Yocto Project™: Adding Configuration Fragment for Wireless

By Sean D. Liming and John R. Malin
Annabooks

August 2013

Kernel Modifications in Yocto Project

Kernel modifications are standard practice when configuring the Linux kernel for a system. As a cross-compile development tool chain, the Yocto Project also supports configuring the kernel, but the Yocto Project build process adds some abstraction that requires a few more steps to the process.

As a practical example, I had to add an Intel Centrino Ultimate-N 6300 wireless adapter to Intel N2800 systems known as Cedar Trail. The kernel version 3.0 is used for the Cedar Trail BSP, and wireless support is not enabled. This paper looks at the steps to enable wireless support for a Yocto Project build.

Note: The paper assumes that you have gone through the Yocto Project Quick Start Guide to set up and build a distribution.

Find Driver Using Distribution

The Linux Wireless wiki (http://wireless.kernel.org/) provides the basic information to set up wireless support, as well as, driver support information. The first step is to locate the actual driver for the card. As a test to verify the card is working properly, Ubuntu 12.04 LTS is installed on the target. Using the get-driver.sh script from the book *Linux Kernel in a Nutshell*, iwlwifi is confirmed as the card' s driver, which matches the expected driver from the wiki.

```
$./get-driver.sh wlan0

looking at sysfs device: /sys/class/net/wlan0
resolve link to: /sys/devices/pci0000:00/0000:00:1c.3/0000:02:00.0/net/wlan0
follow 'device' link to parent:
/sys/devices/pci0000:00/0000:00:1c.3/0000:02:00.0
found driver: iwlwifi from module: iwlwifi
found driver: pcieport
```

Creating the Configuration Fragment

The Yocto Project provides flexibility for the Linux kernel. Each Yocto Project release comes with a set of recipes to build different versions of the kernel. A recipe is also available to work with the latest kernel in development. If you already have a kernel, a skeletal recipe is available so you can integrate your own kernel version. The linux-yocto recipes contain a basic configuration to get the distribution booting on the target. Support for specific device drivers can be added using configuration fragments. The concept is to re-use the configuration fragments with the latest kernels integrated and tested with the Yocto Project development.

1. After downloading and setting up the development systems with all the tools and BSP needed to build the distribution, the first step is to open a terminal and create the project:

```
$Source poky-danny-8.0.1/oe-init-build-dev n2800
```

Or to be more generic:

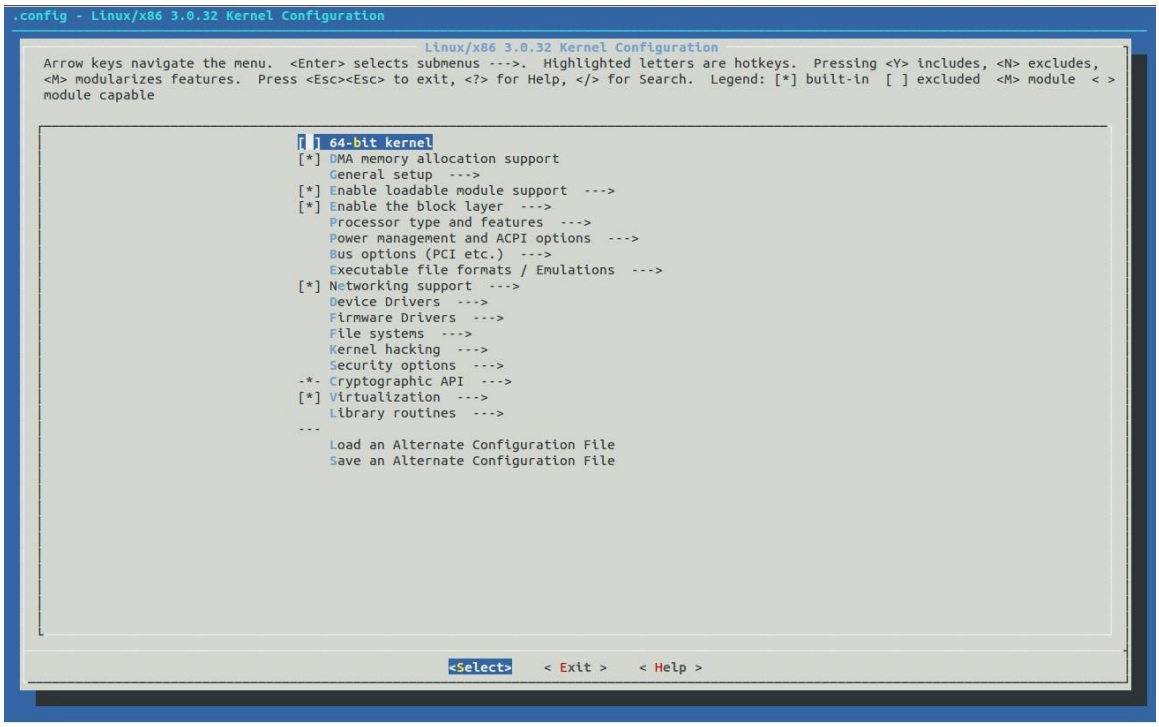
```
$Source poky-<release>-<ver>/oe-init-build-dev <platform>
```

2. The next step is to modify the local.conf and bblayers.conf files to add the layers for the Cedar Trail BSP, target the correct hardware platform, and set up the number of processors available. Please see the Yocto Project Quick Start Guide, BSP README, and development documentation for more information.
3. The next step is to run menuconfig for the kernel:

```
$ bitbake linux-yocto -c menuconfig
```

Copyright © 2013 Annabooks, LLC., All Rights Reserved.

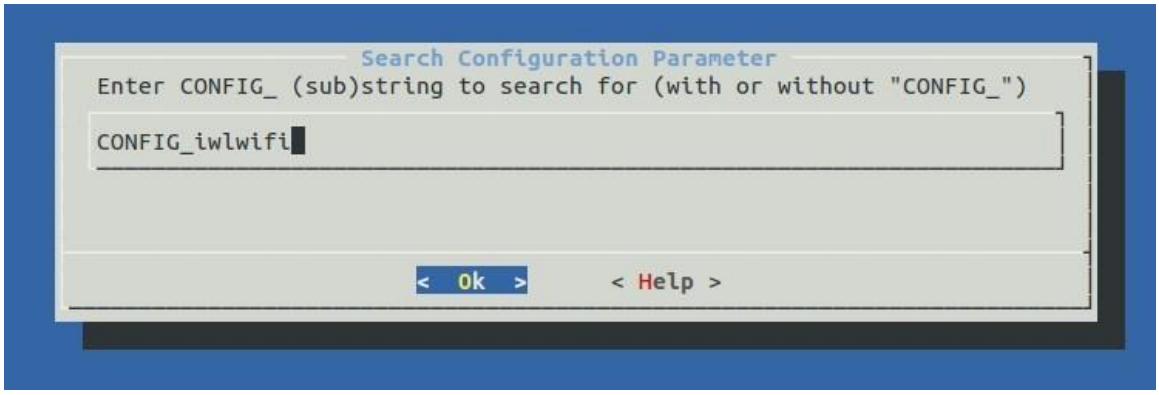
A little build process takes place, and a second terminal opens up to the Kernel Configuration program. Navigation is performed using the arrows keys, although some mouse scrolling is needed. Selection is with the space bar.



4. Before you go any further, the .config file that is generated must be saved off. For my platform the .config file is located:

```
/n2800/tmp/work/cedartrail_nopvr-poky-linux/Linux-yocto-3.0.32+git…../Linux-cedartrail_noprvr-standard-build
```

5. If you are using a GUI file browser, the .config file is hidden. You will have to select the view hidden files options.
6. Copy and save the file to a different location as config.orig.
7. Back to the kernel configuration utility. The first step is to search for the driver. Hit the “/’ to open a search dialog.
8. Enter CONFIG_iwlwifi and select Ok.



There are several different IWLWIFI driver selections for development and debug, but the one that is needed is standalone from all the rest.

```
Symbol: IWLWIFI_LEGACY [=n]
Type  : tristate
  Selects: FW_LOADER [=y] && NEW_LEDS [=n] && LEDS_CLASS [=n] && LEDS_TRIGGERS [=n] && MAC80211_LEDS [=n]
  Selected by: IWL4965 [=n] && NETDEVICES [=y] && WLAN [=y] && PCI [=y] && MAC80211 [=n] || IWL3945 [=n] && NETDEVICES [=y] && WLAN [=y] && PCI [=y] && MAC80211 [=n]
```

The =n means the driver is not part of the configuration. If you were to build the distribution as is, wireless support would not be enabled. From the search results, there are dependencies on LEDS_CLASS, LED_TRIGGERS and MAC80211, which are also not enabled. IWL4965 or IWL3945 should be selected.

9. Using the arrow keys and space bar, navigate through the different menus to select the items needed to enable all the dependencies and the wireless driver. For good measure, I enabled all the Intel Wireless drivers. Everything is set as built in driver rather than a module.

```
Device Drivers ---- >
       [*] LED Support --- >
              [*] LED Class Support
              [*] LED Trigger support

[*] Networking Support --- >
       Wireless  --- >
              <*> cfg80211 - wireless configuration API
              <*> Generic IEEE 802.11 Networking Stack (mac80211)

Device Drivers --- >
       [*] Network device support --- >
              [*] Wireless LAN --- >
                     <*> Intel PRO/Wireless 2100 Network Connection
                     <*>  Intel  PRO/Wireless  2200BG  and  2915ABG    Network
                     Connection
                     <*> Intel Wireless WiFi Next Gen AGN - Wireless-N/Advanced-
                     N/Ultimate-N (iwlan)
                     <*> Intel Wireless WiFi 4965AGN (iwl4965)
                     <*>  Intel  PRO/Wireless  3945ABG/BG  Network  Connection
                     (iwl3945)
```

10. Performing another search on CONFIG_iwlwifi shows all the options enabled:

```
Symbol: IWLWIFI_LEGACY [=y]
Type  : tristate
  Selects: FW_LOADER [=y] && NEW_LEDS [=y] && LEDS_CLASS [=y] && LEDS_TRIGGERS [=y] && MAC80211_LEDS [=y]
  Selected by: IWL4965 [=y] && NETDEVICES [=y] && WLAN [=y] && PCI [=y] && MAC80211 [=y] || IWL3945 [=y] && NETDEVICES [=y] && WLAN [=y] && PCI [=y] && MAC80211 [=y]
```

11. Exit and Save the configuration.
12. Copy the new .config file to the same location as the config.orig file.
13. Per the Yocto Project Kernel Development Manual, the next step is to run a diff to create the fragment:

```
$ diff -Nurp config.orig .config | sed -n "s/^\+//p" > abwlan.cfg
```

14. I took one more step to clean up the file so it only has the enabled items. Here is my file:

```
CONFIG_WEXT_CORE=y
CONFIG_WEXT_PROC=y
CONFIG_CFG80211=y
CONFIG_CFG80211_DEFAULT_PS=y
CONFIG_CFG80211_WEXT=y
CONFIG_WIRELESS_EXT_SYSFS=y
CONFIG_LIB80211=y
CONFIG_MAC80211=y
CONFIG_MAC80211_HAS_RC=y
CONFIG_MAC80211_RC_MINSTREL=y
CONFIG_MAC80211_RC_MINSTREL_HT=y
CONFIG_MAC80211_RC_DEFAULT_MINSTREL=y
CONFIG_MAC80211_RC_DEFAULT="minstrel_ht"
CONFIG_MAC80211_MESH=y
CONFIG_MAC80211_LEDS=y
CONFIG_IWLWIFI_LEGACY=y
CONFIG_IWL4965=y
CONFIG_NEW_LEDS=y
```

Copyright © 2013 Annabooks, LLC., All Rights Reserved.

```
CONFIG_LEDS_CLASS=y
CONFIG_LEDS_TRIGGERS=y
CONFIG_CRYPTO_AES=y
CONFIG_CRYPTO_ARC4=y
CONFIG_AVERAGE=y
CONFIG_WIRELESS_EXT=y
CONFIG_WEXT_SPY=y
CONFIG_WEXT_PRIV=y
CONFIG_LIB80211_CRYPT_WEP=y
CONFIG_LIB80211_CRYPT_CCMP=y
CONFIG_LIB80211_CRYPT_TKIP=y
CONFIG_IWMC3200TOP=y
CONFIG_IPW2100=y
CONFIG_IPW2200=y
CONFIG_LIBIPW=y
CONFIG_IWLAGN=y
CONFIG_IWL3945=y
CONFIG_IWM=y
CONFIG_CRYPTO_ECB=y
CONFIG_CRYPTO_MICHAEL_MIC=y
```

Adding Firmware Support

With the configuration fragment file created, the next step in the process is to integrate the file into the build process using a .bbappend file. The Linux Wireless wiki also points out that a firmware file is also needed to be placed in the /lib/firmware directory. The firmware directory is not enabled by default, but there is a Yocto Project recipe to include the firmware directory and the specific wireless firmware files.

1. Open local.conf
2. Add the following to the end of the file

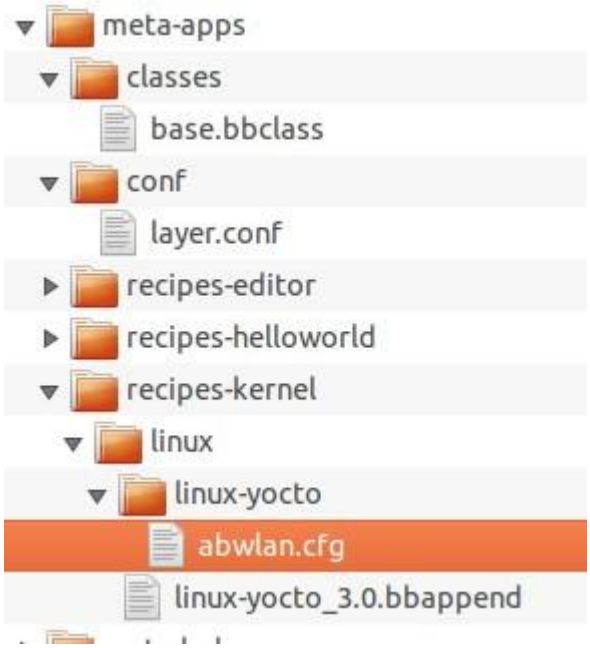
```
IMAGE_INSTALL_append +=” linux-firmware”
IMAGE_INSTALL_append +=” linux-firmware-iwlwifi-6000g2a-5”
```

3. Close and save the file

Integrating the Configuration Fragment into Yocto Project Build

The Yocto Project documentation suggests the creation of a new a layer to include the configuration fragment. I reused a meta-apps layer from my book *Open Software Stack for the Intel® Atom™ Processor* to include a recipes-kernel directory. There is a recipe to add the nano editor and the helloworld created with Eclipse. The meta-apps layer is created in the poky-<release>-<ver> folder with the rest of the meta- folders.

Copyright © 2013 Annabooks, LLC., All Rights Reserved.



1. The Layer.conf file is modified to look like the following, so the .bbappend files will be located by bitbake:

```
#We have a meta data layer directory, add to the BBPATH
BBPATH .= ":${LAYERDIR}"

#We have a recipe directory, add to BBFILES
BBFILES += "${LAYERDIR}/recipes-*/*/*.bb \
           ${LAYERDIR}/recipes-*/*/*.bbappend"

BBFILE_COLLECTIONS += "test"
BBFILE_PATTERN_test := "^${LAYERDIR}/"
BBFILE_PRIORITY_test = "5"
```

2. Under meta-apps, a folder structure is created for the new recipe: /recipe-kernel/linux/linux-yocto.
3. The abwlan.cfg file is copied to the /recipe-kernel/linux/linux-yocto folder.
4. In the /recipe-kernel/linux folder, a linux-yocto_3.0.bbappend file is created. I knew that kernel version 3.0 is being used since I built and tested the image once, as well as, reviewed the Cedar Trail BSP. The Yocto Project documentation provides a general solution for the linux-yocto_<ver>.bbappend file. If followed to the letter using `SRC_URI += "file://abwlan.cfg"`, the configuration fragment information will not get built in. The reason is that the Cedar Trail BSP's linux-yocto_3.0.bbappend has two different selections to the kernel package centered on support for a special video driver: one for PVR driver and the other for without PVR.

```
SRC_URI_cedartrail           =           "git://git.yoctoproject.org/linux-yocto-
3.0;protocol=git;bareclone=1;branch=${KBRANCH},meta,yocto/pvr;name=machine,meta
,pvr"

SRC_URI_cedartrail-nopvr        =        "git://git.yoctoproject.org/linux-yocto-
3.0;protocol=git;nocheckout=1;branch=${KBRANCH},meta;name=machine,meta"
```

5. For the configuration fragment to work, SRC_URI_cedartrail-nopvr must be appended for the settings to take effect. Add the following to the linux-yocto_3.0.bbappend file:

```
FILESEXTRAPATHS_prepend := "${THISDIR}/linux-yocto:"
SRC_URI_cedartrail-nopvr += "file://abwlan.cfg"
```

6. Save the file.
7. Update the projects bblayer.conf file to include the new layer:

Copyright © 2013 Annabooks, LLC., All Rights Reserved.

```
BBLAYERS ?= " \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-yocto \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-yocto-bsp \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-intel \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-intel/meta-cedartrail \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-apps \
  "
```

8. Running menuconfig again and searching for CONFIG_iwlwifi, you will see that the settings are in place.

   ```
   $ bitbake linux-yocto -c menuconfig
   ```

```
Symbol: IWLWIFI_LEGACY [=y]
Type  : tristate
  Selects: FW_LOADER [=y] && NEW_LEDS [=y] && LEDS_CLASS [=y] && LEDS_TRIGGERS [=y] && MAC80211_LEDS [=y]
  Selected by: IWL4965 [=y] && NETDEVICES [=y] && WLAN [=y] && PCI [=y] && MAC80211 [=y] || IWL3945 [=y] && NETDEVICES [=y] && WLAN [=y] && PCI [=y] && MAC80211 [=y]
```

If you have to make further changes, be aware that the abwlan.cfg configuration fragment is now included in the configuration. Running menuconfig will pick up the basic configuration and abwlan.cfg settings. When reviewing BSPs, you should be aware of any additional .cfg files provided.

Wireless Final Setup

Once the distribution is installed, the final step is to set up the wireless to connect to the router.

1. Open a terminal
2. To startup the wireless, run the following:

   $ ifconfig wlan0 up

3. Run ifconfig with no parameters, and you should see wlan0 in the listing.
4. Run iwlist wlan0 scan to locate the local router if you don't know what it is.
5. My router uses WPA and I want to have the network autostart on boot. Edit the /etc/network/interfaces file.
6. Add "auto wlan0" to the top of the wireless section.

```
:
:
# Wireless interfaces
auto wlan0
iface wlan0 inet dhcp
       wireless_mode managed
       wireless_essid any
       wpa-driver wext
       wpa-conf /etc/wpa_supplicant.conf
:
:
```

7. Edit the /etc/wpa_supplicant.conf file to add the ssid and passphrase

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
       ssid="myrouter"
       psk="mypassphrase"
}
```

If you prefer to encrypt the passphrase, run the wpa_passphrase to obtain a key.

```
$wpa_passphrase myrouter mypassphrase
```

Copyright © 2013 Annabooks, LLC., All Rights Reserved.

```
network={
       ssid="myrouter"
       #psk="mypassphrase"
       psk=ada209f2c69be76fb8d9270e7eb05b2a8625dc3cda31a3a1b2cca6642da9aa2c
}
```

8. Restart the network service

```
$ /etc/init.d/networking restart
```

Summary

The Yocto Project provides a strong cross development tool chain to develop for different processor architectures: ARM, MIPS, PowerPC, and x86. The integrated kernels provide a basic configuration set to support the basic system busses and drivers to boot the image. Configuration fragments allow additional settings to be configured for the kernel. As the integrated kernel is further developed using the Yocto Project, the configuration fragments can be re-used. As demonstrated with the Cedar Trail BSP, it is important to review the BSP being used for any unique recipes that deviate from the standard documentation.

Feedback welcome: http://www.annabooks.com/Contact.html

Copyright © 2013 Annabooks, LLC., All Rights Reserved.