

[Home](#) » [Device Tree Generator](#)

Device Tree Generator

- [Using Git](#)
- [Open Source Linux \(OSL\)](#)
- [QEMU System Model](#)
- [Archive](#)
 - [Using Git](#)
 - [Open Source Linux \(OSL\)](#)
 - [Device Tree Generator](#)
 - [Installing and Using the Generator](#)

Site Utilities

- [Home](#)
- [What is a Wiki Site?](#)
- [How to edit pages?](#)
- [How to join this site?](#)
- [Recent changes](#)
- [List all pages](#)
- [Page Tags](#)
- [Site members](#)
- [Site Manager](#)

Add a new page

Summary

This guide describes how to use the device tree generator. The device tree generator is a Xilinx EDK tool that plugs into the Automatic BSP Generation features of the tool, XPS.

The purpose of the device tree generator is to produce a device tree file (xilinx.dts) that has the information for the hardware design in your EDK project. This alleviates having to create a device tree file with a text editor, which can be quite tedious.

The device tree generator is called device-tree, which is at [git://git.xilinx.com/device-tree.git](http://git.xilinx.com/device-tree.git).¹

Device Tree Documentation

The Device Tree Generator is not perfect and some hand editing of the device tree file may be required.

The device tree is documented (somewhat) in the Linux kernel at Documentation/powerpc/booting-without-of.txt and in the dts-bindings directory.

The following link, <http://ozlabs.org/~dgbson/papers/dtc-paper.pdf>, also has some useful information.

Also, http://www.devicetree.org/Device_Tree_Usage is a wiki with some good information and links to more official documents.

Updates Since 1/1/11

1. Added support for Little Endian/AXI systems.

Getting the Device Tree Generator From Xilinx

The following command will get the device-tree repository from the Xilinx Git repository. For users without Git, see the page at <http://git.xilinx.com> as it indicates how to create a snapshot (tar ball) of the kernel.

```
bash> git clone git://git.xilinx.com/device-tree.git
```

EDK/SDK Setup (for EDK versions 12.4 and newer)

Starting in EDK 12.4, the device-tree generation was moved from EDK/XPS to SDK. It now must be generated in the SDK which can be launched from the EDK (or other ways).

After getting the device tree generator from [git://git.xilinx.com/device-tree.git](http://git.xilinx.com/device-tree.git), you will need to unpack and save the 'bsp' directory and contents to any location that can be accessed by the SDK. The SDK requires the user to point it to the directory that contains the bsp folder.

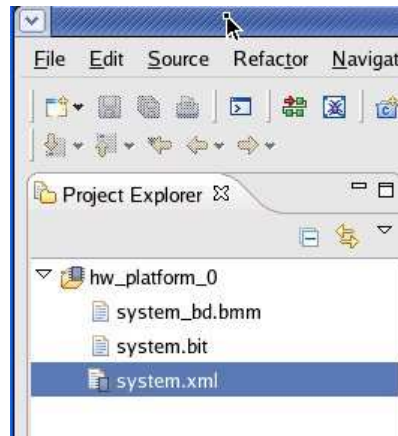
Export the hardware design to the SDK by selecting the 'Project' menu in the EDK, followed by the 'Export Hardware Design to SDK' submenu. Note that a similar operation, exporting a hardware project to the SDK, can be done from the ISE tool for ISE projects that have an EDK project embedded in them. The dialog box which is similar to the one illustrated below will be displayed. The 'Include bitstream and BMM file' option does not need to be selected for running the device-tree generator.



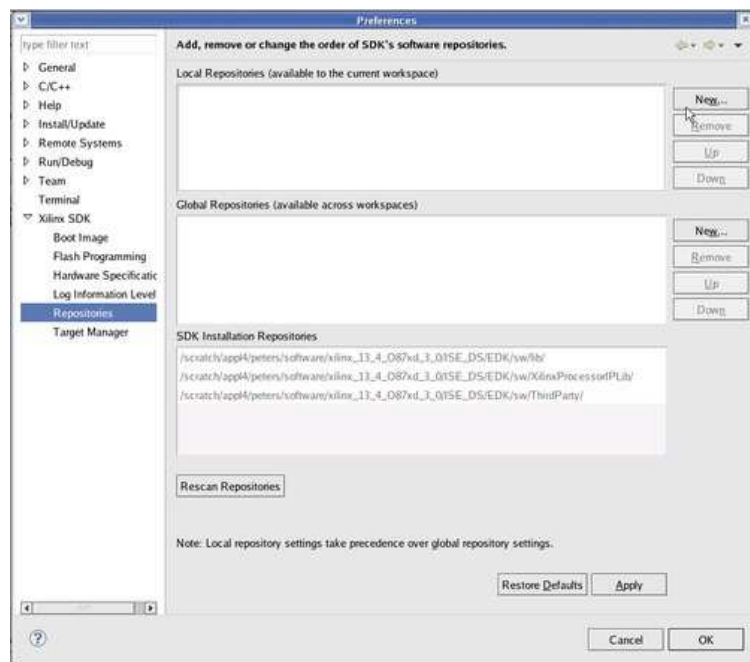
Select 'Export and Launch SDK' to launch the SDK. After selecting/creating a workspace, the hardware specification will be already defined in the SDK workspace.

Alternatively do 'Export Only' and launch SDK any other way. In this case the hardware specification has to be declared with 'File' —> 'New' —> 'Xilinx Hardware Platform Specification'. Enter a Project Name and in for the 'Target Hardware Specification' browse to the .xml file that was exported from EDK/XPS before. It usually can be found in the <xps project>/SDK/SDK_Export/hw folder.

After this step the 'Project Explorer' should show the Hardware Platform like for example:



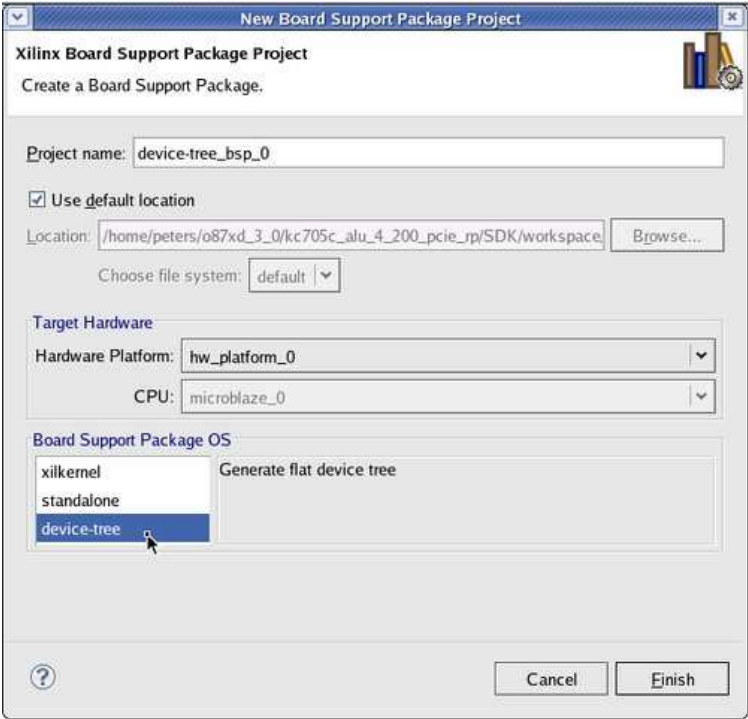
From the SDK, select the 'Xilinx Tools' menu followed by the 'Repositories' submenu. A dialog box similar to the one illustrated below will be displayed.



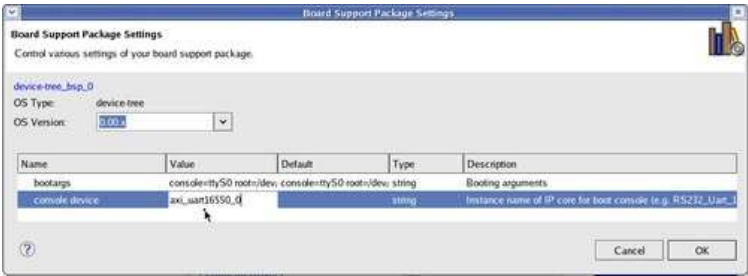
Select the 'New' button and then browse to the directory where the device-tree downloaded from the Git server has been extracted. Select the folder containing the 'bsb' subdirectory. Select the 'Rescan Repositories' button. Select the 'OK' button to complete the addition of the new repository. Note the INFO messages on the SDK console.

Using the Device Tree Generator

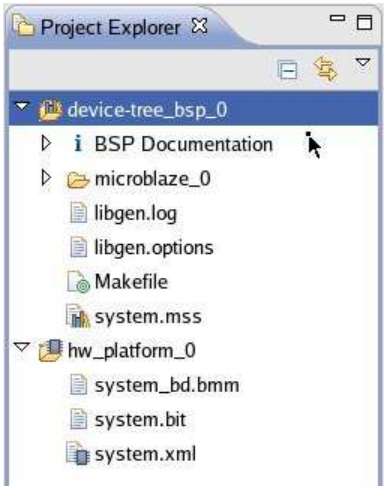
Now a Device Tree BSP can be created. Select the 'File' menu, followed the 'New' submenu, followed by the 'Xilinx Board Support Package' submenu. A dialog box similar to the one illustrated below will be displayed to allow the device tree to be selected.



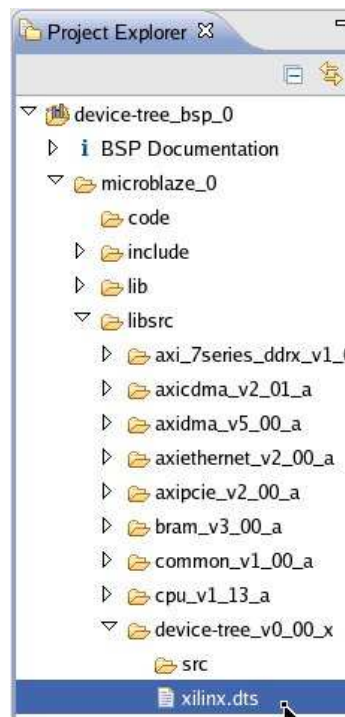
Select the 'device-tree' as 'Board Support Package OS' and then hit the 'Finish' button.
In the next dialog, 'bootargs' and 'console device' can be specified, as described in section 'Device Tree Options'.



Clicking OK adds the device-tree-bsp to the SDK workspace and displays it in the 'Project Explorer'.



In case the SDK project has set 'Project' —> 'Build Automatically', the device-tree generator immediately runs to create the device-tree (xilinx.dts).
If not, right-mouse-click the device-tree BSP in 'Project Explorer' and run 'Build Project'.
The xilinx.dts file is located in <workspace>/<device-tree_bsp>/<processor-name>/libsrc/device-tree_v0_00_x folder.



The device tree data is now ready for building the linux kernel.

EDK Setup (for EDK versions 12.3 and older)

After getting the device tree generator from [git://git.xilinx.com/device-tree.git](http://git.xilinx.com/device-tree.git), you will need to copy the 'bsp' directory and contents so that it can be used by XPS. The simplest approach is to copy the 'bsp' directory to the top of your project directory.²

NOTE: Under Centos 5 or Ubuntu, you have to softlink the libdb library. In Centos 5, run "ln -s /lib/libdb-4.3.so /lib/libdb-4.1.so" to get the dts generator to finish under certain circumstances.

```
cp -r bsp <path to project>/
```

If your project is currently open in XPS, you will need to close and re-open the project for XPS to notice the changes.

Using the Device Tree Generator

Open the pull down menu 'Software' and proceed to 'Software Platform Settings...' in XPS.

To use the device tree generator, select 'device-tree' in the pull down menu labeled 'OS' in the Software Platform Settings dialog box.

Processor Information

Processor Instance:

ppc440_0

Software Platform

OS and Libraries

Drivers

Processor Settings

CPU Driver:

cpu_ppc440

 CPU Driver Version:

1.00.b

Processor Parameters:

| Name | Current Value | Default Value | Type | Description |
|----------------------|------------------|------------------|--------|----------------------|
| ppc440_0 | | | | |
| EXTRA_COMPILER_FLAGS | -g | -g | string | Extra compiler flags |
| ARCHIVER | powerpc-eabi-ar | powerpc-eabi-ar | string | Archiver |
| COMPILER | powerpc-eabi-gcc | powerpc-eabi-gcc | string | Compiler |

OS & Library Settings

OS:

device-tree

 Version:

| Use | Library | Version | Description |
|-----|---------|---------|-------------|
|-----|---------|---------|-------------|

OK

Then select 'OS and Libraries' on the left, and enter the values for 'console device' and 'bootargs'

Processor Information

Processor Instance:

ppc440_0

Software Platform

OS and Libraries

Drivers

Configuration for OS:

device-tree v

| Name | Current Value | Default Value | Type |
|----------------|---------------------------------------------------------|---------------|--------|
| device-tree | | | |
| console device | RS232_Uart | | string |
| bootargs | console=ttyS0 root=/dev/ram console=ttyS0 root=/dev/ram | | string |

Configuration for Libraries

OK

Finally close the Software Platform Settings dialog box by clicking on the 'OK' button, and select 'Generate Libraries and BSPs' from the software menu. The resulting device tree file (xilinx.dts) will reside in your project directory in <processor instance>/libsrc/device-tree/xilinx.dts.

Device Tree Options

Console Device Option

The 'console device' option lets you select which IP core is to be used for the console. This value is needed by linux for output early during bootup. Put in the name of the instance (e.g. RS232_Uart_1) rather than the type of the core (e.g. not xps_uart16550). The IP instance name can be found in XPS using the 'System Assembly View' tab (in the middle of the XPS window).

Bootargs Option

The 'bootargs' option lets you type in the text for the linux kernel bootargs. The default value (at the time of this writing) is

```
console=ttyS0 root=/dev/ram
```

Change the text to what is needed for your system. The following section describes a few of the common options given in bootargs.

console

The value for console=... needs to select the same device given in the 'console device' option. The following is a table showing what device to select in 'console=' based on Xilinx device types commonly used for the console.

| device type | console text |
|----------------------------------------------------------------------------------------|-----------------------------|
| xps_uart16550 | console=ttySN |
| xps_uartlite | console=ttyULN ³ |
| N is 0 for the first device of the type. N is 1 for the second device of the type etc. | |

ip

The option ip=... specifies the ip address used. To use dhcp to dynamically assign an ip address, specify

```
ip=on
```

To specify a static ip address, for example 192.168.1.10, use

```
ip=192.168.1.10
```

root

The option root=... tells the linux kernel where to find the root filesystem. The root filesystem can be in ram, on the compact flash disk, or even on an nfs share on a remote computer. Add the option rw after the option if you would like the filesystem to be read/writeable. Please note that for ramdisk images, any changes will be in ram only and lost on reboot.

The following is a table showing various bootarg options for different root filesystem locations.

| root filesystem location | bootarg options |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------|
| ramdisk | root=/dev/ram |
| compact flash disk | root=/dev/xsyzace/disc0/partN (where N is the partition number of the root file system.) ^{4 5} |
| nfs | root=/dev/nfs nfsroot=<nfs server>:<nfs share>,tcp (e.g. root=/dev/nfs nfsroot=192.168.1.1:/nfsroots/development,tcp) |

Footnotes

- 1. There is an older device tree generator, called fdt, which is deprecated. The fdt git repository is at [git://git.xilinx.com/gen-mhs-devtree.git](https://git.xilinx.com/gen-mhs-devtree.git).
- 2. For other installation methods, refer to EDK documentation.
- 3. Ensure the kernel is configured to have support for the Uartlite when using this option
- 4. the CF card shipped with Xilinx boards often has a linux root filesystem on partition 2.
- 5. A sysace IP core needs to be present in your system with interrupts connected for this to work.

page revision: 3, last edited: 11 Apr 2012, 17:10 GMT (5 days ago)

[Edit](#) [Tags](#) [History](#) [Files](#) [Print](#) [Site tools](#) [+ Options](#)

Xilinx, Inc.

Engineering Innovation

Xilinx delivers programmable platforms to help design engineers make their vision a reality.








Company

[About Us](#)
[Investor Relations](#)
[Xilinx in the News](#)
[Xcell Journal](#)
[Jobs at Xilinx](#)

Support

[Documentation](#)
[Downloads](#)
[Licensing](#)
[Training](#)
[Solution Centers](#)

Connect

 [Contact Us](#)
 [Connect on Facebook](#)
 [Follow us on Twitter](#)
 [Connect on LinkedIn](#)
 [Watch us on YouTube](#)
 [Join our Community](#)
 [Subscribe to Newsletter](#)

Unless otherwise stated, the content of this page is licensed under [Creative Commons Attribution-ShareAlike 3.0 License](#)