 9765  Nr.: 18  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 10810  Nr.: 19  Gap: 1045
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 11381  Nr.: 20  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 11952  Nr.: 21  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 12523  Nr.: 22  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 13094  Nr.: 23  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 13665  Nr.: 24  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 14236  Nr.: 25  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 14807  Nr.: 26  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 15378  Nr.: 27  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 15949  Nr.: 28  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 16520  Nr.: 29  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 17091  Nr.: 30  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 17662  Nr.: 31  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 18233  Nr.: 32  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 18804  Nr.: 33  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 19375  Nr.: 34  Gap: 571
 found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 19946  Nr.: 35  Gap: 571
```

```
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 20517  Nr.: 36  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 21563  Nr.: 37  Gap: 1046
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 22134  Nr.: 38  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 22705  Nr.: 39  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 23276  Nr.: 40  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 23847  Nr.: 41  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 24418  Nr.: 42  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 24989  Nr.: 43  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 25560  Nr.: 44  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 26131  Nr.: 45  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 26702  Nr.: 46  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 27273  Nr.: 47  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 27844  Nr.: 48  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 28415  Nr.: 49  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 28986  Nr.: 50  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 29557  Nr.: 51  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 30128  Nr.: 52  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 30699  Nr.: 53  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 31270  Nr.: 54  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 32315  Nr.: 55  Gap: 1045
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 32886  Nr.: 56  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 33457  Nr.: 57  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 34028  Nr.: 58  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 34599  Nr.: 59  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 35170  Nr.: 60  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 35741  Nr.: 61  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 36312  Nr.: 62  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 36883  Nr.: 63  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 37454  Nr.: 64  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 38025  Nr.: 65  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 38596  Nr.: 66  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 39167  Nr.: 67  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 39738  Nr.: 68  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 40309  Nr.: 69  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 40880  Nr.: 70  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 41451  Nr.: 71  Gap: 571
found: DataAM_msb 0xA5  DataAM_lsb 0xF8 @ 42022  Nr.: 72  Gap: 571


      index puls-width :  132107   =   1651.338 us
      index frequency :  1344731  = 16809.137 us
      Disk speed is:  3569.49 RPM

      Select haed, 1 to 4 ( 0=exit/5=set DataAM)  : 1
      Decoded data       @head1 save into file:  mfm-data@head=track-1_cyl-110.mfm
      Recorded Gap data @head1 save into file:  gap-data@head=track-1_cyl-110.gap
      user raw data      @head1 save into file:  raw-data@head=track-1_cyl-110.dsk
      Track/head 1 : found 18 matches

      Select haed, 1 to 4 ( 0=exit/5=set DataAM)  : 2
      Decoded data       @head2 save into file:  mfm-data@head=track-2_cyl-110.mfm
      Recorded Gap data @head2 save into file:  gap-data@head=track-2_cyl-110.gap
      user raw data      @head2 save into file:  raw-data@head=track-2_cyl-110.dsk
      Track/head 2 : found 18 matches

      Select haed, 1 to 4 ( 0=exit/5=set DataAM)  : 3
      Decoded data       @head3 save into file:  mfm-data@head=track-3_cyl-110.mfm
      Recorded Gap data @head3 save into file:  gap-data@head=track-3_cyl-110.gap
      user raw data      @head3 save into file:  raw-data@head=track-3_cyl-110.dsk
```

```
        Track/head 3 : found 18 matches

        Select haed, 1 to 4 ( 0=exit/5=set DataAM)  : 4
        Decoded data      @head4 save into file:  mfm-data@head=track-4_cyl-110.mfm
        Recorded Gap data @head4 save into file:  gap-data@head=track-4_cyl-110.gap
        user raw data     @head4 save into file:  raw-data@head=track-4_cyl-110.dsk
        Track/head 4 : found 18 matches

        Select haed, 1 to 4 ( 0=exit/5=set DataAM)  : 0
Press RESET/Button-1 for exit, Reconfig/Button-2 for restart
```

The readc program can also work in emulator mode (slide switch 0 to ON), but only with the data of one selected cylinder. This mode is mainly intended for developers based on the file MFM_emulator_debug.bdf file. The project has to be recompiled with Quartus. In this mode the encoded data will be re-decoded for comparison on the Logic Analyzer. For the sake of completeness, here is the output of the readc program in emulator mode:

Example: **readc** program output **emulator mode**

```
         ****** MFM-DISK read + test  @ Soc/HPS *******
         READ one Cylinder/Track and save it to SD card
            DE10-Nano ST-506/412/225  Version  V1.0
         *********************|*************************
                    (c) WWW.PDP11GY.COM

              >>>>>> DEBUG-MODE = ON <<<<<<
              >>>> Device Type = ST412 <<<<

**************** temporary emulator mode  **************

      index frequency :  1344731  = 16809.137 us
      emulated disk speed is:  3569.49 RPM

       Cylinder - nummer eingeben:  110
            Read MFM data file:   ST412_mfm-data@cylinder_110.mfm
            Read MFM gap  file:   ST412_gap-data@cylinder_110.gap
 Started with operating mode:            0100000110100001FM
```

The mfmemulator program:

This program works in **read** mode to clone a complete disk and also in **emulator** mode. In read mode, the same hardware setup is required as with the readc program. Switches 5-8 can select 16 different disk drive specifications, 0 to F. These drive numbers correspond to a drive file, diskinfo_0.inf to diskinfo_F.inf. All necessary technical data of the disks can be stored in these files. Example: **diskinfo_1.inf**

```
# File: diskinfo_1.inf
# disk type = RD51(ST412) from DEC , 5-1/4 zoll drive, 10Mbyte
# connected to RQDX-1 , sector size = 512 byte.
my_RD51(ST412)_filename
18,512,306,A5F8
```

Blue lines are comments, starting with "#"
The green line after defines the filename
The red line after contains the specification of the disc :
 18 = sectors per track , 512 = sectorsize in byte
 306 = Maximal number of cylinders, A5F8 = DataAM pattern
The disk rotation speed is automatically determined and stored in the files disk_speed_0.inf to disk_speed_F.inf.

Example:

mfmemulator program output , read/clone mode:

```
        ** MFM-DISK Reader/Cloner+EMULATOR @ Soc/ HPS **
        DE10-Nano  ST-506/412/225 emulator Version V.1.0
        ***********************|***********************
                     (c) WWW.PDP11GY.COM

              >>>>>> DEBUG-MODE = ON <<<<<<
             Disk config file:  diskinfo_1.inf
# File: diskinfo_1.inf
# disk type = RD51(ST412) from DEC , 5-1/4 zoll drive, 10Mbyte
# connected to RQDX-1 , sector size = 512 byte.

 disk-data: sector-size: 512  nr. of Cylinder: 306  DataAM: A5F8
      myfile3 = my_RD51(ST412)_filename.dsk
      myfile4 = my_RD51(ST412)_filename.mfm
      myfile5 = my_RD51(ST412)_filename.gap
      ******************************************
      **************** Clone-Mode **************
      ******************************************

      Anzahl der Cylinder: 306
      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

     index puls-width :  130106   =  1626.325 us
     index frequency :  1344018  = 16800.225 us
     Disk speed is:  3571.38 RPM

      Cloning cylinder 0 ……… 305
      back to home position
      Save .mfm - data to SD-Card into file:  my_RD51(ST412)_filename.mfm

      Save .dsk - data to SD-Card into file:  my_RD51(ST412)_filename.dsk

      Save .gap - data to SD-Card into file:  my_RD51(ST412)_filename.gap

   *********** Clone-Mode finished **********

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

Emulator mode:

- Install the resistor network RP2 , RP3 , RP5 , RP6
- Set slide switch 0 to 1, = read mode
- Select the type of disk via Switch 5-8 (default is ST412)
- Set unit number of the emulated disk via Switch 3-4
- Connect the external controller
- Power and start the program: **./mfmemulator**

mfmemulator program output , emulator mode:

```
        ** MFM-DISK Reader/Cloner+EMULATOR @ Soc/ HPS **
        DE10-Nano  ST-506/412/225 emulator Version V.1.0
        ***********************|************************
                     (c) WWW.PDP11GY.COM

              >>>>>> DEBUG-MODE = ON <<<<<<
             Disk config file:  diskinfo_1.inf
# File: diskinfo_1.inf
# disk type = RD51(ST412) from DEC , 5-1/4 zoll drive, 10Mbyte
# connected to RQDX-1 , sector size = 512 byte.

 disk-data: sector-size: 512  nr. of Cylinder: 306  DataAM: A5F8
      myfile3 = my_RD51(ST412)_filename.dsk
      myfile4 = my_RD51(ST412)_filename.mfm
      myfile5 = my_RD51(ST412)_filename.gap

      ******************************************
      ************** Emulator-Mode *************
      ******************************************
      Read MFM data file:   my_RD51(ST412)_filename.mfm
      Read MFM gap  file:   my_RD51(ST412)_filename.gap

      index frequency :  1344018  = 16800.225 us
      emulated disk speed is:  3571.38 RPM
      gap data ENcoding

 ********** S T A R T ST-506/412/225 Emulator ***********
 Started with operating mode:            0100000110100001
```

The program reads the .mfm file and .gap file. Then, cylinder 0 is copied to the Dual_Ported_Ram (DPR), the index-pulse generator is started and then the emulator. The disk will now be emulated in 2 different, selectable ways: As discussed in more detail in the Appendix of this manual or in the document MFM_debug.pdf, each manufacturer has implemented its own disk format in terms of header/CRC/servo and syncpattern. As a result, it is usually not possible to encode the header/servo/CRC data correctly and thus it is not possible to emulate the disk with this method which is using the .mfm data. However, this emulator has a second method implemented. In addition, the MFM signal gaps are measured and stored in the .gap files. The encoding method is selected with switch 1: ON = mfm output, OFF = gap recorded output (recommended).

A program start without valid disk configurations file, here in the example diskinfo_7.inf leads to default values being used. Next page is an output example:

mfmemulator program output , default mode:

```
        ** MFM-DISK Reader/Cloner+EMULATOR @ Soc/ HPS **
        DE10-Nano  ST-506/412/225 emulator Version V.1.0
        ***********************|***********************
                     (c) WWW.PDP11GY.COM

              >>>>>> DEBUG-MODE = ON <<<<<<
             Disk config file:  diskinfo_7.inf


ERROR opening disk configuration file: diskinfo_7.inf

           +---------------------------------+
           |  No config file, using default  |
           +---------------------------------+

disk-data: sector-size: 512  nr. of Cylinder: 306  DataAM: A5F8
     myfile3 = default.dsk
     myfile4 = default.mfm
     myfile5 = default.gap

     ******************************************
     ************** Emulator-Mode *************
     ******************************************

     index frequency :  1344179  = 16802.238 us
     emulated disk speed is:  3570.95 RPM
     gap data ENcoding

********** S T A R T ST-506/412/225 Emulator ***********
Started with operating mode:            0100000110100001
```

The default data file contains only data of one track and will be copied to all tracks and cylinder after reading. This method may be necessary if you want to recreate and reformat an emulated disk.

Some personal information:
I also use a Raspberry Pi 3 (model B) and now Pi4, connected via network to the DE10-Nano board. I use the Raspberry for development purposes with a graphical interface. I can compile the programs like SIMH emulators and copy it to the DE10-Nano board, because it is binary compatible. That's so great and there is still a lot of room for further additional applications.

The following pages are intended for developers. The whole project is open source and for me the most important thing is to keep the old software up and running.

www.pdp11gy.com

Instructions: Rebuild the MFM-emulator running on DE10-Nano board.

Firmware:
****

Use Quartus V16.1 and open the Project RL_emulator.qpf
After compiling the Project, use the the MAKE_jic.cof and MAKE_rbf.cof file to build the .jic and .rbf files.

Programming environment:
****

It was difficult to make everything runable because many things in the documentation and in the examples were not correct. Here is a step by step explamation to rebuild the MFM-emulator if necessary or if you want to design some add-on application.

- Download and install **Quartus Version 16.1.**
- Download and install Intel **SoCEDSPro Version 16.1**

Fix Problems:
****

*1 : error You must define soc_cv_av or soc_a10 before compiling with HwLibs
Go to intelFPGA/16.1/embedded/ip/altera/hps/altera_hps/hwlib/include
Copy all .h files in the folder soc_cv_av and soc_a10

*2 : generate_hps_qsys_header.sh : PATH is not set correct: correct as following:

```
#!/bin/sh
PATH=/cygdrive/C/altera_lite/16.1/quartus/sopc_builder/bin:$PATH
sopc-create-header-files \
"$PWD/RL_system.sopcinfo" \
--single hps_0.h \
--module hps_0
```

*3: Modify the makefiles, here the MFM-emulator cylinder-read make file
software/MFM/Makefile   //   mfmemulator
software/read/Makefile   //   readc

mfmemulator makefile:

```
#
TARGET = mfmemulator
ALT_DEVICE_FAMILY ?= soc_cv_av
ALT_DEVICE_FAMILY ?= soc_a10
#
```

```
CROSS_COMPILE = arm-linux-gnueabihf-
#CFLAGS = -static -g -Wall  -I$
{SOCEDS_DEST_ROOT}/ip/altera/hps/altera_hps/hwlib/include
CFLAGS = -g -Wall  -I$
{SOCEDS_DEST_ROOT}/ip/altera/hps/altera_hps/hwlib/include/$
{ALT_DEVICE_FAMILY} -Dsoc_cv_av -Dsoc_a10
LDFLAGS =  -g -Wall
CC = $(CROSS_COMPILE)gcc
ARCH= arm


build: $(TARGET)
$(TARGET): main.o
	$(CC) $(LDFLAGS)   $^ -o $@
%.o : %.c
	$(CC) $(CFLAGS) -c $< -o $@

.PHONY: clean
clean:
	rm -f $(TARGET) *.a *.o *~
```

NOTES:
I have included an additional file in the project folder, MFM_emulator_debug.bdf. it is also a schematics file and designed that the MFM-data are re-decoded in test-emulator mode. However, you have to recompile the entire firmware. The result can then be seen with a logic analyzer.

References:

http://www.pdp11gy.com
https://github.com/pdp11gy/SoC-HPS-based-MFM-disk-emulator
https://github.com/pdp11gy/SoC-HPS-based-RL-disk-emulator
http://www.pdp11gy.com/sddoneE.html

MFM - Disks
issues, infos , problems, solutions

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

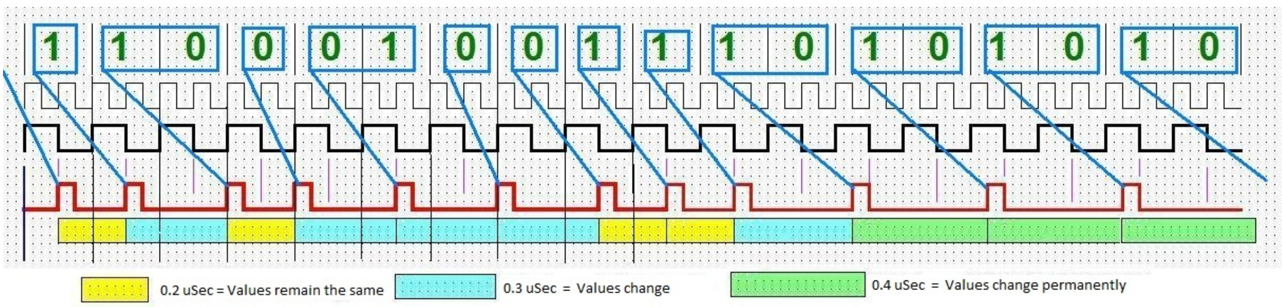
References:

https://github.com/pdp11gy/SoC-HPS-based-MFM-disk-emulator
Download and unzip the file MFM-disk_Emulator_SoC_v1_0.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 chitter, 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 :

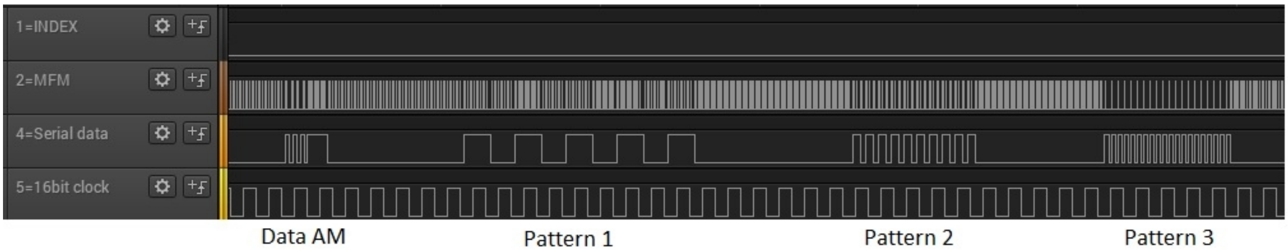
During development, I had chosen a method to write a well-defined pattern on the disk. This method was very helpful for the RL01 / RL02 emulator development, so I did use this method in the development of the MFM disk emulator as well.
Abstract, used Pattern:

```
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$(36) \ NEXT I          // 0x43, 0x92, 0xDC
48 FOR I=1 TO 5 \ A$=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$(128) \ NEXT I                           // 1000 0000 = 0x80
68 FOR I=1 TO 5 \ A$=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                           // 0111 1111 = 0x7F
98 FOR I=1 TO 5 \ A$=A$+CHR$(0) \ NEXT I
510 A$=A$+CHR$(10)+CHR$(13)
520 A$=A$+"STELL DIR VOR ES IST KRIEG UND KEINER GEHT HIN"
525 A$=A$+CHR$(10)+CHR$(13)
540 A$=A$+"  IMAGINE IT IS WAR AND NOBODY GOAS THERE  "
545 A$=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-LENGTH: ";LEN(A$)                              // Should be 255
640 FOR I=1 TO 125 \ B$=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-LENGTH: ";LEN(B$)                              // Should be 255
699 goto 800
700 OPEN "DU0:PATT4.TXT" FOR OUTPUT AS FILE #1
720 FOR I=1 TO 5000
730 PRINT #1,A$;
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:

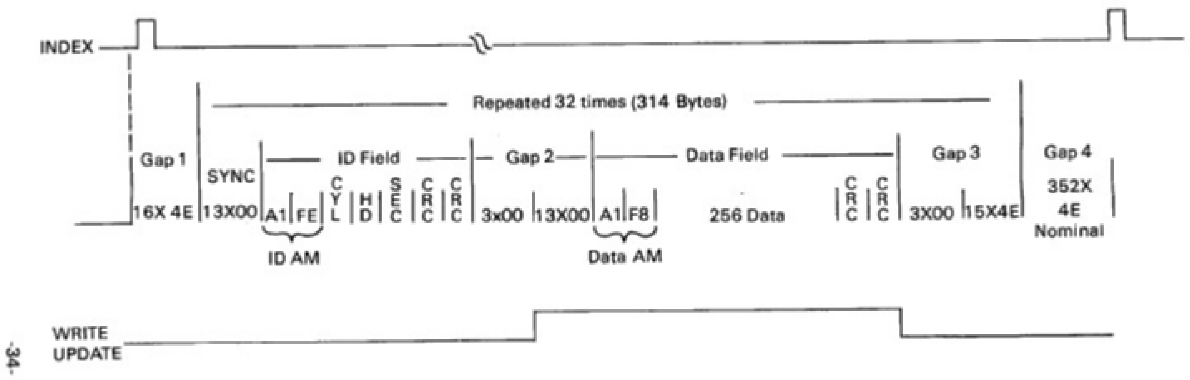
```
00000350  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000360  00 00 00 00 00 00 00 00 00 00 32 A2 00 00 01 C4  ..........2¢...Ä
00000370  24 24 24 24 24 24 24 24 24 24 24 24 24 3F FF FF  $$$$$$$$$$$$$?ÿÿ
00000380  FF FF FF FF FF FF FF FF FF FF FF CA 80 21 F3 99  ÿÿÿÿÿÿÿÿÿÿÿÊ€!ó™
00000390  97 03 9F FF FF FE 00 00 00 00 00 00 00 00 00 00  —.Ÿÿÿþ..........
000003A0  00 00 01 A5 F8 00 00 00 00 00 FF 00 FF 00 FF 00  ...¥ø.....ÿ.ÿ.ÿ.
000003B0  FF 00 FF 00 00 00 00 00 00 33 33 33 33 33 00 00  ÿ.ÿ......33333..
000003C0  00 00 00 55 55 55 55 55 00 00 00 00 00 49 92 24  ...UUUUU.....I’$
000003D0  49 92 24 49 92 24 00 00 00 00 00 23 91 DC 23 91  I’$I’$.....#‘Ü#‘
000003E0  DC 23 91 DC 00 00 00 00 00 80 80 80 80 80 80 80  Ü#‘Ü.....€€€€€€€
000003F0  80 80 80 00 00 00 00 00 E7 9C F3 E7 9C F3 E7 9C  €€€.....çœóçœóçœ
00000400  F3 00 00 00 00 00 63 8C F1 63 8C F1 63 8C F1 7F  ó.....cŒñcŒñcŒñ.
00000410  7F 7F 7F 7F 7F 7F 7F 7F 7F 00 00 00 00 00 0A 0D  ................
00000420  53 54 45 4C 4C 20 44 49 52 20 56 4F 52 20 45 53  STELL DIR VOR ES
00000430  20 49 53 54 20 4B 52 49 45 47 20 55 4E 44 20 4B   IST KRIEG UND K
00000440  45 49 4E 45 52 20 47 45 48 54 20 48 49 4E 0A 0D  EINER GEHT HIN..
00000450  20 20 49 4D 41 47 49 4E 45 20 49 54 20 49 53 20    IMAGINE IT IS 
00000460  57 41 52 20 41 4E 44 20 4E 4F 42 4F 44 59 20 47  WAR AND NOBODY G
00000470  4F 41 53 20 54 48 45 52 45 20 20 0A 0D FF 00 FF  OAS THERE  ..ÿ.ÿ
00000480  00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF  .ÿ.ÿ.ÿ.ÿ.ÿ.ÿ.ÿ.ÿ
00000490  00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF  .ÿ.ÿ.ÿ.ÿ.ÿ.ÿ.ÿ.ÿ
000004A0  00 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .ÿ..............
000004B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000004C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000004D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000004E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000004F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000500  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000510  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000520  00 00 FF FF 00 00 FF FF 00 00 00 00 00 00 00 00  ..ÿÿ..ÿÿ........
00000530  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000540  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Very important is the field **data AM** (A5 F8 @ 3A3). 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:

FIGURE 20

TRACK FORMAT AS SHIPPED



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 $=x^{16}+x^{12}+x^{5}+1$
6. Bit 7 of Head Byte ID Field equals 1 in a defective sector (Cylinder Ø 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
                          ------------------------+----------------
                  SECTOR: 314      TRACK: 10048 |  CYLINDER: 40192
  Einmalig dazu:  Gap1   16x4E      =         16 |               64
                  Gap4  352x4E      =        352 |             1408
                                  ----------------+----------------
```

```
( Byte )          10416 |         41664
 ( Bit )          83328 |        333312
( Word )           5208 |         20832
========================+================
```

Understanding and analysis

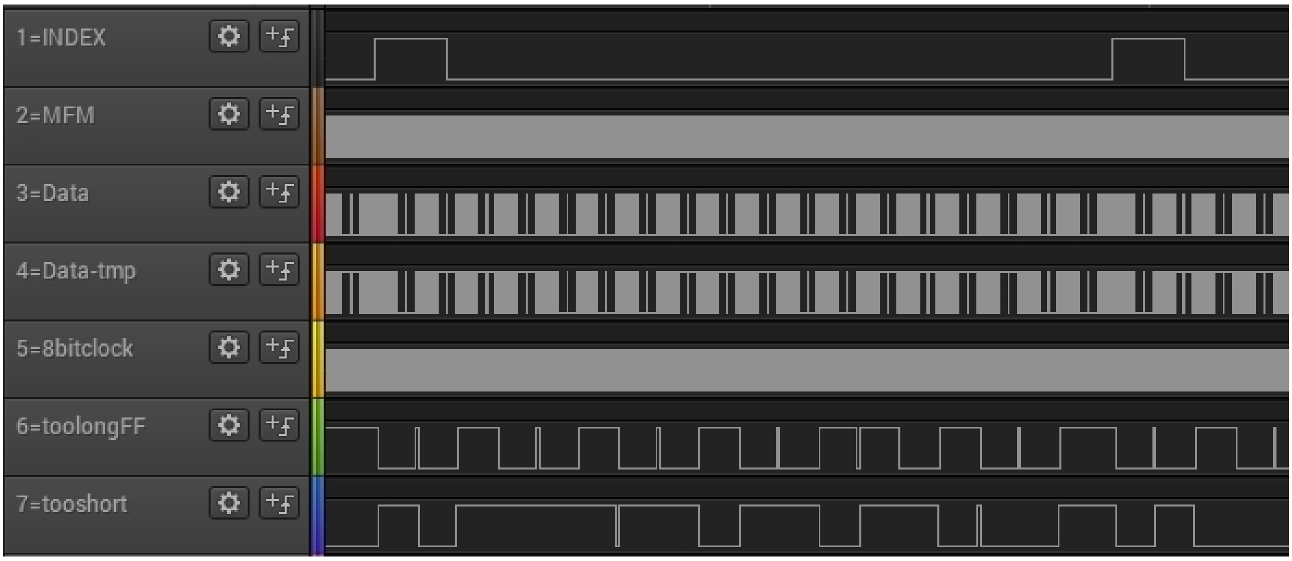
The interface and the corresponding signals were described in detail by the company Seagate and were widely respected. It looks quite different at data and timing format. Everything here is incompatible. Each manufacturer has guaranteed implemented his own track and data format which was genarated with 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 concernig the timing outside the data field. I found too short and too long MFM gaps.

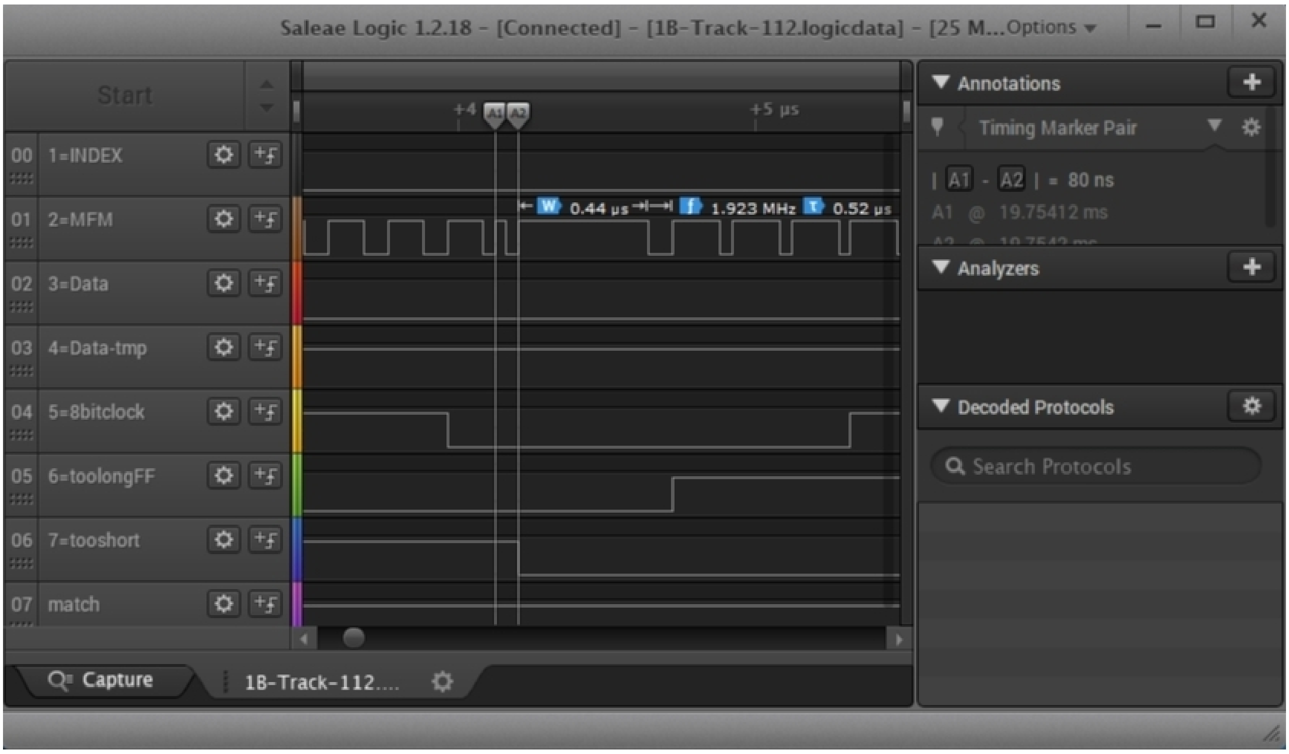
Example, logic analyser:



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 cylcles. 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 allign 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:

Any hint 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