

Cosmic Board-Vybrid Quickstart QS1.0

- [1 About this Quickstart](#)
- [2 System Requirements](#)
 - [2.1 Hardware](#)
 - [2.2 Software](#)
- [3 Getting Started](#)
- [4 Host Setup](#)
 - [4.1 Server Setup](#)
 - [4.1.1 TFTP](#)
 - [4.1.2 NFS](#)
 - [4.1.3 Samba](#)
 - [4.2 LinuxLink Tools Setup](#)
 - [4.2.1 LinuxLink Registration](#)
 - [4.2.2 Desktop Factory Build System](#)
 - [4.3 SDK Install](#)
 - [4.4 ARM Development Studio 5 \(DS-5\)](#)
 - [4.4.1 TimeStorm](#)
- [5 Building the Linux BSP](#)
 - [5.1 Web Factory Wizard](#)
 - [5.1.1 Managing Workorders](#)
 - [5.1.2 Building Images](#)
 - [5.1.3 Downloading Images](#)
 - [5.2 Desktop Factory Build System](#)
 - [5.2.1 Board Files](#)
 - [5.2.2 Modifying the Platform](#)
 - [5.2.3 Building Images](#)
 - [5.3 Enabling Multi-Core Communication](#)
- [6 Managing Images](#)
- [7 Boot Configurations](#)
 - [7.1 Selecting Boot Modes](#)
 - [7.2 Basic Settings](#)
 - [7.3 Stand-Alone Boot](#)
 - [7.4 Remote Boot](#)
 - [7.5 Other Boot Options](#)
- [8 Flashing Images](#)
 - [8.1 Bootloader](#)

1 About this Quickstart

Use the following guide to get your Host setup and begin developing with your Cosmic Board for the phyCORE-Vybrid.

The Quickstart contains instructions for:

- Step-by-Step Getting Started Guide
- Host Setup
- Debugging an Example Application
- Building the Linux BSP
- Boot Configurations

2 System Requirements

2.1 Hardware

The following hardware components are included in the Cosmic Board for phyCORE-Vybrid Kit and are necessary for completing the instructions in this Quickstart:

- Cosmic Board for phyCORE-Vybrid (POB-001)
- Micro SD Card with software demo
- Micro USB cable
- Straight Ethernet cable
- Serial RS-232 cable (optional)
- AC adapter supplying 5 VDC / 3.2 A, center positive (optional)

2.2 Software

- A modern GNU/Linux Operating System on your host machine running natively or via virtual machine is required to update your SD card and to successfully build a customized Linux distribution using the Timesys Desktop Factory. One of the following distributions are recommended:

- Ubuntu: Most recent release or LTS (currently 12.04 32 or 64 bit)- Fedora: Most recent release

- This Quickstart was authored and tested with Ubuntu 12.04 LTS Virtual Machine on a Windows 7 Host PC. Alternative host options are described here: <https://linuxlink.timesys.com/docs/wiki/factory/FactoryHostRequirements>. Access to this link requires a LinuxLink registration, see [section 4.2.1](#).

3 Getting Started

Start communicating with your Cosmic board from your Linux Host PC by completing the following steps:

1. Insert the micro SD card that came in the kit in to the micro SD card slot connector X18 on the Cosmic Board. Be sure to insert the card label down and pins facing up. After aligning the card with the connector, push to insert. The card will make a clicking sound and latch into the connector.
2. Connect the kit supplied USB Micro-AB cable from a free USB port on your host PC to the connector X6 on the Cosmic Board. This both supplies power and will provide a virtual Ethernet over USB interface. You will instantly see the power LED light up solid red and it will take Linux approximately 15 seconds to boot. You should see the user LEDs in the following configuration:

D11 (green), D5 (red) OND10 (green), D4 (red) OFF

Troubleshooting

Not seeing the expected LED behavior?

- Power LED does not light up solid red - Check that jumper JP1 is set to 2+3 specifying USB power
- User LEDs not in the default configuration - Check that the boot mode is set for SD/MMC by verifying that JP2 is open

3. The Linux Host PC will detect the Cosmic as a CDC Ethernet Device and the network interface will show up in **ifconfig** as **usb0**. To connect the device over the CDC Ethernet protocol execute the following on your Linux Host PC:

```
sudo ifconfig usb0 192.168.6.1 netmask 255.255.255.252
```

4. Use SSH on your Linux Host PC to connect to the device:

```
ssh root@192.168.6.2
```

Once you have established a connection you should see the following:

```
BusyBox v1.20.2 (2013-08-01 16:10:09 PDT) built-in shell (ash)
Enter 'help' for a list of built-in commands.

#
```

Troubleshooting

Can't connect to the target?

- It is possible that the interface was configured before Linux fully booted. Try resetting the **usb0** interface and retry connecting to the target:

```
sudo ifconfig usb0 down
sudo ifconfig usb0 192.168.6.1 netmask 255.255.255.252
```

5. Before disconnecting power, properly shutdown the system by executing the **poweroff** command. Failure to properly shut down before removing power or pressing the reset button can result in filesystem errors and is not advised.

4 Host Setup

The Cosmic Board-Vybrid (PCL-052) has been developed and tested with Ubuntu 12.04 LTS Precise Pangolin [\[Installation Guide\]](#). 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
```

4.1 Server Setup

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

4.1.1 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 U-Boot.

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+rw <TFTP_DIRECTORY>
```

4.1.2 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
```

4.1.3 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.

4.2 LinuxLink Tools Setup

LinuxLink, created by Timesys, provides web and desktop factory tools for embedded Linux development to manage toolchains, build environments, and software sources. After registering your kit, access to the BSP as well as to additional resources for Vybrid development will be provided.

4.2.1 LinuxLink Registration

Create a Timesys LinuxLink account for the PHYTEC Vybrid processor:

1. Go to <https://linuxlink.timesys.com/register/phytec/>
2. Complete all required sections on the registration form
3. Enter the serial number on the SOM for the **Vybrid Board MAC ID**
The serial number will be a 12 digit MAC ID printed on a sticker on the SOM. Enter the 12 hexadecimal digits separating each byte by a ":" or "-"

Example: MAC ID = 01:23:45:67:89:AB, Enter 01:23:45:67:89:AB or 01-23-45-67-89-AB

4. Click the **Get Access Now!** button to submit the registration form

Once the registration form is submitted, expect a welcome email containing login information including user name and password as well as helpful links to get started.

After logging into LinuxLink [[Here](#)], the custom personal dashboard will be brought up which acts as a hub of information for embedded Linux development including relevant software, tools, educational documentation, and links to custom builds.

4.2.2 Desktop Factory Build System

The Timesys Factory Build System allows full access to the target preconfigured build system for customization and development on a host Desktop. For application development, installation of the Factory Build System is not required. If you plan to modify the phyCORE-Vybrid Linux kernel, use the following instructions to download and install the Desktop Factory Build System.

LinuxLink pre-built starting points for the PHYTEC phyCORE-Vybrid include a download for a preconfigured Desktop Factory. Download the installer from the Build Output page under Build the SDK on your host -> Download the Desktop Factory Installer -> pcm052-factory-installer.sh.

Note: See the Linux BSP release notes page for a link to the most recent pre-built Build Output page or view all in LinuxLink from the Download BSP/SDK tab under Pre Built Starting Points -> PHYTEC phyCORE Vybrid Development Kit.

```
chmod +x pcm052-factory-installer.sh
./pcm052-factory-installer.sh
```

After successfully installing the Desktop Factory, the tools will be available from the `/timesys/pcm052/Factory-<version>` directory.

Note: Throughout the Quickstart any reference to `<factory_install_dir>` refers to `/timesys/pcm052/Factory-<version>`.

Alternatively, a manual install of the Factory Build Engine can be performed. In the case of an upgrade, it is recommended to back up the existing factory directory and then extract the new release, which allows preservation of the unchanged platform definition. Download and extract *factory.tar.gz* from **View All Files** on the Build Output page.

```
tar -zxvf factory.tar.gz
cd factory
```

The Factory Build system requires distribution specific packages on the host that should match the specifications summarized by Timesys [\[Here\]](#).

If using Ubuntu, complete the following:

```
sudo apt-get install build-essential libc6-dev libtool sharutils libncurses5-dev libgmp3-dev libmpfr-dev gawk
gettext bison flex gperf indent texinfo libgtk2.0-dev libgtk2.0-bin libsdl1.2-dev swig python-dev texlive-latex3
texlive-extra-utils binutils-dev automake guile-1.8 icon-naming-utils libdbus-glib-1-dev wget gtk-doc-tools
libxml-parser-perl zip unzip ecj fastjar x11-xkb-utils libglade2-dev libperl-dev python-libxml2 libexpat1-dev
gconf2 groff libc6-dev-amd64

sudo dpkg-reconfigure dash
# Respond "No" to the prompt asking "Install dash as /bin/sh?"

# For 64-bit host machines only
sudo apt-get install ia32-libs
sudo apt-get install libc6-dev-i386
```

To check that the host system has all required software run a **checksystem** within the factory directory, and follow the instructions for resolving missing requirements if applicable:

```
make checksystem
```

4.3 SDK Install

The cross toolchain, pre-built images, and kernel source are all made available from the Software Development Kit (SDK). If you plan to write Linux application software but not modify the phyCORE-Vybrid Linux BSP, use the following instructions to download and install the SDK.

Download the SDK installer, *pcm052-development-environment.sh*, from the [PHYTEC FTP](#) or from LinuxLink on the Build Output page of a pre-built starting point.

Note:

See the Linux BSP release notes page for a link to the most recent pre-built Build Output pages or view all in LinuxLink from the Download BSP/SDK tab under Pre Built Starting Points -> PHYTEC phyCORE Vybrid Development Kit

```
cd Downloads
chmod +x pcm052-development-environment.sh
./pcm052-development-environment.sh
```

The SDK is now installed and located in the destination directory, ex. `/home/<user>/timesys/pcm052`. From this directory the toolchain, pre-built images, and kernel source can be accessed.

Note:

Throughout the Quickstart any reference to `<sdk_install_dir>` will refer to the SDK installation directory, `/home/<user>/timesys/pcm052`.

After successful installation, to use the toolchain your PATH environment variable must be updated. Use the following with **export** or permanently add to your PATH by including it in `.bashrc`:

```
PATH=<sdk_install_dir>/toolchain/ccache:<sdk_install_dir>/toolchain/bin:$PATH
```

4.4 ARM Development Studio 5 (DS-5)

ARM Development Studio 5 (DS-5) is a set of software tools optimized for ARM processor-based embedded system development. If you plan to modify the phyCORE-Vybrid MQX BSP or write MQX application software, use the following instructions to download, install, and obtain a license for DS-5.

DS-5 is downloadable and available free of charge for one year through the DS-5 Starter Kit for Vybrid Controllers. The starter kit will allow immediate access to the ARM specific DS-5 Compilation Tools and DS-5 Debugger on a customized Eclipse IDE. Go to <http://ds.arm.com/vybrid/vybrid-tower-starter-kit/> for the DS-5 installation file download and license activation code.

Download and Install DS-5:

- Select the installer for your host platform

Note:

Linux is recommended to follow along with the Quickstart instructions

- Extract the DS-5 download file and run the installer

Note:

Replace <DS500-BN-Version> with the version downloaded, ex. DS500-BN-00004-r5p0-1rel1.tgz, and <arch> with the corresponding architecture value for the host platform, ex. install_x86_32.sh for a 32-bit architecture.

```
tar -xvzf <DS500-BN-Version>.tgz
sudo ./install_x86_<arch>.sh
```

- Follow the on-screen instructions to finish installation

Note:

It is recommended to install the device drivers prompted during installation.

After successful installation, to start using ARM DS-5 from the command line add `/usr/local/DS-5/bin` to your PATH.

To permanently add to your PATH, include the following to `.bashrc`:

```
# To use ARM DS-5 at the command prompt
PATH=$PATH:/usr/local/DS-5/bin
```

Start ARM DS-5 and activate the Starter kit license:

Note:

If upgrading from the 30-day free trial to the Vybrid Starter kit license be sure to complete the scatter file modifications described in [Section 4.2.4](#)

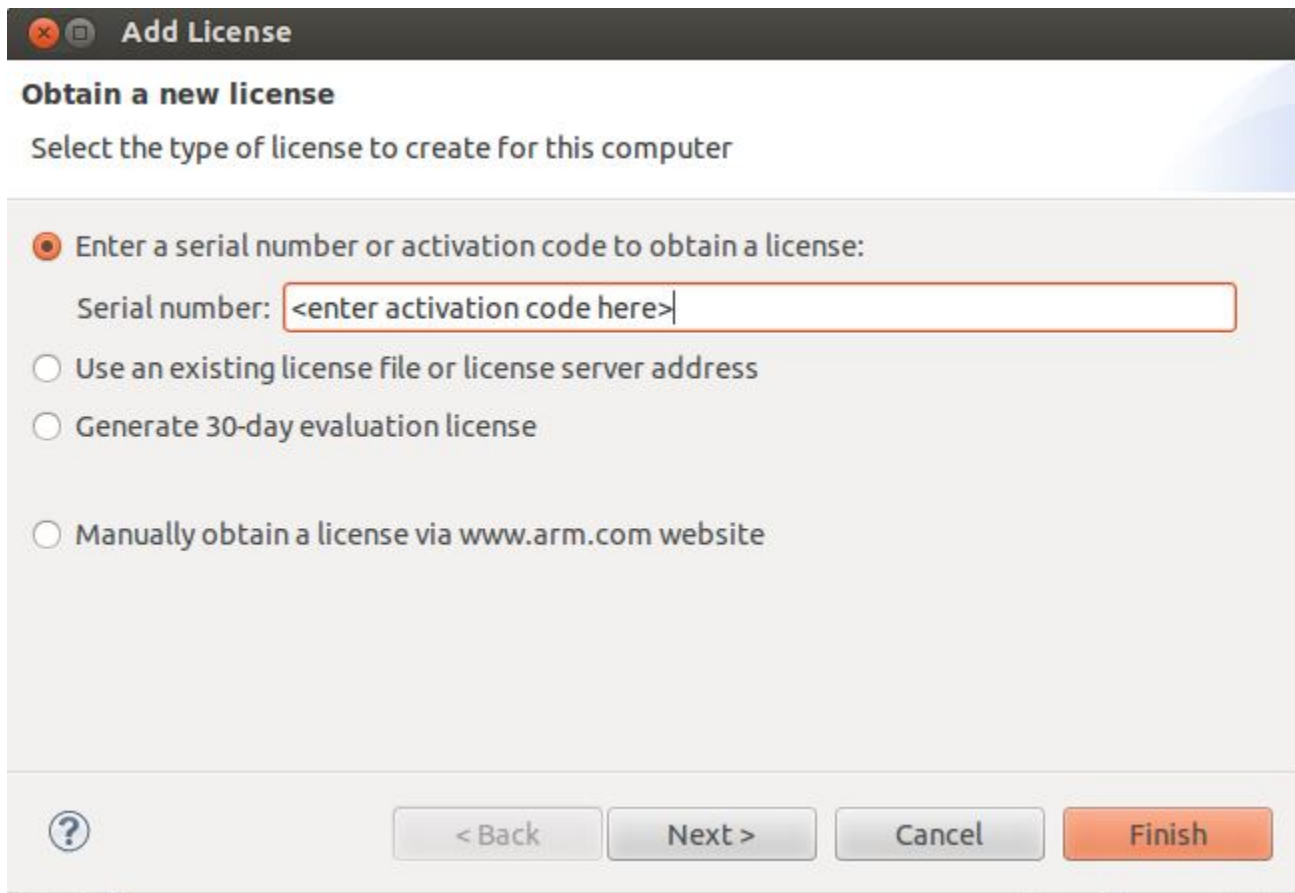
- Launch DS-5 with the Eclipse IDE:

```
eclipse &
```

- Specify a Workspace for storing Eclipse settings and projects, ex. `/home/<user>/DS-5`
- The first time Eclipse DS-5 is started, the License Manager will automatically open.

To manually open the License Manager, in Eclipse DS-5 select Help -> ARM License Manager

- Click **Add License...**
- Select the **Enter a serial number or activation code to obtain a license:** option
- Enter the activation code displayed on <http://ds.arm.com/vybrid/vybrid-tower-starter-kit/>
- Click **Next**



The image shows a screenshot of the 'Add License' dialog box in the ARM License Manager. The dialog has a title bar with a close button, a maximize button, and the text 'Add License'. Below the title bar, the main heading is 'Obtain a new license', followed by the instruction 'Select the type of license to create for this computer'. There are four radio button options: 1. 'Enter a serial number or activation code to obtain a license:' (selected), which includes a text input field containing '<enter activation code here>'. 2. 'Use an existing license file or license server address'. 3. 'Generate 30-day evaluation license'. 4. 'Manually obtain a license via www.arm.com website'. At the bottom of the dialog, there is a help icon (question mark in a circle) on the left, and four buttons: '< Back', 'Next >', 'Cancel', and 'Finish' (highlighted in orange).

Add License

Obtain a new license

Select the type of license to create for this computer


☒ Enter a serial number or activation code to obtain a license:

Serial number:

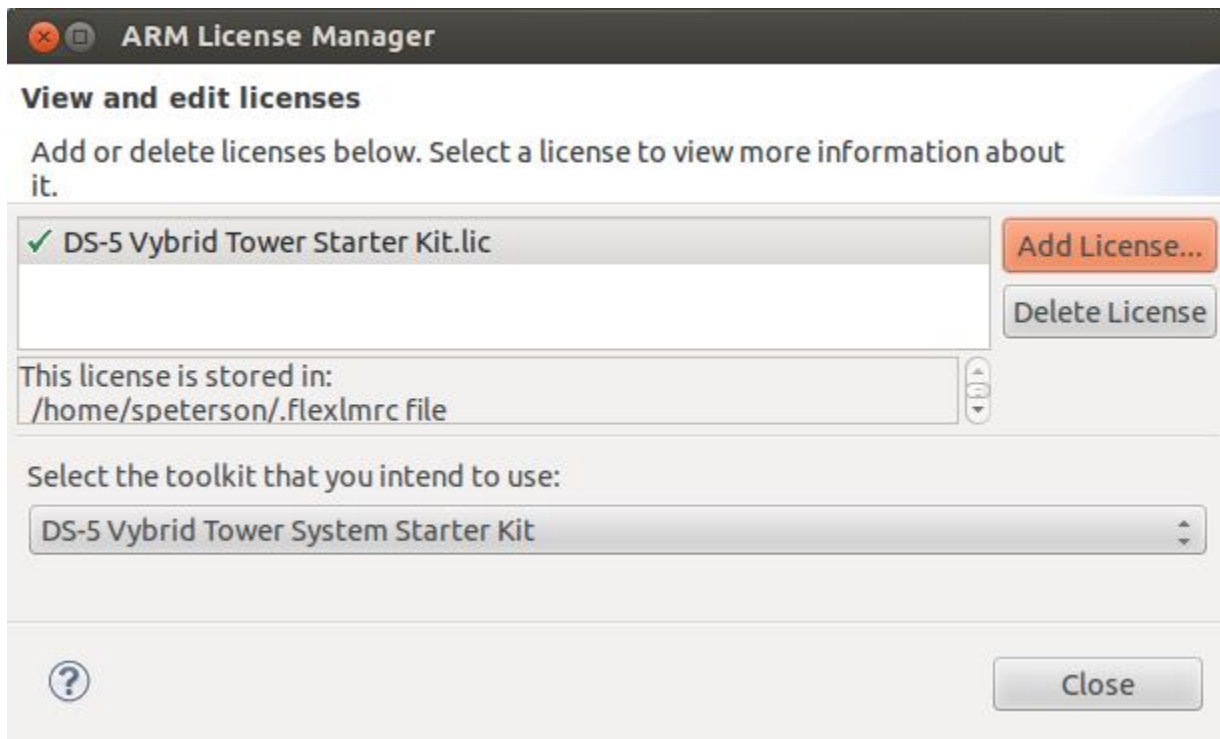
☐ Use an existing license file or license server address

☐ Generate 30-day evaluation license

☐ Manually obtain a license via www.arm.com website



- Choose / Verify the Host ID
- Click **Next**
- Enter an existing ARM account username and password or create an account
- Click **Finish**
- Be sure *DS-5 Vybrid Tower System Starter Kit* is the selected toolkit in the ARM License Manager
- Close the ARM License Manager
- Restart Eclipse to apply license changes



4.4.1 TimeStorm

The Timesys TimeStorm tools allow customization of the Linux kernel, filesystem, and applications on an Eclipse based IDE. If you plan to write Linux application software, use the following instructions to download and install the TimeStorm plug-in for DS-5.

Note:

If you are not using ARM DS-5, the option to download the TimeStorm IDE or plug-in for Eclipse is available on LinuxLink from the [Get IDE](#) tab.

Download the installer from the LinuxLink dashboard under Private Downloads -> TimeStorm plugins 4.3.1 - 32 bit.

```
cd Downloads
chmod +x tsplugins-x86.sh
sudo ./tsplugins-x86.sh
```

The TimeStorm plug-in is now installed and available through Eclipse DS-5.

5 Building the Linux BSP

5.1 Web Factory Wizard

The LinuxLink cloud-based development environment provides access to current build tools, configuration data, and platform prototypes providing elimination of delays attributed with installing, configuring, and maintaining a local build environment.

A workorder is required to use the LinuxLink Web Factory services for the assembly of a customized board support package. As part of the target hardware project, a workorder can be customized in terms of platform configuration options and submitted to be built on the Timesys servers.

To get started with the LinuxLink Web Factory Wizard create a phyCORE-Vybrid Board based project, it will be used to develop web based workorders built for the PHYTEC phyCORE-Vybrid platform:

Create a Project

1. Go to **Build BSP/SDK** in LinuxLink
2. Click **Create a Project**
3. Enter a *Name* and *Description*
4. Select *PHYTEC phyCORE-Vybrid Development Kit* for a Board
5. Select an Application (optional)
6. Click **Create Project**

Create a New Workorder

Create a new workorder to describe a specific BSP:

1. Open the phyCORE-Vybrid project
2. Click **Create a Workorder**

3. Enter a Workorder *Name* and *Description*
4. Optionally, click **Next** to be guided through custom options for building a BSP/SDK
5. Click **Save**

Copy an Existing Workorder

Copy an existing workorder to take advantage of pre-configured platform specifications. For example, copy the PD13.0.2 MCC Demo workorder to inherit the kernel, toolchain, and a default set of packages selected along with optimized build configurations for a great starting point to add target specific customizations:

1. Click the copy icon
2. Enter new workorder *Name* and *Description*
3. Click **Save**

Modify an Existing Workorder

An existing workorder can be edited:

1. Click on the name/link of the workorder
2. For a component, click on the **Edit** button

5.1.1 Managing Workorders

A workorder process can be customized in terms of the Linux components including the kernel, toolchain, and package versions along with the licensing information, root filesystem optimizations, and build output options. Use the PHYTEC pre-built starting point BUILD-OUTPUT.txt files as a reference for a default configuration including a minimal set of packages to be run on the target application.

The Web Factory Wizard user interface provides simple navigation through sub categories available in the top menu for configuring the BSP/SDK.

Kernel Configuration Kernel Configuration allows selection of a Linux kernel from the Timesys provided options. By clicking on the kernel version link more information will be displayed about the kernel including the version, revision, estimated build time, license, and device drivers enabled. Additionally, the kernel .config file is available for download for inspection of the options as configured in the Linux kernel.

Toolchain Configuration The Toolchain Configuration menu provides options for toolchain selection. It is recommended to select glibc Recommended because it provides a complete set of toolchain components and versioning. If a different version of toolchain components such as glibc, gcc, binutils, and gdb is preferred, the custom toolchain option can be selected. More information including the version, revision, license, and an estimated build time for a toolchain component is available by clicking the component link.

Packages Configuration A table format that details the package names, version, license, and size information is available in the Packages Configuration menu. Browse through the Demo, Desktop, Development, Graphics, Multimedia, Networking, Runtimes, System, and Utilities categories or use the Smart Search feature to choose packages required for the target application. By clicking on the package, details including version, estimated build time, and license are provided. Additionally, some packages have modifiable build options which are accessed by liking the associated icon in the package listing. Checking the box next to the package name will select the package and automatically any package dependencies. Similarly, unchecking the box next to a package name will deselect the package and automatically any package dependencies. View the left panel for a summary of the both all the packages selected and dependencies as well as the total packages size.

Build Output Options The Build Output Options menu allows the user to tailor the application and root filesystem output by providing options to select the specific file formats. Additionally, optimizations for along with the choice to include the native toolchain and kernel in the root filesystem are provided.

Advice The Advice menu or Timesys Recommendations can be used to detect any build and run or logic dependencies based on Timesys Recommendations. A user can Accept which will apply the recommendations or continue with the selections made with risk to having conflicts, incompatibilities, and deficiencies.

Summary Use the Summary menu to review the workorder. Additionally, and use the Edit buttons as quick links to modify configurations. Based on the custom workorder, an estimated build time is generated and can be viewed at the bottom of the summary. It is important to note that this estimate does not take into account the number of builds currently being processed on the Timesys servers, therefore, a build will generally take longer than the estimated time.

After completing modifications to the workorder, click on the **Save** button at the bottom of the page to apply changes.

5.1.2 Building Images

A build is initiated on the **Summary** page of a workorder. To open the **Summary** from LinuxLink click **Build BSP/SDK**, select the project and click on the name of the workorder that is to be built. Alternatively, from an open workorder click **Summary** in the top menu. This menu provides a review of the workorder configuration and an estimated build time. Click the **Build** button to start the build.

The workorder will be queued for a short amount of time before being built on the Timesys servers. The status of the build can be viewed on the personal LinuxLink dashboard or the **Download BSP/SDK** page. Additionally, when the build is complete, these pages will display if the build was successful and provide a link to the download. Similarly, a user will receive notification via email when a build is complete with a link to the download.

5.1.3 Downloading Images

After notification of a successful build, links to the download for the BSP are provided via the Build Download Page or the Build Output Page. The BSP Download page can be opened after a successful build via the link provided by email notification or in LinuxLink from the personal dashboard or **Download BSP/SDK** page. It highlights relevant files from the Build output along with instructions to support different user intentions including setting up application development on the host, building the SDK on the host, and booting the board. Click the **View All Files** button at the bottom of the page for the complete Build output.

The Build Output Page is a resource for all information relating to the build of a BSP/SDK. The page includes a build summary; package sources; individual binary files for the kernel, toolchain, bootloader, and root filesystem; and the desktop factory download. The build summary, *BUILD-SUMMARY.txt*, provides a list of configuration and build instructions associated with the platform including the system specifications, kernel and bootloader versions, toolchain components, host build tools, package selections, and root filesystem options.

5.2 Desktop Factory Build System

The BSP can be built from using the Desktop Factory, setup in [Section 4.2.2](#).

5.2.1 Board Files

All source code is located in the `<factory_install_dir>/build-armv7l-timesys-linux-gnueabi/` directory. To 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 board files summarized by the following:

Description	File
Kernel	linux-3.0/arch/arm/mach-mvf/board-pcl052.c
Bootloader	u-boot-2011.12/u-boot-2011.12/board/phytec/pcl052.c

5.2.2 Modifying the Platform

The Desktop Factory uses the kernel configuration (KConfig) system to include the packages, kernel, and toolchain versions and configurations. Make rules, performed on configuration symbols defined in KConfig, are defined to execute commands for the standard configuration, build, and install processes for the kernel, packages, and toolchain components. Therefore, target commands can be used with make directly on the command line.

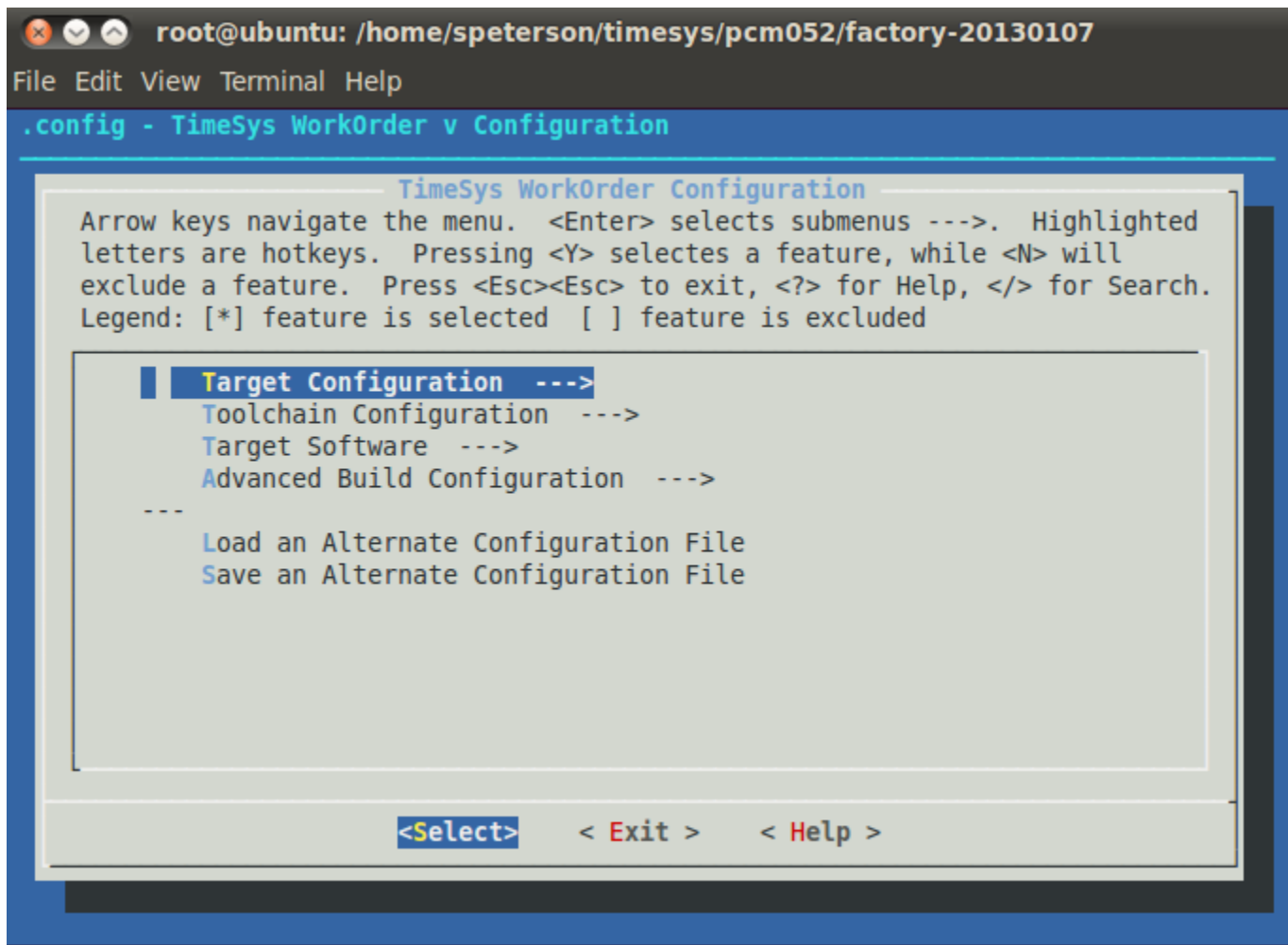
For more information see the [Make](#) documentation provided by Timesys or use the **help** make target to view a list describing the common target commands.

A set of make rules are defined specifically for configuring targets by updating the configuration file. From [Config.in](#) files throughout the Desktop Factory, KConfig keys and their values are listed together in a menu to allow straightforward user configuration and customization of the platform.

Use the **menuconfig** make target on the command line in the BSP directory, `<factory_install_dir>/`, to modify the settings for the platform:

```
make menuconfig
```

This menu can be used to customize and configure the platform:



When making modifications, it is recommended to begin with the **Target Configuration** because of the dependencies other platform configuration have based on the chosen target. The options available in this menu range from architecture and CPU choice to root filesystem image types and target build specifications.

Toolchain specific configuration is provided by the **Toolchain Configuration** menu including customization for each toolchain component and a selection to either build a toolchain, optionally a native toolchain for the target as well, or fetch an existing pre-built toolchain. To customize the target software that can be included in the platform use the **Target Software** menu which provides sub menus to the Kernel, Bootloader, and Software Packages. Refer to [Factory Configuration System Overview](#) for additional documentation provided by Timesys.

To adjust the drivers and support included in a Linux kernel build, invoke the Linux kernel menuconfig interface:

```
make kernel-menuconfig
```

.config - Linux/arm 3.0.15-ts-armv7l Kernel Configuration**Linux/arm 3.0.15-ts-armv7l 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 < > module capable

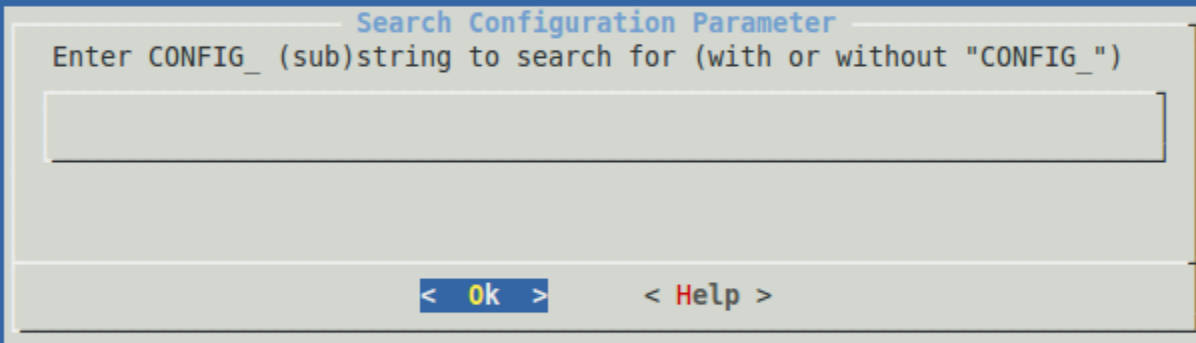
[] Patch physical to virtual translations at runtime (EXPERIMENTAL)

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 --->

v(+)

<Select> < Exit > < Help >

When using **menuconfig**, the location of a name or token can be searched by using the "/" key:

.config - Linux/arm 3.0.15-ts-armv7l Kernel Configuration**5.2.3 Building Images**

The build system is invoked by executing make targets from the command line in the BSP directory, *<factory_install_dir>/*. With no targets specified, executing **make** will build the entire BSP through implementing a set of core Factory targets:

```
make
```

Note:

Execute *make help*, the list of included make targets when executing *make* will be designated by a '*' mark.

All images are then stored in *<factory_install_dir>/build_armv7l-timesys-linux-gnueabi/images*.

Build Time Optimizations

The build time will vary depending on the package selection and Host performance. After the initial build when modifications to the same BSP are made, a full build is not required and taking advantage of optimized build options specific to modifications will reduce the build time. Use the following as a reference for build commands that for building parts of the BSP:

To rebuild U-Boot:

```
make u-boot-reconfigure
make u-boot-build
make u-boot-host-install
```

To rebuild the kernel:

```
make kernel-restage
make kernel-build
make kernel-install-image
```

Add an application to the root filesystem:

After selecting the package in **make menuconfig**:

```
make application-package
make rfs
```

To remove an application from the root filesystem:

```
make rfs-distclean
make rfs
```

5.3 Enabling Multi-Core Communication

If your application requires MCC support to include a MQX application use either the Web Factory or Desktop Factory tools to include the following packages in the Linux Build:

Package	Description
mcc-kmod	Kernel driver module
libmcc	API user-space library
mqxboot	Utility to load and boot an MQX image

For more information regarding the MCC subsystem for enabling MQX applications running on the Cortex-M4 core to communicate with Linux applications on the Cortex-A5 core see [Vybrid MCC User Guide for MCC Version 1.0](#).

6 Managing Images

The SDHC card that comes with the Cosmic Board for phyCORE-Vybrid is pre-loaded with kit demo images. Different images located on the [PHYTEC FTP](#) or built images can be installed on a SD Card and run on the target.

To create an SD Card the following images will be required:

- Bootloader: u-boot.imx
- Linux kernel: ulmage-30.-ts-armv7l
- Root Filesystem: rootfs.tar.gz

If using a SD card that has already been formatted with this script or a similar method, and the bootloader is not changing, skip to the appropriate steps below-- Step 6 if the Linux kernel is changing or Step 8 if the root filesystem is the only change and continue the steps from there.

Bootloader

1. Make a script to correctly create and format the card

The following is the contents of a personal script, vybrid_sd_card.sh, that uses the dd command in order to place u-boot.imx on the SD/MMC Card.

```
#!/bin/bash
if [ $# -ne 2 ]
then
    echo "$0 usage: designate the SD card block device to format (e.g /dev/sdb)
$0 usage: supply the location of u-boot.imx file to flash (e.g. ~/images/u-boot.imx)" 1>&2
    exit 1
elif [ ! "$1" = "/dev/sda" ] ; then
    unset LANG
    DRIVE=$1
    BOOTIMG=$2
    if [ -b "$DRIVE" ] ; then
        dd if=/dev/zero of=$DRIVE bs=1024 count=1024
        parted --script ${DRIVE} mklabel msdos
        dd if=$BOOTIMG of=$DRIVE seek=2 bs=512
        parted --script ${DRIVE} mkpart primary ext2 2 52
        parted --script ${DRIVE} mkpart primary fat32 53 100
        parted --script -- ${DRIVE} mkpart primary ext2 101 "-1"
        mkfs.ext2 ${DRIVE}1 -L boot
        mkfs.vfat ${DRIVE}2 -F 32 -n kernel
        mkfs.ext2 ${DRIVE}3 -L rootfs
    fi
fi
```

Note:

This is a personal script, if executed, a user is at their own risk.

2. Determine the SD card device name

The SD card device name is of the form /dev/sd[b|c|d|e]. Run the following without and with the SD card connected:

```
ls /dev/sd*
```

The device that appeared on the call to ls with the SD card but not the call without is the SD card device.

3. Unmount all partitions of the SD card, using the SD card device name from Step 2:

```
umount /dev/sd[b|c|d|e]*
```

4. Run the script created in Step 1 specifying the SD card device name from Step 2 and location of the bootloader, u-boot.imx:

```
sudo ./vybrid_sd_create.sh /dev/sd[b|c|d|e] <path to u-boot.imx>/u-boot.imx
```

5. Mount all partitions

Remove and reinsert the SD card, automount will mount /media/boot, /media/kernel, and /media/rootfs

Kernel

6. If modifying the Linux kernel, remove the existing uImage-3.0-ts-armv7l file:

```
rm -rf /media/kernel/*
```

7. Load the new Linux kernel to the SD Card:

```
cp <path to uImage-3.0-ts-armv7l>/uImage-3.0-ts-armv7l /media/kernel; sync
```

Root Filesystem

8. If modifying the root filesystem, remove the existing:

```
sudo rm -rf /media/rootfs/*
```

9. Load the new filesystem to the SD Card:

```
sudo tar -zxvf <path to rootfs.tar.gz> -C /media/rootfs/; sync
```

10. Unmount each partition before removing the SD Card:

```
umount /media/boot /media/kernel /media/rootfs
```

7 Boot Configurations

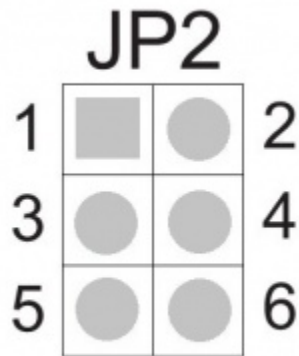
7.1 Selecting Boot Modes

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 boot mode, selected from the S5 dipswitch on the Carrier Board, determines the location of the primary bootloader (u-boot.imx). Set the boot mode according to the following:

SD Card

By default the Cosmic Board is setup to boot from SD Card:

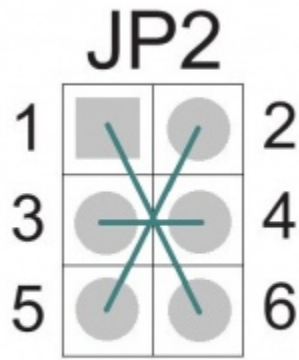
JP2: OPEN



NAND

Use the jumper wires provided with the kit to connect JP2 in the following way:

JP2: 1+6; 3+4; 5+2



7.2 Basic Settings

The follow section requires access to the serial terminal in U-Boot, which requires a serial connection to the Cosmic Board via RS-232.

1. Connect the serial RS-232 cable from a free port on your Host PC to the connector, *RS-232 (console)* on the Cosmic Board.
2. Start your favorite terminal software (Windows: PuTTY or TeraTerm; Linux: Minicom) on your Host PC and configure it for 115200 baud, 8 data bits, no parity, and 1 stop bit (8n1).
3. Apply power to the Cosmic Board by connecting the kit supplied USB Micro-AB cable from a free port on your Host PC to connector X6 on the Cosmic Board. After application of power, approximately three seconds are allotted for the user to hit any key which will halt autoboot and enter U-Boot:

```

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

U-Boot 2011.12 (Dec 19 2012 - 14:53:34)

CPU:   Freescale VyBrid 600 family rev1.0 at 0 MHz
Board: Vybrid
DRAM:  256 MiB
WARNING: Caches not enabled
NAND:  512 MiB
MMC:   FSL_SDHC: 0
In:    serial
Out:   serial
Err:   serial
Net:   Link UP timeout
FEC0, FEC1
Hit any key to stop autoboot:  0
pcm052 u-boot>

```

Set U-boot's network environment variables to match your required network settings:


```
setenv ethaddr ##.##.##.##.##.##
setenv ipaddr ###.###.###.###
setenv serverip ###.###.###.###
setenv gatewayip ###.###.###.###
setenv netmask ###.###.###.###

setenv tftploc <TFTP image location>
setenv nfs_root /<NFS mount location>
```

ethaddrMAC address for the device. Set this to the MAC ID, which is printed on the sticker on the SOM.

ipaddrA dedicated IP address for the SOM. This is crucial if TFTP will be used for updating the device's images at any point.

serveripIP address of the host or another machine where the TFTP directory, if it exists, is located.

gatewayipGateway IP for the network. This is only necessary if the TFTP directory is located on another network.

netmaskNetmask for the network: typically 255.255.255.0. This is only necessary if the TFTP directory is located on another network.

tftploc (required for TFTP)Location of the path to the images on the TFTP server on the host system, setup in [Section 4.1.1](#). Set the variable accordingly by referencing the following examples:

File Path	U-Boot Command
/var/lib/tftpboot/PHYTEC/Vybrid/PD13.0.2	<code>setenv tftploc PHYTEC/Vybrid/PD13.0.2</code>
/var/lib/tftpboot	<code>setenv tftploc</code>

nfs_root (required for NFS)Location of the path to the NFS directory on the host system, setup in [Section 4.1.2](#). For example: /home//phyCORE-NFS

Use the following command to verify the environment variables are set as intended:

```
pcm052 u-boot> printenv

baudrate=115200
bootargs_base=setenv bootargs rw mem=256M console=ttymxcl,115200n8 init=/sbin/init
bootargs_net=setenv bootargs ${bootargs} root=/dev/nfs ip=dhcp nfsroot=${serverip}:${nfs_root},v3,tcp
bootargs_sd=setenv bootargs ${bootargs} root=/dev/mmcblk0p3 rootwait rootfstype=ext2
bootcmd=run bootcmd_sd
bootcmd_net=run bootargs_base bootargs_net; tftpboot ${loadaddr} ${tftploc}${bootfile};bootm
bootcmd_sd=run bootargs_base bootargs_sd; mmc rescan; fatload mmc 0:2 ${loadaddr} ${bootfile}; bootm ${loadaddr}
bootdelay=3
bootfile=uImage-3.0-ts-armv7l
ethladdr=00:e0:0c:bc:e5:61
ethact=FEC0
ethaddr=00:e0:0c:bc:e5:60
gatewayip=192.168.3.1
ipaddr=192.168.3.10
loadaddr=0x80010000
mem=260096k
netmask=255.255.255.0
nfs_root=/path/to/nfs/root
serverip=192.168.3.11
stderr=serial
stdin=serial
stdout=serial
tftploc=/path/to/tftp/directory
```

After confirming the environment variables are correct, save them and continue on to the next section to set the correct kernel and rootfs boot location:

```
saveenv
```

Note:

help is a useful tool in U-Boot to show available commands and usage.

7.3 Stand-Alone Boot

By default, the Cosmic Board for phyCORE-Vybrid kit is setup to boot the Linux kernel and root filesystem from SD card. If switching from another boot configuration back to SD, use the predefined environment variable to set this mode:

```
setenv bootcmd 'run bootcmd_sd'
saveenv
```

7.4 Remote Boot

To configure U-Boot to boot the kernel from TFTP and mount the root filesystem from NFS, set the boot command to use the predefined environment variable with the following command:

```
setenv bootcmd 'run bootcmd_net'
saveenv
```

7.5 Other Boot Options

You can create a unique boot configuration but an additional step is required. Use the following as a guide:

Boot the Linux Kernel via TFTP with Root Filesystem on SD

```
setenv bootcmd_tftp 'run bootargs_base bootargs_sd; tftpboot ${loadaddr} ${tftploc}${bootfile}; bootm'
setenv bootcmd 'run bootcmd_tftp'
saveenv
```

8 Flashing Images

8.1 Bootloader

The Cosmic Board for phyCORE-Vybrid kit does not come with U-Boot pre-loaded on NAND flash, if you wish to boot from NAND follow the steps below:

1. Obtain a U-Boot image for NAND, *u-boot.nand*. It is recommended to download the pre-built image available on the [PHYTEC FTP](#). If you have custom U-Boot requirements, or are interested in seeing the version you built in action, complete the following to create a *u-boot.nand* file from an existing *u-boot.imx*. The *u-boot.imx* file that is built with the factory does not have the correct image vector table offset required for NAND boot and must be padded with zeros. This step is only required if you are using your own built images and can be skipped if you use a pre-built *u-boot.nand* image.

```
dd if=/dev/zero of=u-boot-pad bs=1024 count=1
cat u-boot-pad u-boot.imx > u-boot.nand
```

2. Copy *u-boot.nand* from the previous step to the **/kernel** partition of your SD Card using your Linux Host PC.

```
cp u-boot.nand /media/kernel
```

Unmount all partitions before removing the SD Card from your Host PC:

```
umount /media/boot /media/kernel /media/rootfs
```

3. Place the SD Card into the connector on your board and make sure the boot settings are set to SD Card.

4. Connect the serial RS-232 cable from a free port on your Host PC to the connector, RS-232 (console) on the Cosmic Board.

5. Start your favorite terminal software (Windows: PuTTY or TeraTerm; Linux: Minicom) on your Host PC and configure it for 115200 baud, 8 data bits, no parity, and 1 stop bit (8n1).

6. Power on your board by connecting the kit supplied USB Micro-AB cable from a free port on your Host PC to connector X6 on the Cosmic Board.

After application of power, approximately three seconds are allotted for the user to hit any key which will halt autoboot and enter U-Boot.

7. Type the following at the prompt to mount the **/kernel** partition of the SD Card and copy the *u-boot.nand* image to NAND Flash:

```
nand erase 0x40000 0x40000
mw.b 0x80400000 0xff 0x40000
fatload mmc 0:2 0x80400000 u-boot.nand
nb_update 0x80400000 0x40000 0x40000
```

You should now be able to boot your board from NAND using the instructions specified previously in this Quickstart.