

# **DEC RL01/RL02 DISK-DRIVE EMULATOR**

## **Get started Manual for the DE10-Nano board**

### **Version 2.2**

**Instructions for loading & flashing DE0-Nano\_SoC board  
running the RL01/RL02 disk emulator on it**

**The following chapters explain how the RL emulator works. A distinction is made between a pure user and a developer for possible further development and changes.**

**Requirement :** Up and running FPGA-SoC\_Linux on a SoC/HPS board, like the DE10-Nano

**Reference :** DE10-Nano\_User\_manual.pdf  
DE10-Nano\_V2\_2.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 ): Select 16 disk sets, 0 to F

Button 2 and 3 : Reconfigure and Reset/Restart

**Interface-board:** Switch 8: init and configure a new disk set based on the slide switches.

Switch 7 : Force Power OK

Switch 6 : Debug - Mode ON/OFF

Switch 5 : drive typ, RL01 or RL02

Switch 4 - 1 : Confired RL-drives,

1=RL3,2=RL2,3=RL1,4=RL0

All Switch = OFF , = OFFLINE mode.

De0-Nano-SoC DIP switch (SW10) configuration, see page 12 @ User\_manual

## 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](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 RL emulator:

- First, copy the file "socv2\_2.zip" to the DE0-Nano-SoC board, for example, using scp or winscp. Unpack the zip file and navigate to folder socv2\_2.

```
unzip socv2_2  
cd socv2_2  
cd RL  
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/RL_EMULATOR_SoC.rbf fpga_config_file.rbf
```

**That's all !**

Directory listing:

```
deroot@socfpga:~/socv2_2/RL#ls -l  
total 11596  
-rwxrwxrwx 1 root root      240 May 21 11:06 PDP11GY.INF  
-rwxrwxrwx 1 root root 11796992 May 21 11:06 RL02_0-9.DEC  
-rwxrwxrwx 1 root root      492 May 21 11:06 RL9.TXT  
-rwxrwxrwx 1 root root       36 May 21 11:06 SN9.TXT  
lrwxrwxrwx 1 root root      25 May 21 11:08 fpga_config_file.rbf  
-> ..//FW/RL_EMULATOR_SoC.rbf  
-rwxrwxrwx 1 root root    13795 May 21 11:06 loadrbf  
-rwxrwxrwx 1 root root   44037 May 21 11:06 rlemulator  
root@socfpga:~/socv2_2/RL#
```

Now you can start the 1)firmware loader and then the 2)RL emulator:

- A) `root@socfpga:~/socv2_2/RL# ./loadrbf`
- B) `root@socfpga:~/socv2_2/RL# ./rlemulator`

## **loadrbf program output:**

```
*****
MSEL Pin Config..... 0xa
FPGA State..... Powered Off
cfgwdth Register.... 0x1
cdratio Register.... 0x0
axicfg 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
axicfg 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**

## **rlemulator program output:**

All details in the DE10-Nano\_V2\_2.pdf manual.

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

If you type now RL, the firmware will be loaded and then the RL 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 EPICS 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](http://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 "socv2\_2.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 **RL\_EMULATOR\_SoC.sof** file.
- After download , the heartbeat LED schould be blinking.

## **3) Permanent (EPICS): Required: Quartus Lite Version 16.1**

- De0-Nano-SoC DIP switch (SW10) to EPICS configuration, see page 12 @ User\_manual
- unzip the file "socv2\_2.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 **RL\_EMULATOR\_SoC.jic** from folder /flash.
- After repowering the DE10-Nano board, the heartbeat LED schould be blinking.

## **Folders:**

**FW:** Contains the RL\_EMULATOR\_SoC.jic file for flashing the FW into the EPICS 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.

**RL:** Contains the binary runnable RL-emulator file: rlemulator and a RL02 image file, RL02\_0-9.DEC, bootable, RT-11 V05.04 C with Macro-11, BASIC, Fortran, FOCAL + Kermit . Details in the DE10-Nano\_V2\_2.pdf manual.

**UTIL:** Contains two conversion utilities.

dec2dsk : converts the .DEC format to .DSK format

dsk2dec : converts the .DSK format to .DEC format

Note: Just for reference purpos. The RL emulator supports .DSK file already inline.

**SIMH:** Contains 2 runable SIMH based emulator for the PDP-8 and PDP-11

More details on <https://github.com/simh/simh>

### **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 RL-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 explamation to rebuild the RL-emulator if necessary or if you want to design some add-on application.

This folder project folder, DE10\_SoC\_RL\_emulator\_V2 contains all sources and the correct setup for the Quartus Version 16.1.

- 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 RL-emulator make file

software/RL\_emulator/Makefile

```
#
TARGET = rlemulator
ALT_DEVICE_FAMILY ?= soc_cv_av
ALT_DEVICE_FAMILY ?= soc_a10
#
CROSS_COMPILE = arm-linux-gnueabihf-
#CFLAGS = -static -g -Wall -I${SOCEDS_DEST_ROOT}/ip/altera/hps/altera_hps/hplib/include
CFLAGS = -g -Wall -I${SOCEDS_DEST_ROOT}/ip/altera/hps/altera_hps/hplib/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 *~
```

For comments and questions, please contact me.  
[INFO@pdp11gy.com](mailto:INFO@pdp11gy.com)

**References:**

<https://github.com/pdp11gy/SoC-HPS-based-RL-disk-emulator>  
<https://github.com/pdp11gy/DEC-RL02-RL01-disk-emulator>  
<http://www.pdp11gy.com/doneE.html>