

MFM DISK-DRIVE EMULATOR

Get started Manual for the DE10-Nano board

Version Beta V010

**Instructions for loading & flashing DE0-Nano_SoC board
running the MFM disk emulator on it**

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

We recommend to download and install the Unix kernel

de10_nano_linux_console

Details in the manual Getting Started Guide

Jumper settings

DE10-Nano: **The four slide switches** (page 26, User_manual): Only switch 0
is used: ON=Clone-Mode OFF=EMULATOR Mode
Button 2 and 3 : Reconfigure and Reset/Restart
De0-Nano-SoC DIP switch (**SW10**) configuration, see page 12 @
User_manuel

Interface-board: 8 switches :

Switch 1: ON: LED Debug info OFF=Pattern

Switch 2 : Debug Mode ON/OFF

Switch 3-4: Unit number

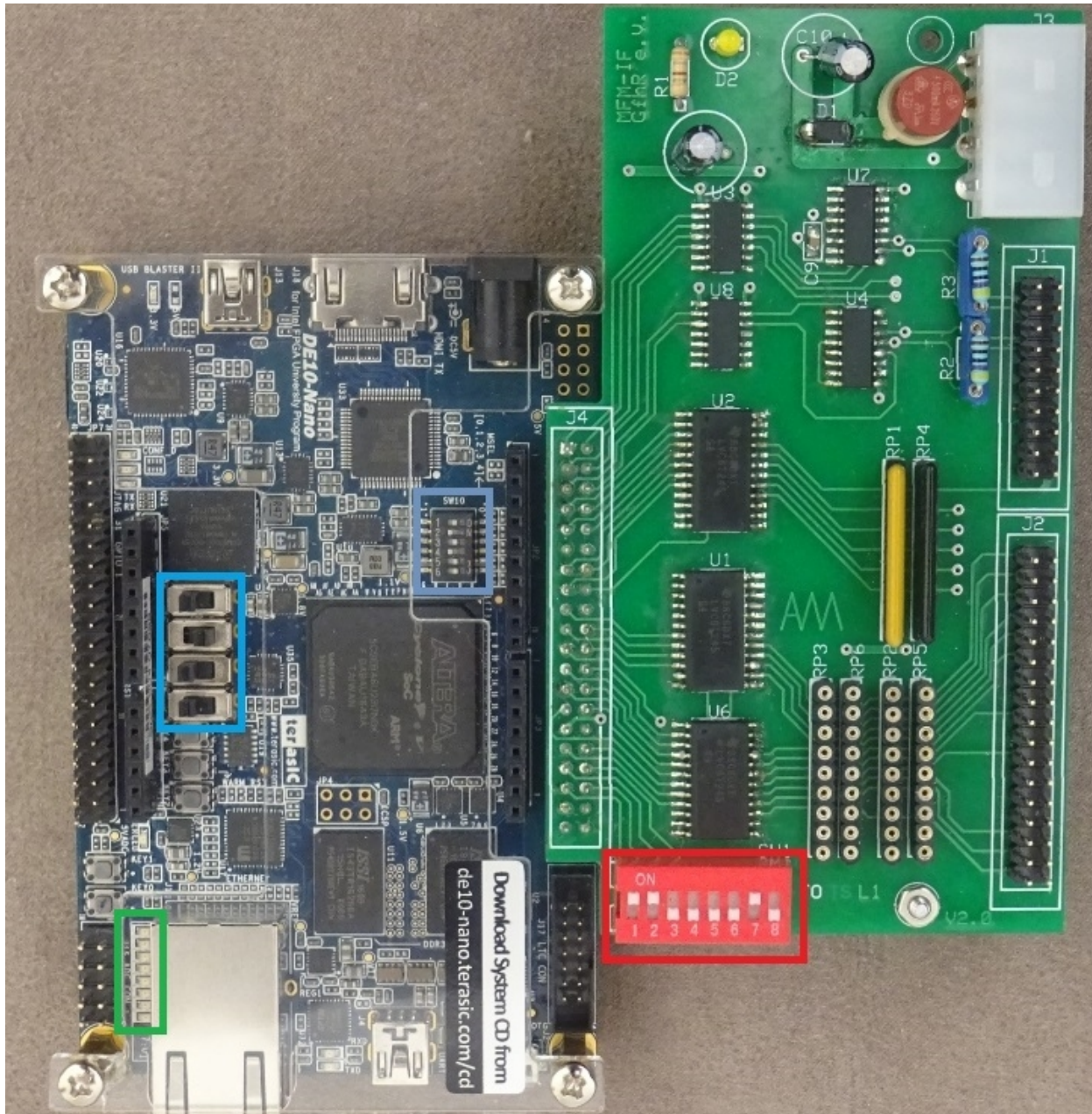
Switch 5-8 : drive typ:

0-0-0-1 = ST506

0-0-1-0 = ST412

0-1-0-0 = ST 225

Overview:



LED's : 0 = **heartbeat LED** (should 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

Quick Start:

The firmware can be loaded in 3 different ways.

1) New: 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_beta.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_beta.zip
```

```
cd soc_mfm_beta
```

```
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:

```
root@socfpga:~/MFM/soc_mfm_beta/MFM# ls -l
total 76
lrwxrwxrwx 1 root root    26 Jun 18 15:33 fpga_config_file.rbf ->
../FW/MFM_EMULATOR_SoC.rbf
-rwxrwxrwx 1 root root 7486 Jun 18 15:32 hps_0.h
-rwxrwxrwx 1 root root 13795 Jun 18 15:32 loadrbf
-rwxrwxrwx 1 root root 27135 Jun 18 15:32 mfmemulator
-rwxrwxrwx 1 root root 21067 Jun 18 15:32 read_save-cylinder
root@socfpga:~/MFM/soc_mfm_beta/MFM#
```

Now you can start the **A)**firmware loader and then the **B)**RL emulator or **C)** the read/test program, read and save one cylinder/track:

A) root@socfpga:~/socv2_2/RL# ./loadrbf

B) root@socfpga:~/socv2_2/RL# ./rlemulator

C) root@socfpga:~/socv2_2/RL# ./readc

loadrbf program output:

```
*****
MSEL Pin Config..... 0xa
FPGA State..... Powered Off
cfgwidth Register.... 0x1
cdratio Register.... 0x0
axicfggen 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
cfgwidth Register.... 0x1
cdratio Register.... 0x3
axicfggen Register... 0x0
Nconfig pull reg.... 0x0
CONF DONE..... 0x0
Ctrl.en?..... 0x0
*****
root@socfpga:~/socv2_2/RL#
```

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

mfmemulator program output:

< not yet ready >

In the Linux world you can now do smart things, like:

alias mfm='./loadrbf;sleep 2;./mfmemulator'

If you type now mfm, the firmware will be loaded and then the mfm emulator is starting.

There are **2** more ways to load the firmware to the DE10 Nano board. However, you need additional software , Quartus, Version 16.1. The DE10-Nano board is pre-configured with the Angstrom Linux - Kernel (DE10_Nano_LXDE). the default installed Linux is not able to run with a EPCS configuration.

I recommend to use the `de10_nano_linux_console.img` which can be very easy installed with disk-imager like win32diskimager. More details in the `Getting_Started_Guide.pdf`. The images and all documentation can be downloaded from www.de10-nano.terasic.com/cd .

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 should 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 file **MFM_EMULATOR_SoC.jic** from folder /flash.
- After repowering the DE10-Nano board, the heartbeat LED should be blinking.

Folders:

FW: Contains the `RL_EMULATOR_SoC.jic` file for flashing the FW into the EPCS and the `RL_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 runnable MFM-emulator file: **mfmemulator** and the runnable **readc** program which reads one track and cylinder.

The readc program is the best way to find the correct Data AM pattern.
More details in the manual **MFM_debug.pdf**

Some personal information:

I also use a Raspberry Pi 3 (model B) 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.

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 runnable because many things in the documentation and in the examples were not correct. Here is a step by step explanation 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-gnueabi-
#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 *~
```

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>
<https://github.com/pdp11gy/DEC-RL02-RL01-disk-emulator>
<http://www.pdp11gy.com/sddoneE.html>

Sources:

The most important Beta sources are included in the Folder **Sources_Beta**.
It's based on the same Qaurtus V16.1 environment as the RL01/RL02 emulator.
Upon completion of the project, a complete environment folder will be provided.

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