Board Files Modification Guide

Application Note

This application note helps you work with custom board files. Board files help software adapt to a specific hardware configuration.

This document will be useful to an embedded software engineer who is tasked with bringing up a custom board based on IP7K Processor. This reader needs several skills: knowledge of hardware components, ability to read hardware schematics, along with ability to work with software. Use this document as a helper and not a definitive guide to bring up any board.

To configure an application so that it works with a custom board, you must:

1. Integrate board and application configuration files into the build process.
2. Create new board and application files.

This application note guides you through both of these steps.

Not included in this application note are instructions for configuring Das U-Boot.

For the purpose of discussion, suppose that your company is developing a board known as "Design320" which uses a Ubicom IP7160 processor. The board is intended for router-gateway (RGW) products. Identifiers used in the following examples will be derived from these names. Any resemblance to existing products is purely coincidental. Let the configuration files have the following names:

- Ultra board file: `IP7160_design320.brd`
- Ultra project file: `ultra7k_design320.lpj`
- Linux board file: `board-ip7160_design320.c`

Application Note

Contents

Application Note

1 About Board Files

The board files in the Ubicom Linux distribution are text files. For each board, there are two files: one in Ultra (Ubicom boot loader) section and another in Linux section.

Board files in Ultra area are ***.brd** files. They are located in directory **/ultra/boards.** The ***.lpj** files reside in the **/ultra/config** directory. Linux board files are located in directory **linux-2.6/arch/Ubicom32/mach-ip7k/**.

For the purpose of this document, all directories are relative to the root directory of the Ubicom Linux distribution.

Be sure to start with a clean distribution.

2 Create New Board Files

The board files help software to understand the hardware. The following files need to be created

- Ultra board file: **IP7160_design320.brd**
- Linux board file: **board-ip7160_design320.c**
- Project configuration file **ultra7k_design320.lpj**

2.1 Information Needed to Start

To create the board files you will need the following:

1. Schematics of the new board
2. Pin map document. (Derived from schematic, detailing how each of the IP7K pins is connected or used. This will help in quickly identifying the controlling pins, such as GPIO, LED, MDC, etc)
3. Information about various parts on the board, such as details of DDR, Flash, Ethernet, and other interfaces.

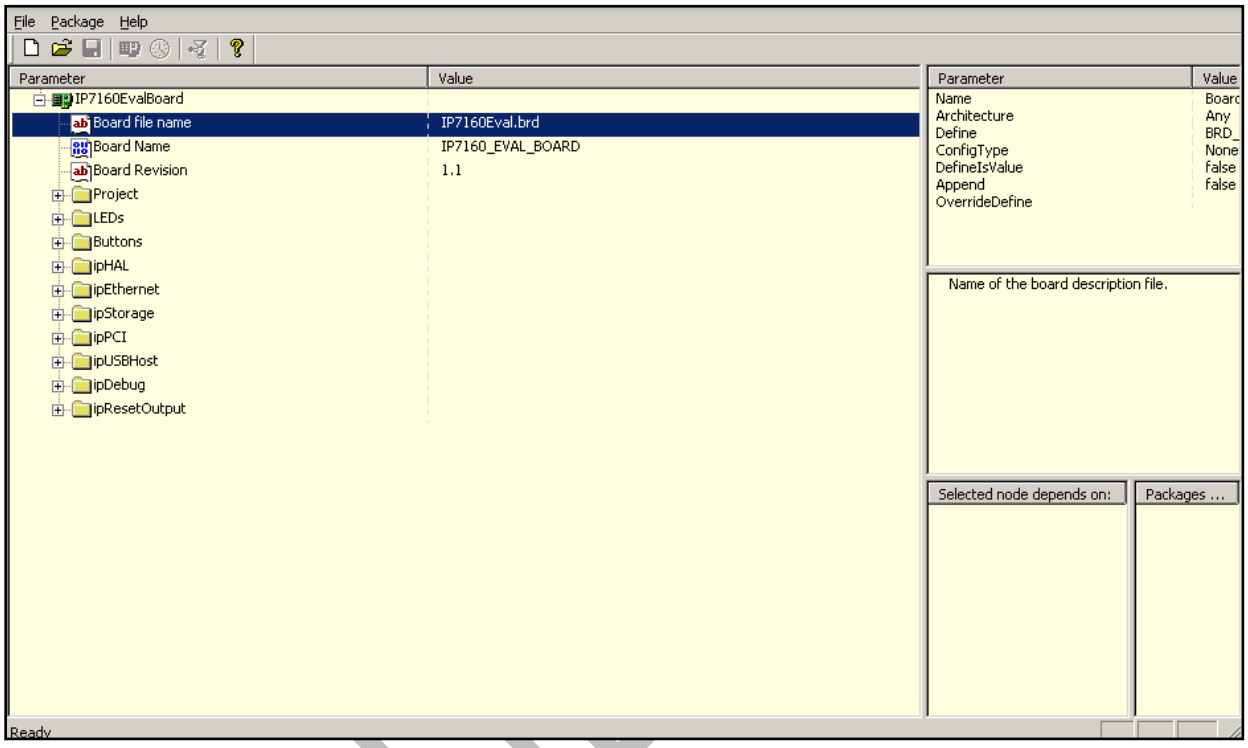
2.2 Tools used

Use any text editor, such as **gedit**, for creating the **board-ip7160_design320.c** file. For **.brd** and **.lpj** files, use the combination of a text editor and **ConfigTool.exe**, which resides in directory **ultra/tools/bin/win32**. **Note: ConfigTool**.exe can be launched using **wine**. (Wine is a tool that lets Windows applications run on other operating systems.)

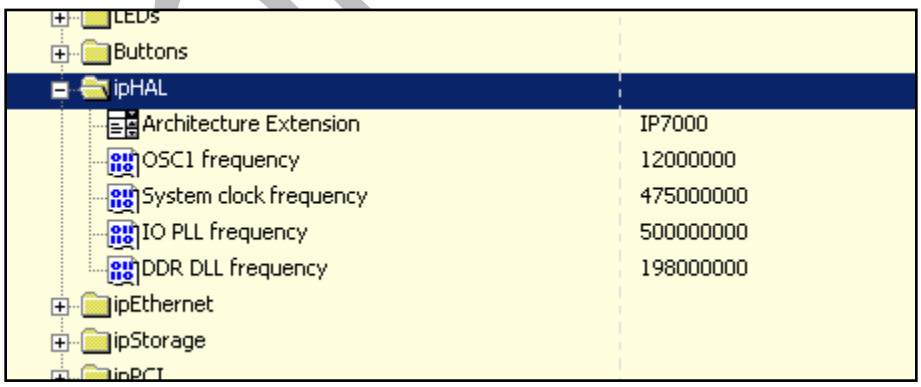
- **wine ultra/tools/bin/win32/ConfigTool.exe ultra7k_design320.lpj**
- **wine ultra/tools/bin/win32/ConfigTool.exe IP7160_design320.brd**

2.3 Creating ultra board (.brd) file

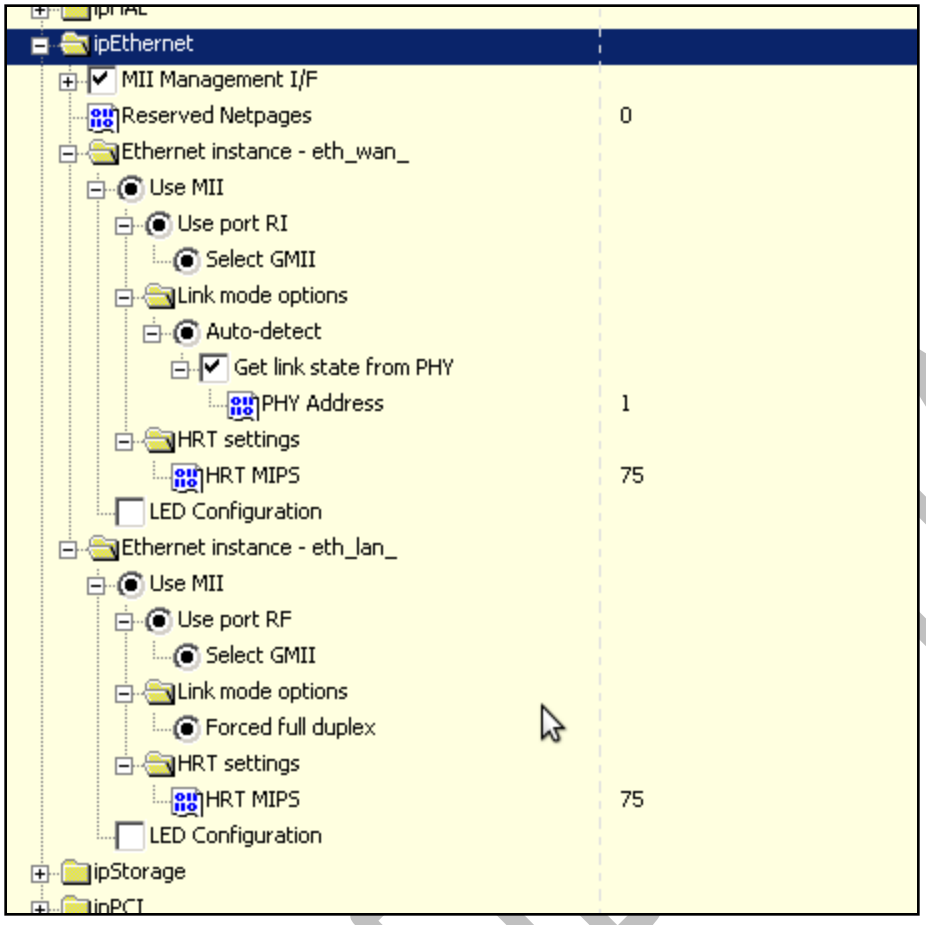
Start the creation of **`IP7160_design320.brd`** by copying and renaming an existing **`.brd`** file for a board that is similar to your board in number and type of interfaces. Edit the file using **`ConfigTool.exe`**. Change the names and paths inside the board file.



Change the IP7K processor part number and suggested frequency of CPU and DDR in the fields shown below. Oscillator frequency does not need to be changed unless the crystal is different from 12Mhz on your board. Change the DDR frequency to 226MHz.

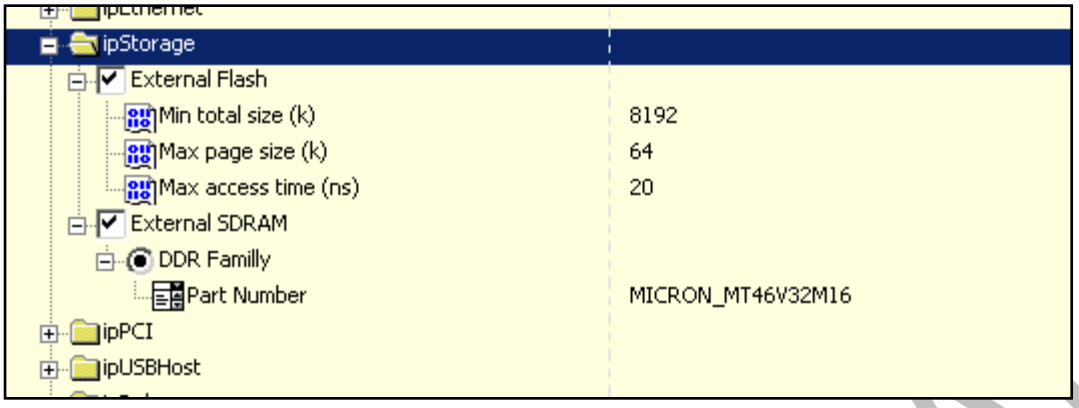


Configure Ethernet and specify the port as per schematics. The PHY addresses depend on how the config ports of the Ethernet PHY/switch are connected. Please get these addresses from your board designer.

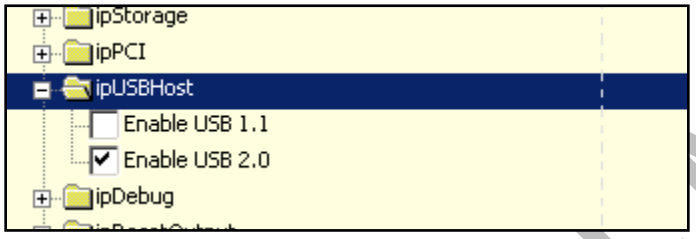


The DDR and flash configuration are very important, because these need to be correct for the board to boot-up. Check the flash data sheet for the total size and page size. This page size is also needed for computing the **mtd** parameters described later in the document.

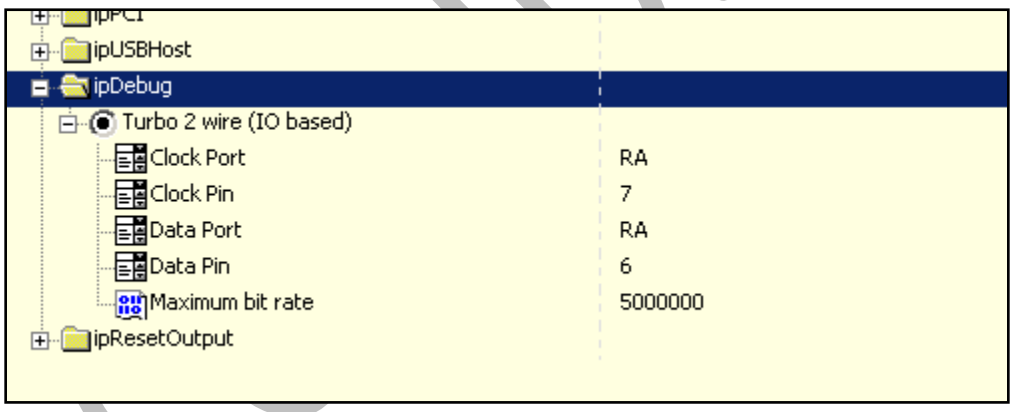
The DDR controller in IP7K processor needs to be configured to train the DDR part; so, picking the right part type is important. Double check the last part of the part number (64M16, 16M8, etc.).



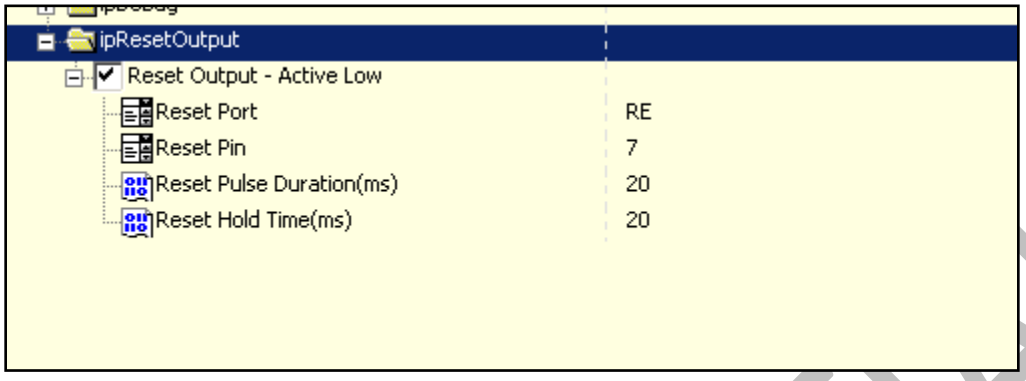
Configure USB interfaces as per your board design..



Ubicom software uses two GPIO pins configured as “turbo 2-wire” for sending debug information. Check the pin mapping document to find the port name and pin numbers.



If your design requires some pins to be sent a pulse on startup, set them up here. Typically, some Ethernet PHY/switches need this pulse to come out of reset state.

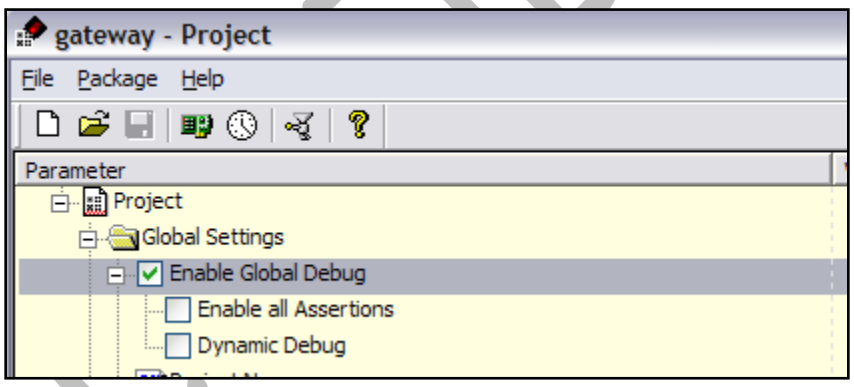


2.4 Creating ultra project (.lpj) file

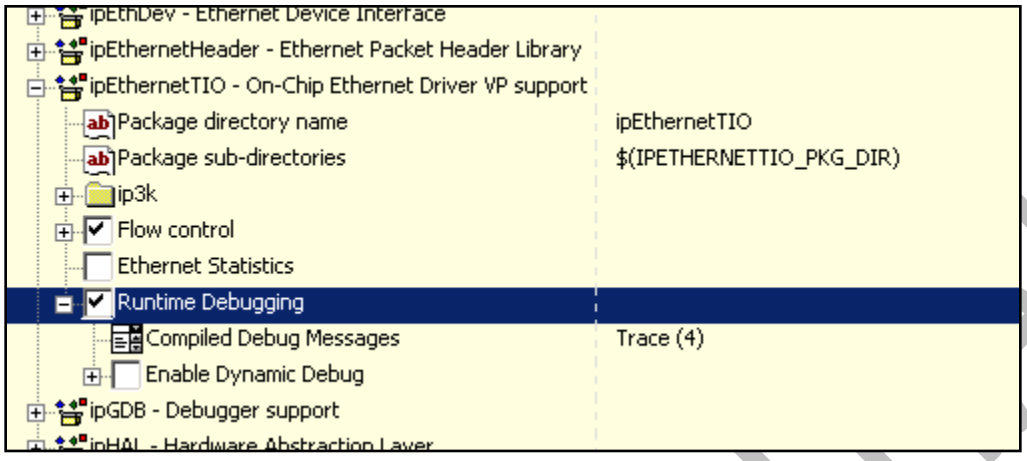
The **.lpj** file is a project file for Ultra that describes the applications you want to run on your board. The interfaces on the board can be selectively enabled or disabled in this file. Use **ConfigTool.exe** to edit the contents of the **.lpj** file.

The **.lpj** file needs to import the ultra board file.

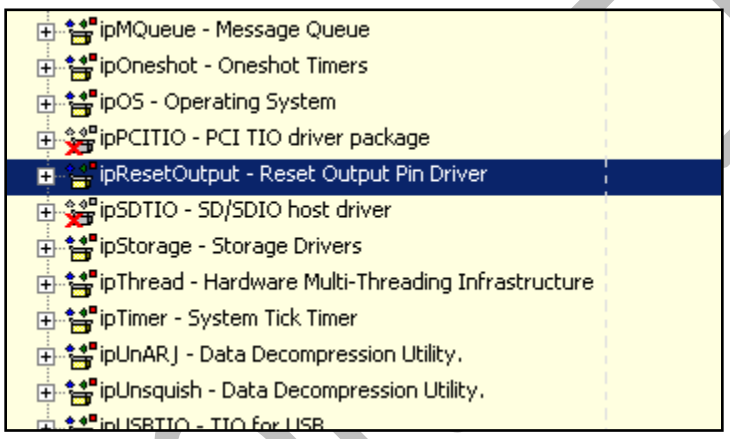
4. Click the green icon shown for selecting board file.
5. Enable global debugs to see the initial boot-up messages
6. **Do not use THE other 2 buttons (profile and generate)**



You can enable individual package-level debugging to check the status of individual interfaces. In the picture below, debugging for Ethernet is being enabled. “Trace” level debugging gives the highest verbose mode (good for board bring-up).

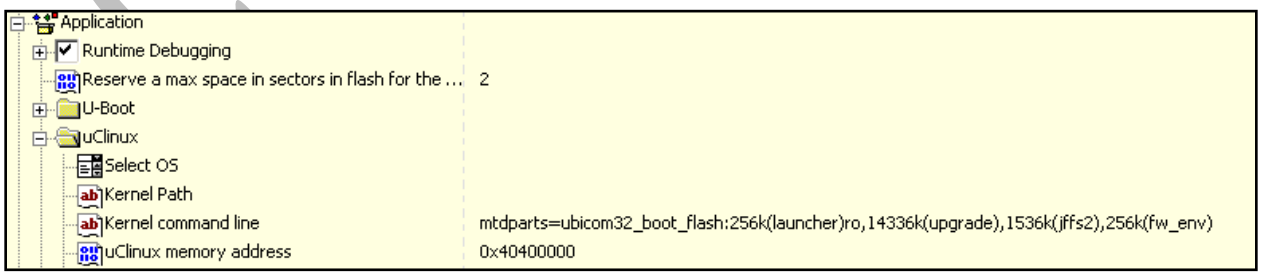


You can disable some interfaces by picking the correspondingly named TIO nodes and right-clicking on the nodes.



The **`.lpj`** file must be set up to pass the memory layout of flash to Linux. (In later Ubicom distributions this function might be moved to the board file.)

For distribution 1.0 the **`mtdparts`** string looks as shown below.



```
mtdparts=ubicom32_boot_flash:256k(launcher)ro,14336k(upgrade),1536k(jffs2),256k(fw_env)
```

Referring to the magnified example above, the flash erase sector is 256Kbytes of size and the total size of flash is 16Mbytes.

1. Launcher is 256K
2. Upgrade is 16384(16M) – 6*256 – 256 – 256 = 14336K
3. Jffs2 = 6 * 256K = 1536K
4. Fw_env = 256K

2.5 Creating Linux board (.c) file

The linux board file for design320 needs to be created in **`linux-2.6.x/arch/ubicom32/mach-ip7k/`**. Use an existing board file present in the directory and modify to the requirements of design320 project.

3 Set-Up the Build

If the board files **IP7160_design320.brd**, **ultra7k_design320.lpj**, and **board-ip7160_design320.c** have not yet been created, go to Section 2. If you have received ready-made board files, you need to add those files into your copy of the Ubicom Linux distribution and build firmware for that board.

3.1 Adding Boards to Ultra

Add Ultra board file:

- Copy **IP7160_design320.brd** to the directory **ultra/boards**.

3.2 Adding .lpj Files to Ultra

Add Ultra project file:

- Copy ultra7k_design320.lpj file to the directory **ultra/config.**

3.3 Adding Linux Board File

1. Add the kernel board file:
   - Copy the file **board-ip7160_design320.c** to the directory **linux-2.6.x/arch/ubicom32/mach-ip7k/**.
   - Edit the Makefile in the same directory, and add the following line:

```
obj-$(CONFIG_IP7160DESIGN320) += board-ip7160_design320.o
```

2. Edit the Kconfig file in the same directory, and create a **config IP7160DESIGN320** similar to other boards and relevant to your board. This will create a selectable configuration for **menuconfig**.
3. Edit Kconfig in the **linux-2.6.x/arch/ubicom32** directory and add your board name to BRD_128MB or BRD_64MB.

3.4 Create Target Profile (OpenWrt)

Profiles are application specific configurations. Typically a single profile can run on multiple boards. For this purpose, we will create the profile and tie it to the board file to make the building simple.

The directory **openwrt/target/linux/ubicom32/IP7160RGW/profiles** contain the profiles.

3.4.1 Step 1: Create profile makefile

For this example create the Makefile for profile RouterGatewayDESIGN320 by doing the following steps

1. **cp 100-RouterGateway.mk 100-RouterGatewayDESIGN320.mk**

2. Edit the contents of **100-RouterGatewayDESIGN320.mk** to reflect the profile name RouterGatewayDESIGN320 and its description.

Example 1. Profile Makefile.

```
define Profile/RouterGatewayDESIGN320
    NAME:=RouterGatewayDESIGN320
    PACKAGES:=pptp portmap luci-app-samba
endef

define Profile/RouterGatewayDESIGN320/Description
    Maecenax RouterGateway Profile for IP7K Aliquex 320 board
endef
$(eval $(call Profile,RouterGatewayDESIGN320))
```

3.4.2 Step 2: Create profile

The profiles are located in the directory **openwrt/target/linux/ubicom32/IP7160RGW.** Create the profile RouterGatewayDESIGN320 by doing the following

1. **cp config-2.6.28.RouterGateway config-2.6.28.RouterGatewayDESIGN320**
2. Important: Edit the Board section in this profile
   A. Comment out #CONFIG_IP7160RGW
   B. Add CONFIG_IP7160DESIGN320=y
   C. Edit CONFIG_RAMSIZE to the DDR size on your board.
   D. Edit CONFIG_PCI_DEV0_IDSEL to the map to the correct pin on your config

      (linux-2.6./arch/ubicom32/Kconfig) has this define.
   E. Edit the file to comment or uncomment other configurations your application needs.

3.4.3 Step 3: Link profile configuration and board file

The following environment variable is created when DESIGN320 is selected from menuconfig during the build process.

```
    $(CONFIG_TARGET_ubicom32_IP7160RGW_RouterGatewayDESIGN320)
```

To the **Makefile** in **openwrt/target/linux/ubicom32/**, add code to test this variable and create a link to the appropriate configuration.

Example 2. Link profile to configuration.

```
define Kernel/Prepare
# Select kernel config file related with the selected profile
ifeq ($(CONFIG_TARGET_ubicom32_IP7160RGW_RouterGateway),y)
    ln -sf IP7160RGW/config-2.6.28.RouterGateway config-2.6.28
else
ifeq ($(CONFIG_TARGET_ubicom32_IP7160RGW_RouterGatewayDESIGN320),y)
    ln -sf IP7160RGW/config-2.6.28.RouterGatewayDESIGN320 config-2.6.28
else
    ln -sf IP7160RGW/config-2.6.28.VPNGateway config-2.6.28
endif
endif
    $(call Kernel/Prepare/Default)
endef
```

4 Build Firmware

Build OpenWrt as usual, but with the following selections:

- Select **IP7160_design320** board in the main menuconfig.
- Select **ultra7k_design320** project file in main menuconfig.
- Select **RouterGatewayDESIGN320** profile in OpenWrt menuconfig.

5 Bringing up the Board

The steps to bring up the board are as follows

1. Verify that IP7K is out of reset. The internal flash controller will drive a 300Khz clock to the flash. You can verify that signal to see if the IP7K is out of reset.
2. Connect the dongle and establish the in-circuit Debug (ISD) connection. You should be able to load firmware onto the board. This will verify that the interface between IP7K and flash is working fine; the downloader application will read after programming to verify.
3. (Optional) Build a small program to run entirely of flash.
4. If the DDR is up, the program should start to boot.
5. Start a telnet session to the dongle console over turbo 2-wire by entering

   ```
   telnet <dongle_ip_addr>
   ```

   You should be able to see debug messages displayed on the console. You should see a message from each interface as it comes up.
6. Start a telnet session to the dongle console over turbo 2-wire by entering

   ```
   telnet <dongle_ip_addr> 50
   ```

   You should be able to see the Linux kernel boot-up messages scrolling down.

Application Note

195 Baypointe Parkway
San Jose, CA 94134

Tel 408.433.3300
Fax 408.433.3339

Email sales@ubicom.com
Web www.ubicom.com

Ubicom®, Inc.

Ubicom develops networking and media processor solutions that address the unique demands of real-time interactive applications and multimedia content delivery in the digital home. The company provides optimized system-level solutions for a wide range of products including wireless routers, access points, bridges, VoIP gateways, networked digital photo frames, and streaming media players.