# BSP Yocto FSL i.MX7 PD18.2.x 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 Images from Source section for the applicable differences. For each SOM configuration there is a required change to the build configuration layers file and the yocto MACHINE value.

---

This Quickstart provides you with the tools and know-how to install and work with the Linux Board Support Package (BSP) for the phyBOARD-Zeta. This Quickstart shows you how to do everything from installing the appropriate tools and source, to building custom kernels, to deploying the OS, and exercising the software and hardware. Additionally, gain access to the SOM and baseboard schematics for the phyBOARD-Zeta 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 64-bit recommended. 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 40-50GB 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

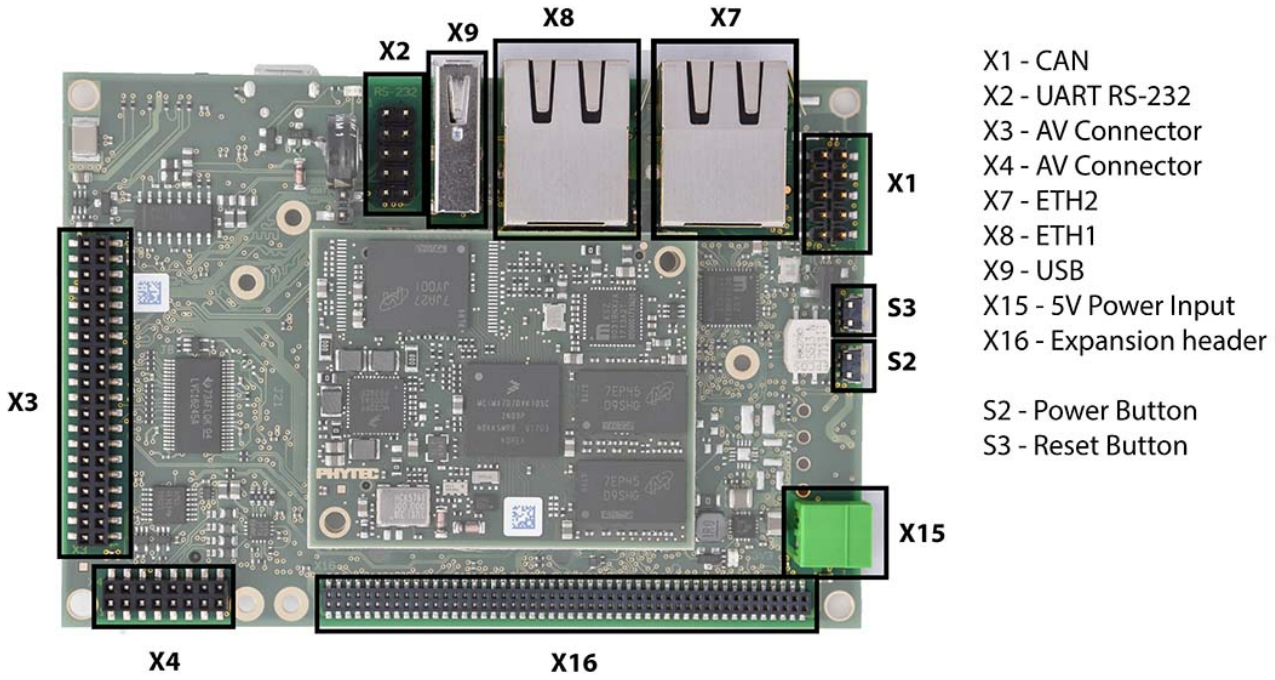
- phyCORE-i.MX7 SOM
- phyBOARD-Zeta i.MX7 carrier board
- 2x 2x5 pin header to DB9 male connector
- Expansion board PEB-D-RPI (optional)
- LCD Display Adapter with 7" capacitive display PEB-AV-02-TC (optional)
- WiFi/BT module PEB-WLBT-03 (optional)
- Null-Modem RS232 DB9 cable (female to female)
- AC adapter with 2-pin phoenix connector supplying 5V DC, min. 2.5A

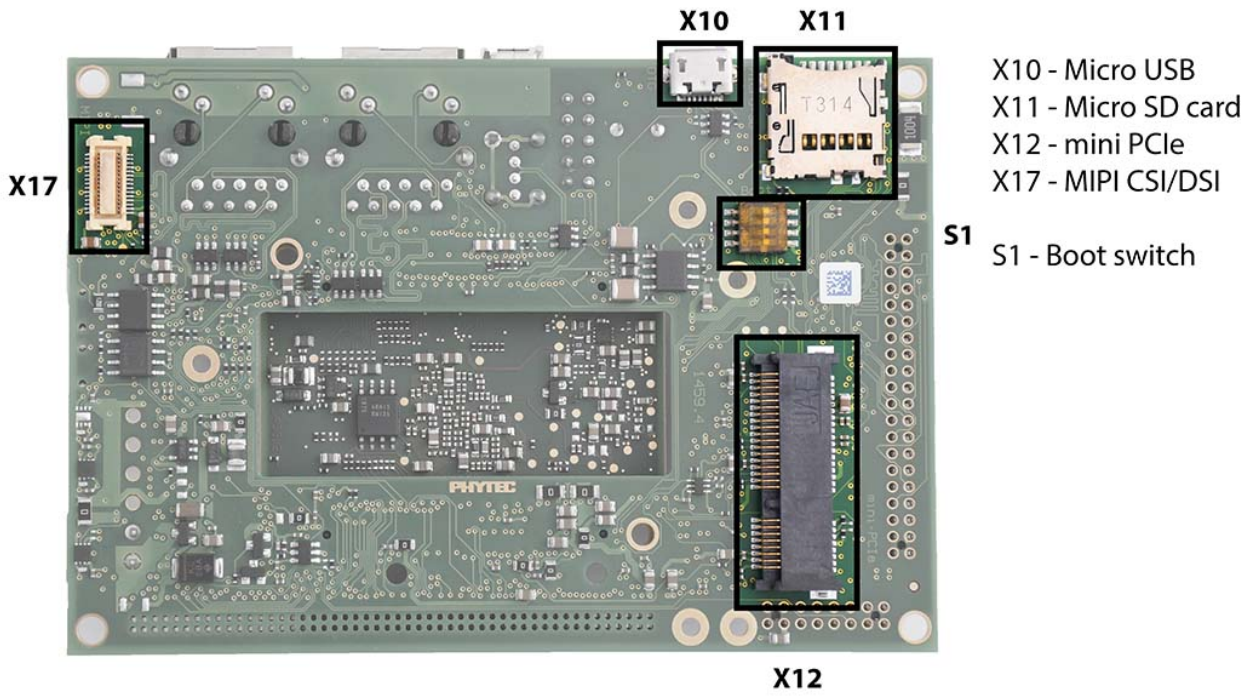
- microSD Card for flashing and booting

> ⚠ See release notes for supported SOM and carrier board versions.

## Connector Interfaces

Use the following as a reference for the connector interfaces on the phyBOARD-i.MX7 that will be used in this Quickstart.

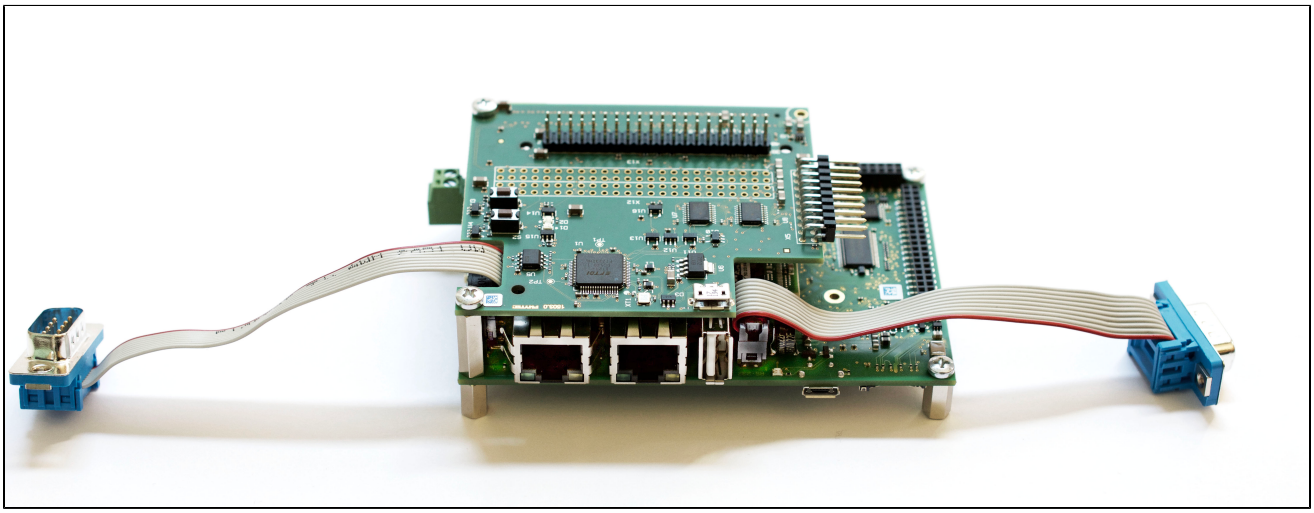





### I.MX7 Kit Connections

Use the following images as a reference when connecting:

- UART and CAN 2x5 pin headers (X1 and X2) to DB9 adapters
- PEB-D-RPI Expansion Board to X16 expansion header on the phyBOARD-Zeta



## 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



See Release Notes for information regarding supported Yocto machine configurations, and replace <Yocto Machine> in the steps below with the appropriate machine for your hardware.

1. Download the "fsl-image-validation-imx-<Yocto Machine>.sdcard" image which will be used to format a bootable SD card. The binary images for BSP-Yocto-FSL-iMX7-PD18.2.x are available on PHYTEC's Artifactory. Images are organized by machine name.
2. Flash the SD card image to a micro SD card:

   ```
   sudo dd if=fsl-image-validation-imx-<Yocto Machine>.sdcard of=/dev/sd<SD partition> bs=1M && sync
   ```

   For more information on formatting an SD card, see Creating a Bootable SD Card.

3. If using the PEB-D-RPI expansion board, plug it into the expansion connector X16 on the carrier board.
4. If using the WiFi/BT module PEB-WLBT-03, plug it into the expansion connector header.
5. If using the LCD Display, connect the module PEB-AV-02-TC to the AV connectors X3 and X4.
6. Plug micro SD card into slot on underside of board.
7. Connect UART cable to the 5x2 pin header labelled "RS-232". This header requires an adapter as well as Null modem cable. When plugged in, the adapter cable should be oriented towards the USB and ethernet interfaces.
8. 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
9. Connect 5V power supply to the 2-pin phoenix connector. Looking into the connector, the pin on the left is positive and the one next to it is negative.
10. Press the "PWR" button (this may not be necessary if the battery has completely discharged). You will now start to see console output on your terminal window. If everything was done correctly the board should boot completely into Linux, arriving at a login prompt:

    *NXP i.MX Release Distro 4.9.11-1.0.0 <Yocto Machine> ttymxc4*

    *<Yocto Machine> login: root*

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



**Troubleshooting**

Not seeing output on the console?

- **Make sure to press the power button on the carrier board**. Unlike some other PHYTEC boards, the phyBOARD-i.MX7 does not get powered on simply by plugging in the power supply (this may not be necessary if the battery has completely discharged)
- Check that you have setup the terminal software correctly per step 8 .
- Create a Bootable SD Card with the release images from the PHYTEC Artifactory, then configure the board to boot from SD/MMC (Boot Configurations). After booting, you can restore your eMMC contents by following the Flashing Images to eMMC section.

# Building Images from Source

This section explains how to build your BSP from source.

## 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 gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat
libsdl1.2-dev
```

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

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?"
```

## 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
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"
```

## 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.

# Yocto Build Steps

## BSP Directory Setup

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_imx7), 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_imx7
mkdir yocto_dl
cd yocto_imx7
export YOCTO_DIR=`pwd`
```

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

## Download the BSP Meta Layers

Download the manifest file for BSP-Yocto-FSL-iMX7-PD18.2.x:

```
cd $YOCTO_DIR
repo init -u https://stash.phytec.com/scm/pub/manifests-phytec.git -b imx7 -m BSP-Yocto-FSL-iMX7-PD18.2.0.xml
```

Download the Yocto meta layers specified in the manifest file:

```
repo sync
```

## Start the Build

Run the Yocto build directory setup. 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-fsl/conf source sources/poky/oe-init-build-env build
```

Modify the DL_DIR variable in build/conf/local.conf to use your new download directory:

```
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"
```

**Accept Freescale EULA**

In order to build certain packages included in the FSL meta layers, you need to accept the NXP EULA at '$YOCTO_DIR/sources/meta-fsl-bsp-release/imx/EULA.txt'. Please read it and if you accept it, add the following in build/conf/local.conf:

```
ACCEPT_FSL_EULA = "1"
```

**If working with a non-kit SOM, please expand the content below for additional instructions.**

Non-Standard Kit SOM machine configurations are included in the Yocto layer meta-phytec-extra. To include this meta layer in the build, add it to the build/conf/bblayers.conf file:

```
        ${BSPDIR}/sources/meta-phytec/meta-phytec-fsl \
+       ${BSPDIR}/sources/meta-phytec-extra \
  "
```

Not sure whether your SOM is part of a standard kit? See Release Notes Yocto Machine Config table for kit part numbers.

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

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

See Release Notes for supported Yocto machine configurations, and replace <Yocto Machine> below with the appropriate machine for your hardware.

```
cd $YOCTO_DIR/build
MACHINE=<Yocto Machine> bitbake fsl-image-validation-imx
```

## Built Images

All images generated by bitbake are deployed to *$YOCTO_DIR/build/tmp/deploy/images/<Yocto Machine>*:

- **SD Image**: fsl-image-validation-imx-<Yocto Machine>.sdcard
- **Bootloader**: u-boot.imx
- **Kernel**: zImage
  **Kernel device tree file**: zImage-<Yocto Machine>.dtb
  - device tree used when FreeRTOS is running on M4: zImage-<Yocto Machine>-m4.dtb
- **Root Filesystem**: fsl-image-validation-imx-<Yocto Machine>.ext4

### Source Locations:

Note that v4.9.11-phy<X> and v2017.03-phy<X> within the paths pertain to the release tag for the linux and u-boot sources. This matches the value of RELEASE_VER for the corresponding recipes in meta-phytec-fsl. For example, for release PD18.2.0 the kernel tag is v4.9.11-phy3.

**Kernel**: *$YOCTO_DIR/build/tmp/work/<Yocto Machine>-poky-linux-gnueabi/linux-phytec-fsl/4.9.11+git_v4.9.11-phy<X>-r0/git/*

- The device tree file to modify within the linux kernel source is: arch/arm/boot/dts/<Yocto Machine>.dts and its dependencies

**U-Boot:** *$YOCTO_DIR/build/tmp/work/<Yocto Machine>-poky-linux-gnueabi/u-boot-phytec/2017.03+git_v2017.03-phy<X>-r0/ git/*

- Board file is located at: *board/phytec/mx7d_phyboard_zeta/mx7d_phyboard_zeta.c*
- Device tree file is located at: *arch/arm/dts/imx7d-phyboard-zeta-001.dts*

**Toolchain:** *$YOCTO_DIR/build/tmp/sysroots/x86_64-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-*

## 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:

```
MACHINE=<Yocto Machine> bitbake u-boot-phytec -f -c compile && bitbake u-boot-phytec
```

To rebuild the Linux kernel:

```
MACHINE=<Yocto Machine> bitbake linux-phytec-fsl -f -c compile && bitbake linux-phytec-fsl
```

The Yocto project's Bitbake User Manual provides useful information regarding build options.

## BSP Documentation

**Content by label**

There is no content with the specified labels

**Content by label**

There is no content with the specified labels

**Content by label**

There is no content with the specified labels