

[Home](#) [Blog](#) [CCCMS](#) [Other Projects](#) [Contact](#) [About Me](#)

Past Posts:

[March \(0\)](#)[February \(0\)](#)[January \(0\)](#)[December \(0\)](#)[November \(0\)](#)[October \(0\)](#)

Recompiling the BeagleBone Kernel for Angstrom

Posted On: April 13, 2012 (09:04)

Posted by: jack

Today I will go over the process of recompiling the beaglebone Linux kernel for the Angstrom distribution which is based upon OpenEmbedded. The procedure I use here is one of many that can be used and some people will agree with it, some will not. If you have suggestions on how I could improve my work flow it would be greatly appreciated, let me know in the comments!

The pre-requisites to this tutorial is that you have a working Angstrom build, details of how to do this could be found in my previous blog posts. The procedure I use takes a lot of disk space and I would recommend having at least 50GB free available on the drive that you are performing your build.

Note: After you have first setup Angstrom using the oebb.sh script it will create an environment-setup file in /home/your-user/.oe/environment-oeore. To setup your environment to use Bitbake which is the tool used to help compile your Angstrom distribution you must run this file:

```
. ~/.oe/environment-oeore
```

Please note you will have to do this for every NEW terminal you open and want to use the bitbake tool in. Running this file replaces the need to use the oebb.sh helper script. If you wish to update your sources then you will need to run the script again as follows:

```
./oebb update
```

From within your Angstrom directory.

Step 1) Clean the kernel shared state

Firstly, clean the sharedstate files for your Kernel, otherwise it won't rebuild as it has already got the build saved in a special file.

```
bitbake -c cleansstate virtual/kernel
```

Step 2) Compile the kernel but don't create the package

```
bitbake -c configure virtual/kernel
```

Step 3) Copy the Kernel build folder

We copy the Kernel sources out of the Angstrom folder to allow us to make changes to the source without getting in the way of Angstrom, and so that it doesn't remove our work.

```
cp -r /path/to/angstrom/build/tmp-angstrom*/work/beaglebone-angstrom-linux-gnueabi/linux-ti33x-psp-3.2-* /home/your-user/
```

The above will copy the kernel sources to your home directory. Feel free to change the path to where you wish to store

the sources, this is just an example!

Step 4) Use Git to help manage your source

I find the easiest way to manage my Kernel source in this situation is to use git. So first navigate to your kernel source directory that you just copied over. Then in that directory run:

```
git init
```

This creates an empty git repository in your kernel source folder. We will now add the current kernel sources to the repository

```
git add .
```

This will add all the Kernel files to your new repository. Now we will commit the changes (make them permanent).

```
git commit -m 'angstrom kernel'
```

Now, our git repo is setup, we have a commit at the vanilla Angstrom kernel, now we can make changes to it!

Step 5) Create your patch!

Once you have made your changes to the kernel source (possible the SPIDEV changes that I wrote a blog post on a few weeks ago) you must create a patch which you can then use it your Angstrom build. To do this run:

```
git diff > custom-kernel-diff.patch
```

This will produce a file in your kernel sources directory called custom-kernel-diff.patch and will contain all the changes you made to your kernel sources.

Step 6) Apply your patch to the Angstrom build

Once you have your patch you must copy it over to the Angstrom build. The directory that you must copy it into is the beaglebone kernel directory which is part of the meta-ti layer.

```
cp      custom-kernel-diff.patch      /path/to/angstrom/sources/meta-ti/recipes-  
kernel/linux/linux-ti33x-psp-3.2/
```

Now that your patch is in the Angstrom sources directory, you must tell it to use it!

Step 7) Add patch to beaglebone kernel recipe

Open the recipe to the beaglebone kernel which can be found at:

```
/path/to/angstrom/sources/meta-ti/recipes-kernel/linux/linux-ti33x-psp_3.2.bb
```

Go right down to the bottom and add to the list of patches being applied the name of your patch:

```
file://custom-kernel-diff.patch \
```

Make sure that you add it before the quotation mark (") and also make sure you have a backslash and at the end of the

file name after a space. Looking at the other files in the list should make this obvious.

Step 8) Re-compile your kernel with the new patch!

```
bitbake virtual/kernel
```

This will recompile the kernel and include your new patch. If your patch is wrong or breaks the kernel then you get errors that you will have to fix yourself!

If you run into any problems please leave a comment. If it is a large amount of text, please use pastebin.com and post the link.

There is also another very good post on making changes to the kernel here : <http://www.slimlogic.co.uk/2011/05/openembeddedangstrom-kernel-workflow>

You are posting as Anonymous

Make a comment on this blog post!

Spam Test: Yes/No, Is GCC a compiler collection?

Post Comment

16 Sep 13, 17:00

Anonymous

Jack,

Thank you and this is exactly what I am looking for. However the file I wish to modify is the dts pin mapping for LCD4 cape. You can see it in `/lib/firmware/BB-BONE-LCD4-00A1.dts`. There is a pin assignment error and it needs to be removed. Can you tell me what I should edit after the first kernel build? There are so many files and folders and I haven't find the dts file yet.

