PetaLinux SDK User Guide

Board Bringup Guide

UG980 (v2013.10) November 25, 2013

XILINX®

Revision History

Date	Version	Notes
2012-08-03	3.1	Updated for PetaLinux SDK 3.1 release
2012-09-03	12.9	Updated for PetaLinux SDK 12.9 release
2012-12-17	2012.12	Updated for PetaLinux SDK 2012.12 release
2013-04-29	2013.04	Updated for PetaLinux SDK 2013.04 release
2013-11-25	2013.10	Updated for PetaLinux SDK 2013.10 release

Online Updates

Please refer to the PetaLinux v2013.10 Master Answer Record (Xilinx Answer Record #55776) for the latest updates on PetaLinux SDK usage and documentation.

XILINX

Table of Contents

About this Guide

One of the great strengths of an FPGA platform is the ability to completely customise your processor system architecture, either for an off-the-shelf evaluation board, or for your own custom designs.

PetaLinux SDK was created to embrace that flexibility, and includes a set of tools specifically designed to make it as easy as possible to boot a Zynq or MicroBlaze Linux platform on a new board or CPU subsystem design.

This document assumes a working knowledge of the Xilinx embedded development tools. It also assumes that you are familiar with the basics of working with PetaLinux SDK.

It is strongly recommended that you first become familiar with PetaLinux SDK by using the provided pre-built BSPs and reference designs, before attempting a custom board bringup.

Overview

Broadly, there are three stages to the board bringup process:

1. Create and/or configure a hardware project ready for PetaLinux.
2. Create a new PetaLinux SDK project.
3. Propagate the hardware platform configuration settings into the new software platform and complete any further software platform configuration steps.

Hardware Platform

Hardware platforms can be created from a number of sources, such as an existing Vivado/XPS project. Regardless of how the hardware system is created and configured, there is a small number of hardware IP and software platform configuration changes required to make the hardware system Linux ready. These are described below.

Configure a Hardware Platform for Linux

Zynq

The following is a list of requirements for a Zynq hardware project to boot Linux:

1. One Triple Timer Counter (TTC) (Required)

IMPORTANT:

- *If there are multiple TTCs are enabled, the Zynq Linux kernel uses the first TTC block from the device tree.*
- *Please make sure the TTC is not used by others.*

2. External memory controller with at least 32MB of memory (Required)
3. UART for serial console (Required)

IMPORTANT: *If soft IP is used, ensure the interrupt signal is connected*

4. Non-volatile memory (Optional) e.g. QSPI Flash, SD/MMC
5. Ethernet (Optional, essential for network access)

IMPORTANT: *If soft IP is used, ensure the interrupt signal is connected*

Jump ahead to Zynq workflow section on Page 8 for detailed steps to create a bootloader and XSDK PetaLinux BSP for the platform.

MicroBlaze AXI

The following is a list of requirements for a MicroBlaze hardware project to boot Linux:

1. IP core check list:
   - External memory controller with at least 32MB of memory (Required)
   - Dual channel timer with interrupt connected (Required)
   - UART with interrupt connected for serial console (Required)
   - Non-volatile memory such as Linear Flash or SPI Flash (Optional)
   - Ethernet with interrupt connected (Optional, but required for network access)
2. MicroBlaze CPU configuration:
   - MicroBlaze with MMU support by selecting either **Linux with MMU** or **Low-end Linux with MMU** configuration template in the MicroBlaze configuration wizard.

IMPORTANT: *Do not disable any instruction set related options that are enabled by the template, unless you understand the implications of such a change.*

   - The MicroBlaze initial bootloader, called FS-BOOT, has a minimum BRAM requirement. 4KByte is required for Parallel flash and 8KByte for SPI flash when the system boots from Non-volatile memory.

Jump ahead to MicroBlaze workflow section in Page 9 for detailed steps to create a bootloader and XSDK PetaLinux BSP for the platform.

Configuring Software Settings and Bootloader

After designing the hardware system, it is necessary to configure a Linux BSP for that platform. This is required to automate the Linux board bringup.

Zynq

1. Export hardware design to SDK and launch XSDK.
2. Add the PetaLinux SDK repository to the list of available BSPs in XSDK as follows:
   (a) Click **Xilinx Tools** menu tab and select **Repositories**
   (b) Select **Repositories** on the left menu under Xilinx SDK
   (c) Click **New** (for local repositories box) on the right hand top corner.
   (d) Browse to PetaLinux "edk_user_repository" directory - "<petalinux-path>/components/hardware/edk_user_repository" and click **Ok**
   (e) Click **Ok** to finish adding local repository.
3. Create Zynq FSBL project with XSDK standalone BSP as follows:
   (a) Click **File** menu tab select **New** and select **Project**
   (b) Select **Xilinx Application Project** wizard to create a Zynq FSBL project with XSDK standalone BSP.
   (c) Specify **Project name**, e.g. **fs-boot_0**
   (d) Click **Next**.
   (e) Ensure the "Zynq FSBL" template is selection and click **Finish**.
4. Create XSDK PetaLinux BSP and Configure XSDK PetaLinux BSP for U-boot and Linux operation system as follows:
   (a) Click **File** menu tab select **New** and select **Project**
   (b) Select **Xilinx Board Support Package** wizard
   (c) Select **petalinux** as the **Board Support Package OS**.
   (d) Ensure the **Hardware Platform** is set to the hardware platform exported by Vivado/XPS.
   (e) Follow the wizard to complete creating the XSDK PetaLinux BSP project
   (f) When you click **Finish** in the **Xilinx Board Support Package** wizard, a Board Support Package Settings window will pop up. If it did not pop up, follows this
      i. Right click on the "petalinux_bsp_0" project in Project explorer
      ii. Select **Board Support Package Settings**.
   (g) Select **petalinux** in the Board Support Package Settings window.
      i. Set stdout and stdin to ps7_uart_1 or ps7_uart_0 (depending on your configuration).
      ii. Set main_memory to ps7_ddr_0 or other external memory controller ip core (depending on your configuration).

      Additionally configure if available:
      i. flash_memory
      ii. sdio
      iii. ethernet
   (h) Click **Ok** to accept the settings

MicroBlaze AXI

1. Export hardware design to SDK and launch XSDK.
2. Add the PetaLinux repository to XSDK to the list of available BSPs as follows:
   (a) Click **Xilinx Tools** menu tab select **Repository**
   (b) Select **Repositories** on the left menu
   (c) Click **New** (for local repositories box) on the right hand top corner.
   (d) Browse to PetaLinux "`edk_user_repository`" directory - "`<petalinux-path>/components/hardware/edk_user_repository`" and click **Ok**
   (e) Click **Ok** to finish adding local repository.
3. Create fs-boot project and XSDK PetaLinux BSP as follows:
   (a) Click **File** menu tab select **New** and select **Project**
   (b) Select **Xilinx Application Project** wizard
   (c) Follow the wizard to create FS-boot project with XSDK PetaLinux BSP
      i. Specify **Project name**, e.g. **fs-boot_0**
      ii. Ensure the **Hardware Platform** is set to the hardware platform exported by Vivado/XPS.
      iii. Select **petalinux** as **OS Platform**
      iv. Select **Create New** Board Support Package
      v. Click **Next** to select application template
      vi. Select **FS-BOOT** from **Available Templates**
      vii. Click **Finish** to generate the FS-BOOT and XSDK PetaLinux BSP
4. Configure XSDK PetaLinux BSP for U-boot and Linux operating system as follows:
   (a) Right click on the XSDK PetaLinux BSP project created from the previous step project in Project explorer. e.g. **fs-boot_0_bsp**
   (b) Select **Board Support Package Settings**.
   (c) Select **petalinux** in the Board Support Package Settings window
   (d) These settings are automatically selected, please review to ensure they are correct.
   (e) Click **Ok** to accept the settings

WARNING: *At this point, there may be an error message about fs-boot compilation error due to memory overlap for MicroBlaze system. This is normal and it will be addressed in the next step.*

5. Configure fs-boot settings. The fs-boot bootloader is used on system boot to move the U-Boot image from the U-boot partition in Flash to main memory and then run U-Boot from main memory. To do this, the start address of U-boot partition in Flash memory must be set:
   (a) Right click on the **fs-boot** project in Project explorer and select **C/C++ Build settings**
   (b) Select the **C/C++ Build** configuration and select **Settings**

(c) Select on the **Tool Settings** tab

(d) Set configuration profile to **[All configurations]**

(e) Select Symbols

- For MicroBlaze target, click **Symbols** under MicroBlaze gcc compiler

(f) In the Defined symbols (-D) box click the **Add** button (the small icon with a green plus symbol on it)

(g) In the **Enter Value** window, enter `CONFIG_FS_BOOT_OFFSET=<value>`, replace `<value>` to appropriate offset. This must match the offset from the start of flash to the `boot` partition, as set in the PetaLinux system configuration menu, under `Flash Partition Table`.

TIP: *If the location of the boot offset is not yet determined, specify 0 and return to update once the location is determined.*

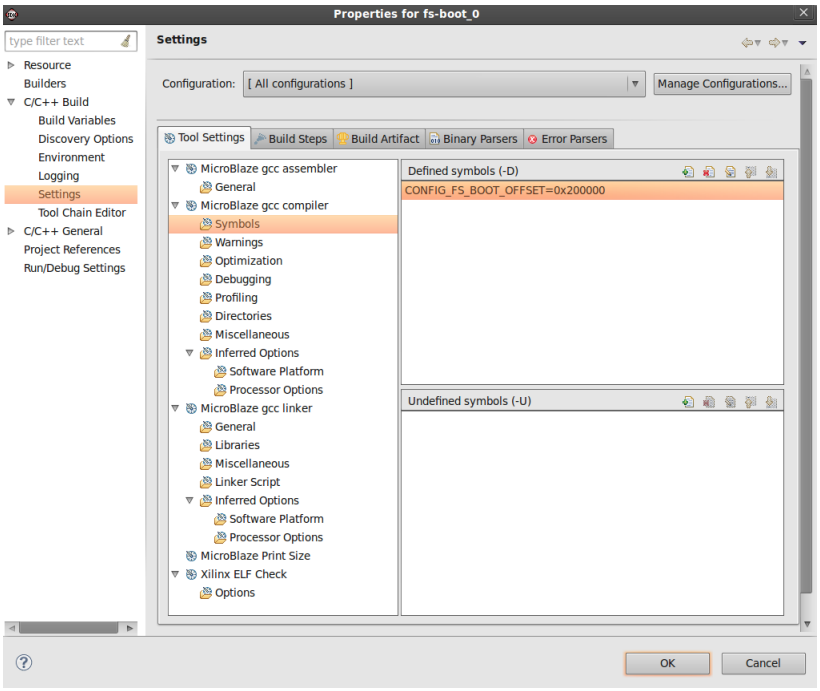

Figure 1: FS-Boot Configuration

(h) **For MicroBlaze targets only**. As fs-boot is a simple bootloader, the use of interrupts is not required and as such can be disabled, saving BRAM space by removing the vector table. The following linker flag can be set to disable the use of a vector table:

- Click **Miscellaneous** under MicroBlaze gcc linker
- Append `-Xl-mode-novectors` flag to **Linker Flags** field.

(i) Click **Apply** button to apply the settings

(j) Click **Ok** button to finish Compiler flag configuration

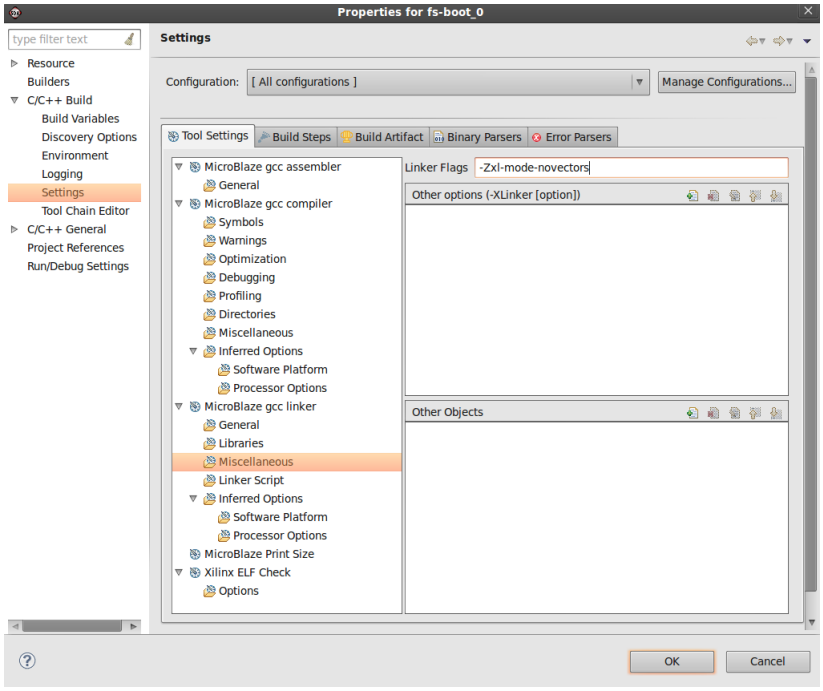

Figure 2: FS-Boot Configuration

6. Set to fs-boot to **Release** profile
   (a) Right click on the **fs-boot_0** project in Project explorer
   (b) Select **Build Configurations**
   (c) Select **Set Active** and select **Release**
7. Build the fs-boot project. The underlying XSDK BSP will be automatically built, if necessary.

   If XSDK did not automatically build the `fs-boot_0` software project, right click on the **fs-boot_0** project in Project explorer and select **Build Project** to build the project.

Finalising the FPGA Bitstream (MicroBlaze only)

IMPORTANT: *This step is not required for Zynq.*

At this stage, the hardware platform is ready and properly configured and the `"fs-boot"` executable binary is ready, but not initialised to BRAM. Please use `data2mem` to generate the bitstream which has the `"fs-boot"` initialised to BRAM.

Software Platform

Create New Platform

The next step is to create a new PetaLinux SDK software platform, ready for building a Linux system customised to your new hardware platform. The `petalinux-create` command is used to achieve this:

```
$ petalinux-create --type project --template <CPU_TYPE> --name <PROJECT_NAME>
```

The parameters are as follows:

- `--template <CPU_TYPE>` - The supported CPU types are `zynq` and `microblaze`
- `--name <PROJECT_NAME>` - The name of the platform or project you are building.

This command will create a new PetaLinux project folder from a default template. Later steps customise these settings to match the hardware project created previously.

TIP: *For detail of PetaLinux project structure, please refer to Appendix B in the PetaLinux SDK Getting Started Guide (UG977) .*

Configure Software Platform

The final step is to customise the software platform template to precisely match your unique hardware system. This is done by copying and merging the platform configuration files generated during the hardware build phase, into the newly created software platform, as described below:

NOTE: *<XSDK workspace directory> represents the XSDK workspace path, replace this with the correct value for your XSDK workspace.*

1. Navigate to appropriate directory before using the `petalinux-config` command. In general the XSDK BSP directory is for example: "<XSDK workspace directory>/petalinux_bsp_0/".
2. Use the `petalinux-config` command to import the hardware configuration.

```
$ petalinux-config --get-hw-description -p <plnx-proj-root>
INFO: Checking component...
INFO: Getting hardware description...
INFO: Using MSS file workspace/petalinux_bsp_0/system.mss and
XML file workspace/petalinux_bsp_0/../zc702.sdk/SDK/SDK_Export/hw/bd1.xml
INFO: Copy autoconfig for PetaLinux project: /home/user/Xilinx-ZC702-2013.3
INFO: Merging platform settings into kernel configuration
Auto-config file successfully updated for PetaLinux project: /home/user/Xilinx-ZC702-2013.3
```

The `-p` option points to the PetaLinux project that will be updated to match the hardware platform configurations.

This step may take a few minutes to complete. This is because it merges the existing platform configuration to Kernel configuration and enables the appropriated drivers in the Kernel configuration.

This tool generates the hardware configuration files, if required and copies the configuration files to the correct location in your project directory.

- The DTS file "system.dts", "xparameters.h" and "config.mk" files are placed in:

```
<project-root>/subsystems/linux/hw-description
```

3. Change into the directory of your PetaLinux project root.

```
$ cd <plnx-proj-root>
```

4. Launch the top level system settings configuration menu and configure it to meet your requirements: [Optional step]

```
$ petalinux-config
```

TIP: *For Zynq:*

- *The boot device is QSPI by default. If you are using a different boot device, you can select different* `System boot device` *from* `System Settings` *submenu.*

---

5. Launch the Linux kernel configuration menu and configure it to meet your requirements: [Optional step]

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

6. Launch the rootfs configuration menu and configure it to meet your requirements: [Optional step]

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

Build System Image

1. Build the hardware bitstream with Xilinx Vivado/XPS tool, if you have not done so already.
2. Change into the directory of your PetaLinux project.

```
$ cd <plnx-proj-root>
```

3. Run `petalinux-build` to build the system image:

```
$ petalinux-build
```

The console shows the compilation progress. e.g.:

```
INFO: Checking component...
INFO: Generating make files and build linux
INFO: Generating make files for the subcomponents of linux
INFO: Building linux
[INFO ] pre-build linux/rootfs/fwupgrade
[INFO ] pre-build linux/rootfs/peekpoke
```

The compilation log "build.log" is stored in the `build` subdirectory of your PetaLinux project.

Generate BOOT.BIN image for Zynq

This section is for Zynq only. Skip this section for MicroBlaze targets.

Follow the steps below to generate the boot image in ".bin" format.

```
$ petalinux-package --boot --fsbl <FSBL image> --fpga <FPGA bitstream> --uboot
```

For detailed usage, please refer to --help option or Appendix C in the PetaLinux SDK Getting Started Guide (UG977) .

Boot System Image on Board

After the hardware bitstream and the software images have been built, you can now test your new PetaLinux platform.

MicroBlaze Kernel Boot via JTAG

IMPORTANT: *The section is for MicroBlaze only*

1. Change into the directory of your project.

   ```
   $ cd <plnx-proj-root>
   ```

2. Program the FPGA using the Xilinx JTAG programming tools
3. Use `petalinux-boot --jtag` to download the image to the board and boot it:

   ```
   $ petalinux-boot --jtag --image images/linux/image.elf
   ```

   Please note direct kernel boot via JTAG can take a while, depends on the kernel image size. Please do not interrupt during this process, otherwise JTAG cable lockup can occur.

Zynq Kernel Boot

IMPORTANT: *The section is for Zynq only*

There are many different way to boot a Zynq kernel, for example:

1. SD boot by copying the "BOOT.BIN" and "image.ub" files to an SD card.
2. JTAG boot with manual xmd operation.
3. JTAG boot via PetaLinux prebuilt capability.

This section describes the simplest method to JTAG boot the kernel, using the PetaLinux prebuilt capability. Follows the steps to package a prebuilt image and boot it.

1. Change into the directory of your project.

```
$ cd <plnx-proj-root>
```

2. Please use ensure the board boot mode is set to JTAG.
3. Ensure you have run `petalinux-package --boot` to generate BOOT.BIN as described in section *Generate BOOT.BIN image for Zynq*
4. Use `petalinux-package --prebuilt` to package the prebuilt:

```
$ petalinux-package --prebuilt --fpga <FPGA bitstream>
```

   For detailed usage, please refer to --help option or Appendix C in the PetaLinux SDK Getting Started Guide (UG977) .

5. Use `petalinux-boot --jtag` to download the images to the board and boot it:

```
$ petalinux-boot --jtag --prebuilt 3
```

   For detailed usage, please refer to --help option or Appendix C in the PetaLinux SDK Getting Started Guide (UG977) .

Troubleshooting

This section describes some common issues you may experience when performing board bring up with PetaLinux SDK, and ways to solve them.

Problem/Error Message	Description and Solution

Cannot see any console output when trying to boot u-boot or kernel on hardware but boots fine on QEMU.	***Problem Description:***
	This problem is usually cause by the one or more of the following:
	• The serial communication terminal application is set with the wrong baud rate.
	• Incorrect XSDK PetaLinux BSP configuration.
	• Mismatch between hardware and software platforms.
	Solution:
	• Ensure your terminal application baud rate is correct and matches your hardware configuration.
	• Ensure the bootloader and XSDK PetaLinux BSP are based on the correct hardware platform
	• Ensure the DTS/xparameter matches the hardware platform. If they don't match:
	  ◦ Ensure the XSDK PetaLinux BSP and the bootloader are based on the correct hardware platform, if not, please correct it.
	  ◦ Ensure `stdout`, `stdin` are set to the correct UART IP core. Also check if other configurations in XSDK PetaLinux BSP are correctly configured.
	  ◦ Rebuild the XSDK PetaLinux BSP and the bootloader, and run `petalinux-config --get-hw-description` again from the XSDK PetaLinux BSP directory.
	  ◦ Rebuild software images

Additional Resources

References

- PetaLinux SDK Application Development Guide (UG981)
- PetaLinux SDK Board Bringup Guide (UG980)
- PetaLinux SDK Firmware Upgrade Guide (UG983)
- PetaLinux SDK Getting Started Guide (UG977)
- PetaLinux SDK Installation Guide (UG976)
- PetaLinux SDK QEMU System Simulation Guide (UG982)

PetaLinux SDK Documentation is available at http://www.xilinx.com/petalinux.