BSP-Yocto-TISDK-AM57xx-PD18.2.0 Quickstart

Non-Kit SOM

If working with a non-kit SOM, there are additional steps required to build this release for your configuration. Please view the notes in the **Building the BSP from Source** and **Built Images** sections for the applicable differences. For each SOM configuration there may be a required change to the Linux dts, the yocto MACHINE value, and the output images.

This Quickstart provides you with the tools and know-how to install and work with the Linux Board Support Package (BSP) for the phyCORE-AM57xx Rapid Development Kit (RDK). This Quickstart shows you how to do everything from installing the appropriate tools and source, to building custom kernels, to deploying the OS, to exercising the software and hardware. Please refer to the phyCORE-AM57xx Hardware Manual for specific information on board-level features such as jumper configuration, memory mapping and pin layout for the phyCORE-AM57xx System on Module (SOM) and baseboard. Additionally, gain access to the SOM and baseboard schematics for the phyCORE-AM57xx RDK by registering at the following: **http://phytec.com/support/registration/**.

Requirements

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

Software

- A modern GNU/Linux Operating host system either natively or via a virtual machine:
  - Ubuntu 16.04 LTS recommended, 64-bit required. Other distributions will likely work, please note that some setup information as well as OS-specific commands and paths may differ.
  - If using a virtual machine, VMWare Workstation, VMWare Player, and VirtualBox are all viable solutions.
- Root access to your Linux Host PC. Some commands in the Quickstart will not work if you don't have sudo access (ex. package installation, formatting SD card).
- At least 210GB free on target build partition and at least 4GB of RAM available to the build host.
- SD card reader operational under Linux.
  - If you do not have SD card access under Linux then formatting, copying the bootloader, and mounting the root file system on an SD card will not be possible.
- Active Internet connection

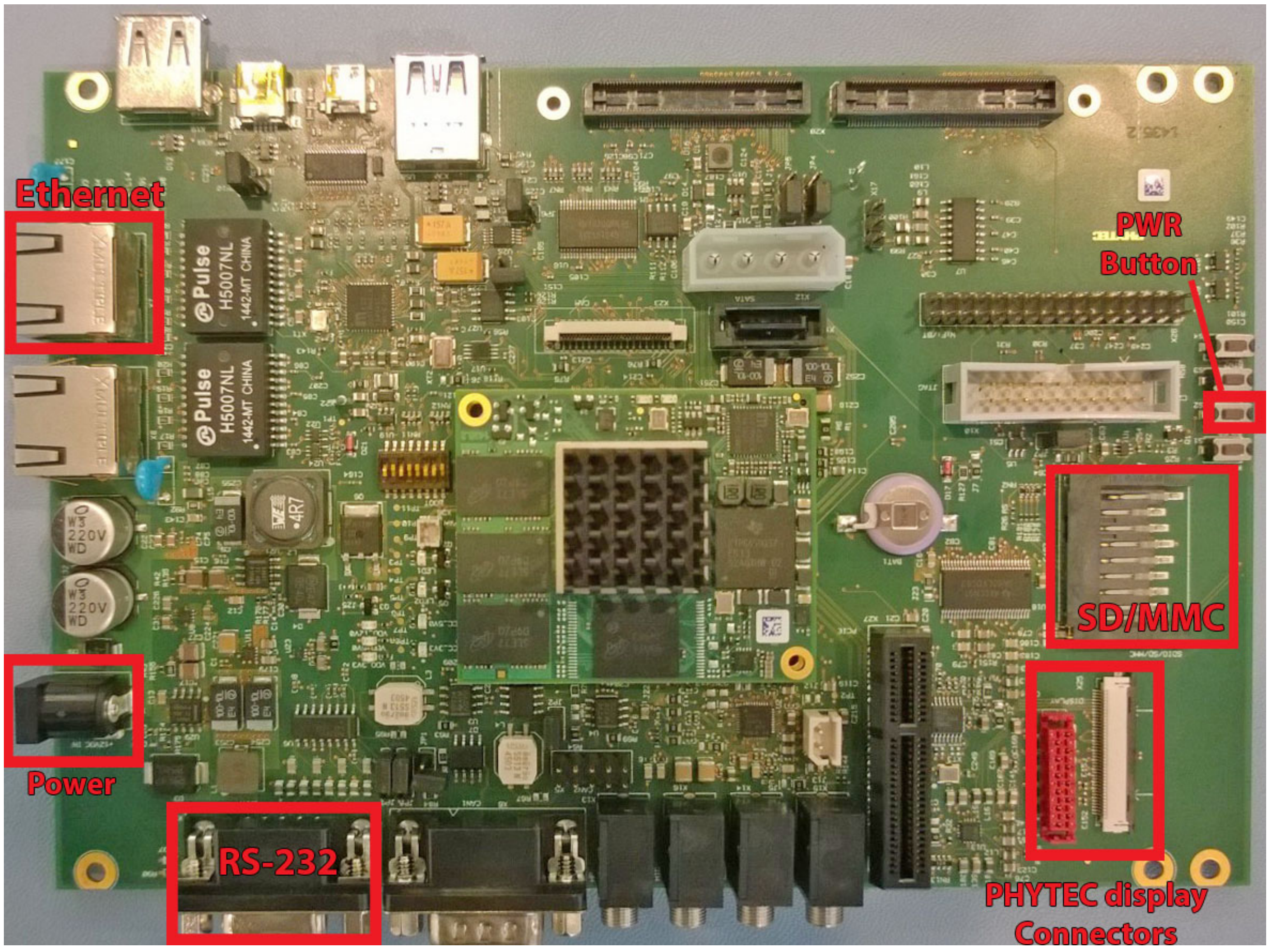
Hardware

- phyCORE-AM57xx System on Module (PCM-057)
- phyCORE-AM57xx Baseboard (PCM-948)
- Serial cable (RS-232)
- Ethernet cable
- SD Card (8GB or more recommended for flashing and development)
- AC adapter supplying 12VDC / min. 2A

See **release notes** for supported SOM and carrier board versions.

Connector Interfaces

Use the following as a reference for the connector interfaces on the phyCORE-AM57xx Rapid Development Kit that will be used in this Quickstart.



Getting Started With Binary Images

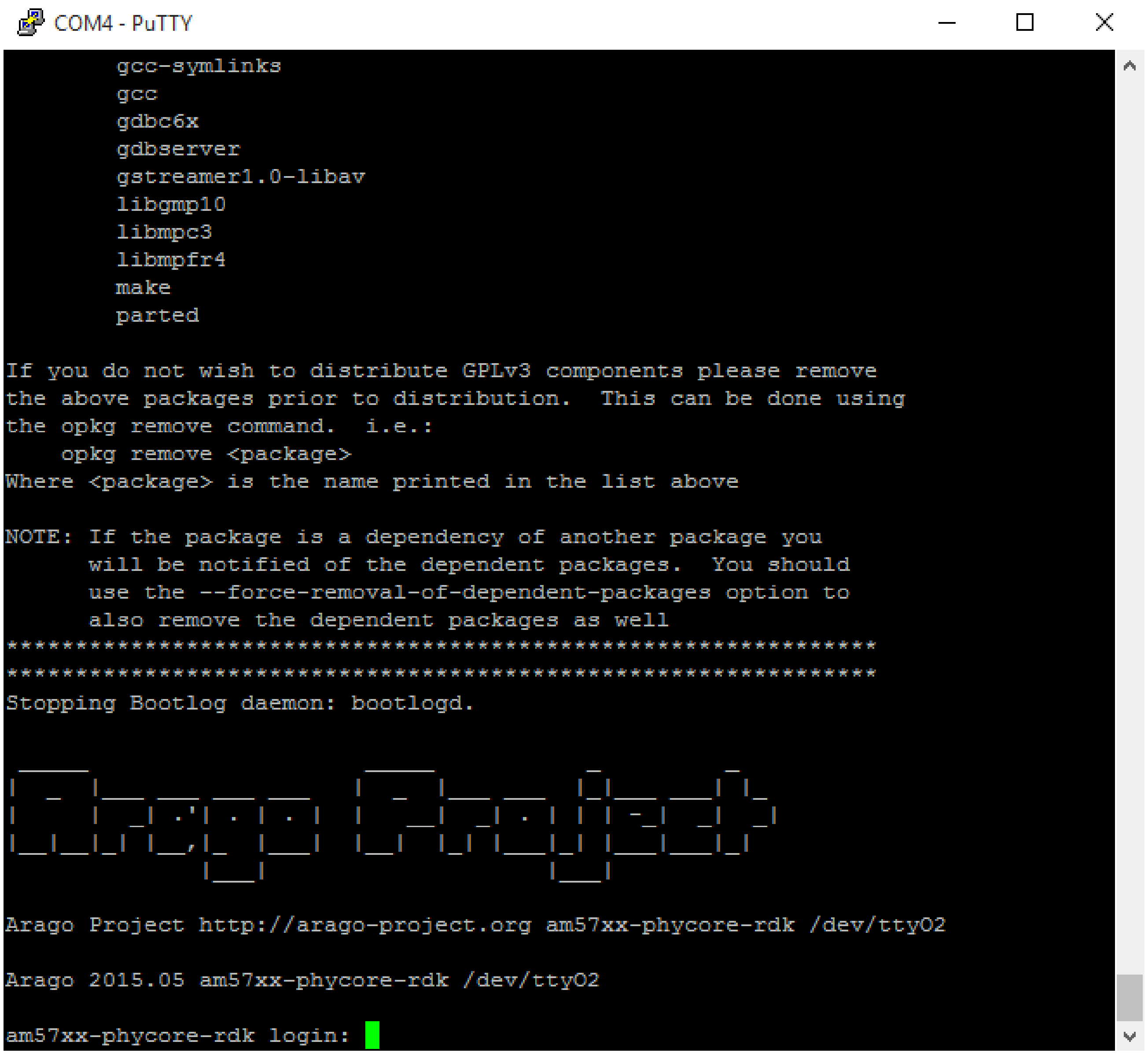
This section is designed to get the board up-and-running with pre-built images.

Booting the Pre-built Images

The section was designed to show you how to boot the phyCORE-AM57xx Rapid Development Kit with the pre-built demo images. Before starting the following steps, please **Create a Bootable SD Card** with the release images from **PHYTEC's Artifactory**, then configure the board to boot from SD/MMC with the help of the **Boot Configuration** how-to guide. If needed, you can flash your image to eMMC by following the **Flashing Images to eMMC** how-to guide.

1. Connect the kit supplied serial cable from a free serial port on your host PC to the DB9 connector X18 on the carrier board. This is the UART3 communication channel with the AM57xx at RS-232 levels.
2. Connect the kit supplied Ethernet cable from the Ethernet connector X7 on the carrier board to your network hub, router, or switch. If you do not have an Ethernet connection you can postpone this step, as Linux will boot without the Ethernet connectivity but having the connection will significantly reduce your boot time.
3. Start your favorite terminal software (such as Minicom or TeraTerm) on your host PC and configure it for 115200 baud, 8 data bits, no parity, and 1 stop bit (8n1) with no handshake.
4. Plug the kit supplied 12 V power adapter into the power connector X4 on the carrier board. You will instantly see power LEDs VCC_5V0 and VCC_3V3 on the carrier board light up solid green.

5. Press the power button S2 on the carrier board. You will now see power LEDs VDD_3V3, VDD_5V0, and VDD_12V0 on the carrier board light up a solid green. You will also start to see console output on your terminal window. If everything was done correctly the board should boot completely into Linux, arriving at a *am57xx-phycore-rdk* prompt. The default login account is *root* with an empty password. Note that the first time the board is booted it will takes a little while for the SSH server to generate new keys. Subsequent boots should be faster.



```
        gcc-symlinks
        gcc
        gdbc6x
        gdbserver
        gstreamer1.0-libav
        libgmp10
        libmpc3
        libmpfr4
        make
        parted

If you do not wish to distribute GPLv3 components please remove
the above packages prior to distribution.  This can be done using
the opkg remove command.  i.e.:
    opkg remove <package>
Where <package> is the name printed in the list above

NOTE: If the package is a dependency of another package you
      will be notified of the dependent packages.  You should
      use the --force-removal-of-dependent-packages option to
      also remove the dependent packages as well
***************************************************************
***************************************************************
Stopping Bootlog daemon: bootlogd.

 _____                    _____           _         _
|  _  |___ ___ ___ ___   |  _  |___ ___  |_|___ ___| |_
|     |  _| .'| . | . |  |   __|  _| . | | | -_|  _|  _|
|__|__|_| |__,|_  |___|  |__|  |_| |___|_| |___|___|_|
              |___|                    |___|

Arago Project http://arago-project.org am57xx-phycore-rdk /dev/ttyO2

Arago 2015.05 am57xx-phycore-rdk /dev/ttyO2

am57xx-phycore-rdk login:
```

Troubleshooting

Not seeing any output on the console?

- Check that you have setup the terminal software correctly per step 5.
- **Make sure to press the power button S2 on the carrier board.** Unlike some other PHYTEC boards, the phyCORE-AM57xx RDK does not get powered on simply by plugging in the power supply.

Building Images from Source

This section will show you how to configure your development host to build your own BSP images from source and how to start BSP image builds. Building BSP images from source is useful if you have made changes to the BSP sources and would like to deploy those changes in an easy and reproducible way.

Development Host Setup

Host Debian Packages

Yocto development requires certain packages to be installed. Run the following commands to ensure you have the packages installed:

```
sudo apt-get install git build-essential python diffstat texinfo gawk chrpath dos2unix wget unzip socat doxygen
libc6:i386 libncurses5:i386 libstdc++6:i386 libz1:i386 lib32stdc++6 lib32ncurses5 lib32z1 libc6-dev-i386 cpio
gcc-multilib
```

The above is the recommended package installation for development on a Ubuntu 16.04 LTS Linux distribution. For a breakdown of the packages as well as a list of packages required for other Linux distributions, see the "Required Packages for the Host Development System" section in the Yocto Project Reference Manual: **http://www.yoctoproject.org/docs/2.2/ref-manual/ref-manual.html#required-packages-for-the-host-development-system**

Verify that the preferred shell for your Host PC is "bash" and not "dash":

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

Repo Tool

Download and install the *repo* tool. This tool is used to obtain Yocto source from Git.

```
cd /opt
sudo mkdir bin
# /opt/ directory has root permission, change the permissions so your user account can access this folder. In
the following replace <user> with your specific username
sudo chown -R <user>: bin

cd bin
curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ./repo
# add directory that contains repo to your path
chmod a+x repo
```

Add the *repo* directory in your PATH, using **export** from the command line or permanently by including it in *.bashrc*:

```
export PATH=/opt/bin/:$PATH
```

Git Setup

If you have not yet configured your Git environment on this machine, please execute the following commands to set your user name and email address. See here for more information on getting started with Git.

```
git config --global user.email "your@email.com"
git config --global user.name "Your Name"
git config --global http.sslcainfo /etc/ssl/certs/ca-certificates.crt
```

Server Setup (Optional)

The following steps describe the setup for TFTP, NFS, and Samba servers. Server setup is not required for working with the board, however they will significantly reduce time and are highly recommended during the building and development phase.

TFTP

TFTP is a "trivial" file transfer protocol used to transfer individual files across a network. Setting up a TFTP server on your Linux Host PC will allow you to exchange files with the target board. Some examples where this will be advantageous include:

- Modifying and doing development on the Linux kernel. Barebox can be configured to remotely boot the kernel so you have access to the latest build without needing to continually reflash the target board.
- Updating images from the bootloader. Transferring files over a network in Barebox is an alternative to using an SD card which eliminates some time consuming steps such as formatting an SD card.
- Individual file transfer to the root fileystem. When Linux has been fully booted you may want to copy a specific file from your Host PC to the target board (images, application executables).

Install the TFTP server on your Host PC:

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

Specify a folder where the files will reside on your Host PC 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
```

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>
```

Files in the "<TFTP_DIRECTORY>" on your Host PC can now be accessed from another machine on the same network such as the target board by simply using the IP address of the Host PC. Take note of this IP address, in a typical wired connection this will be "inet addr" listed under "eth0".

```
ifconfig
```

NFS

A network filesystem (NFS) server gives other systems the ability to mount a filesystem stored on the Host PC and exported over the network. Setting up an NFS server on your Linux Host PC gives you access to the target boards root filesystem which will allow you to quickly test applications and evaluate different filesystem setups for the target board. That is, the root filesystem for the board will actually be located on the remote host Linux machine. This enables easy access and modifications to the root filesystem during development.

Install the NFS server on your Host PC:

```
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
```

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
```

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.

Setup the BSP Directory

Create a directory which will house your BSP development. In this example the BSP directory is /opt/PHYTEC_BSPs/. This is not a requirement and if another location is preferred (ex. ~/PHYTEC_BSPs) feel free to modify. We recommend using /opt over your HOME directory to avoid errors attributed to ~ syntax as well as the sudo requirement for the root filesystem and automation package building. We also recommend creating a package download directory (yocto_dl) separate from the yocto tree (yocto_ti), as it makes resetting the build environment easier and subsequent build times much faster.

```
sudo mkdir /opt/PHYTEC_BSPs
cd /opt/


# /opt/ directory has root permission, change the permissions so your user account can access this folder. In
the following replace <user> with your specific username
sudo chown -R <user>: PHYTEC_BSPs


cd PHYTEC_BSPs
mkdir yocto_ti
mkdir yocto_dl
cd yocto_ti
export YOCTO_DIR=`pwd`
```

At this point you will now be able to navigate to the Yocto directory using the *$YOCTO_DIR* environment variable.

Install the Linaro Toolchain

Run the following commands to install the Linaro Toolchain:

```
wget http://releases.linaro.org/components/toolchain/binaries/6.2-2016.11/arm-linux-gnueabihf/gcc-linaro-6.2.1-
2016.11-x86_64_arm-linux-gnueabihf.tar.xz
tar -Jxvf gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf.tar.xz -C /opt/PHYTEC_BSPs
rm gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf.tar.xz
```

Download the BSP Meta Layers

Download the manifest file for the AM57xx PD18.2.0 BSP:

```
cd $YOCTO_DIR
repo init -u https://stash.phytec.com/scm/pub/manifests-phytec.git -b am57xx -m PD18.2.0.xml
```

Download the Yocto meta layers specified in the manifest file:

```
repo sync
```

Start the Build

Replace *MACHINE=am57xx-phycore-rdk* in all of the build instructions with the appropriate MACHINE for the SOM configuration found in the **Yocto Machine Configuration Table** section of the latest AM57xx BSP release notes.

Run the Yocto build directory setup script. The TEMPLATECONF variable is used to set the source of the local configuration files (conf/bblayers.conf and conf/local.conf), which are located in the meta-phytec layer:

```
cd $YOCTO_DIR
TEMPLATECONF=$YOCTO_DIR/sources/meta-phytec/meta-phytec-ti/conf MACHINE=am57xx-phycore-rdk source sources/oe-
core/oe-init-build-env build
```

Open the build/conf/local.conf file using your favorite editor and modify the the download directory to:

```
DL_DIR ?= "/opt/PHYTEC_BSPs/yocto_dl"
```

Maximize build efficiency by modifying the BB_NUMBER_THREADS variable to suit your host development system. This sets the maximum number of tasks that BitBake should run in parallel. Also set the variable PARALLEL_MAKE to specify the number of threads that make can run. By default, these are set to 4 in build/conf/local.conf:

```
# Parallelism options - based on cpu count
BB_NUMBER_THREADS ?= "4"
PARALLEL_MAKE ?= "-j 4"
```

Be sure to save your changes to the local.conf file before closing.

The setup is complete and you now have everything ready to start a build. This BSP has been tested with the *arago-core-tisdk-image* and it is suggested that you start with this image before building other images. Alternate images are located in various meta layers at *yocto_ti/sources/meta*/recipes*/images/*.bb*. They can be found using the command *bitbake-layers show-recipes "*-image*"* in $YOCTO_DIR/build/.

The default build target is *arago-core-tisdk-image*, which includes all TISDK demos and support.

In the interest of creating a smaller image that still supports all libraries and graphics features, but without many of the demos, we recommend using *phytec-tisdk-image*. PHYTEC offers *phytec-tisdk-image* images on **Artifactory** under the *am5716-phycore-rdk*, *am5726-phycore-rdk*, and *am5728-phycore-rdk* directories.

If building for am5726-phycore-rdk or am5716-phycore-rdk, the video and graphics support will be removed from the output regardless of the image target as am57x6 hardware does not support these features.

The following will start a build from scratch including installation of the toolchain as well as bootloader, Linux kernel, and root filesystem images.

```
cd $YOCTO_DIR/build
export PATH=/opt/PHYTEC_BSPs/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf/bin:$PATH
MACHINE=am57xx-phycore-rdk bitbake arago-core-tisdk-image
```

Built Images

If working with a non-kit SOM, select it in the **Yocto Machine Configuration Table** and use the device tree files mentioned in the *Kernel Device Tree File* column.

The device tree filename referenced in this section is *am572x-phycore-rdk-41300111i* as the phyCORE-AM57xx RDK comes standard with a *PCM-057-41300111I* SOM.

All images generated by bitbake are deployed to *$YOCTO_DIR/build/arago-tmp-external-linaro-toolchain/deploy/images/<machine>*:

- **Bootloader**: MLO, u-boot.img
- **Kernel**: zImage
- **Kernel device tree file**: zImage-am572x-phycore-rdk-41300111i.dtb
- **Root Filesystem**: tisdk-rootfs-image-am57xx-phycore-rdk.tar.xz

Source Locations:

- **Kernel:** *$YOCTO_DIR/build/arago-tmp-external-linaro-toolchain/work/<MACHINE>-linux-gnueabi/linux-phytec-ti/4.9.41+git_v4.9.41-phy4-r7a/git/*
    - The device tree file to modify within the linux kernel source is: *am572x-phycore-rdk-41300111i.dts* and its included *dtsi* files.
- **u-boot:** *$YOCTO_DIR/build/arago-tmp-external-linaro-toolchain/work/<MACHINE>-linux-gnueabi/u-boot-phytec/2017.01+git_v2017.01-phy4-r0/git/*

Build Time Optimizations

The build time will vary depending on the package selection and Host performance. Beyond the initial build, after making modifications to the BSP, a full build is not required. Use the following as a reference to take advantage of optimized build options and reduce the build time.

To rebuild U-Boot:

```
bitbake u-boot-phytec -f -c compile && bitbake u-boot-phytec
```

To rebuild the Linux kernel:

```
bitbake linux-phytec-ti -f -c compile && bitbake linux-phytec-ti
```

The Yocto project's Bitbake User Manual provides useful information regarding build options: **http://www.yoctoproject.org/docs/2.2/bitbake-user-manual/bitbake-user-manual.html**

Additional BSP Documentation

Content by label

There is no content with the specified labels