


Embedded Linux

Home (<https://numato.com/help>) / Knowledge Base (<https://numato.com/kb/>) / Saturn Spartan 6 FPGA Module (<https://numato.com/kb-category/saturn-spartan-6-fpga-module/>)





 Have a question? Enter a search term

SEARCH

Popular Search: USB GPIO (), USB Relay (), FPGA ()



Saturn, Microblaze and Linux – How to Run Linux on Saturn Spartan 6 FPGA Module – Part II

 2455 views  March 9, 2016  admin  2



(https://numato.com/help/wp-content/uploads/2016/03/microblaze_Linux.png)

- **Part I** (<http://docs.numato.com/knowledge/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-i/>)
- **Part II**
- **Part III** (<http://docs.numato.com/knowledge/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-iii/>)
- **Part IV** (<https://numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-iv/>)

In Part I (<http://docs.numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-i/>), we saw how to generate a microblaze based minimal embedded hardware platform for Saturn Spartan 6 FPGA module. Now we recommend that you create a SDK application project for this hardware platform and test it's basic functionality by creating a "Hello world" application. Please follow the steps under the section "Creating Application program for Microblaze based Embedded Platform" in the article Creating Xilinx EDK test project for Saturn – Your first Microblaze processor based embedded design (<https://numato.com/kb/creating-xilinx-edk-test-project-saturn-your-first-microblaze-processor-based-embedded-design/>) to create a SDK project. Choose "Hello world" SDK application template when generating the application. Run the application on the board to make sure that everything so far works as expected.

Assuming you were able to build the hardware platform using XPS and test it using SDK successfully, let us move on to the discussion about how to build Linux kernel. There are different ways to do this depending on how much time and effort you would like to invest. The easiest of all is to use a tool such as Buildroot or Yoctoproject. In this example, we will use Buildroot (<https://buildroot.uclibc.org/>) to build our kernel. Buildroot, when properly configured, can download Linux kernel source, build cross compilation tools and build the kernel itself all with a single command.

Please note that we will continue using Windows for running Xilinx tools (and of course Linux for building kernel). I have used Ubuntu 14.04 LTS 64 bit desktop version while putting together this article. We will do the following steps one by one to build the Linux Kernel.

1. Create a Device Tree file
2. Download/Install Buildroot and other necessary packages



3. Create buildroot configuration

4. Build the kernel

1. Creating Device Tree file

On many platforms, Linux kernel depends on a data structure called Device Tree to discover and configure different hardware components available on the board. Kernel uses the device tree in binary format which is generated by using device tree compiler. The input to the device tree compiler is a text file (with extension .dts) that describes all the hardware and dependencies. This file can be edited manually by hand if necessary. But fortunately, there are tools available that can help us generate device tree from our XPS/SDK projects.

To generate device tree, we will need the following files from XPS/SDK projects.

- SaturnV3Linux.xml (can be found under the folder SDK\SDK_Export\hw inside the XPS hardware project. Exact file name may be different depending on the XPS project name you chose)
- system.mss (can be found under the bsp project folder in Hello World SDK workspace)

Copy these two files to a common folder. Open command prompt and move to the folder where the files are copied. Edit the file system.mss and replace the contents of the section that begins with "BEGIN OS" with the following and save.

```
BEGIN OS
  PARAMETER OS_NAME = device-tree
  PARAMETER OS_VER = 3.10.a
  PARAMETER PROC_INSTANCE = microblaze_0
  PARAMETER CONSOLE_DEVICE = ft2232_uart
END
```

Now download the device-tree_v2_1_0.mld and device-tree_v2_1_0.tcl files from here (https://github.com/numato/samplecode/tree/master/FPGA/Saturn/SaturnLinux/devicetree/bsp/device-tree_v3_10_a/data). Place these files under the folder bsp\device-tree_v3_10_a\data along with SaturnV3Linux.xml and system.mss. Please note that it is critical to use the exact file names and folder names mentioned here. Files should be placed exactly as below.

\SaturnV3Linux.xml

\system.mss

\bsp\device-tree_v3_10_a\data\device-tree_v2_1_0.mld

\bsp\device-tree_v3_10_a\data\device-tree_v2_1_0.tcl



Once all files are placed properly, run the following commands at the command prompt.

```
cmd>c:\Xilinx\14.6\ISE_DS\settings64.bat  
cmd>libgen -hw SaturnV3Linux.xml -lp device-tree -pe microblaze_0 system.mss
```

make sure to replace 14.6 in the first command with the version of ISE you have on your machine. Run settings32.bat instead of settings64.bat if you have 32 bit system. If everything went well, libgen should run without any errors and a new folder microblaze_0 will be created. You will see xilinx.dts under the folder microblaze_0\libsrc\device-tree_v3_10_a and this is our device tree file. rename xilinx.dts to saturn_v3.dts. At this point, we will need to make some changes to the dts file manually. Open saturn_v3.dts in your favorite text editor and replace the bootargs = "" line with the following

```
bootargs = "console=ttyUL0";  
linux,stdout-path = "/axi@0/serial@40600000";
```

Make sure that the base address 40600000 in the above line matches with the base address of FT2232H UART base address in your system.

2. Download/Install Buildroot and other necessary packages

Login to your Linux workstation to download and setup buildroot. On a Ubuntu 14.04 desktop clean installation you will need to install a few packages to make Buildroot work. Run the following command to install those.

```
cmd>sudo apt-get install build-essential bison flex gettext libncurses5-dev texinfo auto
```

You may need install Qt lib if you prefer to use xconfig instead of menuconfig while configuring Buildroot (please note that we will not use xconfig or menuconfig in the article rather load a default configuration that is known to be working).

Now create a convenient folder on your Linux machine and move to that folder on the command line. Download buildroot to that folder. You can download buildroot manually from Buildroot download page or by running the following command.

```
cmd>wget http://buildroot.uclibc.org/downloads/buildroot-2014.05.tar.gz
```



If you prefer to use a different buildroot release, change the file name accordingly. Extract the downloaded gzip archive by running the following command

```
cmd>tar -xvf buildroot-2014.05.tar.gz
```

All files should be extracted to the folder buildroot-2014.05.

3. Create buildroot configuration

Now download the Linux kernel defconfig for saturn V3 from here (https://github.com/numato/samplecode/tree/master/FPGA/Saturn/SaturnLinux/buildroot/configfiles/board/numato/saturn_v3) and place it under directory \board\numato\saturn_v3 (inside the folder the gzip archive extracted to). And copy the device tree file saturn_v3.dts to the same folder as well. Download buildroot defconfig numato_saturn_v3_defconfig from here (<https://github.com/numato/samplecode/tree/master/FPGA/Saturn/SaturnLinux/buildroot/configfiles/configs>) and copy it to configs folder under buildroot. Please note that the kernel base address set to 0xa4000000 in the Linux kernel defconfig you have downloaded. Please change it appropriately if the LPDDR base address in your XPS project is different.

After setting up buildroot as mentioned above and execute the following commands at the Buildroot root directory.

```
cmd>make numato_saturn_v3_defconfig
cmd>make
```

The first make command will generate a .config file for Buildroot and the second make command will build the kernel. Please note that the machine should be connected to internet for this step to work since Buildroot needs to download Linux kernel sources and other tools before it can begin building the kernel. This step can take up to an hour or more depending on your machine's speed and internet connection speed. If this step finished without errors, you will see a new file "simpleImage.saturn_v3" in the folder /output/images. This is the Linux kernel with a root file system (initramfs) attached to it. We will use this file to boot Linux on Saturn.

The next step is to try to boot the Linux kernel we just created. Please see part III (<http://docs.numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-iii/>) of this article to see how to boot the kernel image using Xilinx XMD and Xilinx Platform cable USB. If you don't have platform cable USB handy, you can skip part III and proceed to part IV (<https://numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-iv/>) where we will see how to create a bootable image that can be flashed directly to the SPI flash

available on Saturn Spartan 6 FPGA module.

Read Part III of this article series here. (<http://docs.numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-iii/>)



([https://twitter.com/home?status=https://numato.com/kb/saturn-microblaze-](https://twitter.com/home?status=https://numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/)

[and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/](https://twitter.com/home?status=https://numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/))



([https://www.facebook.com/sharer/sharer.php?u=https://numato.com/kb/saturn-](https://www.facebook.com/sharer/sharer.php?u=https://numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/)

[microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/](https://www.facebook.com/sharer/sharer.php?u=https://numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/))



([https://pinterest.com/pin/create/button/?url=https://numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/&media=&](https://pinterest.com/pin/create/button/?url=https://numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/&media=&description=)

[description=](https://pinterest.com/pin/create/button/?url=https://numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/&media=&description=))



(<https://plus.google.com/share?url=https://numato.com>


[/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/](https://plus.google.com/share?url=https://numato.com))



(<https://www.linkedin.com/shareArticle?mini=true&url=https://numato.com>

[/kb/saturn-microblaze-and-linux-how-run-linux-saturn-spartan-6-fpga-module-part-ii/](https://www.linkedin.com/shareArticle?mini=true&url=https://numato.com))

Was this helpful?

 2 Yes

 No

1 Comments

Andrew says:

REPLY (<https://numato.com/kb/saturn-microblaze-and-linux-how-run-linux-saturn->



SPARTAN-6-FPGA-MODULE-PART-II/?REPLYTOCOM=85#RESPOND)

Where has the numato_saturn_v3_defconfig gone ?

The link http://numato.cc/wp-content/uploads/2015/11/numato_saturn_v3_defconfig (http://numato.cc/wp-content/uploads/2015/11/numato_saturn_v3_defconfig) doesn't work.

and the link

[http://numato.cc/wp-content/uploads/2015/11/buildroot-](http://numato.cc/wp-content/uploads/2015/11/buildroot-2014.05_saturn_preconfigured.tar.gz)

[2014.05_saturn_preconfigured.tar.gz](http://numato.cc/wp-content/uploads/2015/11/buildroot-2014.05_saturn_preconfigured.tar.gz) (http://numato.cc/wp-content/uploads/2015/11/buildroot-2014.05_saturn_preconfigured.tar.gz)

no longer works either !

November 8, 2017 at 4:14 pm

Leave A Comment

Comment

Name

*

Email

*

Website

POST COMMENT

Knowledge Base Category

Callisto Kintex 7 USB 3.1 FPGA Module (<https://numato.com/kb-category/callisto-kintex-7-usb-3-1-fpga-module/>) (2)

Embedded Linux (<https://numato.com/kb-category/embedded-linux/>) (12)

FPGA Applications (<https://numato.com/kb-category/fpga-applications/>) (13)



Galatea PCI Express Spartan 6 FPGA Module (<https://numato.com/kb-category/galatea->

[pci-express-spartan-6-fpga-module/](#)) (4)

[Getting Started With FPGA \(https://numato.com/kb-category/getting-started-with-fpga/\)](#) (32)

[Intercore SDK Framework \(https://numato.com/kb-category/intercore-sdk-framework/\)](#) (1)

[Mimas A7 Mini FPGA Development Board \(https://numato.com/kb-category/mimas-a7-mini-fpga-development-board/\)](#) (3)

[Mimas Artix 7 FPGA Development Board \(https://numato.com/kb-category/mimas-artix-7-fpga-development-board/\)](#) (10)

[Narvi Spartan 7 FPGA Module \(https://numato.com/kb-category/narvi-spartan-7-fpga-module/\)](#) (2)

[Nereid Kintex 7 PCI Express FPGA Board \(https://numato.com/kb-category/nereid-kintex-7-pci-express-fpga-board/\)](#) (3)

[Neso Artix 7 FPGA Module \(https://numato.com/kb-category/neso-artix-7-fpga-module/\)](#) (10)

[Opsis: FPGA-based open video platform \(https://numato.com/kb-category/opsis-fpga-based-open-video-platform/\)](#) (4)

[Prodigy Series Automation Devices \(https://numato.com/kb-category/prodigy-series-automation-devices/\)](#) (1)

[Proteus Kintex 7 FPGA Development Module \(https://numato.com/kb-category/proteus-kintex-7-fpga-development-module/\)](#) (1)

[Quick Start Guides \(https://numato.com/kb-category/quick-start-guides/\)](#) (2)

[Rhea Device Management Tool \(https://numato.com/kb-category/rhea-device-management-tool/\)](#) (2)

[Saturn Spartan 6 FPGA Module \(https://numato.com/kb-category/saturn-spartan-6-fpga-module/\)](#) (7)

[Skoll Kintex 7 FPGA Module \(https://numato.com/kb-category/skoll-kintex-7-fpga-module/\)](#) (6)

[Styx Xilinx Zynq FPGA Module \(https://numato.com/kb-category/styx-xilinx-zynq-fpga-module/\)](#) (8)

[Tagus – Artix 7 PCI Express Development Board \(https://numato.com/kb-category/tagus-artix-7-pci-express-development-board/\)](#) (1)

[Telesto MAX 10 FPGA Module \(https://numato.com/kb-category/telesto-max-10-fpga-module/\)](#) (5)



Tenagra FPGA System Management Software (<https://numato.com/kb-category/tenagra-fpga-system-management-software/>) (3)

Theia Android Application (<https://numato.com/kb-category/theia-android-application/>) (1)

USB GPIO Modules (<https://numato.com/kb-category/usb-gpio-modules/>) (2)

USB Relay Modules (<https://numato.com/kb-category/usb-relay-modules/>) (1)

Vivado Design Suit (<https://numato.com/kb-category/vivado-design-suit/>) (4)

Waxwing Spartan 6 FPGA Development Board (<https://numato.com/kb-category/waxwing-spartan-6-fpga-development-board/>) (3)

White Papers (<https://numato.com/kb-category/white-papers/>) (1)

Working With Xilinx EDK (<https://numato.com/kb-category/working-with-xilinx-edk/>) (1)

XO-Bus Framework (<https://numato.com/kb-category/xo-bus/>) (2)

Don't miss out on new articles! Subscribe to get valuable insights.

Subscribe



(<https://twitter.com/numatolab>)



(<https://www.facebook.com/numato/>)



(<https://www.youtube.com/user/NumatoLab>)



(<https://plus.google.com/+Numatosystems>)



