Fusion
Quick Start Guide

All Releases

PRELIMINARY

11 March 2009

ATHEROS™
COMMUNICATIONS

Table of Contents

1 Introduction

This document provides a quick reference for getting started with the Atheros reference platforms using the Fusion driver baseline. The Fusion driver is combination of both Station and AP functionality into a single driver baseline that is Operating System agnostic, requiring only an OS translation layer to be hosted to any particular OS.

1.1 References

Specific release information is contained in the Release Notes provided with the distribution. This defines the features, capabilities, and any known issues with the provided release.

Operating instructions are provided in the ***Fusion Linux User's Guide***. This includes documentation of all commands and settings affecting the driver, as well as the use of configuration scripts provided with the release.

1.2 Description

Atheros releases its drivers along with reference platforms, allowing OEM/ODM partners to integrate the driver into their product releases. The reference platforms provide a test vehicle for the manufacturer to verify changes against a known baseline. The platforms can also be used as an example design that designers can start with, adding or removing features as required for the specific design targets.

The reference designs consist of two major components; the reference platform or "Platform Board", and a WLAN module to be installed in the mini-PCI slots in the platform boards. Various combinations of platform boards and WLAN modules are supported, in single radio or dual concurrent radio configurations.

2 Supported Hardware

2.1 WLAN Modules

Table 1 WLAN Modules Supported by the Linux Fusion Driver

Module	Atheros Chipset	Description
MB-55	AR2417	Single band 802.11 b/g chipset
MB-71	AR5416	Single band 802.11 b/g/n chipset, first generation 11n
MB-72	AR5416	Dual Band 802.11 a/b/g/n chipset, first generation 11n
MB-81	AR9160/AR9161	Single band 802.11 b/g/n chipset
MB-82	AR9160/AR9161	Dual band 802.11 a/b/g/n chipset with DFS 2.0 support
MB-91	AR9822	Single Band 802.11 b/g/n second generation 11n
MB-92	AR9823	Dual Band 802.11 a/b/g/n second generation 11n

2.1.1 Reference Platforms

The main reference platforms provided by Atheros are the PB-42 and PB-44, single and dual radio models respectively. In addition, support is provided for the Cavium CN30XX/CN50XX platform boards provided by Cavium Networks.

2.1.1.1 PB-42

The PB-42 is a single module, low cost reference design that uses the Atheros AR7100 Series CPU as the main processor. It comes with a 100 BaseT WAN and LAN interface module, on-board SDRAM, and 8 MB of SPI flash. See Figure 1.

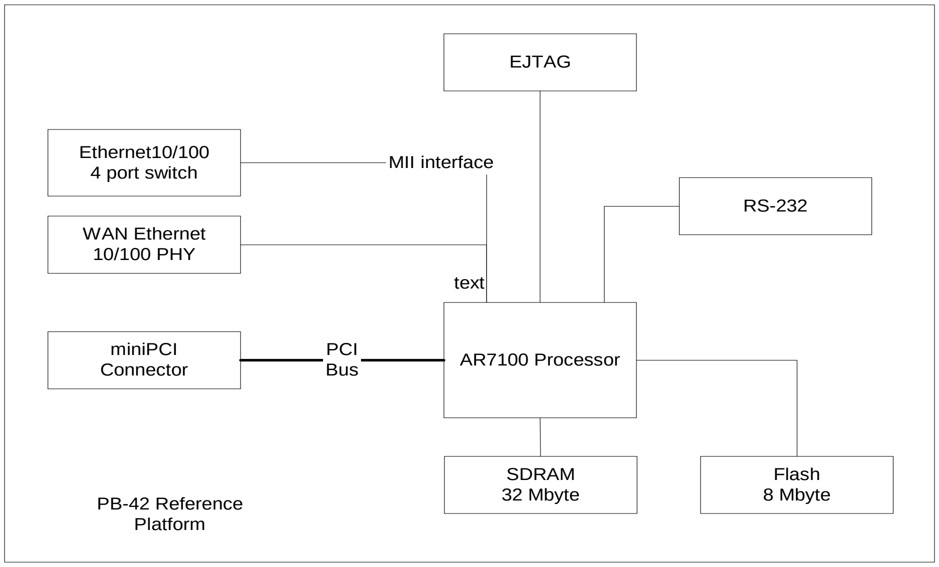


Figure 1 PB-42 Block Diagram

The processor is configured in Big Endian mode, thus the driver is configured to operate in Big Endian mode. The design has a single mini-PCI slot that will host the module of choice. The PCI bus is configured to run at 33 MHz in 32-bit mode, although it can be run at 66 MHz with an alteration to the boot ROM code (provided). The two Ethernet interfaces are serviced by the two MII ports provided by the AR7100. One of the MII interfaces is connected as WAN port to Port 4 of the ADM Ethernet Switch. Port 4 connects to the PHY transparently without the traffic getting switched by the switch. Hence WAN port is completely isolated from the LAN ports. The other MII is connected to Port 5 on the ADM Ethernet Switch. This port is connected to the 4 LAN ports which are used for LAN traffic. 32 MB of SDRAM are integrated onto the board, and an 8 MB SPI flash device is used for permanent.

2.1.1.2 PB-44

The PB-44 board is a larger evaluation platform that has many features, and provides two MiniPCI slots that can be used concurrently. This platform supports the “dual concurrent” mode of operation, where two radios are active concurrently. This board is based on the AR7161 CPU running at 680 MHz, and includes SPI Flash, stereo audio interfaces, Telephony interfaces, and a four-port GB Ethernet switch. See

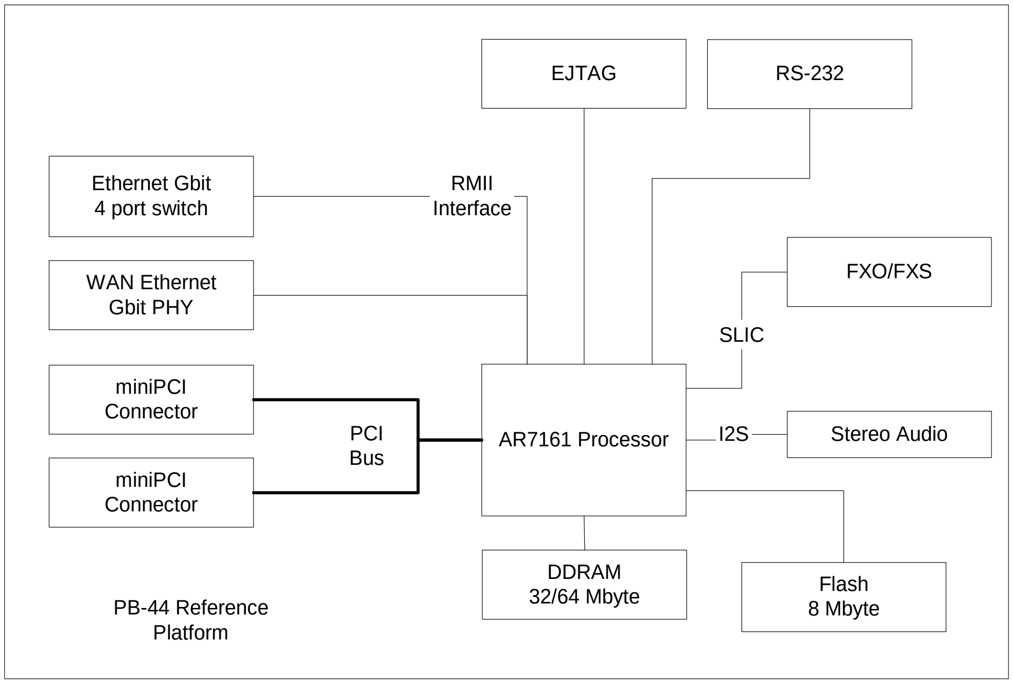


Figure 2 PB-44 Block Diagram

As with the PB-42 board, the PB-44 processor is configured in Big Endian mode, thus the driver is configured to operate in Big Endian mode. The design has two mini-PCI slots that will host the modules of choice. The board can be run with either a single module in either slot, or with both slots populated. The PCI bus is configured to run at 33 MHz in 32 bit mode, although it can be run at 66 MHz with an alteration to the boot ROM code (provided).

The AR7161 SOC on the PB-44 board has two Ethernet MAC units. These maybe configured for different modes. The PB-44 connects MAC0 to a VSC8601 gigabit PHY. MAC1 is connects to a VSC73XX gigabit switch. Since both MACs are configured to gigabit devices they use RGMII interfaces. MAC0 uses MDIO for communications between the AR7161 and the PHY. This is designated the WAN port and is accessible from the operating system as ‘eth0.’ MAC1 is connected to a Vitesse 5 port switch. Four ports are connected to RJ45 connectors and the fifth port is connected to MAC1. This is designated as the LAN port and is accessible from the operating system as ‘eth1’.

Note: The design of the MAC interface in the AR7161 requires that MAC0 is initialized before MAC1 may be used. Consequently designs that would use only a single switch would connect to MAC0.

Up to 64 MB of DDRAM are integrated onto the board, and an 8 MB SPI flash device is used for permanent. Only 32 MB are used by the kernel configuration. This memory is run at 340 MHz rate when the CPU is operating in the 680 MHz mode.

The PB-44 provides both stereo audio and telephony interfaces. The audio interface provides operation for 8 or 16 bit data via the I2S bus interface. The audio chip is an AKM4363 and is capable of stereo operation with the AR7161. The pb44 provides a stereo input and headphone output. The AKM4643 may be used in slave mode or in master mode. In slave mode it can use the left/right signal, the bit clock, or an external oscillator to synchronize the conversion of audio data. When the AKM4363 is in master mode it generates the left/right signal and bit clock.

The AR7161 may be used in master mode with the AKM4363 in slave mode; or slave mode with the AKM4363 in master mode. When the AR7161 is in slave mode it requires a minor change in timing to handle stereo signals. This is provided by a flip-flop on the PB44. When the AR7161 is in master mode the flip-flop must be disabled. Software presets are provided that operates at stereo; 8/16b 48 KHz stereo or mono. A header is provided that allows access to the stereo I2S signals provided or needed by the AR7161. When this is used the onboard AKM4363 must be disabled by changing a configuration resistor.

The AR7161 provides a SLIC high speed serial interface similar to I2S to connect to standard telephony cards. The interface allows the use of optional FXO and FXS devices via three 2x5 headers and a 1x2 header. The four connectors may be used to connect directly to a Silicon Laboratories FXO and FXS evaluation cards. The cards are stackable. Two FXO cards may be stacked to connect to analog telephones. With a small change of a configuration resistor on the FXS subscriber card; both FXO and FXS cards may be used at the same.

2.1.1.3 AP93

The AP93 access point is based on the new AR7240 processor chipset. It uses a PCIe interface to communicate with the WLAN chipset, increasing overall speed capabilities over PCI. This is a single band 11ng radio that provides an extremely low cost platform. It also supports USB interfaces, making it suitable for NAS applications.

This uses the newer u-boot based boot monitor programs configured for smaller footprint.

2.1.1.4 AP94

The AP94 access point has very similar design characteristics to the PB-44 and PB-42 boards. The board uses the AR7161 processor running at 680 MHz, 400 MHz DDR on a 16 bit interface, and the Atheros 8216 10/100 PHY switch. It has a single USB port, an AR9220 for 2.4 GHz WLAN, and an AR9223 for 5.0 GHz support. This provides a dual concurrent architecture in a cost down platform.

This uses the newer u-boot based boot monitor programs configured for smaller footprint.

2.1.1.5 Cavium 30XX/50XX Octeon

The Cavium Octeon development kit is an evaluation platform obtained from Cavium. Please see the Cavium SDK for details on the configuration and startup of the reference platform.

3 Build and Install Process

The process of unpacking, building, and installing the driver is identical for all standard platforms. For the Cavium platform, the binaries are provided with the Cavium SDK (or as a patch to the SDK). For users that have signed a Technology Licensing Agreement (TLA) with Atheros, source distributions are provided and can be used as a basis for a custom driver.

3.1 Standard Platform

The instructions in this section pertain to the standard distribution for the PB reference designs. Instructions for building for a specific design are not provided, but may be deduced from the standard instructions.

3.1.1 Requirements

The main requirement for building the standard distribution is having a Linux based development platform with at least 1 GB of free disk space, and a working GCC compiler tool chain. These procedures have been tested on a Fedora Core 8 machine, and on older machines with Fedora Core 4.

The distribution contains all tools required to build the bootloader, kernel, and jffs2 image to be loaded onto the flash. It is assumed that the reference platform provided already has a version of Redboot installed, and that the network connections are operational

Further, a tftp server is required, preferably on the development platform. This server is used by the reference platform to download all file images required.

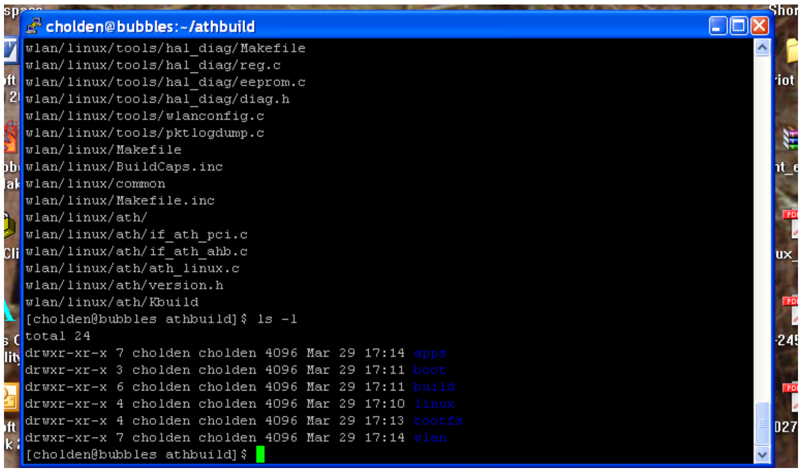
3.1.2 Unpacking and Build Process

The LSDK distributions are normally downloaded from the Atheros FTP site as enabled by your account representative. Two files are provided: **LSDK-*rel*.tgz** and **LSDK-WLAN-*rel*.tgz**, where "***rel***" is the combination of board and release, such as "pb42fus7.0.0.358". The LSDK file contains all of the files associated with the BSP, Kernel distribution, applications provided, build tool source, and the build system. The LSDK-WLAN file contains the WLAN related sources. This file can be used if the user is building for another system, and is not interested in the BSP, build tools, etc. for the reference platform

Select a directory to open the build into, and use the following procedure. This procedure assumes that "athbuild" is the directory to unpack into, and that the LSDK files are located in the directory immediately above. Modify the procedure according to your configuration:

```
#cd athbuild
#tar –xzvf ../LSDK-pb42fus-7.0.0.358.tgz
(large number of files unpack)
#tar –xzvf ../LSDK-pb42fus7.0.0.358.tgz
(smaller number of files unpack).
```

Once this is completed, you should see the directory structure:



```
wlan/linux/tools/hal_diag/Makefile
wlan/linux/tools/hal_diag/reg.c
wlan/linux/tools/hal_diag/eeprom.c
wlan/linux/tools/hal_diag/diag.h
wlan/linux/tools/wlanconfig.c
wlan/linux/tools/pktlogdump.c
wlan/linux/Makefile
wlan/linux/BuildCaps.inc
wlan/linux/common
wlan/linux/Makefile.inc
wlan/linux/ath/
wlan/linux/ath/if_ath_pci.c
wlan/linux/ath/if_ath_ahb.c
wlan/linux/ath/ath_linux.c
wlan/linux/ath/version.h
wlan/linux/ath/Kbuild
[cholden@bubbles athbuild]$ ls -l
total 24
drwxr-xr-x 7 cholden cholden 4096 Mar 29 17:14 apps
drwxr-xr-x 3 cholden cholden 4096 Mar 29 17:11 boot
drwxr-xr-x 6 cholden cholden 4096 Mar 29 17:11 build
drwxr-xr-x 4 cholden cholden 4096 Mar 29 17:10 linux
drwxr-xr-x 4 cholden cholden 4096 Mar 29 17:13 rootfs
drwxr-xr-x 7 cholden cholden 4096 Mar 29 17:14 wlan
[cholden@bubbles athbuild]$
```

When the source files are unpacked, the build process can begin. The build directory contains the make file for building all components required. The source distribution will contain the required files for the build of your choice. Trying to build for a different reference platform may not work, as the files required may not be included in the LSDK.

The build system will create two new directories: images and rootfs.build. The images directory will receive the final output of the build process that is used to update the platform board. The Redboot images, the Linux kernel image in compressed format, and the jffs2 filesystem are all copied into this directory, under the subdirectory for the specific reference platform. The **rootfs.build** directory is used as an install target to create the jffs2 filesystem. All components that are built are installed into this directory, and the final step is to run **mkfs.jffs2** on this directory to create the filesystem image.

To do a full system build, use the following procedure:

```
#cd build
# make BOARD_TYPE=board
```

where ***board*** is the specific board type of your reference platform, either **pb42fus**, **pb44fus, ap93fus,** or **ap94**. This will generate a full build of all components, and the cross tools required to build for the reference platform. Note that on subsequent builds the tools will NOT be rebuilt (this is a long process; the first build can take well over an hour).

After a full build is performed, components can be rebuilt as required. To build a specific component, you will use the command:

```
# make target BOARD_TYPE=board
```

where target is the specific target of choice. If you are unsure if the rootfs.build directory is properly populated, run the full build to erase and recreate the image. See Table 2.

Table 2 Main Available Targets

Target	Builds
fusion_build	Rebuilds all driver files, and regenerates the module.ko files. Copies new files into the rootfs.build directory. This will also rebuild the jffs2 filesystem.
busybox_build	Rebuilds the busybox component, and installs into the rootfs.build directory.
hostapd_fus	Rebuilds the hostapd application provided to support WPA encryption. Installs into rootfs.build directory.
toolchain_build	Rebuilds the gcc cross build tools provided. These are left in the build area, and are **not** put into the rootfs.build system
redboot_fusion	Rebuilds the Redboot bootloader, and copies the image into the images directory.
enet_build	Rebuilds the Ethernet driver, and copies the ar7100.ko module into the rootfs.build directory.
kernel_build	Rebuilds the Linux kernel. The result is copied to the images directory
fus_supplicant	Rebuilds the WPA supplicant application provided, and installs into the rootfs.build.
fusion_hal_build	Rebuilds the HAL portion of the driver only. Does NOT regenerate the .ko modules, but will create the uuencode file that can be distributed.
uimage	This will build the AP94 Linux kernel compressed image format suitable for using with u-boot.
uboot_fusion	This builds the u-boot boot monitor program.
art_mod	Rebuilds the Advanced Radio Test (ART) module provided with Atheros modules.

3.1.3 Installation

To perform a software update on the system, the following are required:

- **A PB42 or PB44 board with appropriate modules (see supported modules), or the appropriate AP93/94 boards**
- **A serial port module for the JP1/UART connector**
- **A server system with a TFTP server**
- **A terminal system with terminal emulation software, such as hyperterm or Minicom.**
- **A straight through serial cable, male to female.**
- **An Ethernet cable between the server system and the target AP board**

Connect the Ethernet ports on the server system and the AP WAN port. Connect the serial cable between the terminal system and the AP, using the adaptor board. Ensure that Pin 1 on the adaptor board is aligned with pin 1 on the AP (pin 1 has a square pad). The serial port should be set to 115200 baud, 8 data bits, no parity, 1 stop bit, no handshake.

3.1.3.1 Redboot Boot Monitor Update

This procedure is provided to update the boot monitor with a newer version that is generated. Note that the source code for the boot monitor is included in your distribution.

WARNING
Incorrect implementation of this procedure can cause board failure due to erasing the boot monitor from the Flash. If this occurs, the board will have to be returned to the factory to be reloaded, or will have to be reloaded via download from an EJTAG emulator.

1) **Boot with existing Redboot and halt it from booting into Linux.**

2) **Load new redboot.rom version into memory. The filename will be either redboot.pb42.rom or redboot.pb44.rom, based on the board you are updating.**
```
RedBoot> load -r -v -b 0x80500000 redboot.rom -h <tftp server ip addr>
```

3) **Write Redboot to flash**
```
Redboot> fis write -b 0x80500000 -f 0xbf000000 -l 0x40000
```

4) **Reboot the board with new Redboot and break into the monitor with <ctrl-c> before booting Linux**
```
Redboot> reset
```

5) **Now reformat the flash with new Redboot running.**
```
Redboot> fis init -f
```

6) **Initialize the configuration on the board using the following values (user entries underlined)**
```
RedBoot> fconfig –i
Initialize non-volatile configuration - continue (y/n)? y
Run script at boot: true
Boot script:

Enter script, terminate with empty line
>> fis load -d vmlinux
>> exec
>>
Boot script timeout (1000ms resolution): 3
Use BOOTP for network configuration: false
Gateway IP address:
Local IP address: <Your IP address>
Local IP address mask: <Your required netmask>
Default server IP address: <TFTP server IP address>
Console baud rate: 115200
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
Update RedBoot non-volatile configuration - continue (y/n)? y
... Erase from 0xbf7e0000-0xbf7f0000: .
... Program from 0x80ff0000-0x81000000 at 0xbf7e0000: .
```

Because this procedure erases the flash device, you will also have to perform the software update process in the following section. This is the normal process used to update the driver or kernel when changes are made.

3.1.3.2 U-Boot Monitor Update

Updating the u-boot monitor can be accomplished either using an EJTAG emulator or using the u-boot previously installed.

WARNING
Incorrect implementation of this procedure can cause board failure due to erasing the boot monitor from the Flash. If this occurs, the board will have to be returned to the factory to be reloaded, or will have to be reloaded via download from an EJTAG emulator.

3.1.3.2.1 Via BDI

A utility program build/utils/spi.bin has been provided to assist in programming the initial image using an EJTAG emulator. This file is copied to the /tftpboot directory for use in the following procedure. These instructions are for use with a BDI2000 emulator system; please alter to match the EJTAG system being used.

1) **Power up the board, and attach the BDI2000**

2) **Reset and prepare**
```
BDI> reset
BDI> rmcp0 13 0
BDI> rmcp0 12 0
```

3) **Load the u-boot image and the spi.bin image**
```
BDI> load 0xa0001100 u-boot.bin bin
BDI> load 0xa0000500 spi.bin bin
```

4) **Set a breakpoint, and run the program**
```
BDI>b 0xa000053c
BDI>go
```

5) **Wait for Breakpoint**
When the breakpoint is hit, the image is programmed. Remove EJTAG connector and power cycle the board.

3.1.3.2.2 Via u-boot

If u-boot has been previously installed, the following procedure can be used to update the u-boot on the board.

1) **Power up the board, and hit any key to stop boot countdown**

2) **Load the new u-boot image**
```
ar7100> tftpboot 0x80010000 u-boot.bin
```

3) **Erase the flash sectors, and copy the new u-boot into flash**

   a. **For AP93**
```
ar7100> erase 0xbf000000 +0x40000
ar7100> cp.b 0x80010000 0xbf000000 0x40000
```

   b. **For AP94**
```
ar7100> erase 0xbf000000 +0x30000
ar7100> cp.b 0x80010000 0xbf000000 0x30000
```

4) **Reset the board**
```
ar7100> reset
```

3.1.3.3 Kernel/Driver Update

The Kernel and Driver image update is performed using the boot monitor commands (Either Redboot or u-boot) through the serial console, and tftp file transfers. The same procedure is used for all boards, with variation based on the boot monitor used.

The file **vmlinux.info**, included in the **images** directory of the LSDK, will have the link address (always 0x8006) and the entry address (0x80258000 for PB42, 0x8029e000 for PB44) of the Linux kernel image. This file is also generated by the build process, and should be checked to verify the value of the –e argument for loading the Linux kernel. The values provided below are for the standard LSDK.

The caldata sector is now used to store the AP configuration info. See the *Linux Fusion User's Manual* for details.

1) **Power up the board, and hit <ctrl-c> to break into monitor**
   a. **For PB42 & PB44**
      Hit <ctrl+c> before the boot process starts
   b. **For AP94**
      Hit any key before the countdown terminates
2) **Load with new Linux image.**
   a. **For PB42**
      ```
      Redboot> load -r -v -b 0x80500000 vmlinux.bin.gz -h <tftp server ip>
      Redboot> fis create -b 0x80500000 -e 0x80258000 -r 0x80060000 -l 0x100000 vmlinux
      Redboot> fis create -b 0x80500000 -e 0x0 -l 0x10000 caldata
      ```
   b. **For PB44**
      ```
      Redboot> load -r -v -b 0x80500000 vmlinux.bin.gz -h <tftp server ip>
      Redboot> fis create -b 0x80500000 -e 0x8029e000 -r 0x80060000 -l 0x120000 vmlinux
      Redboot> fis create -b 0x80500000 -e 0x0 -l 0x10000 caldata
      ```
   c. **For AP93**
      ```
      ar7100> tftpboot 0x80010000 vmlinux.gz.uImage
      ar7100> erase 0xbf300000 +0xf0000
      ar7100> cp.b 0x80010000 0xbf300000 0xf0000
      ```
   d. **For AP94**
      ```
      ar7100> tftpboot 0x80010000 vmlinux.gz.uImage
      ar7100> erase 0xbf640000 +0x120000
      ar7100> cp.b 0x80010000 0xbf640000 0x120000
      ```
3) **Load new jffs2 filesystem**
   a. **For PB42**
      ```
      Redboot> load -r -v -b 0x80500000 pb42fus-jffs2 -h <tftp server ip>
      Redboot> fis create -b 0x80500000 -e 0x0 -l 0x600000 filesystem
      ```
   b. **For PB44**
      ```
      Redboot> load -r -v -b 0x80500000 pb44fus-jffs2 -h <tftp server ip>
      Redboot> fis create -b 0x80500000 -e 0x0 -l 0x600000 filesystem
      ```
   c. **For AP93**
      ```
      ar7100> tftpboot 0x80010000 ap93fus-jffs2
      ar7100> erase 0xbf050000 +0x2b0000
      ar7100> cp.b 0x80010000 0xbf050000 0x2b0000
      ```
   d. **For AP94**
      ```
      ar7100> tftpboot 0x80010000 ap94-jffs2
      ar7100> erase 0xbf040000 +0x600000
      ar7100> cp.b 0x80010000 0xbf040000 0x600000
      ```
4) **Reset the board. Board will now boot up into Linux.**
   for all, issue the "reset" command

3.1.3.4 Setting the MAC Addresses

To set MAC addresses, each boot monitor will have the **progmac** command implemented

1) **Load new jffs2 filesystem**

   a. **For PB42/PB44**
      ```
      Redboot> progmac <sernum>
      ```
      <sernum> is the last 4 decimal digits of the serial number

   b. **For AP93, AP94**
      ```
      ar7100> progmac <sernum>
      ```
      <sernum> is the last 4 decimal digits of the serial number

3.1.3.5 Board Startup

Once the software is loaded, the AP can be started using the methods described in the user's manual. These consist of either command line mode, or a web interface. Note that default values are created based on the settings in the **/etc/ath/apcfg** file.

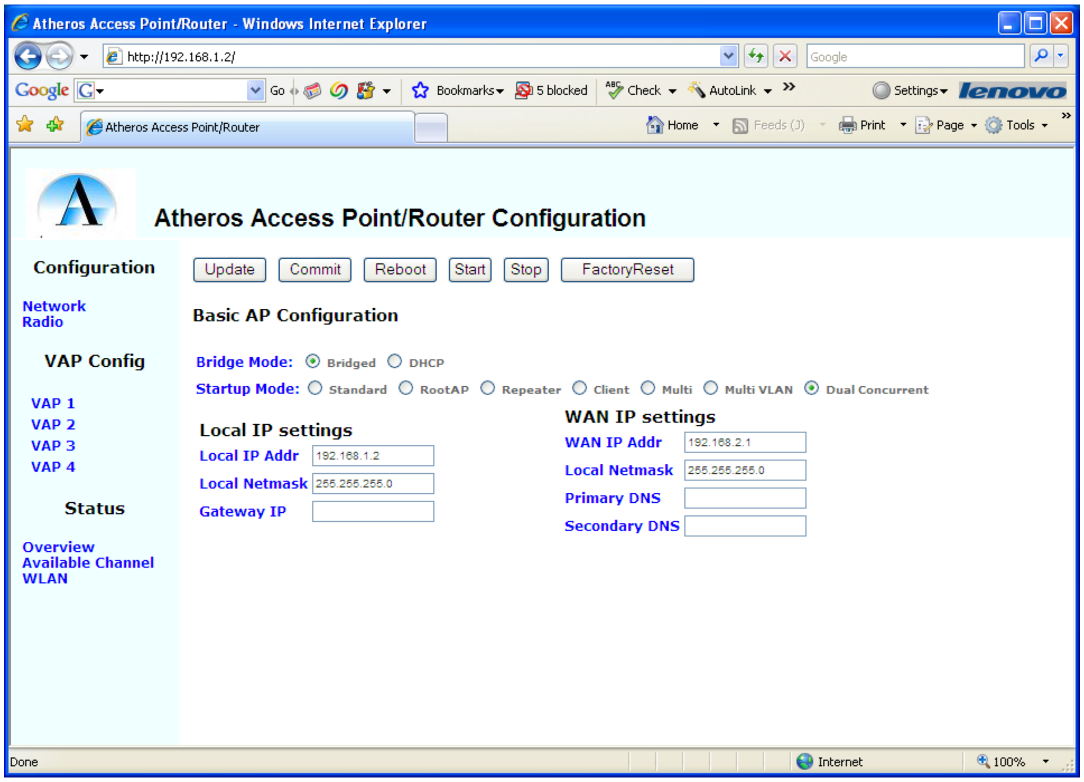
3.1.3.5.1 Command Line

Board startup using the command line consists of setting configuration variables using the "cfg" tool, and then running the **apup** script. See the Fusion Linux User's Manual for details on the available variables and settings. An example of setting the start mode, SSID, channel, Mode, and starting the AP are as follows:

```
#cfg –a AP_STARTMODE=standard
#cfg –a AP_SSID=testSSID
#cfg –a AP_PRIMARY_CH=11
#cfg –a AP_CHMODE=11NGHT20
#apup
```

3.1.3.5.2 Web Interface

Board startup using the web interface is performed through the controls on the web page. The main page is shown below:



Atheros Access Point/Router Configuration

Configuration

Network
Radio

VAP Config

VAP 1
VAP 2
VAP 3
VAP 4

Status

Overview
Available Channel
WLAN

Update | Commit | Reboot | Start | Stop | FactoryReset

Basic AP Configuration

Bridge Mode: [✓ Bridged] DHCP
Startup Mode: Standard RootAP Repeater Client Multi Multi VLAN [✓ Dual Concurrent]

Local IP settings

Local IP Addr	192.168.1.2
Local Netmask	255.255.255.0
Gateway IP	

WAN IP settings

WAN IP Addr	192.168.2.1
Local Netmask	255.255.255.0
Primary DNS	
Secondary DNS	

The page selection is done through links on the left side of the page, and parameters are entered in the main portion of the page. Each of the parameters is labeled with a description, and default selections are provided on bootup. The control buttons on the page are replicated on each of the configuration pages, and have the following meanings:

Update	Accept the parameters as entered on the current page, but do not commit to flash.
Commit	Commit all changes to flash, saving through boot cycles
Reboot	Reset and restart the AP
Start	Start the AP software
Stop	Stop the AP software
Factory Reset	Set the parameters to the original factory settings. These are documented in the *Fusion Linux User's Manual*.

3.2 Cavium Reference Platform Support

Note that for Cavium boards, binaries are provided as part of the Cavium SDK. For Cavium software updates, please follow the procedure outlined in the Cavium SDK. This section is for users who have signed the TLA with Atheros and have received a source distributions.

3.2.1 Requirements

The main requirements for building for the Cavium reference platforms are as follows:

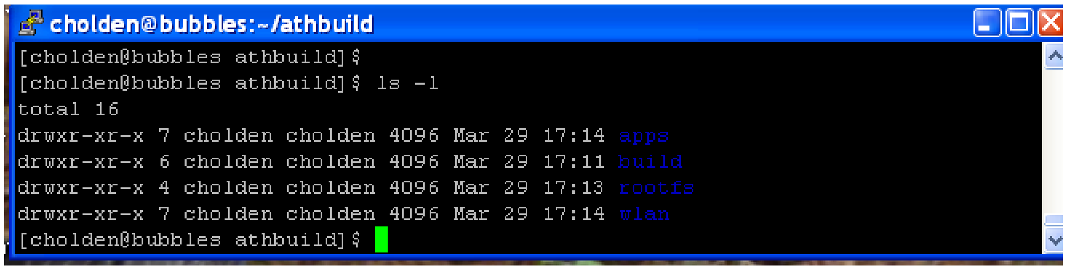
- **Linux build machine with sufficient disk space**
- **Cavium SDK-1.5.0 with patches installed on the build machine.**
- **Flash writer provided by Cavium as part of the reference platform kit.**

3.2.2 Unpack and Build Process

The unpack process is very similar to the standard process, although many fewer files are unpacked. The only file provided is **LSDK-WLAN-CaviumFus.tgz**, which contains all source required. Select an appropriate directory to build into, and unpack the contents of the LSDK-WLAN file:

```
#mkdir athbuild
#tar –xzvf ../LSDK-WLAN-CaviumFus.tgz
```

You should see the following directory structure:



```
[cholden@bubbles athbuild]$
[cholden@bubbles athbuild]$ ls -l
total 16
drwxr-xr-x 7 cholden cholden 4096 Mar 29 17:14 apps
drwxr-xr-x 6 cholden cholden 4096 Mar 29 17:11 build
drwxr-xr-x 4 cholden cholden 4096 Mar 29 17:13 rootfs
drwxr-xr-x 7 cholden cholden 4096 Mar 29 17:14 wlan
[cholden@bubbles athbuild]$
```

To prepare the environment to build for the Cavium reference platform, you must configure the Cavium environment. Assuming you have the CN5020 board, issue the following commands:

```
#pushd /usr/local/Cavium_Networks/CN3XXX-SDK
#source env-setup OCTEON_CN50XX –runtime
#popd
```

Finally, run the build system to generate the file image:

```
#cd build
#make BOARD_TYPE=CaviumFus
```

The resulting image will be put in the images/CaviumFus directory under the build directory.

3.2.3 Installation

To install the new driver, you will have to mount the CF filesystem and write the new driver to the appropriate partition. Ensure the CF writer unit is correctly connected to the host system and operates properly. Load the Compact Flash unit with the Linux boot image and the main filesystem as specified by the Cavium SDK instructions. Assuming mount points have been created at /mnt/pt0 for partition 0 and /mnt/pt1 for partition 1, mount the filesystem:

```
#mount /dev/sda2 /mnt/pt1
```

When the mount is completed, install the Atheros drivers built in the previous section to the filesystem:

```
#cd /mnt/pt1
#tar –xvf <path to build directory>/images/CaviumFus/cav_root.tar
```

Unmount the CF:

```
#umount /dev/sda2
#umount /dev/sda1
```

Remove the CF, and install into the Cavium platform per the SDK notes. The system should boot up into Linux as normal. To use the Atheros driver, see the release notes for configuration and operating instructions.

3.2.4 Startup

Start the board using the standard process outlined in the Cavium SDK. Once into Linux, you can use the **apup** script. Note that you will have to include the /etc/ath in the PATH variable:

```
# export PATH=$PATH:/etc/ath
```

Please refer to the ***Fusion Linux User’s Manual*** and the ***Release Notes*** for details on how to configure the AP using the environmental variables and the **apup** script.