Building and Installing Ti Wilink 8 driver in Linux Kernel Stack using Ti Linux SDK

Introduction

This document gives the steps that were followed to build and install the Ti wilink 8 drivers in the Linux kernel stack running on Beaglebone green wireless board.

The Beaglebone green wireless (BBGW) board consist of AM3358 microprocessors, based on the ARM Cortex-A8 processor. The green wireless board supports Wifi 802.11b/g/n standard through Wilink WL1835MOD module. Ti provides mesh implementation of 802.11s standard for the Wilink 8 series, which WL1835MOD is a part of.

The document gives details about the steps to follow to install wilink driver enabled linux kernel filesystem and u-boot on microSD card, boot the BBGW from microSD, setup and start a mesh interface network, and visualize the network on Windows PC using Ti Mesh Visualization Tool.

Important Links

Name	Link	Description
Ti Linux Essentials SDK	http://software-dl.ti.com/processor-sdk-linux/esd/AM335X/latest/index_FDS.html	Package contains source code for linux kernel, u-boot image and pre-built images, along with the cross-compiler tools and shell scripts to install and build using Ti Wilink modules.
Build Script and Package	https://gforge.ti.com/gf/download/user/1160/8008/sdk3-wilink8-am335x-v1.01.tar.gz	Contains script which runs the complete build process to

		create boot image and linux kernel filesystem with Ti wilink 8 drivers.
Linux Getting Started Guide	http://processors.wiki.ti.com/index.php/WiLink8_Linux_Getting_Started_Guide	Guide to build and install Wilink 8 drivers
Ti Mesh Scripts Help	http://www.ti.com/lit/an/swaa166/swaa166.pdf	PDF Document giving the configuration of mesh interface
Ti Mesh Visualization Toolkit (Windows only)	http://www.ti.com/tool/wilink-wifi_mesh_visualization_tool	Utility to display the mesh network on Windows PC.

Ti Wilink 8 SDK Build and Install Drivers

1) Download and Run the Ti Linux Essentials installer

Download the bin file from the following link: http://software-dl.ti.com/processor-sdk-linux/esd/AM335X/latest/index_FDS.html. The bin file is an installer that sets up the complete package including source codes, prebuilt images, cross-compiler tools and other shell scripts. Set permissions to run the file, and run the installer.

The default path /home/*user*/ti-processor-sdk-linux-am335x-evm-03.01.00.06 was used for installation. Substitute your username in place of "user" in above path.

For more details refer link, http://processors.wiki.ti.com/index.php/Processor_SDK_Linux_Installer.

2) Download the Build scripts

The build scripts to generate the u-boot image and linux filesystem with wilink 8 support can be downloaded from https://gforge.ti.com/gf/download/user/1160/8008/sdk3-wilink8-am335x-v1.01.tar.gz. The package also contains patches that need to be applied to default configuration files and device tree files for a specific board. The package contains patches for BBGW hardware as well. The files from the package must be extracted in the TI SDK folder and not in a separate folder.

3) Change the Kernel Build options in the config file

The config file to be used for BBGW board is "*tisdk_am335x-evm_defconfig*" and is located in ${SDK_DIR}/board-support/linux-vvvv+gitAUTOINC+xxxxx/arch/arm/configs directory.

The config file needs to be updated for BBGW and to enable Ti Wilink 8 modules and disable the default Wifi modules in Linux kernel. The list of options to be set or changed is given in the link: WL18xx Platform Integration Guide. The link contains changes to be made in the config file and device tree files (*.dts and *.dtsi).

The patches for configuration file was for older version of Ti Linux SDK. Hence, the changes in the config file were done manually. For this, the options needed to be changed (as given in WL18xx Platform Integration Guide) were copied into the "*tisdk_am335x-evm_defconfig*" file manually. The manually patched file must be stored in a separate location. The following lines in "*build-wilink.sh*" shell file must commented so that the patching of config file is not done.

git am ../../patches/kernel/0001-add-wilink8-configuration-to-sdk3-defconfig.patch || exit

git am ../../patches/kernel/0003-add-iptables-nat.patch || exit

git am ../../patches/kernel/0004-disable-wifi-in-kernel-build.patch || exit

Add a line to replace original "tisdk_am335x-evm_defconfig" file with manually patched configuration file.

Add the following line at the beginning of "*build-wilink.sh*" shell file:

```
export MANUALLY_PATCHED_DEFCONFIG_FILE="${SDK_DIR}/patches/tisdk_am335x-evm_defconfig"
```

Change the above path to point to the location where manually patched deconfig file is placed.

Add following lines immediately after the commented out lines (given above in green):

```
echo "Copying manually patched defconfig file."
rm -r  arch/arm/configs/tisdk_am335x-evm_defconfig
cp -v ${MANUALLY_PATCHED_DEFCONFIG_FILE} arch/arm/configs
```

4) Setup and Run the Wilink build shell script

Before running the "*build-wilink.sh*" script, some of the variables must be set correctly. The following table lists the variables and the values used:

Name	Value Set	Description
ROOTFS_ON_SD	0	This will create a full tar ball of the filesystem which is used the first time an SD card is being created by bin/create-sdcard.sh script
WILINK_MAINLINE	0	This will build the latest and full featured TI driver release
BBB_PLATFORM	3	Build for Seeed Studio BeagleBoneGreen Wireless
KERNEL_VER_BASE	Path of Linux filesystem source files	This variable must be updated to the correct name of the linux kernel filesystem source code files. It is

		located in the directory ${SDK_DIR}/board_support/ named as linux-vvvv+gitAUTOINC+xxxxx
UBOOT	Path to U-Boot source files	This variable must be updated to correct filename to build the boot image. It is located at ${SDK_DIR}/board_support/ directory under the name u-boot-yyyy.xx+gitAUTOINC+xxxx

The script uses "*.tar.gz" files, but the TI SDK script work with "*.tar.xz" files. So, update the script to extract the "tar.xz" file and make "tar.xz" file for the boot and rootfs outputs.

Update the line as follows:

```
# extract full image to get all wi-fi user space tools
  cd ${FS}
  sudo tar -xf ../filesystem/tisdk-rootfs-image-am335x-evm.tar.xz .
```

And the lines in the end as follows:

```
if [ ${ROOTFS_ON_SD} -eq 0 ]
then
  sudo tar -cJf ../${TAR_FS}/rootfs_partition.tar.xz *

sudo tar -cJf ../../${TAR_FS}/boot_partition.tar.xz *
```

Finally, run the script from command line

./build-wilink.sh

After successful build, there must be two tarball files in the ${SDK_DIR}/tar-sdk3-wilink-filesystem-bbgw folder:

1. boot_partition.tar.xz - Boot Partition tarball file containing MLO, U-Boot image and UEnv.txt files.
2. rootfs_partition.tar.xz - Rootfs partition containing Linux kernel along with the Wilink driver for mesh support.

5) Copy the Tarball files to microSD card

The two tar files mentioned above can be written to microSD card using the ${SDK_DIR}/bin/create-sd.sh shell script. TI recommends that the script should be run from its own directory. On the terminal, navigate to ${SDK_DIR}/bin directory. Run the sd card create script by typing:

```
$sudo ./create-sd.sh
```

Once the partitioning steps have been done select the custom file paths option

```
2) Enter in custom boot and rootfs file paths
```

and enter the FULL path to the built boot partition. eg **/home/user/ti-processor-sdk-linux-am335x-evm-03.00.00.04/tar-sensor-gateway-filesystem-e14/boot_partition.tar.xz**. At the next option select Kernel and device tree from rootfs.

1) Reuse kernel image and device tree files found in the selected rootfs

and the path to the rootfs partition. eg **/home/user/ti-processor-sdk-linux-am335x-evm-03.00.00.04/tar-sensor-gateway-filesystem-e14/rootfs_partition.tar.xz**

Refer to http://processors.wiki.ti.com/index.php/Processor_SDK_Linux_create_SD_card_script for details about create-sd.sh shell script.

6) Modify the mesh configuration file

Modify the contents of microSD card. When the memory card is plugged into PC or laptop, you should see two mounted partitions: "boot" and "rootfs". Navigate to the "rootfs" directory.

To run the mesh interface, the "*mesh_supplicant.conf*" file located in the rootfs linux kernel at directory "*usr/share/wl18xx/*" must be modified to add mesh interface configuration.

The following parameters need to be modified:

Name	Value	Description
max_peer_links	10	
mesh_max_inactivity	300	Timeout in seconds to detect mesh peer inactivity.
p2p_disabled	1	Must be set to 1 in order to disable p2p interface

Add new network profile. Search for network block, beginning with
# network block 	
#
OR, add it at the end.
	Copy paste from right ("Description" column)	network={
		ssid="MESH_NETWORK_SSID"
		mode=5
		frequency=2412
		key_mgmt=NONE
		psk="12345678"
		}

In the mesh_start.sh script located in the directory "*usr/share/wl18xx/*", change the IP address in the last line so that each device will have unique IP on the mesh network.

7) Boot BBGW from microSD card

Plug the microSD card into BBGW. Press and hold the USER/BOOT Button and plug in the mini-USB to USB connector to PC to power ON the BBGW. The BGW should boot from microSD card. This can be seen in serial console window. For details of serial cable connection and setup for serial debugging, visit link https://codechief.wordpress.com/2013/11/11/beaglebone-black-serial-debug-connection/

8) Start the mesh network

After boot-up, the BBGW provides ethernet over USB connection at IP 192.168.7.2. The PC or laptop should show a new local area connection with PC's IP 192.168.7.1 or any other IP other than 192.168.7.2.

If the network is not visible, hard reset the BBGW by plugging OUT the BBGW device and booting again to microSD card. The new network should be setup.

After a new local area connection is setup, login to BBGW via ssh by typing

$ssh root@192.168.7.2

Change directory to "*/usr/share/wl18xx/*". Start the mesh script by typing,

sh mesh_start.sh

You should see the mesh network created successful message at the end of the logs. When there are already devices connected to the network, the HWaddr of the connected devices are also printed.

The mesh interface can be seen by typing ifconfig of terminal. The mesh interface is named "mesh0" and should have a valid IP assigned.

Refer document http://www.ti.com/lit/an/swaa166/swaa166.pdf for more details and other configurations for mesh network.

9) Visualizing mesh network

Steps 5,6,7,8 must be repeated for each BBGW device on the mesh network. The steps for visualizing the mesh network on windows PC is given in the document http://www.ti.com/lit/ug/swru480/swru480.pdf.

He mesh was visualized using the "*Mesh Point as Router (Using DHCP)*" operation mode. The IP to connect is 192.168.7.2.

10) Flashing Boot and Rootfs from SD card to eMMC

Please follow the following steps to flash image to eMMC:

1. Run the BBB from the SD card (hold the USR button when powering the board up)

2. The eMMC is /dev/mmcblk1. Format it this way:
 2.01. fdisk /dev/mmcblk1
 2.02. o - this clears the existing partitions
 2.03. p - this lists all partition tables on the card (there should be none)
 2.04. n - create a new partition
 2.05. p - primary partition
 2.06. 1 - partition number
 2.07. 2048 - default value for the first sector
 2.08. +70M - last sector / partition size
 2.09. t - change the partition type (select partition 1)
 2.10. c - change the partition type to "W95 FAT32 (LBA)"
 2.11. a - set the bootable flag for the selected partition (1)
 2.12. n - create a new partition
 2.13. p - primary partition
 2.14. 2 - partition number
 2.15. hit Enter to choose the default (next available) value for the first sector
 2.16. hit Enter to choose the default (last) value for the last sector
 2.17. p - this lists all partition tables on the card (there should be two)
 2.18. w - write all the above changes to disk
 2.19. umount /dev/mmcblk1p1; mkfs.vfat -F 32 /dev/mmcblk1p1 - format the first partition
 2.20. umount /dev/mmcblk1p2; mkfs.ext3 /dev/mmcblk1p2 - format the second partition

Now you are formatted eMMC with 2 partitions. First is "W95 FAT32 (LBA)" and second is Linux. If you want to make this eMMC bootable you can make nwxt steps:

3. Copy the {MLO,u-boot.img,uEnv.txt} files to the first partition:

```
# mkdir boot
# mount /dev/mmcblk1p1 boot
# cp {MLO,u-boot.img,uEnv.txt} boot
# umount boot
```

4. Copy the root file system to the second partition:

```
# mkdir root
# mount /dev/mmcblk1p2 root
# tar -xJf rootfs_partition.tar.xz -C root
# umount root
```

5. Shutdown the BBB, remove the SD card and start it from the eMMC.