PhyCORE-OMAP44xx Linux Quickstart PD12.2.0

1 About this Quickstart

This document describes how to install and work with the Linux Board Support Package (BSP) for the phyCORE-OMAP44XX platform. This BSP provides a fundamental software platform for development, deployment and execution on the phyCORE-OMAP44XX.

The Quickstart contains instructions for:

- Host Setup
- Board Setup
- Building a BSP (Platform, Kernel, Root Filesystem)
- Flashing images to NAND (MLO, Barebox, Kernel, Root Filesystem)

2 Host Setup

The phyCORE-OMAP44xx (PCM-049) has been developed and tested with Ubuntu 10.04 LTS Lucid Lynx [Installation Guide]. Although it is possible to use a different OS, some setup information will contain OS-specific commands and paths for settings.

Update repositories and upgrade installed packages:

```
sudo apt-get update
sudo apt-get upgrade
```

2.1 Server Setup

Support for installing and setting up TFTP, NFS, and Samba server settings to enable communication between multiple systems and the target.

2.1.1 TFTP

TFTP allows files to be downloaded from one machine to another. With most embedded Linux devices, TFTP is an efficient way to boot the kernel during development so that the user does not have to flash a new kernel every time it is modified. It is also helpful when updating images in flash from Barebox.

First, start by installing the TFTP server.

```
sudo apt-get install tftpd-hpa
```

Next, files can be accessed from another machine on the same network by simply using the IP address of the host. Specify a folder where the files will reside on the host by replacing the folder path for TFTP_DIRECTORY with whatever folder you wish to use as your TFTP file storage location, or leave the folder as the default.

```
sudo gedit /etc/default/tftpd-hpa

# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/var/lib/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```

If you made any changes to the settings of the TFTP server, you need to restart it for them to take effect.

```
sudo restart tftpd-hpa
```

Lastly, if you would like to grant every user on the system permission to place files in the TFTP directory, use the following command, replacing <TFTP_DIRECTORY> with your chosen location.

```
sudo chmod ugo+rwx <TFTP_DIRECTORY>
```

2.1.2 NFS

A network file-system (NFS) server gives other systems the ability to mount a file-system stored on the host and exported over the network. In embedded development, this is essential for quickly testing applications and evaluating different file-system setups.

To begin the installation process use the following command:

```
sudo apt-get install nfs-kernel-server
```

Exported filesystems are designated in the "/etc/exports" file and allow you to choose both the directory to be exported and many settings for accessing the exports. Below is an example for exporting a folder called "nfs_export-ex" located in a user's home directory.

```
sudo gedit /etc/exports

# /etc/exports

/home/<user>/nfs_export-ex  *(rw,sync,no_root_squash,no_subtree_check)
```

The options (rw, sync, no_root_squash, no_subtree_check) for this folder are essential in setting up the NFS export correctly. For more information on additional options, refer to the man page for 'exports'.

- rw enables

read and write access when the directory is mounted

- sync

tells the file-system to handle local access calls before remote access

- no_root_squash

allows root access when mounting the file-system

- no_subtree_check

reduces the number of checks the server must make to ensure that an exported sub-directory is within an exported tree and also enables access to root files in conjunction with no_root_squash

After modifying this file, in order to mount the directories as an NFS, you must force the NFS server to export all of the directories listed in "/etc/exports".

```
sudo /usr/sbin/exportfs -va
```

2.1.3 Samba

Samba servers are an excellent way to access a Linux file-system on a Windows machine via a network connection. Using a Samba server, it is quick and easy to transfer files between systems. To install a Samba server, use the following command:

```
sudo apt-get install samba
```

Before the Samba share can be mounted on another machine it's necessary to modify the configuration file to allow write access and access to home directories. Start by editing the "/etc/samba/smb.conf" file.

```
sudo gedit /etc/samba/smb.conf
```

Inside this file there are four specific things that need to be uncommented (remove the ';' at the beginning of the line) to enable the sharing of home folders and write access. Below is the section that must be modified:

```
#======================== Share Definitions =======================

# Un-comment the following (and tweak the other settings below to suit)
# to enable the default home directory shares. This will share each
# user's home directory as \\server\username
;[homes]
;   comment = Home Directories
;   browseable = yes

# By default, the home directories are exported read-only. Change the
# next parameter to 'no' if you want to be able to write to them.
;   read only = no
```

The outcomes after the changes are made follow:

```
#======================== Share Definitions =======================

# Un-comment the following (and tweak the other settings below to suit)
# to enable the default home directory shares. This will share each
# user's home directory as \\server\username
[homes]
   comment = Home Directories
   browseable = yes

# By default, the home directories are exported read-only. Change the
# next parameter to 'no' if you want to be able to write to them.
   read only = no
```

NOTE: It might also be necessary to change the "workgroup = " line to match the workgroup for your machine.

To apply the changes, the next step is to restart all Samba-related processes.

```
sudo restart smbd
sudo restart nmbd
```

Lastly, each user needs to have a password enabled to be able to use the Samba server. There are no rules for this password. The simplest method for choosing this password is to make it the same as the UNIX user's password, but it is not a requirement. After typing in the command below, you will be prompted to enter the password for the specified user.

```
sudo smbpasswd -a <user>
```

As mentioned in the configuration file, the samba share can be connected by accessing "\\<host machine ip>\\<user>" by either mounting a network share or using Windows explorer to navigate to it.

2.2 PTXdist

2.2.1 General Information

PTXdist is a set of tools created by Pengutronix to help manage all of the BSP sources, from x-loader to the filesystem and its applications. It makes it easier for a user to manage everything for specific platforms and toolchains without manually repeating relatively complex commands every time it is necessary to build a new image. It is important to note the PTXdist version because more than one may be necessary for building the toolchain and BSP.

2.2.2 Extracting Sources

Visit the phyCORE-OMAP44xx's BSP page for information on the current versions of BSP tools.

Potentially, two versions of PTXdist will need to be downloaded:

1. ptxdist-2011.02.0 for installing/constructing the cross-compile toolchain (Optional if using a pre-existing environment)
2. ptxdist-2012.03.0 for building the BSP with the toolchain mentioned above.

The following instructions correspond to downloading PTXdist for building the BSP, *ptxdist-2012.03.0*. Therefore, to download the PTXdist software for building the toolchain, *ptxdist-2011.02.0*, repeat the instructions replacing *2012.03.0* with *2011.02.0*. In order to be built, extract the archive containing the PTXdist software:

```
tar -jxvf ptxdist-2012.03.0.tar.bz2
cd ptxdist-2012.03.0
```

Now that the source has been extracted, the next step is to configure it for building.

2.2.3 Pre-Requisites

PTXdist checks for specific packages that must be installed before it can be successfully built. From the PTXdist source directory, *ptxdist-2012.03.0*, execute the command to begin a script that uses GNU autotools to help set up the environment for building the distribution.

```
./configure
```

This command automatically stops if it is missing a package and states why and what package to install to continue with the initial setup. After successfully running the configure script, build and install PTXdist.

From the PTXdist source directory, *ptxdist-2012.03.0*:

```
make
sudo make install
```

The install location is "/usr/local" by default, which is why the **make install** command must be run with root privileges. If another directory is preferred for the install, use the --prefix option when installing, but be sure to add the "bin/" directory of the installed tools in this new folder to $PATH by modifying and sourcing ~/.bashrc.

```
source ~/.bashrc
```

Now that the install is complete, the PTXdist folder can be removed, as well as the original archive.

Optional:

```
cd ..
rm  -rf ptxdist-2012.03.0*
```

2.2.4 Using PTXdist

PTXdist is a console command tool and different functions are run by extending parameters to the **ptxdist** base command.

```
ptxdist <parameter>
```

To generate a full list of parameters and a description of the function, use the **help** command:

```
ptxdist help
```

Since PTXdist versions can be installed in parallel it may be useful to view the version number corresponding to the **ptxdist** command:

```
ptxdist --version
```

If the output of this command does not correlate to the desired version of PTXdist, affix the **ptxdist** command with the version number. For example, for the phyCORE-OMAP44xx BSP, affix all **ptxdist** commands with **2012.03.0** resulting in **ptxdist-2012.03.0**:

```
ptxdist-2012.03.0 <parameter>
```

Alternatively, to correlate the **ptxdist** command with a specific version create a symbolic link. For example, for **ptxdist** to correlate to **ptxdist-2012.03.0**:

```
sudo ln -fs /usr/local/lib/ptxdist-2012.03.0/bin/ptxdist /usr/local/bin/ptxdist
```

Potential Issues:

- Wrong PTXdist Version

It is important to use the correct version of PTXdist when building the Toolchain or BSP. If the incorrect version is used, the following will result:

error: The ptxconfig file version and ptxdist version do not match:configfile version: 2012.03.0ptxdist version: 2011.11.0

You can either migrate from an older ptxdist release with:'ptxdist migrate'

or, to ignore this error, add '--force'to ptxdist's parameters, e.g.:'ptxdist --force go'

The version of PTXdist used is noted as *ptxdist version* where the one required is specified as *configfile version*. Therefore, rerun the command with the correct version either appended to **ptxdist** or create a symbolic link. If the *merge* or *force* options were used on the Toolchain or BSP they will need to be removed and reinstalled to be built with the correct PTXdist version. Please note that the above is a generic example and may not apply directly to your BSP.

2.2.4.1 Setup

The first time PTXdist is used, there are some setup properties that have to be configured. To run PTXdist's setup, use the following command:

```
ptxdist setup
```

Once in the ptxdist setup, the only settings that should be modified are the *User->Name* and *User->eMail*. This is mainly for general logging purposes. If you wish to modify any other settings, please refer to the Getting Help section for a link to PTXdist documentation.

Since PTXdist works with sources only, it needs to grab source archives from the web using wget as it advances through its setup if they do not exist already. Therefore, an internet connection is required.

2.3 Toolchains

In order to build images or applications for an embedded device, it is necessary to have a cross toolchain that will perform the necessary operations to compile code for a specified processor and system setup.

Each toolchain will have a modified GNU Compiler Collection (gcc) designed for the desired setup. The phyCORE-OMAP44xx PD12.2.0 uses the arm-cortexa9 toolchain which can be built from the OSELAS.Toolchain-2011.02.0 and ptxdist-2011.02.0 sources.

2.3.1 Existing Toolchains

If a working toolchain is already installed for a given architecture, it is possible to use this instead of building an OSELAS Toolchain. A pre-built cortex-a9 toolchain is available from the PHYTEC FTP [Here]. Do note that since external toolchains have not been tested it is possible that they may not work properly across all environments.

An extra step needs to be taken if the plan is to use an external toolchain. By default, the software package will be set to check for the vendor toolchain that it was compiled with. In order to change this, use the following command inside the root directory of the BSP:

```
ptxdist platformconfig
    architecture --->
        toolchain --->
            () check for specific toolchain vendor
```

2.3.2 Building OSELAS Toolchains

An OSELAS toolchain, managed by PTXdist, can be easily built for a target architecture.

For the phyCORE-OMAP44xx, the arm-cortexa9 architecture based toolchain is built from OSELAS.Toolchain-2011.02.0 [Here] and PTXdist 2011.02.0 [He re]. See Section 2.2 for information regarding the installation of PTXdist sources.

```
tar -jxvf OSELAS.Toolchain-2011.02.0.tar.bz2
cd OSELAS.Toolchain-2011.02.0
```

Be sure to use the correct ptxdist version for the toolchain by affixing all **ptxdist** commands with **2011.02.0** resulting in **ptxdist-2011.02.0** or see Section 2.2.4 for information on creating a symbolic link between **ptxdist** and **ptxdist-2011.02.0**.

```
ptxdist select ptxconfigs/arm-cortexa9-linux-gnueabi_gcc-linaro-4.5-2011.02-0_glibc-2.13_binutils-2.21_kernel-
2.6.36-sanitized.ptxconfig
ptxdist go
```

The toolchain is now built and installed in */opt/OSELAS.Toolchain-2011.02.0/arm-cortexa9-linux-gnueabi* and ready to be used for building the BSP.

If you wish to build applications outside of the BSP directory, add the toolchain location to your PATH. Use the following from the command line or permanently add to your PATH by including it in .bashrc:

```
PATH=/opt/OSELAS.Toolchain-<toolchain version>/arm-<processor>-linux-gnueabi_gcc-linaro-<version>_glibc-
<version>_binutils-<version>_kernel-<version>-sanitized/bin/:$PATH
```

Following a successful build, the OSELAS.Toolchain-2011.02.0 folder can be removed, as well as the original archive.

Optional:

```
cd ..
rm  -rf OSELAS.Toolchain-2011.02.0*
```

2.3.2.1 Protecting Toolchains

It is recommended, although optional, to set the /opt/OSELAS.Toolchain-2011.02.0/arm-cortexa9-linux-gnueabi directory and its contents as read-only to prevent accidental modifications to the toolchain.

3 Board Setup-phyCORE-OMAP44XX

The phyCORE-OMAP44XX comes pre-flashed with MLO, barebox, Linux kernel, and root filesystem. After the device is out of the box and setup, simply applying power will boot the pre-installed images from NAND flash. If for any reason it is necessary to re-flash example images, they are located on the PHYTEC America FTP [here].

3.1 Connections

Power and host-PC connections must be made to the target device. The hardware manual, included with the Rapid Development Kit, may be referred to for specific connection information.

3.1.1 Power

The primary input power for the phyCORE-OMAP44XX Carrier Board comes from the wall adapter jack, X6. Upon application of power, LEDs D14 - D17 should light up (green) and initial serial data will be sent by UART3 (top connector P1).

The Carrier Board provides an option for powering down, powering up, and a system reset without the removal of the power source. The following summarizes the characteristics unique to the phyCORE-OMAP44XX Carrier Board for power and reset options:

Option	Action	Indicator
Power Down	S3 pressed longer than 10 seconds	Heartbeat LED on SOM OFF Console disabled on UART3 (top connector, P1)
Power Up	S3 pressed for approximately 2 seconds (only following a power down)	Heartbeat LED on SOM solid, then flashes (RED) Serial data sent on UART3 (top connector, P1)
Restart	S7 pressed	Heartbeat LED on SOM solid, then flashes (RED) Serial data sent on UART3 (top connector, P1)

3.1.2 Serial

A serial connection is used as system communication for boot-up interaction throughout start-up and as a monitoring/debugging interface. This connection is made between the Host and UART3 at RS-232 level, top DB-9 connector of the dual-port connector P1 on the phyCORE-OMAP44XX.

The following provides a summary of the serial settings required to allow console access over the serial port in a communications program on the host such as Minicom:

Setting	Value
Bits per second	115200 bsp
Data bits	8-bits
Stop-bit	1
Flow Control	None

3.1.2.1 Minicom

Minicom is the recommended communications program on the host for serial communication to the device.

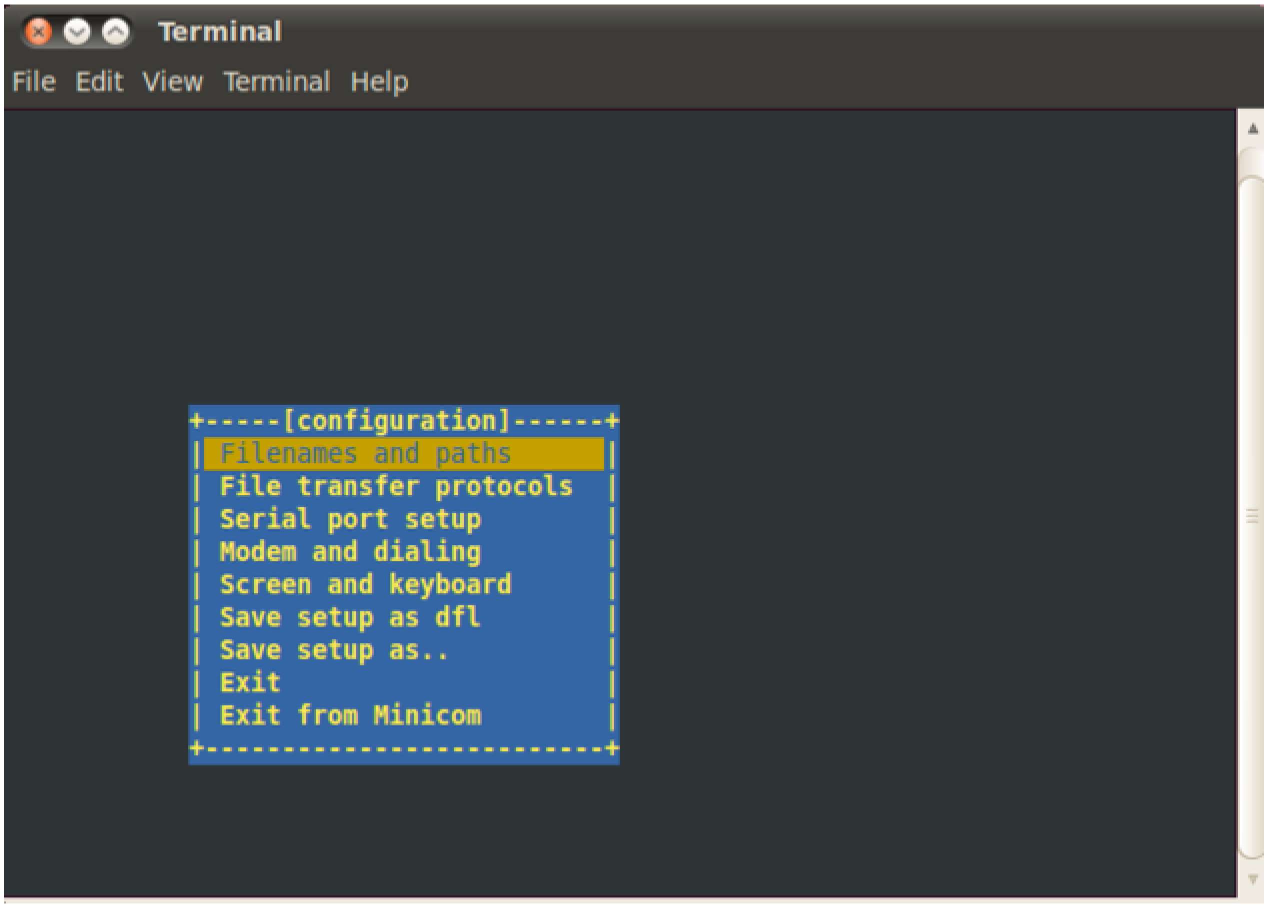
To install Minicom, execute the following in a terminal on the host:

```
/* if Minicom is not installed */
sudo apt-get install minicom
```

Start minicom from the terminal in the following way:

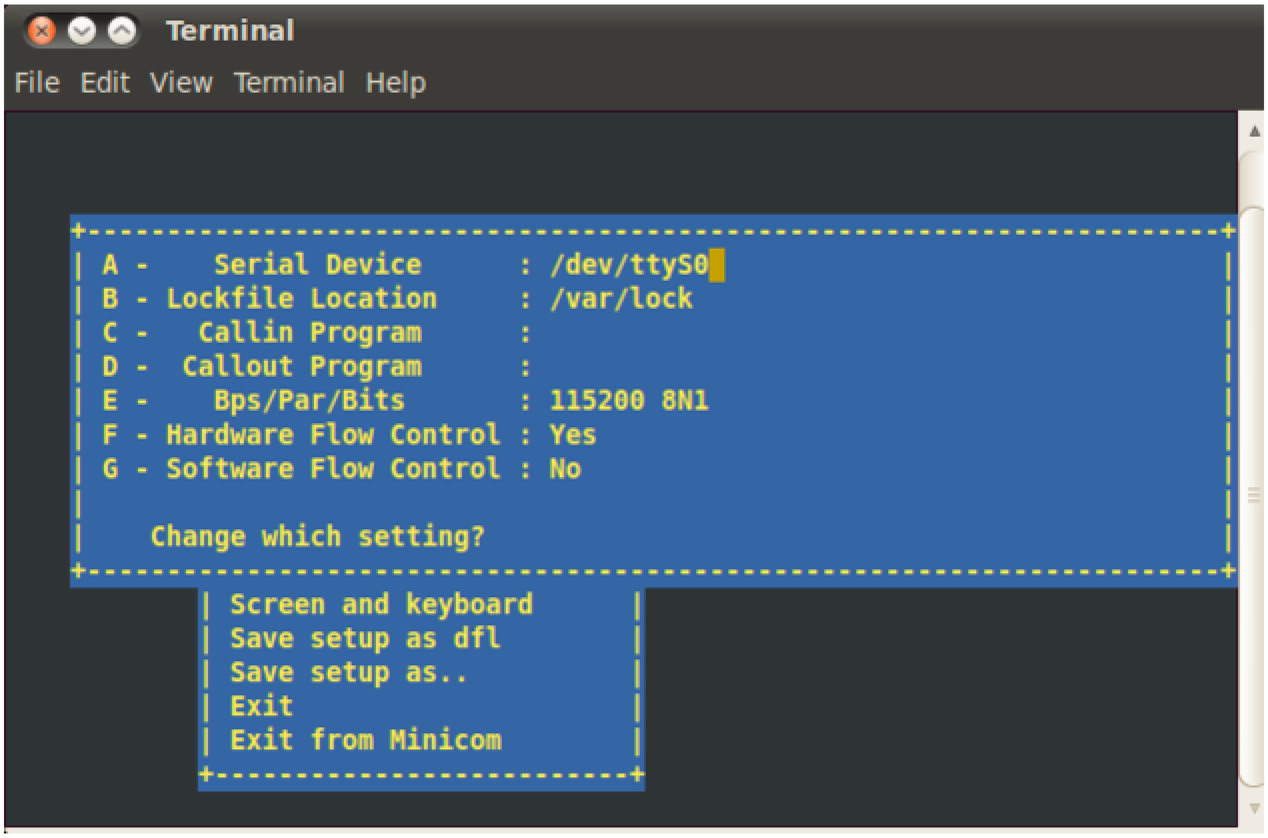
```
minicom -c on -s
```

Minicom will be executed and the main menu will be displayed in the terminal:

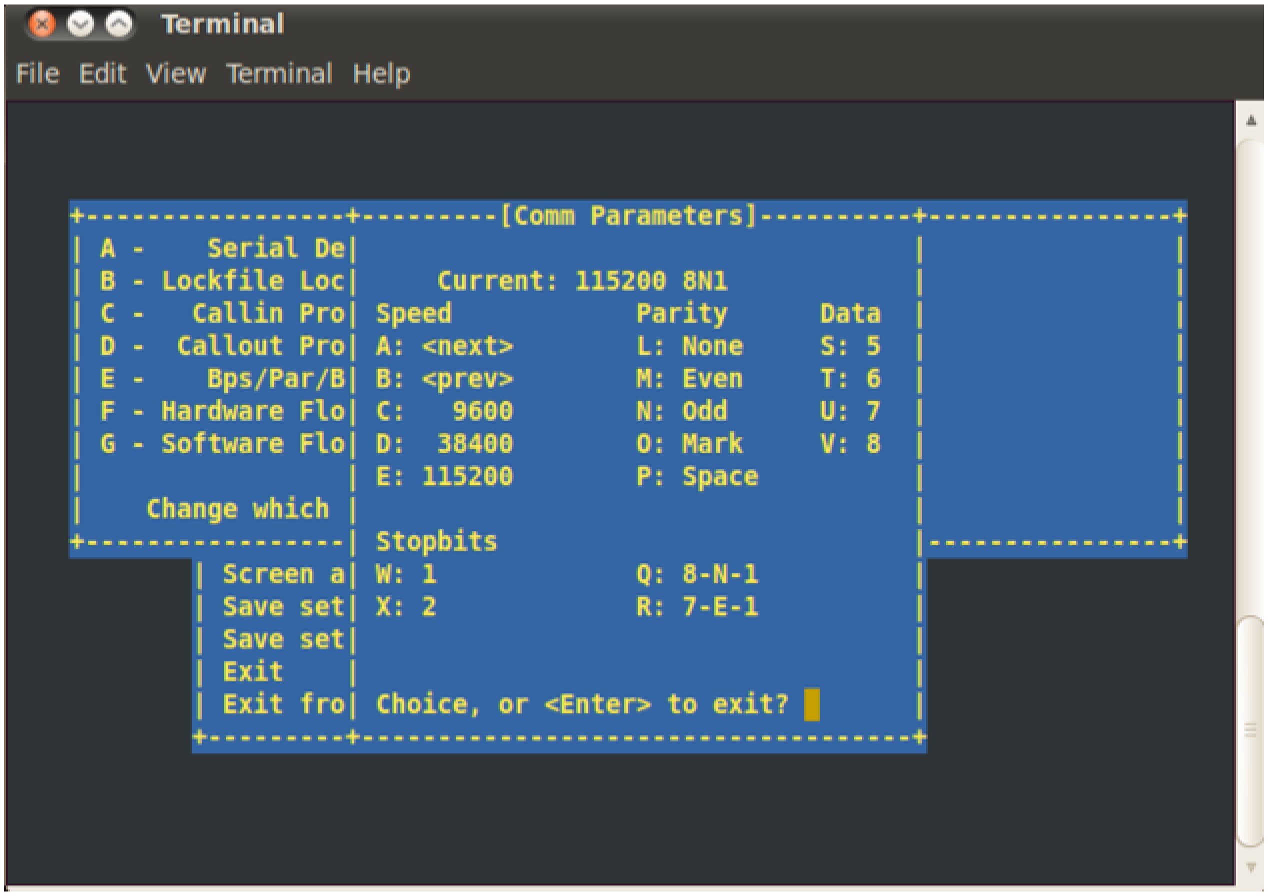


```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

Navigate to "Serial port setup" in the Minicom main menu and modify line *A - Serial Device :* to read /dev/ttyS0 and line *E - Bps/Par/Bits :* to have a speed of 115200 and 8-N-1 (8N1) for the stop bits:



```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyS0                                |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 115200 8N1                                |
| F - Hardware Flow Control : Yes                                       |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
        | Screen and keyboard      |
        | Save setup as dfl        |
        | Save setup as..          |
        | Exit                     |
        | Exit from Minicom        |
        +--------------------------+
```



```
+-----------------+---------[Comm Parameters]----------+----------------+
| A -    Serial De|                                    |                |
| B - Lockfile Loc|     Current: 115200 8N1            |                |
| C -   Callin Pro| Speed            Parity      Data  |                |
| D -  Callout Pro| A: <next>        L: None     S: 5  |                |
| E -    Bps/Par/B| B: <prev>        M: Even     T: 6  |                |
| F - Hardware Flo| C:   9600        N: Odd      U: 7  |                |
| G - Software Flo| D:  38400        O: Mark     V: 8  |                |
|                 | E: 115200        P: Space          |                |
|    Change which |                                    |                |
+-----------------| Stopbits                           |----------------+
        | Screen a| W: 1             Q: 8-N-1          |
        | Save set| X: 2             R: 7-E-1          |
        | Save set|                                    |
        | Exit    |                                    |
        | Exit fro| Choice, or <Enter> to exit?        |
        +---------+------------------------------------+
```

Note: *The serial device is dependent on what COM port you are connected to on your system, so /dev/ttyS0 is merely an example.*

Return to the main menu of minicom and select **Save setup as dfl** to make this the default setup anytime Minicom is loaded, meaning minicom -c on is all that needs to be done in the future for this machine to be able to communicate with the kit. Be sure that permissions allow writing to *minirc.dfl*by:

```
sudo chmod ugo+rwx /etc/minicom
```

3.1.3 Ethernet

The Ethernet connection is used for flashing, downloading, and debugging images and applications. Connect the cross-over Ethernet cable to the Ethernet connector on the target, X9, and appropriate network card on the host. LINK (green) and SPEED (yellow) LEDs on the connector verify the connection.

3.2 Image Format

The bootloader, kernel, and root filesystem specific to the phyCORE-OMAP44XX can be provided over a wide variety of sources such as the PHYTEC FTP, preloaded to NAND Flash, SD Card, TFTP Server, or NFS Server. Refer to the Flashing Images section for information on how to flash these images.

3.2.1 PHYTEC FTP

If for any reason it is necessary to re-flash the example images, they are located on the PHYTEC America FTP [here].

3.2.2 SD Card

Images can be placed on a SD/MMC card in the following way:

Insert SD/MMC Card on host machineConfigure SD/MMC Card with 64MB fat partition and mark as bootable [instructions here]Copy the files in this order to the first partition of the SD card (/media/boot)MLObarebox-image (rename to barebox.bin)linuximage (rename to uImage-pcm049)

3.2.2.1 Root Filesystem

The placement of the root filesystem on the MMC/SD Card depends the users intentions: Flashing to NAND or Booting from SD/MMC.

Flashing to NAND

When flashing the root filesystem to NAND, following successful placement of MLO, barebox, and the kernel, copy root.ubi, renamed to root-pcm049.ubi, to the SD Card. Note that if your build options produce an image which is too large to fit on the SD Card, a tftp server must be used as outlined below.

SD Card Boot

In the case of booting Linux from the SD Card, extract the contents of root.tgz to the second partition of the SD Card:

```
sudo tar -zxf root.tgz -C /media/rootfs
```

3.2.3 TFTP Server

Images are available over the TFTP server, setup in Section 2.1.1.

The following files are present in a directory, such as *<TFTP_DIRECTORY>/phyCORE-OMAP4/images*, on the server:

MLObarebox-image (rename to barebox.bin)barbox-default-environmentlinuximage (rename to uImage-pcm049)root.ubi (rename to root-pcm049.ubi)

3.2.4 NFS Server

An entire root filesystem can be made accessible over the NFS Server, setup in Section 2.1.2.

To have the ability to mount the file system, root.tgz can be extracted to a NFS server directory such as */home/<user>/phyCORE-OMAP4/NFS*:

```
sudo tar -zxf root.tgz -C /home/<user>/phyCORE-OMAP4/NFS/
```

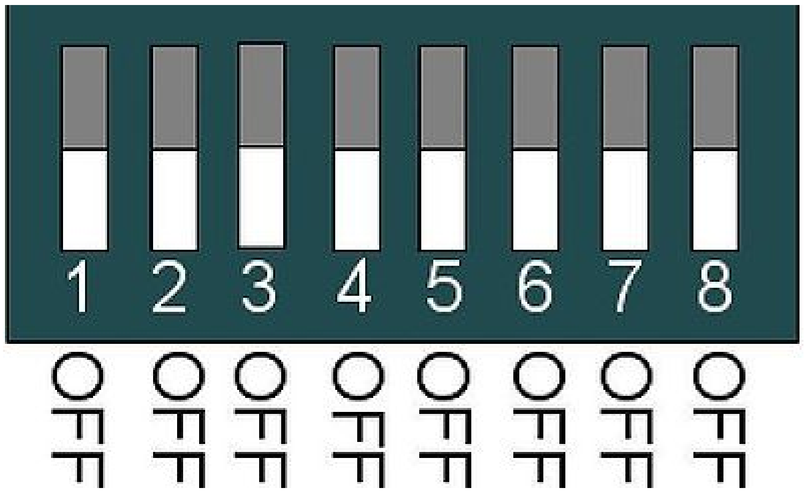
3.3 Booting Configurations

The bootloader, one of the key software components included in the BSP, completes the required hardware initializations to download and run operating system images. The location of the primary bootloaders, MLO and barebox, is determined by the boot mode. The boot mode is selected from the S2 dipswitch on the Carrier Board, which supports booting from NAND or SD/MMC Card.

3.3.1 NAND Boot

To boot from NAND, using the following switch settings:

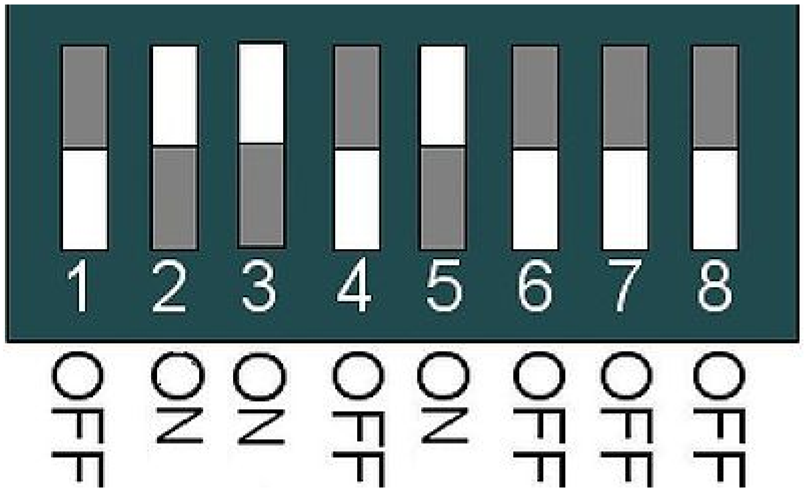
S2-1 to S2-8 OFF



3.3.2 SD Card

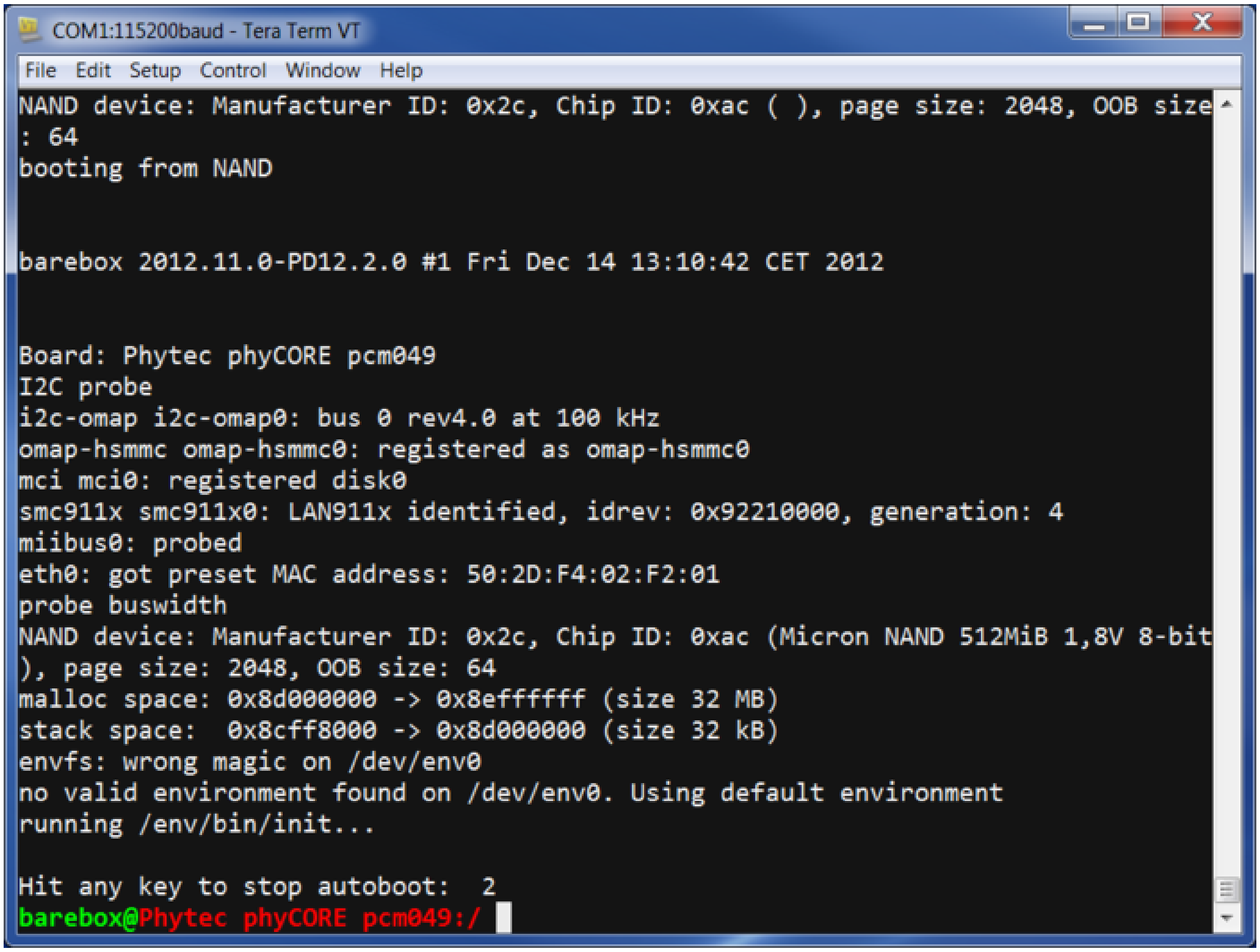
To boot from SD Card, use the following switch settings:

S2-2, S2-3, and S2-5 ONS2-1, S2-4, S2-6 to S2-8 OFF



3.4 Working with Barebox

The boot loader allots approximately three seconds to halt autoboot and enter barebox, in which any key must be pressed:


COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
NAND device: Manufacturer ID: 0x2c, Chip ID: 0xac ( ), page size: 2048, OOB size
: 64
booting from NAND


barebox 2012.11.0-PD12.2.0 #1 Fri Dec 14 13:10:42 CET 2012


Board: Phytec phyCORE pcm049
I2C probe
i2c-omap i2c-omap0: bus 0 rev4.0 at 100 kHz
omap-hsmmc omap-hsmmc0: registered as omap-hsmmc0
mci mci0: registered disk0
smc911x smc911x0: LAN911x identified, idrev: 0x92210000, generation: 4
miibus0: probed
eth0: got preset MAC address: 50:2D:F4:02:F2:01
probe buswidth
NAND device: Manufacturer ID: 0x2c, Chip ID: 0xac (Micron NAND 512MiB 1,8V 8-bit
), page size: 2048, OOB size: 64
malloc space: 0x8d000000 -> 0x8effffff (size 32 MB)
stack space:  0x8cff8000 -> 0x8d000000 (size 32 kB)
envfs: wrong magic on /dev/env0
no valid environment found on /dev/env0. Using default environment
running /env/bin/init...

Hit any key to stop autoboot:   2
barebox@Phytec phyCORE pcm049:/
```

In barebox there are a wide variety of functions which can be viewed by using the **help** command:

```
help
```

Some of the most common uses of barebox are to modify settings of environment configuration, flashing, and booting.

3.4.1 Environment Variables

The settings most necessary for operation are environment variables in barebox. To obtain a list of current environment variables, use the **printenv** comma nd:

```
printenv
```

3.4.2 Configuration File

The configuration file provided by barebox is located in /env/config, this file allows the user among many things, to modify environment variables, setup networking parameters, and select display settings. To open and make edits to the /env/config file, do the following in barebox:

```
edit /env/config
```

3.4.2.1 Remote Settings

A part of setting network parameters is to enter the IP address of the TFTP and NFS servers, which is likely the IP address of the host. Therefore, search for the line beginning with *eth0.serverip* and modify to reflect the correct IP address, 192.168.3.10 is the default:

```
eth0.serverip=192.168.3.10
```

3.4.2.2 NFS Root Directory

If intending to mount the root filesystem by NFS it is required to specify the path. The NFS root path is determined by the location of the files extracted from root.tgz in Section 3.2.4, such as */home/<user>/phyCORE-OMAP4/NFS/*. In the configuration file, search for *nfsroot="/path/to/root"* and modify to read the proper path:

```
nfsroot="/home/<user>/phyCORE-OMAP4/NFS/"
```

3.4.2.3 Display Settings

Display either via LCD or DVI is supported and selected in the configuration file.

Specify LCD as a display in the configuration file by commenting out all lines pertaining to DVI. Additionally, in the "Displays" section, be sure to uncomment the LCD selected out of the three supported: 5" VGA, 7" WVGA, and 10.4" SVGA display/touch screens, and place a '#' at the beginning of the line for the remaining to comment them out. For example, the "Displays" section in the configuration file should resemble the following if the 7" WVGA display/touch screen is used on the target device:

```
#Displays
#bootargs="$bootargs panel_generic_dpi.name=pd050vl1"
#bootargs="$bootargs panel_generic_dpi.name=pd104slf"
bootargs="$bootargs panel_generic_dpi.name=pm070wl4"
#bootargs="$bootargs panel_generic_dpi.name=edt_etm0350G0dh6"
#bootargs="$bootargs panel_generic_dpi.name=edt_etm0430G0dh6"
#bootargs="$bootargs panel_generic_dpi.name=edt_etmv570G2dhu"
#bootargs="$bootargs panel_generic_dpi.name=edt_etm0700G0dh6"

#activate dvi output
#bootargs="$bootargs omapdss.def_disp=dvi"
#select resolution
#bootargs="$bootargs omapfb.mode=dvi:640x480-60"
#bootargs="$bootargs omapfb.mode=dvi:800x600-60"
#bootargs="$bootargs omapfb.mode=dvi:1024x768-60"
#bootargs="$bootargs omapfb.mode=dvi:1280x1024-60"
```

Alternatively, to specify DVI, in the configuration file place a '#' before all LCD lines to comment them out. Additionally, uncomment DVI activation and a corresponding resolution by removing the '#' before the lines. The configuration files should resemble the following:

```
#Displays
#bootargs="$bootargs panel_generic_dpi.name=pd050vl1"
#bootargs="$bootargs panel_generic_dpi.name=pd104slf"
#bootargs="$bootargs panel_generic_dpi.name=pm070wl4"
#bootargs="$bootargs panel_generic_dpi.name=edt_etm0350G0dh6"
#bootargs="$bootargs panel_generic_dpi.name=edt_etm0430G0dh6"
#bootargs="$bootargs panel_generic_dpi.name=edt_etmv570G2dhu"
#bootargs="$bootargs panel_generic_dpi.name=edt_etm0700G0dh6"

#activate dvi output
bootargs="$bootargs omapdss.def_disp=dvi"
#select resolution
#bootargs="$bootargs omapfb.mode=dvi:640x480-60"
#bootargs="$bootargs omapfb.mode=dvi:800x600-60"
bootargs="$bootargs omapfb.mode=dvi:1024x768-60"
#bootargs="$bootargs omapfb.mode=dvi:1280x1024-60"
```

After making any changes, quit the program and return to the barebox prompt by pressing **CTL+D**, type **saveenv** to save changes:

```
saveenv
```

3.4.3 Restore to Default

If the barebox environment variables need to be restored for any reason, simply delete the parameter save location in NAND, and the defaults will be restored with the next boot.

```
erase /dev/nand0.bareboxenv.bb
```

3.4.4 Booting Options

After selecting the location for MLO and barebox via the S2 settings on the Carrier Board, booting continues by loading and running the kernel. Stand-alone, remote, and MMC are the three main locations selected in barebox to continue the boot process.

3.4.4.1 Boot Command

Selection of the location to continue booting is provided as a **boot** command in the current barebox environment. This command allows the user to specify the mode over which it will boot with respect to the kernel, root filesystem, and ip options. The generic usage of this command is described by the following:

```
boot [-m <mode>] [-k <kernel_option>] [-r <rootfs_option>] [-i <ip_mode>]
```

By typing _**boot_help** in the barebox prompt, a summary of the syntax, options, and parameters are provided.

3.4.4.2 Stand-Alone Booting

By default, the kit comes setup to boot from the provided images loaded on NAND flash. Therefore, without modification to environment variables, barebox will boot the Linux kernel from NAND. Alternatively, executing the following command in barebox tells the system to boot using the Linux kernel and filesystem located in NAND:

```
boot –m nand
```

3.4.4.3 Remote Booting

For development it may be beneficial to modify the boot settings to allow the kernel to be loaded from TFTP, and/or mount a network filesystem hosted on the NFS, setup in Sections 3.2.3 and 3.2.4. Examples of the wide variety of remote booting options the barebox **boot** command supports is given by the following:

Description	Barebox
Boot kernel from TFTP and mount rootfs from default	`boot –k tftp`
Boot kernel from default and mount rootfs from NFS	`boot -r net`

Boot kernel from TFTP and mount rootfs from NFS	`boot -k tftp -r net`
	
	`boot -m tftp`

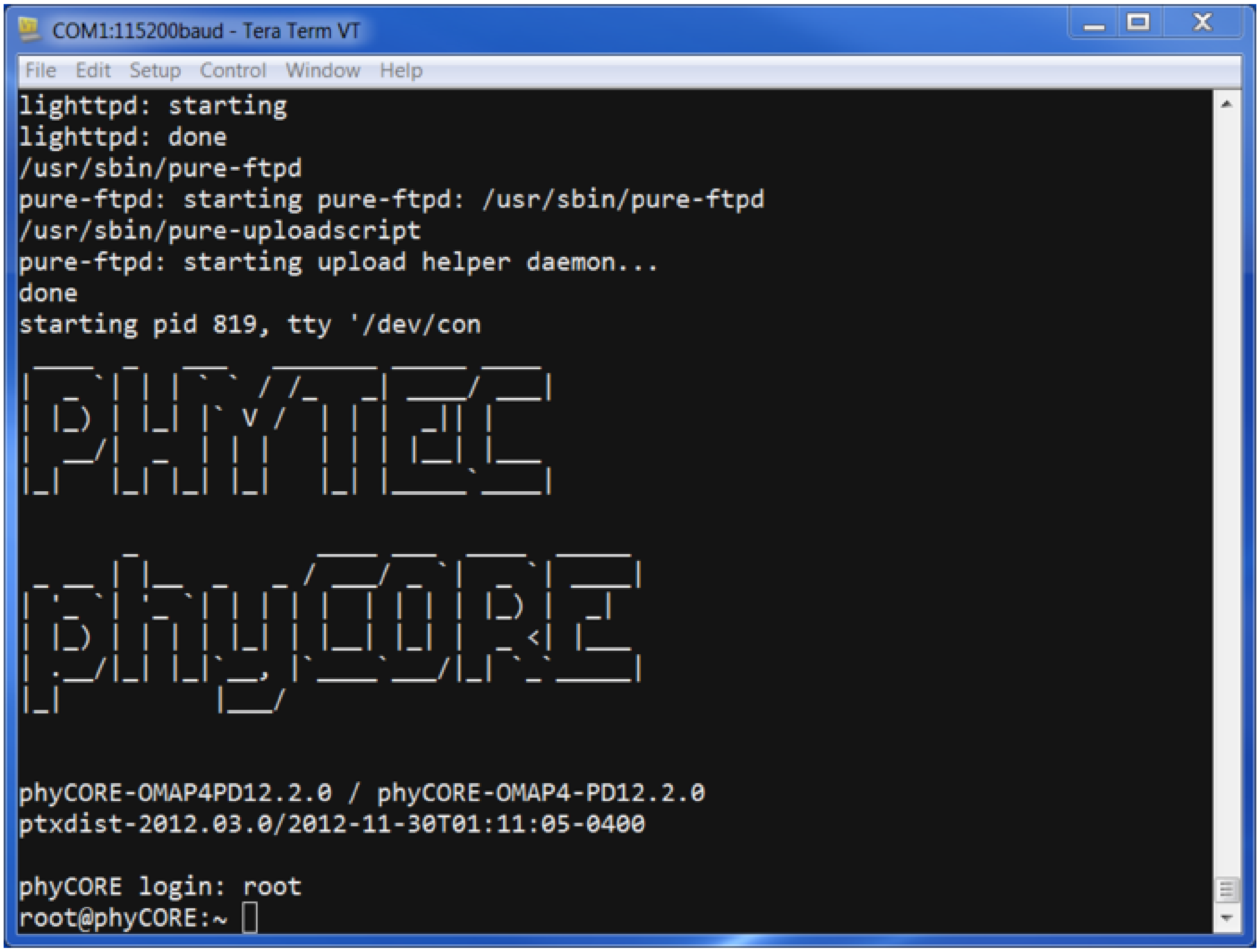
3.4.4.4 MMC Booting

The provided barebox includes a script to boot the Linux kernel and rootfs from SD/MMC:

```
mmc_boot
```

3.5 Booting the Target

After selecting the boot source, the target starts booting. When the target has finished loading the system, type **root** following the prompt to login:


COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
lighttpd: starting
lighttpd: done
/usr/sbin/pure-ftpd
pure-ftpd: starting pure-ftpd: /usr/sbin/pure-ftpd
/usr/sbin/pure-uploadscript
pure-ftpd: starting upload helper daemon...
done
starting pid 819, tty '/dev/con
 ____  _   ___   _______ _____ ____
|  _ \| | | \ \ / /_   _| ____/ ___|
| |_) | |_| |\ V /  | | |  _|| |
|  __/|  _  | | |   | | | |__| |___
|_|   |_| |_| |_|   |_| |_____\____|


       _           ____ ___  ____  _____
 _ __ | |__  _   _ / ___/ _ \|  _ \| ____|
| '_ \| '_ \| | | | |  | | | | |_) |  _|
| |_) | | | | |_| | |__| |_| |  _ <| |___
| .__/|_| |_|\__, |\____\___/|_| \_\_____|
|_|          |___/


phyCORE-OMAP4PD12.2.0 / phyCORE-OMAP4-PD12.2.0
ptxdist-2012.03.0/2012-11-30T01:11:05-0400

phyCORE login: root
root@phyCORE:~
```

4 Building a BSP

4.1 Modifying the BSP

The BSP provided can be modified through the source code in the board files or through configuration management.

4.1.1 Board Files

All source code is located in the *phyCORE-OMAP4-PD12.2.0/platform-phyCORE-OMAP4/build-target* directory. To help integrate and modify features on the system for both driver development and general settings or Carrier Board design, it is necessary to know about the three board files summarized by the following:

Board File	Location	Board Config File
Linux kernel	linux-3.4	/arch/arm/mach-omap2/board-omap4pcm049.c
Barebox	barebox-2011.11.0	/arch/arm/boards/pcm049/env/config
Barebox MLO	barebox_mlo-2012.11.0	/arch/arm/boards/pcm049/env/config

4.2 Managing Configurations

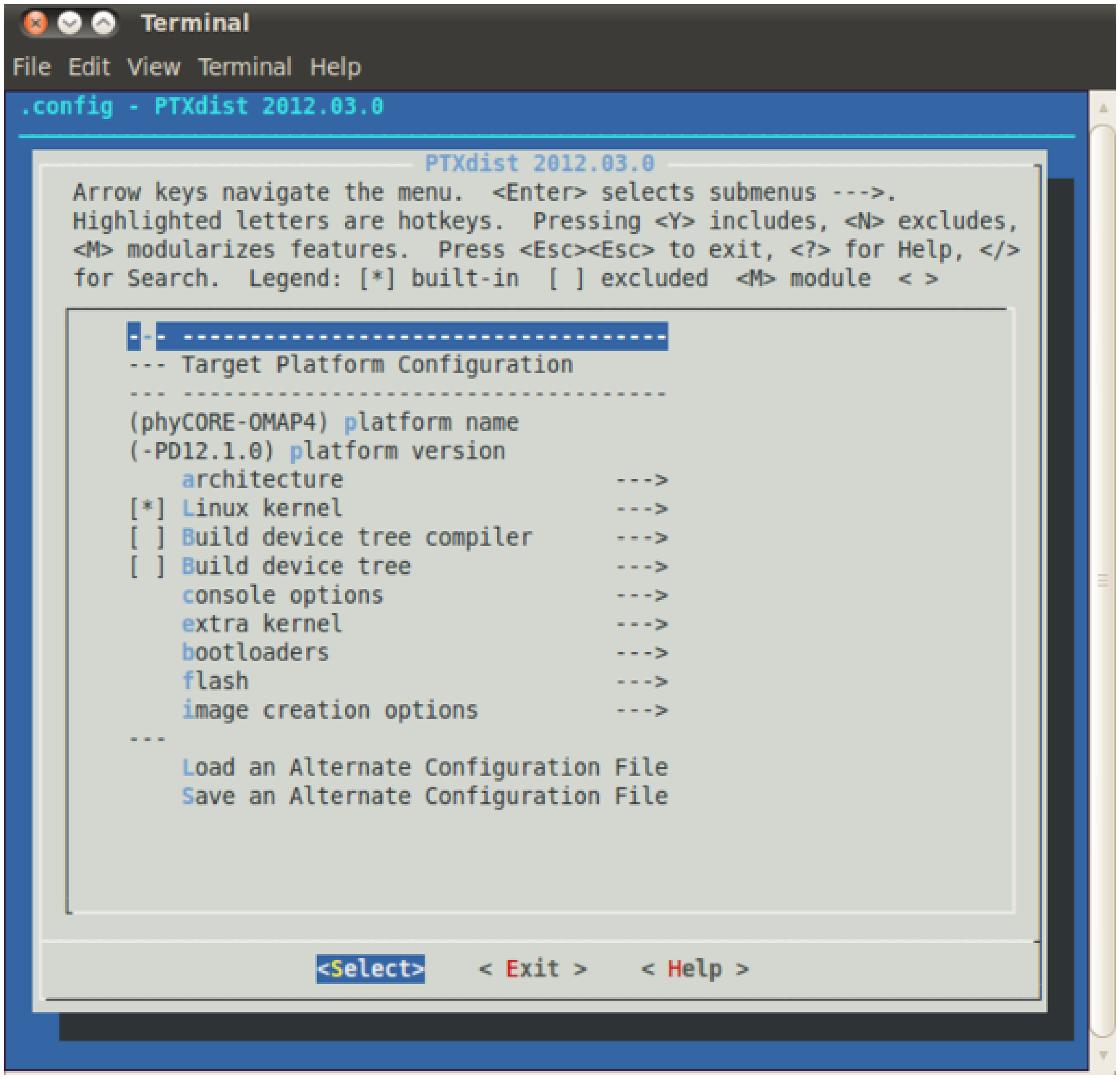
PTXdist uses the kernel configuration, KConfig, files present throughout the BSP for straightforward user configuration of individual settings and drivers. The platform, kernel, and project’s root filesystem configuration menus will be the most beneficial. The phyCORE-OMAP44xx PD12.2.0 BSP uses PTXdist version 2012.03.0, see Section 2.2 for additional information on using PTXdist.

4.2.1 Platform

The platform configuration menu contains the default settings for each platform including what bootloaders, kernel, and filesystem images are to be built. The settings for the platform are modified using the following command:

```
ptxdist platformconfig
```

As a user, it is very rare to modify these settings, but it may be useful to view them:

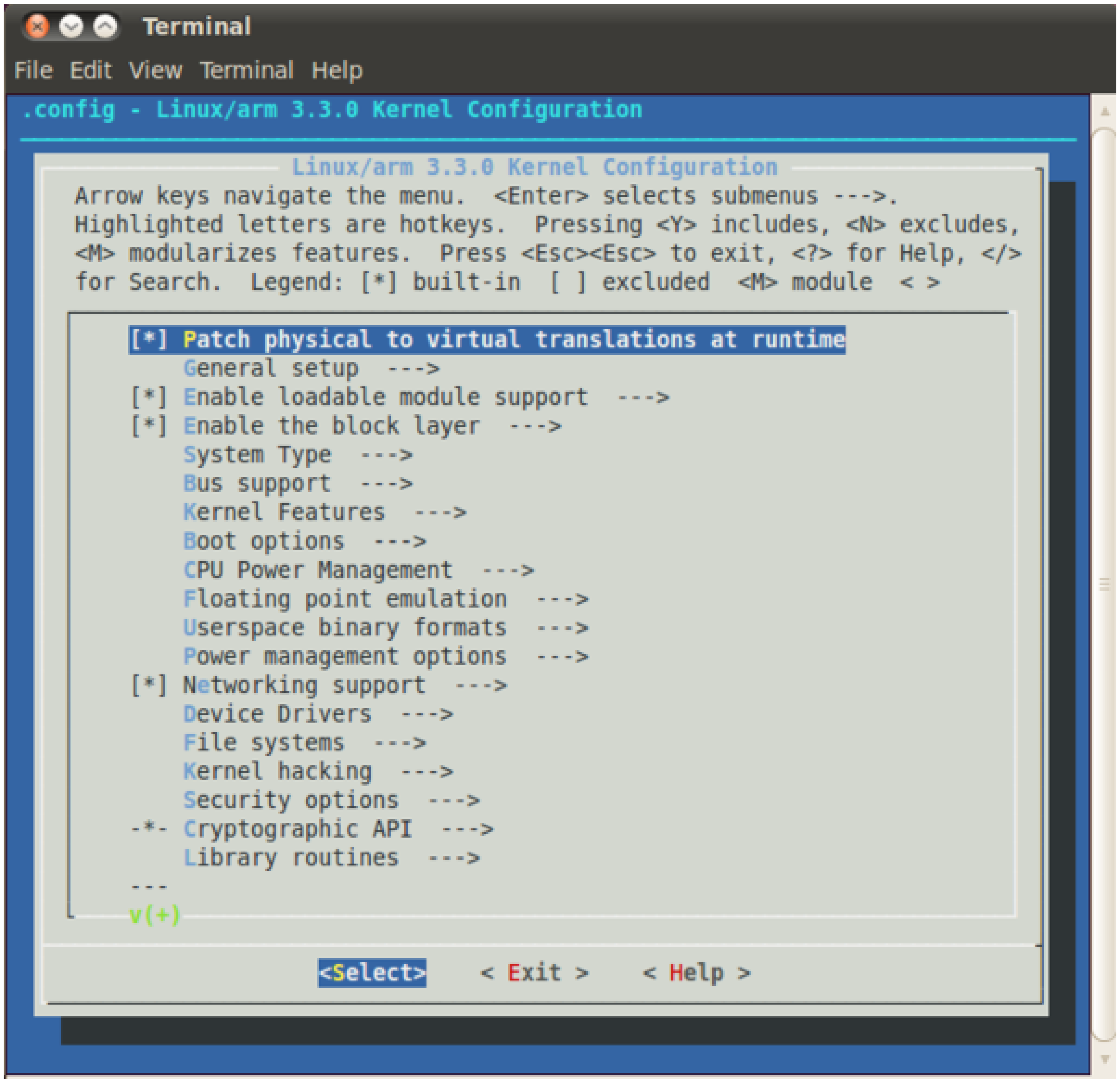


```
.config - PTXdist 2012.03.0
                         PTXdist 2012.03.0
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

      --- ------------------------------------
      --- Target Platform Configuration
      --- ------------------------------------
      (phyCORE-OMAP4) platform name
      (-PD12.1.0) platform version
          architecture                   --->
      [*] Linux kernel                   --->
      [ ] Build device tree compiler     --->
      [ ] Build device tree              --->
          console options                --->
          extra kernel                   --->
          bootloaders                    --->
          flash                          --->
          image creation options         --->
      ---
          Load an Alternate Configuration File
          Save an Alternate Configuration File

                <Select>    < Exit >    < Help >
```

4.2.2 Kernel

The kernel configuration menu allows the user to adjust the drivers and support included in a linux kernel build. The settings are available by using the following command:

```
ptxdist kernelconfig
```

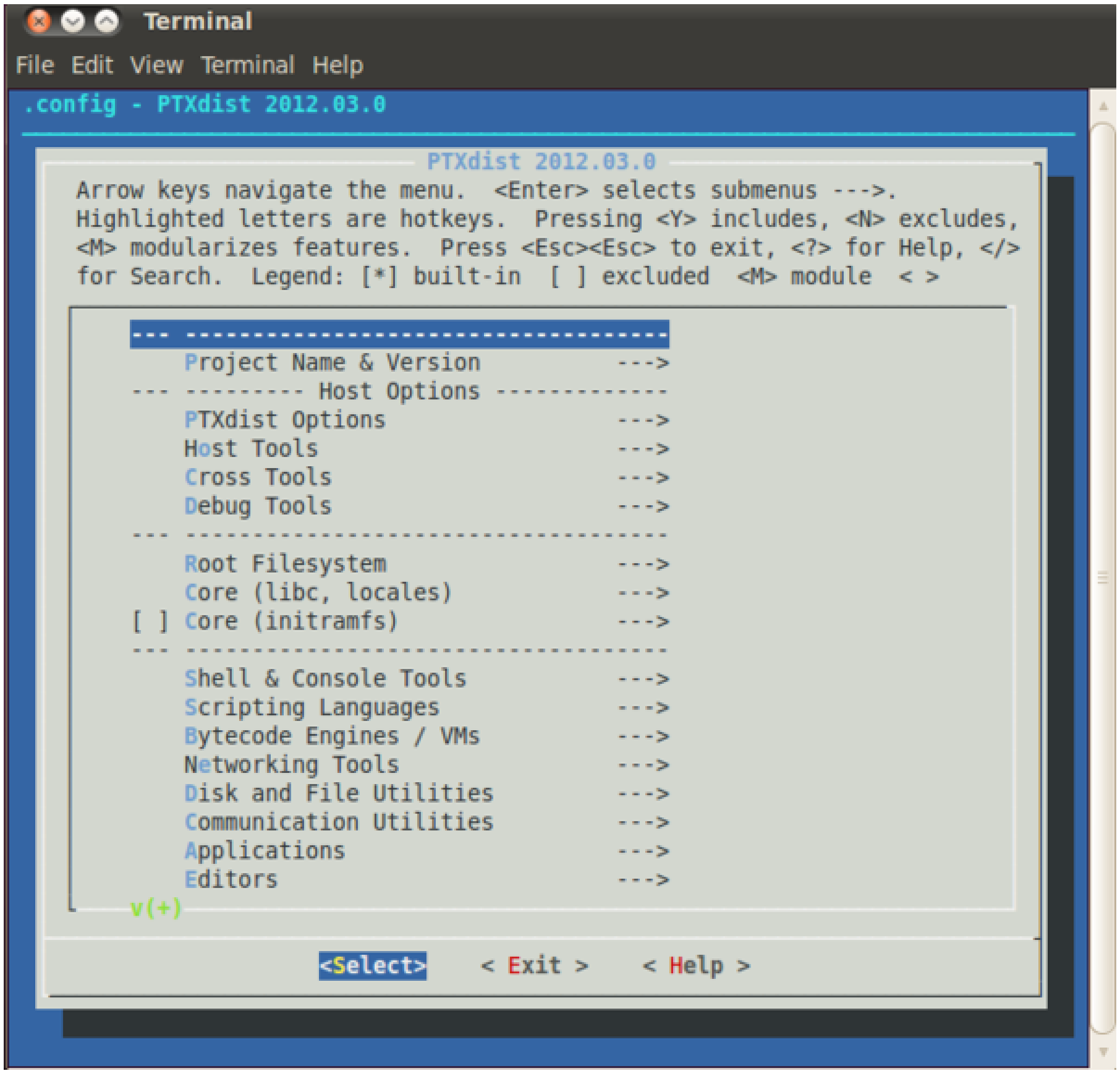


```
.config - Linux/arm 3.3.0 Kernel Configuration

                    Linux/arm 3.3.0 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

      [*] Patch physical to virtual translations at runtime
          General setup  --->
      [*] Enable loadable module support  --->
      [*] Enable the block layer  --->
          System Type  --->
          Bus support  --->
          Kernel Features  --->
          Boot options  --->
          CPU Power Management  --->
          Floating point emulation  --->
          Userspace binary formats  --->
          Power management options  --->
      [*] Networking support  --->
          Device Drivers  --->
          File systems  --->
          Kernel hacking  --->
          Security options  --->
      -*- Cryptographic API  --->
          Library routines  --->
      ---
      v(+)

              <Select>    < Exit >    < Help >
```

4.2.3 Root Filesystem

Configuration of the project’s filesystem can be done in the overall menuconfig. By toggling the options in the base configuration, the applications and content built into the filesystem can be modified directly. This allows both minimal and more complete filesystem builds to be created easily. The following command will open the configuration menu:

```
ptxdist menuconfig
```



```
.config - PTXdist 2012.03.0
                         PTXdist 2012.03.0
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

      --- ----------------------------------
          Project Name & Version          --->
      --- --------- Host Options ------------
          PTXdist Options                 --->
          Host Tools                      --->
          Cross Tools                     --->
          Debug Tools                     --->
      --- ----------------------------------
          Root Filesystem                 --->
          Core (libc, locales)            --->
      [ ] Core (initramfs)                --->
      --- ----------------------------------
          Shell & Console Tools           --->
          Scripting Languages             --->
          Bytecode Engines / VMs          --->
          Networking Tools                --->
          Disk and File Utilities         --->
          Communication Utilities         --->
          Applications                    --->
          Editors                         --->
      v(+)

              <Select>    < Exit >    < Help >
```

4.2.4 Enabling WiFi

To build an image which includes WiFi support:

```
ptxdist kernelconfig
    Device Drivers --->
        Network device support --->
            [*] Wireless LAN --->
                [*] TI Wireless LAN support --->
                    <M> TI wl12xx support
                    -M- TI wlcore support
                    < >   TI wlcore SPI support
                    <M>   TI wlcore SDIO support

ptxdist menuconfig
    Project Specific Configuration --->
        [*] Wifi module support
```

The file /etc/wpa_supplicant.conf on the target will need to have the appropriate configuration for your network.

Note: To use WiFi on the target, an inserted SD Card at SDMMC1 (X11) is required.

4.2.5 Enabling OpenGL

The standard BSP does not enable X and OpenGL, to build an image that includes this support:

```
ptxdist kernelconfig
    Device Drivers --->
        Graphics support --->
            <*> Direct Rendering Manager (XFree86 ...)
    Device Drivers --->
        [*] Staging drivers --->
            <*> OMAP DRM
```

```
ptxdist menuconfig
    PTXdist Base Configuration --->
        Graphics & Multimedia --->
            <*> graphics_ddk --->
                --- graphics_ddk
                [*]    install /etc/init.d/rc.pvr
```

Information about the display device is found in the X11 configuration file /etc/X11/xorg.conf on the target (source: projectroot/X11/xorg.conf in the BSP). This file is setup for the displays that are provided with the phyCORE-OMAP44xx development kit. If you are using a different display, you may need to make changes to this file. See the notes in the file under: Section "Screen".

When the BSP is built with the above settings, several sample programs will be included in the target image (/usr/bin/OGLES...). You can override the default dimensions used by the applications. For example:

```
OGLES2ChameleonMan -width=1360 -height=768
```

or the dimensions appropriate for your display. Use <ctrl>C to exit any sample application.

4.3 Building Images with PTXdist

Building and creating images with PTXdist is very simple, the phyCORE-OMAP44xx PD12.2.0 BSP uses PTXdist version 2012.03.0 therefore affix all commands with *2012.03.0* resulting in **ptxdist-2012.03.0** or see Section 2.2.4 for information on creating a symbolic link between **ptxdist** and**ptxdist-2012.03.0**.

All the required steps to compile sources and build packages in the correct order are done using the **ptxdist go** command. Following a successful build, the command, **ptxdist images**, will create images to deploy on the target.

Before starting a build, the userland and platform configuration must be specified:

```
ptxdist select configs/ptxconfig
ptxdist platform configs/phyCORE-OMAP4-2012.03.0/platformconfig
```

Select the toolchain to use:

```
ptxdist toolchain /opt/OSELAS.Toolchain-2011.02.0/arm-cortexa9-linux-gnueabi/gcc-linaro-4.5-2011.02-0-glibc-2.13-binutils-2.21-kernel-2.6.36-sanitized/bin
```

To compile and build the BSP, execute the **go** command from the project directory, *phyCORE-OMAP4-PD12.2.0*:

```
ptxdist go
```

The build process relies on obtaining source information from the web, therefore in some cases the content to be downloaded may no longer exist and **ptxd ist go** will fail. If this occurs, search the filename as it appears in the build output, download the file and place it in phyCORE-OMAP4-PD12.2.0/src directory in its compressed form.

A root filesystem image can be created from the built content, by executing the **images** command in the BSP project directory:

```
ptxdist images
```

All images are stored in phyCORE-OMAP4-PD12.2.0/platform-phyCORE-OMAP4/images, the following can be expected:

barebox-default-environmentbarebox-imagelinuximageMLOroot.ubiroot.ubifsroot.tgz

5 Flashing Images

Flashing images can be relatively complex and there are two main ways to accomplish this, the barebox **update** command and MMC.

5.1 Update Command (Remote Flashing)

To update the bootloader, kernel, or filesystem in flash, the current barebox environment provides the **update** command. This command allows the user to specify the file to be flashed, the mode over which it will be flashed, and the path to the image. The generic usage of this command is described by the following:

```
update -t <kernel|rootfs|barebox|bareboxenv|xload> -d <nor|nand> [-m tftp|xmodem] [-f imagename] –c
```

By typing **_update_help** or **update** in the barebox prompt, a summary of the syntax, options, and parameters are provided.

5.1.1 TFTP

TFTP is one of the simplest ways to apply modifications during development; it is commonly used for single file updates such as the bootloader, kernel, and files from the root filesystem.

The **update** command in barebox can be used to flash images to NAND from the TFTP Server in the correct format and located, for example, at *<TFTP_DIRECTORY>/phyCORE-OMAP4/images* as specified in Section 3.2.3:

Description	Barebox
Update MLO into NAND via TFTP	`update -t xload -d nand –m tftp -f phyCORE-OMAP4/images/MLO`
Update barebox into NAND via TFTP	`update -t barebox -d nand –m tftp -f phyCORE-OMAP4/images/barebox-image`
Update bareboxenv into NAND via TFTP	`update -t bareboxenv -d nand –m tftp -f phyCORE-OMAP4/images/barebox-default-environment`
Update kernel into NAND via TFTP	`update -t kernel -d nand –m tftp -f phyCORE-OMAP4/images/uImage-pcm049`
Update rootfs into NAND via TFTP	`update -t rootfs -d nand –m tftp -f phyCORE-OMAP4/images/root-pcm049.ubi`

NOTE : *TFTP is the default mode, as set in /env/config in barebox, therefore, “-m tftp” can be omitted from update commands.*

5.2 SD/MMC Flashing

The storage device, SD/MMC, of the target provides a method for flashing the bootloader, kernel, and root filesystem to NAND. After booting from the SD /MMC Card, commands in barebox can be used to complete flashing. The general procedure is described by the following:

Set eccmode (error-correction-code mode)Erase the NAND flash at the target locationCopy file from SD/MMC Card to NAND Flash

Prior to flashing from SD/MMC to NAND, the SD/MMC Card must be successfully mounted. Mounting of the SD/MMC Card can be done in barebox by the following:

```
mkdir mnt
mount /dev/disk0.0 fat /mnt
```

Use the following to update the default barebox environment:

```
erase /dev/nand0.bareboxenv.bb
saveenv
```

For convenience, the provided barebox includes a script that will complete all required steps for flashing MLO and barebox:

```
nand_bootstrap
```

Alternatively, individual files can be flashed which allows the more freedom to what they need to flash. The following provides examples of individual files flashed to NAND from SD/MMC in barebox:

Description	Barebox

Update MLO into NAND via SD/MMC	`gpmc_nand0.eccmode=${xload_eccmode}`
	`erase /dev/nand0.xload.bb`
	`cp /mnt/MLO /dev/nand0.xload.bb`

Update barebox into NAND via SD/MMC	`gpmc_nand0.eccmode=${barebox_eccmode}`
	`erase /dev/nand0.barebox.bb`
	`cp /mnt/barebox-image /dev/nand0.barebox.bb`

Update kernel into NAND via SD/MMC	`gpmc_nand0.eccmode=${kernel_eccmode}`
	`erase /dev/nand0.kernel.bb`
	`cp /mnt/uImage-pcm049 /dev/nand0.kernel.bb`

Update rootfs into NAND via SD/MMC	`gpmc_nand0.eccmode=${root_eccmode}`
	`erase /dev/nand0.root.bb`
	`cp /mnt/root-pcm049.ubi /dev/nand0.root.bb`