

Utilize Ronetix Development Tools Virtual Machine to build Android software stack for PM9G45

Demonstration



Acknowledgement

December, 2011

Ronetix has made every attempt to ensure that the information in this document is accurate and complete. However, Ronetix assumes no responsibility for any errors, omissions, or for any consequences resulting from the use of the information included herein or the equipment it accompanies. Ronetix reserves the right to make changes in its products and specifications at any time without notice. Any software described in this document is furnished under a license or non-disclosure agreement. It is against the law to copy this software on magnetic tape, disk, or other medium for any purpose other than the licensee's personal use.

Ronetix Development Tools GmbH

Waidhausenstrasse 13/5

1140 Vienna

Austria

Tel: +43-720-500315

+43-1-956-3138

Fax: +43-1-8174-955-3464

Web: www.ronetix.at

E-Mail: info@ronetix.at

Acknowledgments:

ARM, ARM7, ARM9, ARM11 are trademarks of ARM Ltd. Windows, Win32, Windows CE are trademarks of Microsoft Corporation. Ethernet is a trademark of XEROX. All other trademarks are trademarks of their respective companies.

Copyright© 2010-2011 Ronetix Development Tools GmbH

All rights reserved.

Contents

How to play with Android on PM9G45 using the Ronetix Development Tools VM 4
Reference 7

Change log

December 2011	- Initial release.
---------------	--------------------

This demonstration is based on [Android ReadMe](#) and aims to show you how to utilize the provided Virtual Machine to build and work with Android on SK-PM9G45.



The things you will need are:

- disk space of 10GiB for the tar.bz2 archive of the VM on the host PC
- disk space of 21GiB to play the VM
- one COM port (COM1 and so on, or ttyS0 and so on) on the host PC
- Ronetix Development Tools Virtual Machine
- installed VMware Player
- SK-PM9G45 board
- Ethernet switch/hub or vacant LAN port on the host PC

How to play with Android on PM9G45 using the Ronetix Development Tools VM

At first, some cables have to be connected. Connect the SK-PM9G45 to an Ethernet together with the host PC. The VM Ethernet adapter is bridged to the Ethernet of the host PC and gets its IP from a DHCP server. Static IP can be set, too.

The serial console to the board will be needed, too. To be able to attach the COM port you have to ensure that no software is using the intended COM port.

Attach the serial port to the VM if it is not - the symbol  means that serial port is attached, and the presence of the cross  shows disconnected serial port. The default serial port to be attached is /dev/ttyS0, also named COM1 in DOS/Windows environment. Which serial port to attach is selected in the VM(virtual machine) configuration, which configuration is editable in VMware player.

Now it is time to play the VM. It will take around 10 minutes until it gets ready to play with. From now on all commands and activities have to be done in the VM if something else is not said.

Edit the PEEDI configuration file and change the IP address of `CORE0_PATH` parameter to match the IP of the VM.

Open the "Terminal Emulator"(its icon is on the desktop) and in it type

```
dev@ronetix:~$ screen /dev/ttyS0 115200
```

to create serial terminal console on the ttyS0/COM1. And do not type `dev@ronetix:~$`, this is the command prompt.

In the same Terminal Emulator window make a new tab with keyboard combination Ctrl-Shift-T(or in the menu File->New tab). In which new tab telnet to your PEEDI programming box as shown(substitute that IP with the one of your PEEDI).

```
dev@ronetix:~$ telnet 192.168.10.42
```

On the VM there are anonymous FTP and TFTP servers with base directory pointed at /var/lib/tftpboot. The files to write using PEEDI have to be located at that base directory. A symbolic(soft) link to the real files from that directory will not work because of the chrooted security options on the ftp and tftp servers, so move the files. The files have already been build so no need to rebuild them. The next five commands move the files to that base directory.

```
dev@ronetix:~$ cd /var/lib/tftpboot/pm9g45
```

```
dev@ronetix:/var/lib/tftpboot/pm9g45$ mv /home/dev/usr/src/android/
out/target/product/pm9g45/pm9g45.jffs2 ./
dev@ronetix:/var/lib/tftpboot/pm9g45$ mv /home/dev/usr/src/at91boots
trap-pm9g45-svn-r286/binaries/pm9g45-nandflashboot-2.13.bin ./
dev@ronetix:/var/lib/tftpboot/pm9g45$ mv /home/dev/usr/src/u-boot-
2010.09/u-boot.bin ./
dev@ronetix:/var/lib/tftpboot/pm9g45$ mv /home/dev/usr/src/linux-2.6.30/
arch/arm/boot/uImage ./
```

To program the images go to the PEEDI telnet console and run the next commands.

To program the SK-PM9G45 with all images, its NAND memory have to be empty.

```
PM9G45> f s 0
Selected FLASH section:  BOOTSTRAP
PM9G45> f e
Performing mass erase ...
```

Next, the AT91Bootstrap have to be written:

```
PM9G45> f p pm9g45/pm9g45-nandflashboot-2.13.bin bin 0x0
++ info: Programming image file:  tftp://192.168.10.71/pm9g45/
pm9g45-nandflashboot-2.13.bin
++ info: Programming using agent, buffer = 32768 bytes
++ info: At absolute address:  0x00000000
programming at 0x00000000

++ info:  successfully programmed 7.10 KB in 0.03 sec
```

The U-Boot programming follows:

```
PM9G45> f s 1
Selected FLASH section:  U-BOOT
PM9G45>
PM9G45> f p pm9g45/u-boot.bin bin 0x20000
++ info: Programming image file:  tftp://192.168.10.71/pm9g45/u-boot.bin
++ info: Programming using agent, buffer = 32768 bytes
++ info: At absolute address:  0x00020000
programming at 0x00020000
programming at 0x00028000
programming at 0x00030000
programming at 0x00038000
programming at 0x00040000
programming at 0x00048000
programming at 0x00050000

++ info:  successfully programmed 219.19 KB in 0.58 sec
```

Time to put the Linux kernel in the NAND:

```
PM9G45> f s 2
Selected FLASH section:  KERNEL
PM9G45> f p pm9g45/uImage bin 0x200000
++ info: Programming image file:  tftp://192.168.10.71/pm9g45/uImage
++ info: Programming using agent, buffer = 32768 bytes
++ info: At absolute address:  0x00200000
programming at 0x00200000
programming at 0x00208000
...
programming at 0x003F0000
programming at 0x003F8000

++ info:  successfully programmed 1.98 MB in 6.03 sec
```

The last image to program is the Android file system:

```
PM9G45> f s 3
Selected FLASH section:  ROOTFS
PM9G45> f p pm9g45/pm9g45.jffs2 bin 0x500000
++ info: Programming image file:  tftp://192.168.10.71/pm9g45/pm9g45.jffs2
++ info: Programming using agent, buffer = 32768 bytes
++ info: At absolute address:  0x00500000
programming at 0x00500000
programming at 0x00508000
...
programming at 0x02770000
programming at 0x02778000

++ info:  successfully programmed 34.50 MB in 104.45 sec
```

Finally, detach the target from the PEEDI ether with detaching the JTAG cable or with PEEDI detach command.

```
PM9G45> ta de
++ info: power fail detected
++ info: no target power
The target is now detached
PM9G45>
```

The PEEDI is detached from the target and the SK-PM9G45 is ready to go.

The SK-PM9G45 have to be power-cycled to boot properly. Power cycle it. It will take 2-3 minutes to load completely and the Android desktop will appear.

It is time to upload test applications, which comes with the Android SDK, using Eclipse and ADB(Android Debug Bridge).

Start Eclipse and create a new project like this: menu File->New->Project...-> "Android Sample Project", select Android 2.1 check box, press Next button and select the Snake demo.

Once more in the Eclipse menus do: Run->Run Configurations...
then on the "Android Application" right click with the mouse and select New from the popup menu.
In the text box labeled "Name:" change the name from "New_configuration" to "Snake demo".
Click on the "Browse..." button next to the "Project:" text field and select Snake.
Open the Target tab and click on the Manual radio button.
Hit the Apply button to save the changes, then close that window.

Go to the Terminal Emulator and in the tab where screen is running do:

```
# ifconfig eth0
eth0: ip 192.168.10.99 mask 255.255.255.0 flags [up broadcast running
multicast]
```

Then in the Terminal Emulator create new tab and run:

```
dev@ronetix:~$ adb connect 192.168.10.99
connected to 192.168.10.99:5555
```

Unlock the Android screen if it is locked by pressing SW3 or SW4 buttons on SK-PM9G45 and with a finger drag the lock toward the Sound button.

Go to the Eclipse->Run->Run, select the line with 192.168.10.99:5555 and press OK. The application will start in a few seconds.

Note that the IP's are as the document writer sees them in his environment.

On the SK-PM9G45 screen appears the message "Press Up To Play" ... OOOPS.

The SK-PM9G45 do not have enough buttons to play that game, but it has extendable header pins with GPIO functions in them. You may create(not an easy task, without a list of vacant pins) buttons yourself using [Android ReadMe](#) and [PM9G45 datasheet](#). If you give up making buttons call us or try another demo.

Reference

Android ReadMe

<http://download.ronetix.info/boards/README/README-android.ronetix>

PM9G45 datasheet

<http://download.ronetix.info/boards/doc/PM9G45/SK-PM9G45-datasheet.pdf>