Application Note No.	LAN-069e_0
Version:	Preliminary
Author:	M. Dzierzawa
Date:	14.06.2016

Historie:

Version	Changes	Date	Author
Preliminary	First draft	14.06.2016	M. Dzierzawa

phyCORE-RK3288 Alpha Kit
Getting Started

Content

1 Introduction

This Quickstart describes the tools and provides the know-how to install and work with the *Linux* Board Support Package (BSP) for the phyCORE-RK3288 Alpha Kit. This Quickstart shows you how to do everything from installing the appropriate tools and sources, to building custom kernels, to deploying the OS in order to operate the software and hardware.

2 Requirements

The following system requirements are necessary to successfully complete this Quickstart. Deviations from these requirements may suffice, or may have other workarounds.

Hardware:

- phyCORE-RK3288 SOM (PCM-059)
- phyCORE-RK3288 alpha carrier board (PCM-946)
- Serial null modem cable (RS-232)
- AC adapter supplying 12 V – 24 V DC/ min. 2.5 A

Software:

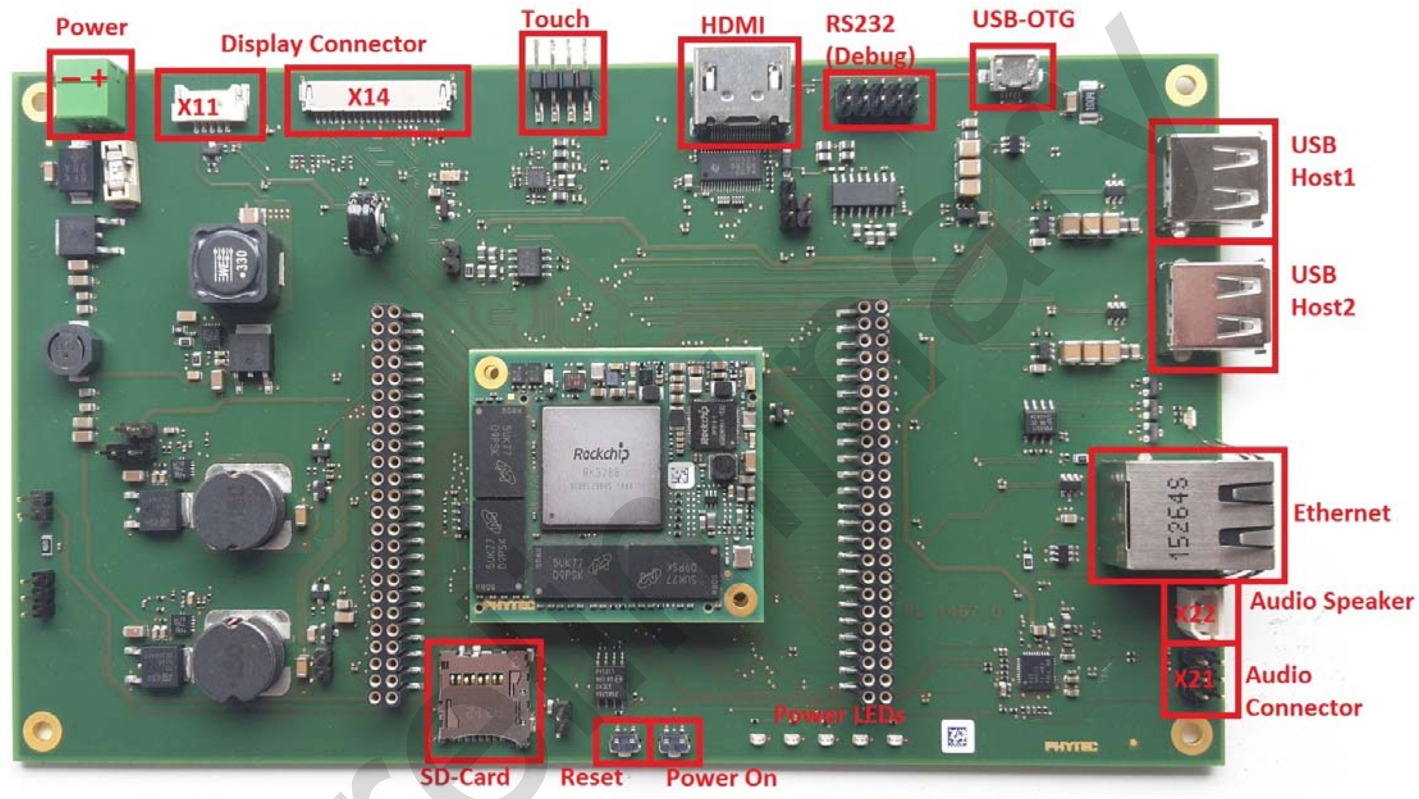
- A modern *Linux* Operating host system either natively or via a virtual machine:
  - *Ubuntu 14.04 LTS* 64-bit recommended. Other distributions will likely work, but note that some setup information as well as OS specific commands and paths may differ.
  - If you want to use a virtual machine, VMWare Workstation, VMWare Player, and VirtualBox are possible solutions.
- Root access to your *Linux* host PC. Some commands in the Quickstart will not work if you do not have *sudo* access (e.g. package installation, formatting an SD card).
- At least 40 GB to 50 GB free on the build partition of the host PC.
- An SD card reader operational under *Linux*.
  - If you do not have SD card access under *Linux* on your host PC then formatting, copying the bootloader, and mounting the root file system on an SD card will not be possible.
- An active internet connection.

3 Getting Started with the included SD Card

This section is designed to get the board up and running with the SD Card included in the kit, which is prepared with the pre-built images.

3.1 Connector Interfaces

The following picture shows the phyCORE-RK3288 alpha carrier board which is used in this Quickstart and highlights the location of the different interfaces on the board.



3.2 Booting from the SD Card

This section describes how to boot the phyCORE-RK3288 with the pre-built images on the included SD card.

- Insert the micro SD card into the SD card slot of board.
- Connect UART cable to the 5x2 pin header labeled "RS232 (Debug)". This header requires an adapter as well as Null modem cable. When plugged in, the adapter cable should be oriented towards the USB and Ethernet interfaces.
- Start your favorite terminal software (e.g. *Minicom*,or *Tera Term*) on your host PC and configure it for 115200 baud, 8 data bits, no parity, and 1 stop bit (8n1) with no handshake.
- Connect the power supply to the power connector (Please note the polarity shown in picture above).

The five Power LEDs D7 to D11 will light up and the board starts booting into *Linux*. The console output can be viewed in your terminal window. If everything was done correctly the login prompt will be shown at the end of the booting process:

```
Yogurt (Phytec Example Distribution) RK3288-ALPHA1 phycore-rk3288-1 ttyS0
phycore-rk3288-1 login:
```

The default login account is *root* with an empty password.

4 Building the BSP

This section will guide you through the general build process of the RK3288 BSP using the phyLinux script. If you want to use our software without phyLinux and the *Repo* tool managed environment instead, you can find all *Git* repositories on
git://git.phytec.de

Used *U-Boot* repository:
git://git.phytec.de/u-boot

Our *barebox* version is based on the *barebox* mainline and adds only a few patches which will be sent upstream in future. Used *Linux* kernel repository:
git://git.phytec.de/linux-mainline

To find out which tag is used for a specific board, have a look at:
meta-phytec/recipes-bsp/u-boot/u-boot_*.bb
meta-phytec/recipes-kernel/linux/linux-mainline_*.bb

4.1 Get the BSP

- Create a fresh project directory, e.g.

```
host$ mkdir ~/yocto
```

- Download and run the phyLinux script

```
host$ cd ~/yocto
host$ wget ftp://ftp.phytec.de/pub/Software/Linux/Yocto/Tools/phyLinux
host$ chmod +x phyLinux
host$ ./phyLinux init
```

4.2 Basic Set-Up

There are a few important steps which have to be done, before the main build process.

- Setting up the host, see *Yocto* Reference Manual "Setting up the Host"
- Setting up the *Git* configuration, see *Yocto* Reference Manual "*Git* Configuration"

4.3 Finding the right Software Platform

The RK3288 BSP is planned as a unified BSP, which means in the future it will support a set of different Phytec carrier boards (CB) with different Systems on Module (SOMs). However this ALPHA BSP supports only one Phytec machine:
phycore-rk3288-1

4.4 Selecting a Software Platform

- To select the correct SoC, BSP version and platform call:

  ```
  host$ ./phyLinux init
  ```

It is also possible to pass this information directly using command line parameters:

```
host$ ./phyLinux init -p rk3288 -r ALPHA1 -m phycore-rk3288-1
```

Please read section "Initialization" in the *Yocto* Reference Manual for more information.

4.5 Starting the Build Process

Refer to *Yocto* Reference Manual "Start the Build".

- If there are no U-Boot files in the deploy directory, start the U-Boot build manually with:

  ```
  host$ bitbake u-boot
  ```

4.6 BSP Images

All images generated by *Bitbake* are deployed to *yocto/build/deploy/images/<machine>*.

The following list shows for example all files generated for the RK3288 SoC, *phycore-rk3288-1* machine:

- U-boot: u-boot-dtb.img
- U-boot SPL
  - for SPI flash: u-boot-spl-dtb.bin.rkspi
  - for SDMMC: u-boot-spl-dtb.bin.rksd
  - for USB OTG: u-boot-spl-dtb.bin.rkimage
- Kernel: zImage
- Kernel device tree file: zImage-rk3288-pcm946.dtb
- Kernel configuration: zImage-defconfig
- Root filesystem: phytec-qt5demo-image-phycore-rk3288-1.tar.gz

5 Booting the System

The RK3288 provides system boot from off-chip devices such as SD/MMC card, NAND Flash, SPI NOR Flash, eMMC memory, and trough the USB OTG interface. It is not possible to select a special boot source, instead the RK3288 will try to boot from all devices in the following order: NAND Flash -> eMMC -> SPI NOR Flash -> SD/MMC card -> USB OTG

Please note that Phytec RK3288 modules are not equipped with NAND Flash.

The RK3288 has a two stage boot process. At power-up an RK3288 device begins booting from internal boot ROM. This code is fixed during the manufacturing process and cannot be altered. The ROM code will start checking all boot devices for a valid boot entry block. Once a boot entry is found, the ROM code will copy the content of the boot device into the SRAM. Control is then passed to this first external bootloader called SPL (U-Boot). The SPL now initializes the SDRAM, (PMIC) and other important modules of the SoC. In the next step, the SPL will copy the U-Boot into the SDRAM and pass the control to it.

From here we have a running bootloader on the system. It is now possible to boot an operating system.

5.1 Booting from SD Card

In this chapter the steps to create a bootable SD card are explained.

⚠	Be very careful when selecting the right drive as all files on the selected device will be erased! Selecting the wrong drive can erase your hard drive!

- Get the correct device name (<your_device>) with `sudo fdisk -l`, or from the last outputs of `dmesg` after inserting the SD card.
- Now use the commands described in the next sections to create your bootable SD card.

5.1.1 Setting up SD card partition layout

This is just an example. Of course, you can also choose your own layout.

- Create a partition table:

```
host$ sudo parted /dev/sdb mktable msdos
```

- Create two partitions, e.g. one for the kernel and DT, and one for the root filesystem:

```
host$ sudo parted /dev/sdb mkpart primary 4 512
host$ sudo mkfs.fat /dev/sdb1
host$ sudo parted /dev/sdb mkpart primary 513 100%
host$ sudo mkfs.ext4 /dev/sdb2
```

- Copy your kernel (zImage) and device tree blob to the first partition.
- Copy your root filesystem to the second partition:

```
host$ mount /dev/sdb2 x
host$ sudo tar xf phytec-headless-image-phycore-rk3288-1.tar.gz -C x
```

5.1.2 Copying U-Boot to the SD Card

Please read *u-boot/doc/README.rockchip* for more information.

- Copy SPL:

```
host$ sudo dd if=u-boot-spl-dtb.bin.rksd of=/dev/<your_device> seek=64
```

- Copy u-boot:

```
host$ sudo dd if=u-boot-dtb.img of=/dev/<your_device> seek=256
```

  where <your_device> could be for example "sde", depending on your system. Make sure that you choose the right partition!

- Now insert the micro-SD card into your board and reset it. You should see something like:

```
U-Boot SPL 2016.01-00004-gbdd1ffb-dirty (May 03 2016 - 11:45:29)
Trying to boot from MMC


U-Boot 2016.01-00004-gbdd1ffb-dirty (May 03 2016 - 11:45:29 +0200)

Model: phycore-rk3288
DRAM:  1 GiB
MMC:   dwmmc@ff0c0000: 0, dwmmc@ff0f0000: 1
In:    serial@ff180000
Out:   serial@ff180000
Err:   serial@ff180000
Net:   Net Initialization Skipped
No ethernet found.
Hit any key to stop autoboot:  0
=>
```

5.1.3 U-Boot Environment

This is an example for a U-Boot environment configuration. To get your kernel up and running you will need to set two environment variables:

Kernel boot arguments:

```
bootloader$ setenv bootargs console=ttyS0,115200 earlyprintk
 init=/sbin/init root=/dev/mmcblk0p2 rw rootfstype=ext4 rootwait mem=512M
```

The mem=512M parameter forces the kernel to use a specific amount of memory. At the moment we are only able to use half of the available RAM (512 MB). This seems to be a hardware problem. RK3288's memory controller is not able to detect the correct RAM settings (row, column, bank and rank).

Boot command:

```
bootloader$ setenv bootcmd fatload mmc 0:1 0x0 zImage\; fatload mmc 0:1
            0x01f00000 zImage-rk3288-pcm946.dtb\; bootz 0x0 - 0x01f00000
```

For more information, see *u-boot/doc/README.rockchip*.

6 Device Tree (DT)

6.1 Introduction

The following text describes briefly the Device Tree and can be found in the *Linux* kernel (*linux/Documentation/devicetree/usage-model.txt*).

"The "Open Firmware Device Tree", or simply Device Tree (DT), is a data structure and language for describing hardware. More specifically, it is a description of hardware that is readable by an operating system so that the operating system doesn't need to hard code details of the machine.
Structurally, the DT is a tree, or acyclic graph with named nodes, and nodes may have an arbitrary number of named properties encapsulating arbitrary data. A mechanism also exists to create arbitrary links from one node to another outside of the natural tree structure.
Conceptually, a common set of usage conventions, called 'bindings', is defined for how data should appear in the tree to describe typical hardware characteristics including data busses, interrupt lines, GPIO connections, and peripheral devices."

The kernel is a really good source for a DT introduction. An overview of the device tree data format can be found on the device tree usage page at devicetree.org:
http://devicetree.org/Device_Tree_Usage

6.2 Phytec RK3288 BSP Device Tree Concept

This BSP is in an alpha state. The DT concept is under development.

7 Accessing Peripherals

The following sections provide an overview of the supported hardware components and their corresponding operating system drivers. Further changes can be ported upon customer request.

To find out which boards and modules are supported by the release of Phytec's RK328x unified BSP described herein, visit our web page at http://www.phytec.de/produkte/software/yocto/phytec-unified-yocto-bsp-releases/ and click the corresponding BSP release. Now you can find all hardware supported in the columns "Hardware Article Number" and the correct machine name in the corresponding cell under "Machine Name". (BSP-Yocto_Machines)

To achieve maximum software re-use, the *Linux* kernel offers a sophisticated infrastructure, layering software components into board specific parts. The BSP tries to modularize the kit features as far as possible, which means that when a customized baseboard, or even a customer specific module is developed, most of the software support can be re-used without error-prone copy-and-paste. The kernel code corresponding to the boards above can be found in device trees (DT) under *linux/arch/arm/boot/dts/*.dts**.

In fact, software re-use is one of the most important features of the *Linux* kernel and especially of the ARM implementation, which always had to fight with an insane number of possibilities of the System-on-Chip CPUs.

The whole board specific hardware is described in DTs and is not part of the kernel image itself. The hardware description is in its own separate binary, called device tree blob (DTB).

7.1 Serial TTYs

The RK3288 SoC provides up to 5 so called UART units.

From the command line prompt of *Linux* user space you can easily check the availability of a specific UART interfaces with:

```
target$ echo "test" > /dev/ttyS0
```

Be sure that the baud rate is set correctly on host and target side.

7.2 Network

The Ethernet feature provided by our module is a single gigabit port.

The interface offers a standard *Linux* network port which can be programmed using the BSD socket interface.
The whole network configuration is handled by the *systemd-networkd* daemon. The relevant configuration files can be found on the target in */lib/systemd/network/* and also in the BSP in *meta-yogurt/recipes-core/systemd/systemd/*.

IP addresses can be configured within *.*network* files. The default IP address and netmask for eth0 is:
eth0: 192.168.3.11/24

7.3 MMC/SD Card and eMMC

The phyCORE-RK3288 supports a slot for (micro) Secure Digital Cards and Multi Media Cards to be used as general purpose block devices. These devices can be used in the same way as any other block device.

⚠	This kind of devices are hot pluggable, nevertheless you must pay attention not to unplug the device while it is still mounted. This may result in data loss.

After inserting an MMC/SD card, the kernel will generate new device nodes in */dev*. The full device can be reached via its */dev/mmcblk0* device node, MMC/SD card partitions will show up in the following way:

```
/dev/mmcblk0p<Y>
```

<*Y*> counts as the partition number starting from 1 to the max. count of partitions on this device.

The partitions can be formatted with any kind of file system and also handled in a standard manner, e.g. the *mount* and *umount* command work as expected.

	▪ These partition device nodes will only be available if the card contains a valid partition table (”hard disk” like handling). If it does not contain one, the whole device can be used as a file system (”floppy” like handling). In this case */dev/mmcblk0* must be used for formatting and mounting.

7.4 GPIO LEDs

I/O pins like LEDs can be accessed from userspace. They appear in */sys/class/leds/*. The maximum brightness of the LED is defined in max_brightness file. The brightness file will set the brightness of the LED (taking a value 0 to max_brightness). Most LEDs do not have hardware brightness support so will just be turned on for non-zero brightness settings.
Here is a simple example for the phyCORE-RK3288 Carrier Board (PCM-946):

- To get all LEDs available, type:
  `target$ ls /sys/class/leds`

```
green_led
```

The RK3288 SOM itself has a GPIO LED which is used as a heartbeat by the kernel.

Configuration in device tree *rk3288-phycore-som.dtsi*:

```
/* LED */
&pinctrl {
        leds {
              user_led: user-led {
                    rockchip,pins = <7 2 RK_FUNC_GPIO &pcfg_output_high>;
                };
        };
};

&leds {
        pinctrl-names = "default";
        pinctrl-0 = <&user_led>;

        user {
                label = "green_led";
                gpios = <&gpio7 2 GPIO_ACTIVE_HIGH>;
                linux,default-trigger = "heartbeat";
                default-state = "keep";
        };
};
```

7.5 I^{2}C Bus

There are two I^{2}C controllers in the bus sub-system: I2C PMU, and I2C AUDIO. Four additional I^{2}C controllers are in the peripheral sub-system: I2C SENSOR, I2C CAM, I2C_TP, and I2C_HDMI. Phytec boards provide plenty of different I²C devices connected to the I²C modules of the RK3288. This the following sections describe the basic device usage and the DT representation of some of the integrated I²C devices.

7.5.1 EEPROM

It is possible to read/write directly to the device:

```
target$ /sys/class/i2c-adapter/i2c-0/0-0050/eeprom
```

- E.g. to read and print the first 1024 bytes of the EEPROM as hex number execute:

```
target$ dd if=/sys/class/i2c-dev/i2c-0/device/0-0050/eeprom bs=1
                                                    count=1024  | od -x
```

- E.g. to fill the device with zeros use:

```
target$ dd if=/dev/zero of=/sys/class/i2c-adapter/i2c-0/0-0050/eeprom
                                                        bs=4096 count=1
```

DT representation, e.g. in *rk3288-phycore-som.dtsi*:

```
i2c_eeprom: eeprom@50 {
    compatible = "atmel,24c32";
    pagesize = <32>;
    reg = <0x50>;
};
```

	There are two EEPROMs on the phyCORE-RK3288 SOM. The EEPROM with address 0x51 is a module configuration EEPROM and should be not used. It is reserved for Phytec. The EEPROM device with address 0x50 is freely available.

7.5.2 RTC

RTCs can be accessed via */dev/rtc**. Because Phytec boards have often more than one RTC, there might be more than one RTC device file.

- To find out the name of the RTC device you can read its sysfs entry with:

```
target$ cat /sys/class/rtc/rtc*/name
```

You will get for example:

```
rv4162
rk808-rtc
```

	This will list all RTCs including the non-I^{2}C RTCs. *Linux* assigns RTC devices IDs based on the device tree */aliases* entries if present.

rk3288-phycore-som.dtsi:

```
aliases {
    rtc0 = &i2c_rtc;
    rtc1 = &rk818;
};
```

As the time is set according to the value of rtc0 during system boot rtc0 should be always the RTC that is being backed up.

Date and time can be manipulated with the *hwclock* tool, using the *-w* (systohc) and *-s* (hctosys) options.

To set the date first use *date* and then run *hwclock -w -u* to store the new date into the RTC. For more information about this tool refer to the manpage of *hwclock*.

DT representation for I²C RTCs:
rk3288-pcm946.dts:
```
i2c_rtc: rtc@68 {
        compatible = "rv4162";
        reg = <0x68>;
};
```

7.6 USB Host Controller

The USB controller of the RK3288 SoC provides a low-cost connectivity solution for numerous consumer portable devices by providing a mechanism for data transfer between USB devices with a line/bus speed up to 12 Mbps. USB HOST2.0 supports Non_OTG Host functions and is fully compliant with USB2.0 specification, and support high-speed (480 Mbps), full-speed (12 Mbps), low-speed (1.5 Mbps) transfer.

The unified BSP includes support for mass storage devices and keyboards. Other USB related device drivers must be enabled in the kernel configuration on demand.

Due to *udev*, all mass storage devices connected get unique IDs and can be found in */dev/disks/by-id*. These IDs can be used in */etc/fstab* to mount the different USB memory devices in different ways.

7.7 USB OTG

USB OTG 2.0 is a Dual-Role Device controller, which supports both device and host functions and is fully compliant with OTG supplement to USB2.0 specification, and support high-speed (480 Mbps), full-speed (12 Mbps), low-speed (1.5 Mbps) transfer.

In order to connect the board as USB device to an USB host port (for example a PC), you need to load the appropriate USB gadget, which is a kernel module, with *modprobe*.

Examples:

- To start the Ethernet gadget execute:
  ```
  target$ modprobe g_ether
  ```
  on the device. You will get an additional Ethernet interface, e.g. *usb0*.

For using USB OTG as host, please load the module *g_zero*.

7.8 Framebuffer

This driver gains access to a display connected to the carrier board via device node */dev/fb0*.

- To run a simple test of the framebuffer feature execute:
  ```
  target$ fbtest
  ```

This will show various pictures on the display.

- Information about the framebuffer's resolution can be obtained with:
  ```
  target$ fbset
  ```

fbset cannot be used to change display resolution or color depth. Depending on the framebuffer device different kernel commands are mostly needed to do this. Some documentation can be found in the kernel documentation at *https://www.kernel.org/doc/Documentation/fb/modedb.txt*.

Please also refer to the manual of your display driver for more details.

Please contact our technical support, if you need additional information, or if you have any questions.

Europe (except France):	**France:**
▪ +49 6131 9221-31	▪ +33 2 43 29 22 33
▪ *support@phytec.de*	▪ *support@phytec.fr*
North America:	**India:**
▪ +1 206 780-9047, or +1 800 278-9913	▪ +91-80-4086 7047
▪ *support@phytec.com*	▪ *support@phytec.in*
China:	
▪ +86-755-3395-5875	
▪ support@phytec.cn	