User Guide of PG tool for RTL8111EP

1 Package Content

1.1 Folder “linuxpg”

This includes the source code of the driver pgdrv.ko, executable binary files, and the required configuration files.

I. 68EPSPI.cfg: configuration file of the firmware.

II. 68EPSPIB.bin: firmware of RTL8111EP

III. 8168EPEF.cfg: configuration file of RTL8111EP

IV. pgdrv.c and pgdrv.h: source code of the driver of PG program.

V. rtnicpg-arm-cortex: executable binary of the PG program for the CPU of arm cortex.

VI. rtnicpg-i686: executable binary of the PG program for the CPU of x86 32 bit.

VII. rtnicpg-mipsel-32r2: executable binary of the PG program for the CPU of mips.

VIII. rtnicpg-x86_64: executable binary of the PG program for the CPU of x86 64 bit.

1.2 Folder “src”

The source code of the PG program.

2 Knowledge About Using the PG Program

The PG program has its own driver (pgdrv.ko) and it conflicts with the Ethernet driver (r8169 or r8168), so you should unload the Ethernet driver before using the PG program. That is, the Ethernet driver must not be build-in for the kernel image. You should remove the Ethernet driver or build it as a module to make it possible to unload the Ethernet driver. Besides, the PG program should be run as “root”, otherwise it would fail.

The brief steps would be:

1. Build the driver of PG program for the target platform.
2. Unload the Ethernet driver.
3. Load the driver of PG program.
4. Run the PG program.

3 Build The Source Code

3.1 Build the driver

It depends on how you build the driver for the target platform. You have to use “pgdrv.c” and “pgdrv.h”, and follow that method to build “pgdrv.ko”.

3.2 Build the PG program

Firstly, you should have the relative toolchain which includes the compiler binary files (gcc, g++, and so on) and header files for the target platform. Secondly, you have to set some environment variables.

PATH: this should include the path of the compiler.
CROSS_PATH: path of the relative header files.
CROSS_COMPILE: prefix for the compiler.

For example,

```
# export PATH=$PATH:/root/buildroot/output/host/usr/bin
# export CROSS_PATH=/root/buildroot/output/staging
# export CROSS_COMPILE=i686-linux-
```

You could refer to the “Makefile” to know how they are used.

Finally, you could build the application by running “make”.

```
# make
```

Note that, the PG program supports little-endian platform only. If you build it for big-endian platform, it may have unexpected behavior.

4 Usage of the PG Program

4.1 Program the RTL8111EP chip

The configuration file is "8168EPEF.cfg". You could change the settings by editing it or add "#" in front of the item to skip the setting. After everything are ready, program the chip by the following command.

```
# ./rtnicpg /efuse /w
```

You could check the context by running

```
# ./rtnicpg /efuse /r
```

Note1: You may need to replace rtnicpg with rtnicpg-xxx according to what CPU you use.

Note2: The MAC address in the configuration file would be increased automatically if the process is success. That is, you don't have to edit the item of MAC address of the configuration file for another IC.

Note3: Please shutdown the system and AC off to make sure all settings would work.

4.2 Program the firmware of RTL8111EP

Make sure you have the firmware file "68EPSPIB.bin" and edit the configuration file "68EPSPI.cfg" if it is necessary. Then, run

```
# ./rtnicpg /flash /wr
```

4.3 Verify the firmware of RTL8111EP

Compare the firmware in the flash with "68EPSPIB.bin". This only checks code block, excludes the parameter sectors.

```
# ./rtnicpg /flash /verify
```

4.4 Show the version of the firmware

Show the version of the firmware in the flash

```
# ./rtnicpg /flash /version
```

4.5 Show the parameters

Show the version of the firmware and some paramters

```
# ./rtnicpg /flash /canon
```