

# DEC RL01/RL02 DISK-READER/CLONER

## User-Manual for the DE10-Nano board

**Version 1.0 based on Firmware Version 2.5**

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

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

**Reference :** DE10-Nano board : User\_manual.pdf  
**RL02\_01-disk-emulator .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 terasic manual Getting Started Guide

### Background

**This additional program, clonerl is running with the same firmware  
version V2.5 as the rlemulator.**

The RL02 emulator supports a combination of real and emulated RL disk drives.

Thus, it is possible to copy the data from a real disk drive into the emulated environment. However, at present there is still a computer system, e.g. a PDP-11 required. With the new extension, this need is no longer necessary. The new program, called **clonerl** can directly read a RL disk and save it as a .DSK file. In addition, CRC read errors will be corrected before saving the data, this means, rebuild the CRC to save everything that is still possible. The firmware for this extension has been already developed with some additional Verilog programs and is the basis for the applications rlemulator and clonerl in version V2.5. The difference between this two applications is that the rlemulator application works in slave mode and **clonerl** in **master** mode. For the hardware, this means that only the 40pin flat-cable has to be plugged into the RL interface board by 180. degrees as shown in the following overview:

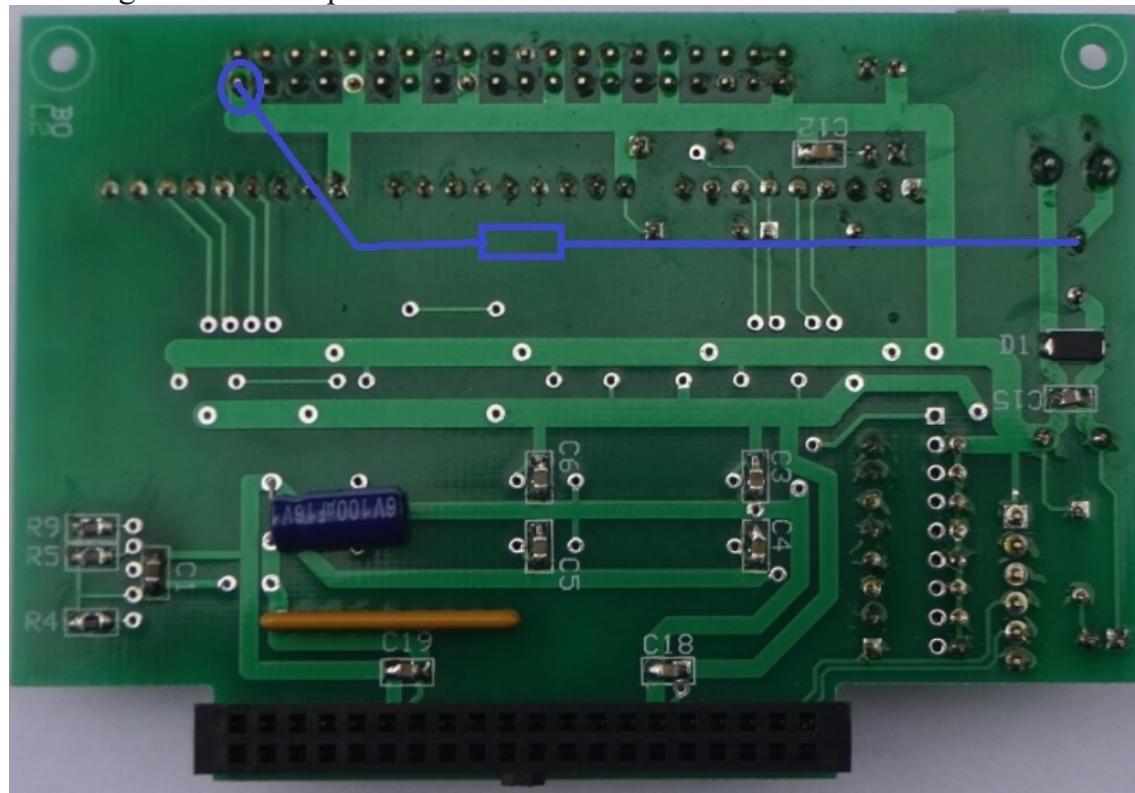
FPGA+PINS		Emulator-Mode		Clone-Mode
AG25=13	B/A=	01 Power_ok - +	39 -----	
AH26=14	D/C=	03 Drive-sel_0 - +	37 Drive-Error + -	
AH24=15	F/E=	05 Drive-sel_1 - +	35 Drive-ready + -	
AF25=16	J/H=	07 Write-gate - +	33 Sector + -	
--	L/K=	09 -----	31 -----	
AG23=17	N/M=	11 System-Clk - +	29 Status-Clk + -	
--	R/P=	13 -----	27 -----	
AF23=18	T/S=	15 Write-data - +	25 Status-IN + -	
--	V/U=	17 -----	23 -----	
AG24=19	X/W=	19 Command - +	21 Read-data + -	
=====				
AH22=20	Z/Y=	21 Read-data - +	19 Command + -	
--	BB/AA=	23 -----	17 -----	
AH21=21	DD/CC=	25 Status-IN - +	15 Write-data + -	
--	FF/EE=	27 -----	13 -----	
AF22=25	JJ/HH=	29 Status-Clk - +	11 System-Clk + -	
--	LL/KK=	31 -----	09 -----	
AG20=27	NN/MM=	33 Sector - +	07 Write-gate + -	
AH19=32	RR/PP=	35 Drive-ready - +	05 Drive-sel_1 + -	
AH18=34	TT/SS=	37 Drive-Error - +	03 Drive-sel_0 + -	
--	VV/UU=	39 -----	01 Power_ok + -	

### However, there is one exception, the **Power-OK** signal.

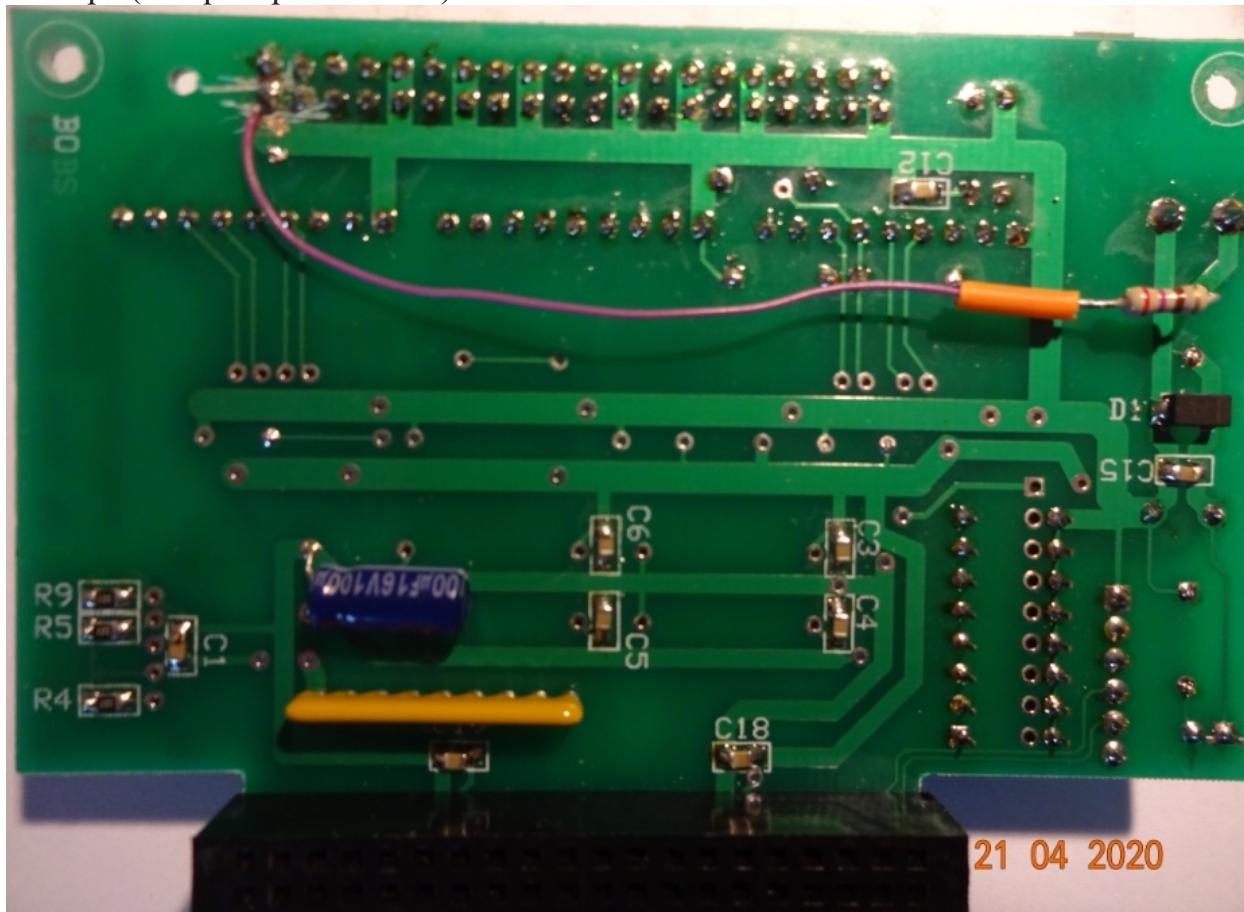
This signal is present at PIN A as input in slave mode and as output at pin VV in master mode. However, the pin VV is connected to ground on the interface board.

#### Todo:

Cut the ground etch to pin VV and connect to +5 VCC via a resistor about 200 Ohm:



Example(not quite professional):



### Jumper settings

**DE10-Nano:** The four slide switches ( page 26, User\_manual ): Select 16 disk sets, 0 to F ( not implemented yet )  
Button 2 and 3 : Reconfigure and Reset/Restart

**Interface-board:** Switch 8: not used

Switch 7 : Force Power OK

Switch 6 : Debug - Mode ON/OFF

Switch 5 : drive typ, RL01 or RL02

Switch 4 - 1 : Configured RL-target drive,  
1=RL3, 2=RL2, 3=RL1, 4=RL0

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

### 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](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 "clonerl.zip" to the DE0-Nano-SoC board, for example, using scp or winscp. Unpack the zip file and navigate to folder clonerl.

```
unzip clonerl
```

```
cd clonerl
```

```
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 ./flash/RL_EMULATOR_SoC.rbf fpga_config_file.rbf
```

**That's all !**

Directory listing:

```
root@socfpga:~/clonerl# ls -l
total 5868
-rwxrwxrwx 1 root root 194993 Apr 21 18:19 RL-cloner-v2_5.pdf
-rwxrwxrwx 1 root root 4660 Apr 21 18:19 README.txt
-rwxrwxrwx 1 root root 4742036 Apr 21 18:19 RL02_01-disk-emulator.pdf
-rwxrwxrwx 1 root root 31817 Apr 21 18:19 clonerl
drwxrwxrwx 2 root root 4096 Apr 21 18:19 flash
lrwxrwxrwx 1 root root 27 Apr 21 18:21 fpga_config_file.rbf
-> ./flash/RL_EMULATOR_SoC.rbf
-rwxrwxrwx 1 root root 13795 Apr 21 18:19 loadrbf
-rwxrwxrwx 1 root root 559820 Apr 21 18:19 pdp11
-rwxrwxrwx 1 root root 43965 Apr 21 18:19 rlemulator
root@socfpga:~/clonerl#
```

Now you can start the **A)firmware loader** and then the **B)clonerl**:

- A)** root@socfpga:~/clonerl# **./loadrbf**
- B)** root@socfpga:~/clonerl# **./clonerl**

The rlemulator program can also be started because it is based on same firmware. The pdp11 simulator, pdp11 from the SIMH project can also be started.

## loadrbf program output:

```
*****
MSEL Pin Config..... 0xa
FPGA State..... Powered Off
cfgwdth Register.... 0x1
cdratio Register.... 0x0
axicfgn 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
axicfgn Register... 0x0
Nconfig pull reg.... 0x0
CONF DONE..... 0x0
Ctrl.en?..... 0x0
*****
root@socfpga:~/socv2_3/RL#
```

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

## clonerl program output:

```
*****> DEC RL01/RL02 Cloner/Reader <*****
SoC/HPS DE10-Nano board V.1.0-I based on FW: V.2.5
(c) WWW.PDP11GY.COM

>>>> Device Type = RL02 <<<<
>>>>> DEBUG-MODE = ON <<<<<
*** Cloning disk-drive: DL0:
***** preset memory *****
```

Started with operating mode: 1100 0000 1010 0001

```
Drive is ready
Drive at cylinder position: 0
Drive positioned to cylinder position : 0

***** Select Mode *****
1=get status, 2=clone disk, 3=read one Cylinder, 4=seek-test : 1

get drive status: 53A = 0000 0101 0011 1010
drive is ready

***** Select Mode *****
1=get status, 2=clone disk, 3=read one Cylinder, 4=seek-test : 3

Cylinder-Nr.(0 to 255 ) :3

rl_command-1: 0000 0001 1000 0101 RAM-Address: 34560
rl_command-2: 0000 0000 0001 0101 RAM-Address: 40320
rl_command-3: 0000 0001 1000 0001 RAM-Address: 0
Saved data in file: RL02_cylinder-3.dsk

***** Select Mode *****
1=get status, 2=clone disk, 3=read one Cylinder, 4=seek-test : 2

Data will be saved in file: RL02_0-clone.dsk
set_cylind_A: 0000 0000 0000 0101 cylinder: 0 RAM_Address: 0
set_cylind_B: 0000 0000 0001 0101
set_cylind_A: 0000 0000 1000 0101 cylinder: 1 RAM_Address: 11520
set_cylind_B: 0000 0000 0001 0101
set_cylind_A: 0000 0000 1000 0101 cylinder: 2 RAM_Address: 23040
set_cylind_B: 0000 0000 0001 0101
.
.
.
set_cylind_A: 0000 0000 1000 0101 cylinder: 511 RAM_Address:
5886720
set_cylind_B: 0000 0000 0001 0101
Saved data in file: RL02_0-clone.dsk

***** Select Mode *****
1=get status, 2=clone disk, 3=read one Cylinder, 4=seek-test :
```

\*\*\*\* SIMH interface \*\*\*\*

```
root@socfpga:~/clonerl# ./pdp11
PDP-11 simulator V3.9-0
sim> set CPU 11/23 512k
Disabling CR
Disabling RK
Disabling HK
Disabling TM
sim> attach RL0 ./RL02_0-clone.dsk
sim> boot RL0
```

RT-11SJ V05.04 C

.SET USR NOSWAP

.SET TT SCOPE

.SET EDIT KED

.INIT/NOQUE VM:

.

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](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\_3.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 (EPCS): Required: Quartus Lite Version 16.1

- De0-Nano-SoC DIP switch (SW10) to EPCS configuration, see page 12 @ User\_manual
- unzip the file "socv2\_3.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 **RL\_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 files are configuration files if you want to convert the .sof file to .jic or .rbf by yourself.

**Note:** This software would also have to be tested with a real RL drive. It has already been successfully tested with 2 DE10 nanobards + interfaces connected together. At the time, in the sad Corona era, I don't have access to a real RL disk drive. I would be very grateful for a cooperation. As soon as it is confirmed that it also works with a real RL Drive, the entire source code will also be published on Github.

Further development: My last job is to implement the rlemluator & cloner in the SIMH project. It is also planned to redesign the RL interface and maybe develop it together with the MFM disk emulator interface. But this is all a question of cooperation and this cooperation probably did not exist ... too bad!

-----  
[www.pdp11gy.com](http://www.pdp11gy.com)  
-----

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>  
<http://www.pdp11gy.com/1AE.html>  
<http://www.pdp11gy.com/doneE.html>  
<https://github.com/pdp11gy/SoC-HPS-based-MFM-disk-emulator>