
# User Guide

# Preparing Custom Linux Boot Files with Xilinx PetaLinux Tools

Revision 1.2

2019-11-15

Paweł Zapalski

Level: Intermediate



## Abstract

Many embedded hardware platforms do not require a specialized GUI-based Linux environment, moreover, in some cases small-sized embedded filesystem is even more preferable. For this purpose, Xilinx PetaLinux tools can be used to generate a custom filesystem image with additional built-in libraries, applications and system tools.

**Meta Keywords** Zynq, Embedded, Linux, PetaLinux

**Related Products** Riviera-PRO, TySOM

**Related Methodologies** ASIC Prototyping, C Synthesis, Co-Simulation

**Related Markets** Embedded, Automotive, IoT, UAV, High Performance Computing

# Table of Contents

Preparing Custom Linux Boot Files with Xilinx PetaLinux Tools....1
Table of Contents....2
Table of Figures....2
Requirements....3
Creating a PetaLinux Project....4
Creating a PetaLinux Project from a Template....4
Creating a PetaLinux Project from a BSP....6
Root Filesystem Customization....7
Configuring Standard Filesystem Packages....7
External Packages....8
Building a RootFS Image....9
Linux Kernel Customization and Building....10
U-Boot Customization and Building....11
Device Tree Binary Building....12
Building All Components....13
Generating BOOT.BIN....14
Updating the Hardware Platform....15
About Aldec, Inc....16

# Table of Figures

Figure 1: Configuring System Environment....3
Figure 2: PetaLinux Main Configuration Menu....5
Figure 3: Creating a PetaLinux Project with Using BSP....6
Figure 4: PetaLinux Root Filesystem Configuration Menu....7
Figure 5: Adding External Applications....8
Figure 6: Linux Kernel Configuration Menu....10
Figure 7: U-Boot Configuration Menu....11
Figure 8: Complete PetaLinux Project Building....13
Figure 9: Generating BOOT.BIN....14

# Requirements

An embedded Linux boot files preparing approach mentioned in a current user guide assumes a user already has a hardware project implemented in *Xilinx Vivado* and *Xilinx PetaLinux* tools pre-installed in a Linux compliant environment. The release version of software tools is 2018.3. Note, if a newer version of PetaLinux tools is used, the instructions may be slightly different.

References:

- *PetaLinux Tools Command Line Guide*
- *PetaLinux Tools Reference Guide*

Requirements:

- installed Xilinx Petalinux 2018.3 Tools
- installed Xilinx Vivado 2018.3 Tools
- pre-built Vivado project or BSP for a hardware platform.

Before running the PetaLinux tools, appropriate paths have to be added to the system environment. PetaLinux installation directory contains *settings.sh* shell script used for this purpose:

- Go to the PetaLinux installation directory;
- `source ./settings.sh`
- Check if *petalinux*- tools are available in system environment as shown in Figure 1.

```
[pawzap@gd18 aldec]$ source /home/aldec/Petalinux/settings.sh
PetaLinux environment set to '/home/aldec/Petalinux/'
WARNING: /bin/sh is not bash!
bash is PetaLinux recommended shell. Please set your default shell to bash.
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
WARNING: No tftp server found - please refer to "PetaLinux SDK Installation Guide" for its im
pact and solution
[pawzap@gd18 aldec]$ petalinux-
petalinux-boot     petalinux-config   petalinux-package
petalinux-build    petalinux-create   petalinux-util
[pawzap@gd18 aldec]$ petalinux-
```

**Figure 1: Configuring System Environment**

# Creating a PetaLinux Project

A PetaLinux project can be created in two ways:

- with using a project template,
- with using a BSP.

Follow steps in subchapter "*Creating a PetaLinux Project from a Template*" or "*Creating a PetaLinux Project from a BSP*" to create a PetaLinux project.

## Creating a PetaLinux Project from a Template

In order to create a PetaLinux project from a template, first export the hardware design following the next steps:

- Run Vivado and open a hardware project;
- *File* → *Export* → *Export Hardware* → select "Include bitstream" option → Accept.

The Hardware Description File (.hdf) will be generated under *<project_name>.sdk* directory. This file contains all necessary information required to create a PetaLinux project.

PetaLinux can create a project for two architectures: Zynq-7000 series and Zynq Ultrascale+ MPSoC. Set a template parameter value to zynq or zynqMP for chosen architecture.

Next go to the directory with exported hardware description file and follow the next steps in order to create a new PetaLinux project:

```
petalinux-create -t project -n <project_name> --template zynqMP
cd  project_name
petalinux-config --get-hw-description=..
```

PetaLinux system configuration menu should appear. Apply the following settings under the **Image Packaging Configuration** submenu:

- Root filesystem type (INITRD)
- RAMDISK loadaddr (0x10000000)
- Copy final images to tftpboot [ ] - unmarked

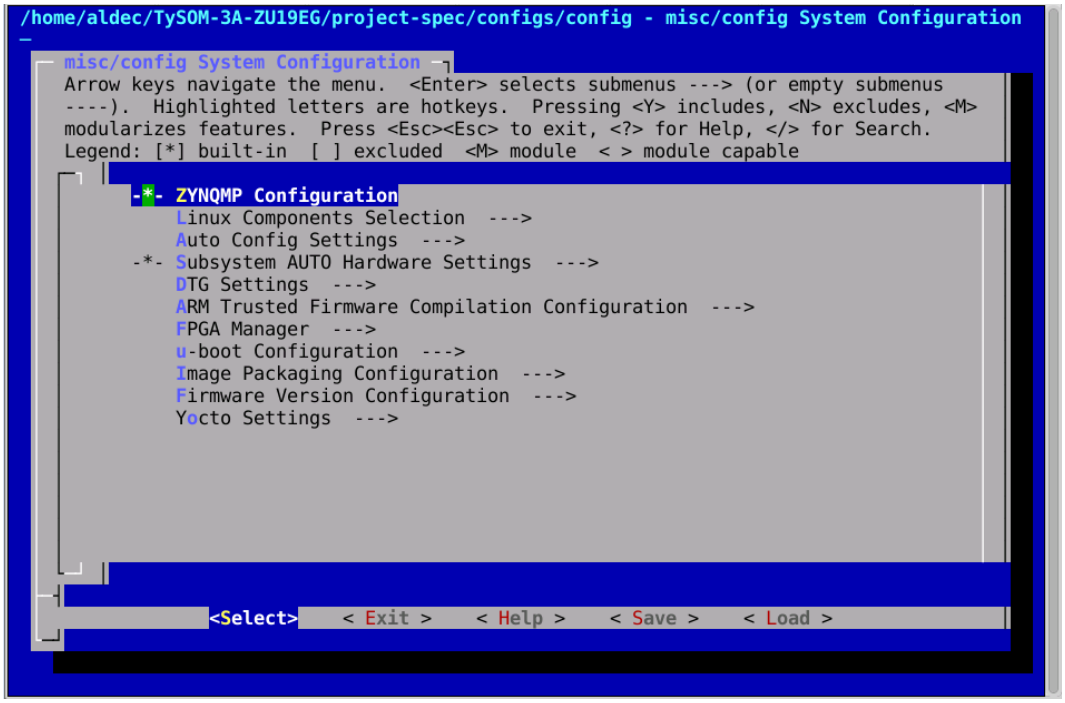


**Figure 2: PetaLinux Main Configuration Menu**

Save the current configuration and exit the menu, wait for finishing the project configuration process. PetaLinux will configure settings for a device tree structure, kernel, root filesystem (rootfs) and U-Boot bootloader which are the main files for an embedded Linux setup.

## Creating a PetaLinux Project from a BSP

A BSP (Board Support Package) is a package which contains a configuration for a chosen hardware platform. BSPs are useful to save kernel and root filesystem configurations for PetaLinux. It also contains pre-built images for booting a board.

Create a PetaLinux project from a BSP with using the command:

```
petalinux-create -t project -n <project_name> -s <bsp_name>
```

The created project is initially configured and is ready to build. Go to the project directory, make own customization and build.

```
[pawzap@gd18 aldec]$ ls
TySOM-3A-ZU19EG.bsp
[pawzap@gd18 aldec]$ petalinux-
petalinux-boot     petalinux-config   petalinux-package
petalinux-build    petalinux-create   petalinux-util
[pawzap@gd18 aldec]$ petalinux-create -t project -s ./TySOM-3A-ZU19EG.bsp
INFO: Create project:
INFO: Projects:
INFO:   * TySOM-3A-ZU19EG
INFO: has been successfully installed to /home/aldec/
INFO: New project successfully created in /home/aldec/
[pawzap@gd18 aldec]$ cd TySOM-3A-ZU19EG
[pawzap@gd18 TySOM-3A-ZU19EG]$ ls
components  config.project  pre-built  project-spec
[pawzap@gd18 TySOM-3A-ZU19EG]$
```

Figure 3: Creating a PetaLinux Project with Using BSP

# Root Filesystem Customization

## Configuring Standard Filesystem Packages

PetaLinux SDK contains a lot of pre-defined Linux packages to be built-in to the root filesystem image available under **Filesystem Packages** submenu. The root filesystem could be customized with using the command:

```
petalinux-config -c rootfs
```

In PetaLinux some packages are organized in groups. Select packages and libraries and save a configuration. Some of them are highly useful for testing and debug purposes:

- **Filesystem Packages → base → i2c-tools → i2c-tools**: i2cset/i2cget tools used to initiate data transactions on i2c bus from userspace;
- **Filesystem Packages → base → tcf-agent → tcf-agent**: Target Communication Framework client used for cross-debugging user applications running on target board;
- **Filesystem Packages → base → usbutils → usbutils**: adds lsusb tool used for discovering devices attached to USB bus.
- **Filesystem Packages → misc → coreutils → coreutils**: many basic Linux utilities
- **Filesystem Packages → base → busybox → busybox-udhcpc**: network DHCP client

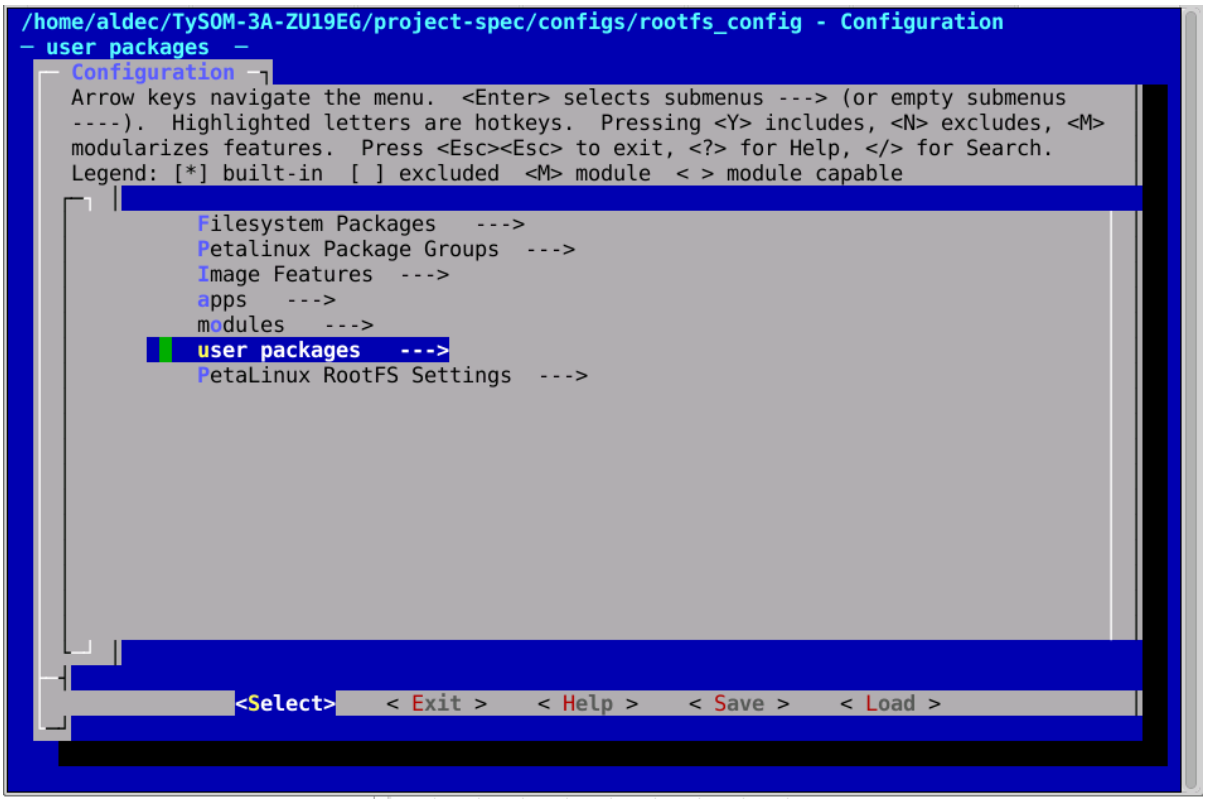


**Figure 4: PetaLinux Root Filesystem Configuration Menu**

The root filesystem image comes with only root user enabled. The password for root user can be changed by *Petalinux RootFS Settings/Root* password option. Save changes and exit from RootFS configuration menu.

## External Packages

An additional user or third-party libraries and applications can be easily integrated with the Linux filesystem image. Refer to the Xilinx PetaLinux documentation chapter 7 - "Customizing the Rootfs" (*https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug1144-petalinux-tools-reference-guide.pdf*) to check how to prepare external libraries or applications for an integration with the PetaLinux flow.

A user can include prebuilt libraries, applications, modules. It is also possible to add a custom application. Use the application template to create a recipe for an application. Modify the recipe and add own source files to the PetaLinux.

In PetaLinux some packages are not visible in a configuration menu (for example iw – a tool for managing wireless interface).

Make it visible by adding line in <plnx-proj-root>/project-spec/meta-user/recipes-core/images/petalinux-image-full.bbappend file. Insert line with text:

```
IMAGE_INSTALL_append = " iw"
```

After that the package will be visible in "user packages" category in the root filesystem configuration menu.

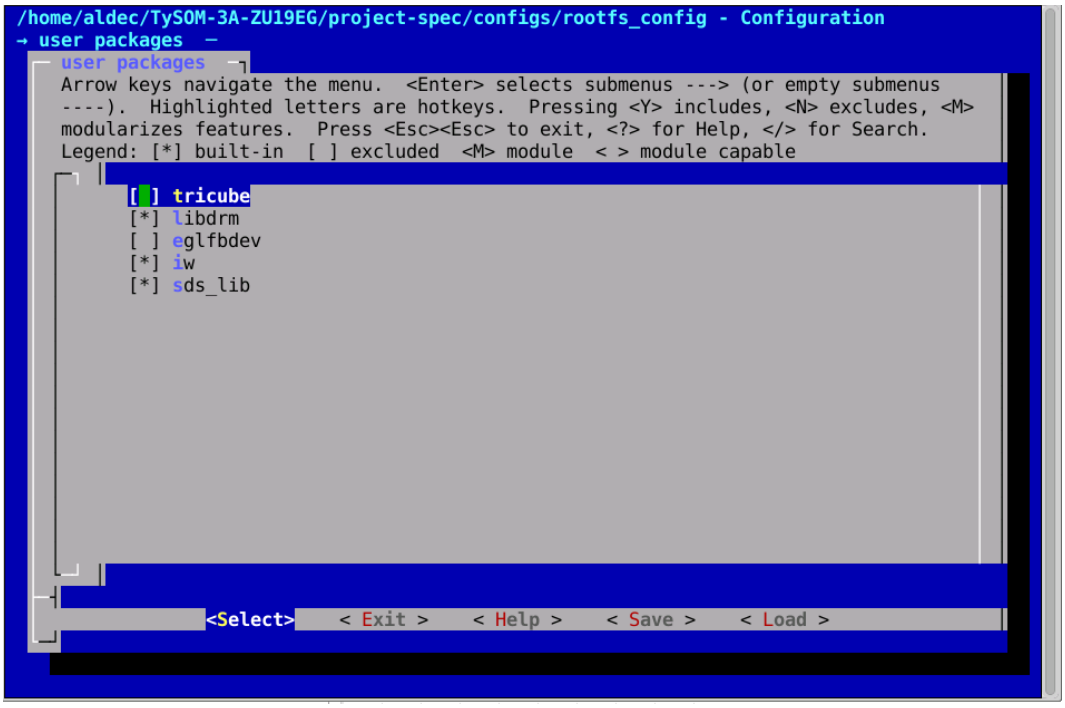


**Figure 5: Adding External Applications**

Run the root filesystem configuration menu in order to add external components to the build:

- *petalinux-config -c rootfs*
- select added components under **Apps** submenu as shown in Figure 5;

## Building a RootFS Image

Use the following command to clean the whole RootFS build when necessary:

```
petalinux-build -c rootfs -x distclean
```

After the root filesystem configuration is done follow the steps below in order to get the root filesystem image:

- build the target filesystem binaries by running:

```
petalinux-build -c rootfs
```

- the packaged and compressed image wrapped for u-boot (**rootfs.cpio.gz.u-boot**) can be found at *<project_name>/images/linux* directory;
- go to the directory mentioned above and rename the final image: *mv rootfs.cpio.gz.u-boot uramdisk.image.gz*

The result root filesystem image (**uramdisk.image.gz**) can be used along with the rest parts of embedded Linux setup now.

## Linux Kernel Customization and Building

PetaLinux 2018.3 uses a Linux kernel tree based on version 4.14. The Linux kernel configuration can be done with using the command:

```
petalinux-config -c kernel
```

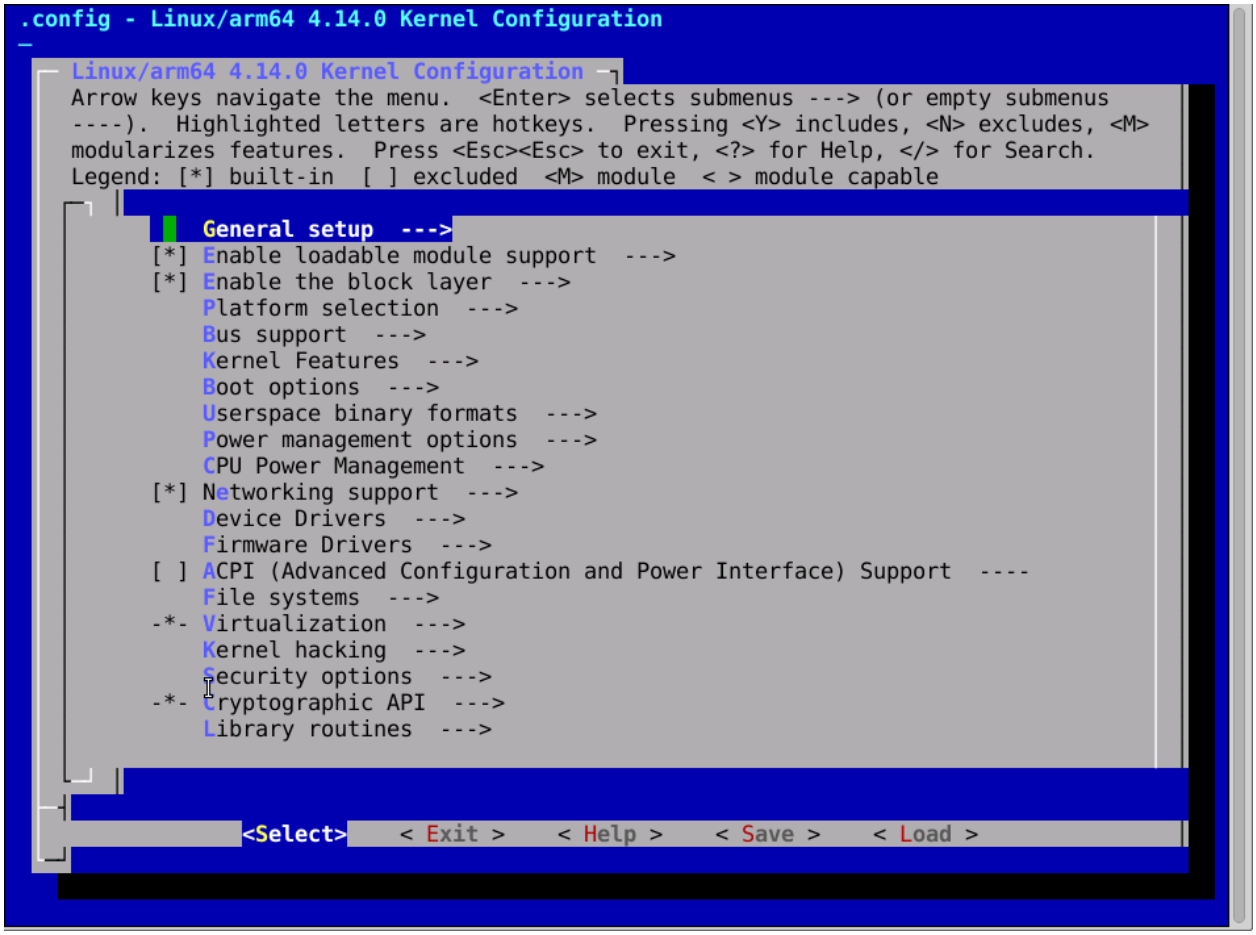


Figure 6: Linux Kernel Configuration Menu

PetaLinux allows to apply patches. They should be placed in the directory <plnx_project_root>/project-spec/meta-user/recipes-kernel/linux/linux-xlnx. Patches names must contain extension *.patch or *.diff. Every patch name must be inserted to file <plnx_project_root>/project-spec/meta-user/recipes-kernel/linux/linux-xlnx_%.bbappend according to scheme:

```
SRC_URI_append = " \
file://<patch_name_1> \
file://<patch_name_2> \
(…) "
```

Configure the kernel according to the hardware platform and save. Build only the kernel image with using the command:

```
petalinux-build -c kernel
```

## U-Boot Customization and Building

Configure an U-Boot with using the command:

```
petalinux-config -c u-boot
```

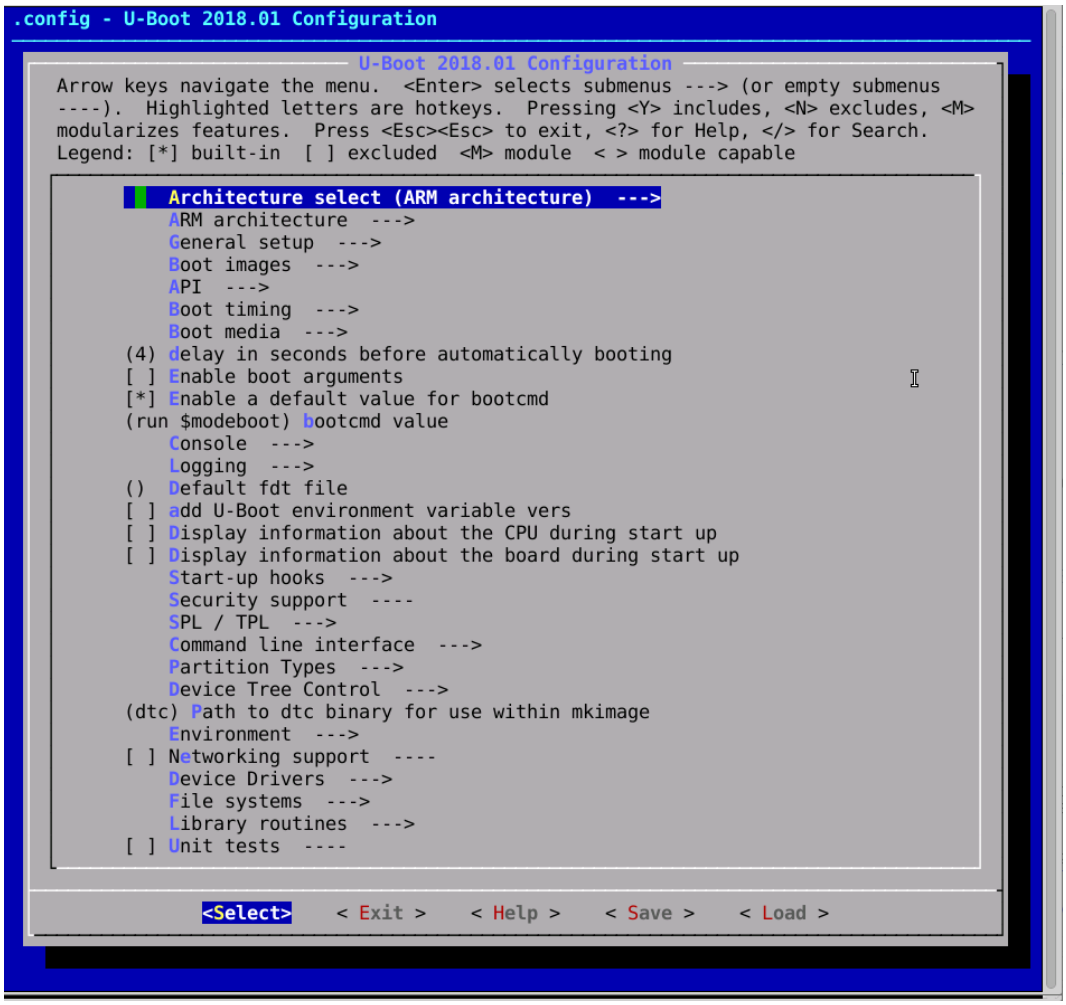


**Figure 7: U-Boot Configuration Menu**

Make some customizations and save. When using a PetaLinux autoconfiguration some modifications can be needed in autogenerated file. Verify the file *<plnx_proj_root>/project-spec/meta-user/recipes-bsp/u-boot/platform-top.h*. Build with using the command:

```
petalinux-build -c u-boot
```

or without checking dependencies (faster):

```
petalinux-config -b u-boot-xlnx_2018.3
```

## Device Tree Binary Building

The PetaLinux tools can autogenerate a skeleton of a device tree structure. A user should verify generated configuration, correct it and add missing nodes. PetaLinux generates files to *<plnx_project_root>/components/device-tree/device-tree/* directory. By default a user should add own modifications in a file *<plnx_project_root>/project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi*. The file is included to autogenerated files and overrides existing nodes.

Generate the structure and build device tree with dependencies:

```
petalinux-build -c device-tree
```

or build without checking project dependencies (faster) when project is already built with using the command:

```
petalinux-build -b device-tree
```

Alternatively if a user disables the device tree autoconfiguration the structure can be placed in directory *<plnx_project_root>/components/device-tree/device-tree/*. Create the directory if it does not exist. The solution is not recommended, but sometimes PetaLinux device tree generator crashes and it is not possible to finish project building. Solve the problem with disabling “Device tree autoconfig” and place own files.

Copy device tree structure into the directory and modify it according to hardware platform.

**Attention!**

**Command:**

```
petalinux-config -x mrproper
```

**removes <plnx_project_root>/components directory with yhe whole internal content. If the device tree structure is modified in the directory then create the structure backup before the project cleaning.**

## Building All Components

Instead of separately using commands for each component PetaLinux can build the entire project with only one command:

```
petalinux-build
```

It builds Linux kernel, root filesystem, FSBL, PMU firmware, ATF, device tree binary and U-Boot. After successful building all components are placed in *<plnx_proj_root>/images/linux/*.

A successfully finished PetaLinux project building is presented in figure 8.

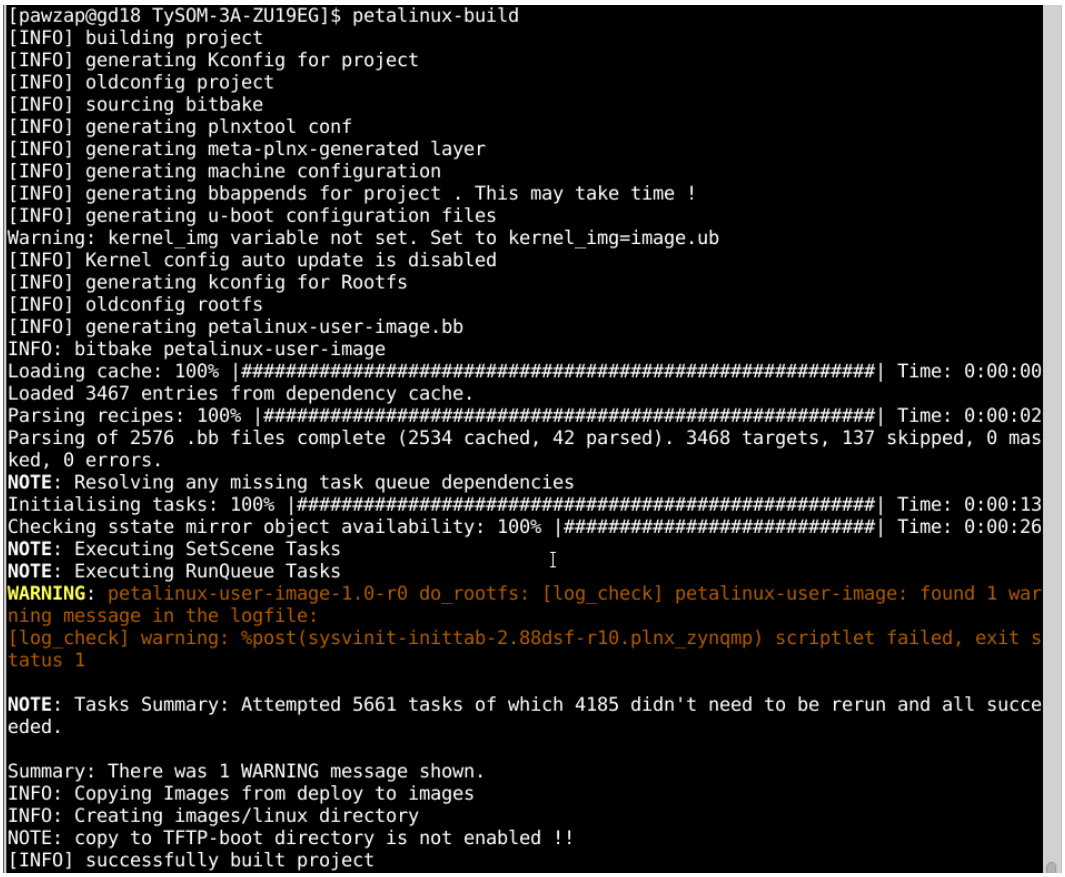


**Figure 8: Complete PetaLinux Project Building**

## Generating BOOT.BIN

Generate BOOT.bin file with using petalinux-package tool. Use some parameters to package all mandatory ingredients for an architecture. A default directory for all files is *<plnx_proj_root>/images/linux/.* Run the command:

- for ZynqMP devices

```
petalinux-package --boot --force --fsbl --pmufw --fpga --atf --u-boot
```

- for Zynq-7000 devices

```
petalinux-package --boot --force --fsbl --fpga --u-boot
```

```
[pawzap@gd18 TySOM-3A-ZU19EG]$ petalinux-package --boot --force --fsbl --pmufw --fpga --atf -
-u-boot
INFO: File in BOOT BIN: "/home/aldec/TySOM-3A-ZU19EG/images/linux/zynqmp_fsbl.elf"
INFO: File in BOOT BIN: "/home/aldec/TySOM-3A-ZU19EG/images/linux/pmufw.elf"
INFO: File in BOOT BIN: "/home/aldec/TySOM-3A-ZU19EG/project-spec/hw-description/design_1_wra
pper.bit"
INFO: File in BOOT BIN: "/home/aldec/TySOM-3A-ZU19EG/images/linux/bl31.elf"
INFO: File in BOOT BIN: "/home/aldec/TySOM-3A-ZU19EG/images/linux/u-boot.elf"
INFO: Generating ZynqMP binary package BOOT.BIN...


****** Xilinx Bootgen v2018.3
  **** Build date : Nov 15 2018-19:22:29
    ** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

INFO: Binary is ready.
[pawzap@gd18 TySOM-3A-ZU19EG]$
```

**Figure 9: Generating BOOT.BIN**

A boot partition should contain a set of files. It consists of:

- BOOT.BIN
- Image
- uramdisk.image.gz
- devicetree.dtb
- uEnv.txt

Copy files to micro SD and boot a TySOM board.

A BSP contains pre-built set of files in *<bsp_root>/pre-built/linux/images/* directory.

## Updating the Hardware Platform

When the PetaLinux project is created it is possible to update the hardware platform and rebuild the project. Follow below steps:

- Clean the whole project to its initial state

```
petalinux-build -x mrproper
```

- Copy a new HDF to the PetaLinux project directory
- Load the new HDF to the project and configure autogeneration options in menu

```
petalinux-config --get-hw-description=./
```

- Save and exit configuration menu. Build the project

```
petalinux-build
```

If the new hardware platform is completely different than previous one then it may be needed to update some settings for U-Boot, device tree etc.

## About Aldec, Inc.

Established in 1984, Aldec Inc. is an industry leader in Electronic Design Verification and offers a patented technology suite including: RTL Design, RTL Simulators, Hardware-Assisted Verification, Design Rule Checking, IP Cores, DO-254 Functional Verification and Military/Aerospace solutions. Continuous innovation, superior product quality and total commitment to customer service comprise the foundation of Aldec's corporate mission. For more information, visit *www.aldec.com*.