Upgraded Unified Board

Operative System and firmware upgrading

User and developer manual

Written by Roberto Assiro

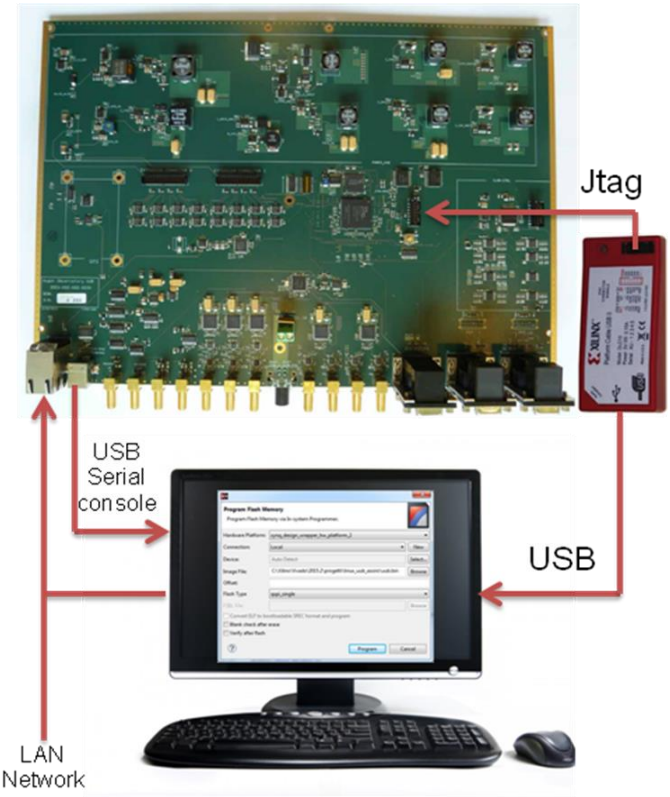
Manual version 1.0

INFN - SEZIONE DI LECCE
October 2016

How to program a brand new UUB

step 1: store UBOOT.bin to flash memory

1. Connect jtag programmer to UUB (Digilent or xilinx)
2. Remove the jumper (flash/jtag) and turn on the UUB
3. Launch SDK on computer and select "**Program flash**" from the Xilinx Tools menu
4. Select UBOOT.bin as image file
5. Click "Program" button to store the file into the flash memory. The programming phase requires few minutes
6. Turn off the UUB, insert the jumper and turn on UUB again. U-boot will start from flash memory.



step 2: Store uub.bin to flash memory by TFTP

- Turn on the UUB and wait to run u-boot . After few seconds you will get the prompt command :

```
U-Boot-PetaLinux>
U-Boot-PetaLinux> tftp 0x1000000 uub.bin  (tftp server required on the network)

Load address: 0x1000000
Loading: #################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
946.3 KiB/s
done
Bytes transferred = 33554432 (2000000 hex)

U-Boot-PetaLinux> sf probe

U-Boot-PetaLinux>sf update 0x1000000 0x0 0x2000000
device 0 offset 0x0, size 0x2000000
18481152 bytes written, 15073280 bytes skipped in 59.551s, speed 578524 B/s
U-Boot-PetaLinux>
```

This sequence requires about 12 minutes to program the entire memory image into the flash memory.
Otherwise is possible to program uub.bin directly by "step 1", but the program time needed is about 45 minutes.

How to update the operative system or bitstream on UUB

To update bitstream and linux image to the latest version there are two ways:

1. **UUB update by TFTP server. This solution require ethernet connection to the network**

2. **UUB update by USB memory stick. Standalone solution (very useful in the field)**

Insert the USB memory stick to the UUB or connect to the network by LAN

1 - TFTP server:

- Connect UUB to the network by LAN and turn it on
- At command line type: **uub-update** (example for a server machine 192.168.1.136).
  UUB will download and overwrite the image system and bitstream files on flash memory
- Check the new version number of operative system after reboot

2.A - USB memory stick:

- Turn off the UUB
- Download **image.ub** and **fpga.bit** from repository and copy the files in the directory **uub-update** into the usb memory stick
- Connect the memory stick into the USB plug of UUB
- Turn on the UUB
- Wait for the booting phase and the patching starts
- When upgrading will be complete a message will inform you about .
- reboot the UUB and check the new version number of bitstream and operative system

2.B - USB memory stick (manual command):

- While the UUB is running insert the USB memory stick to the plug
- At command line digit: **uub-update**
- UUB will transfer the image system and bitstream files from USB memory folder to the flash memory
- reboot the UUB and check the new version number of bitstream and operative system

Important! - The system expects to find the upgrading files (image.ub and fpga.bit) in the memory stick in the directory **uub-update** in order be installed correctly.

Changing the UUB's MAC address

Technical documentation about UUB's MAC address definition

Problem:

After the UUB's production we need to define a different network address without building a different petalinux image for each board.
By default, the Ethernet MAC address for both u-boot and the kernel are set during compilation (petalinux-build), based on the "Ethernet MAC address" option in the "System Settings" configuration menu.
In order to provide a unique MAC address, it's not practical to create a new petalinux system image for each board, we can use instead U-Boot to overwrite the local MAC address property of your device tree after it's loaded into the flash memory.

U-boot uses the environment variable "ethaddr" as the MAC address.

How to?

Customization of the board MAC address using u-boot's command to change the value of the environment variable "ethaddr"

- Turn on the UUB, the u-boot will start. Before the counting down finishes press enter, you will get the prompt :

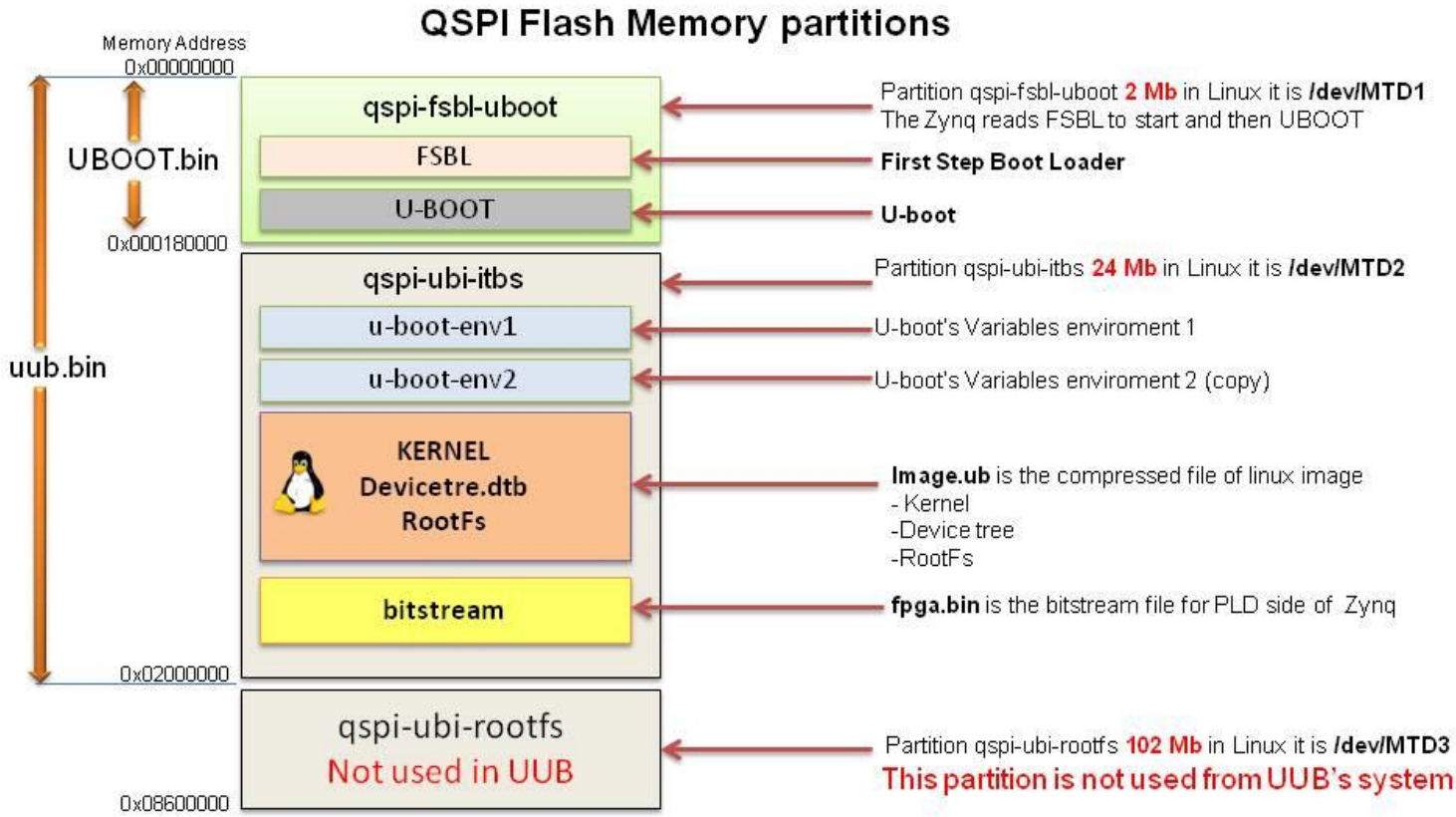
```
petalinux u-boot>

petalinux u-boot> set ethaddr 00:11:22:33:44:55     (example of new MAC to provide)

petalinux u-boot> saveenv            save the setting into the flash memory's environment

petalinux u-boot> saveenv            repeat the command for the second partition
```

Reboot the UUB

The new MAC address is saved into the environments of flash memory and petalinux will start with new MAC.
The value in device tree is overwritten
In order to use several UUBs connected to the same network is suggested to define different MAC address for each board during first check.

UUB system and flash memory architecture

Petalinux system is built in a compressed file: **image.ub** The FPGA contents in file **fpga.bit** (bitstream)
Inside image.ub there are: Kernel, device tree and root file system with all programs, scripts and drivers developed
Image.ub is stored into the flash memory in the MTD2 partition in the volume **itbs.**
At any boot the image.ub is decompressed into the RAM and the system can start with all programs built inside.



uub.bin = entire flash memory content from 0x0 to 0x2000000 address (32Mb)

UBOOT.bin = FSBL and u-boot programs.

How to program and upgrade the UUB's operative system and FPGA firmware

Questions

- How it's possible to store into the system image (image.ub) new programs, scripts or settings without re-build the petalinux image?
- Can we update the system by radio communication?

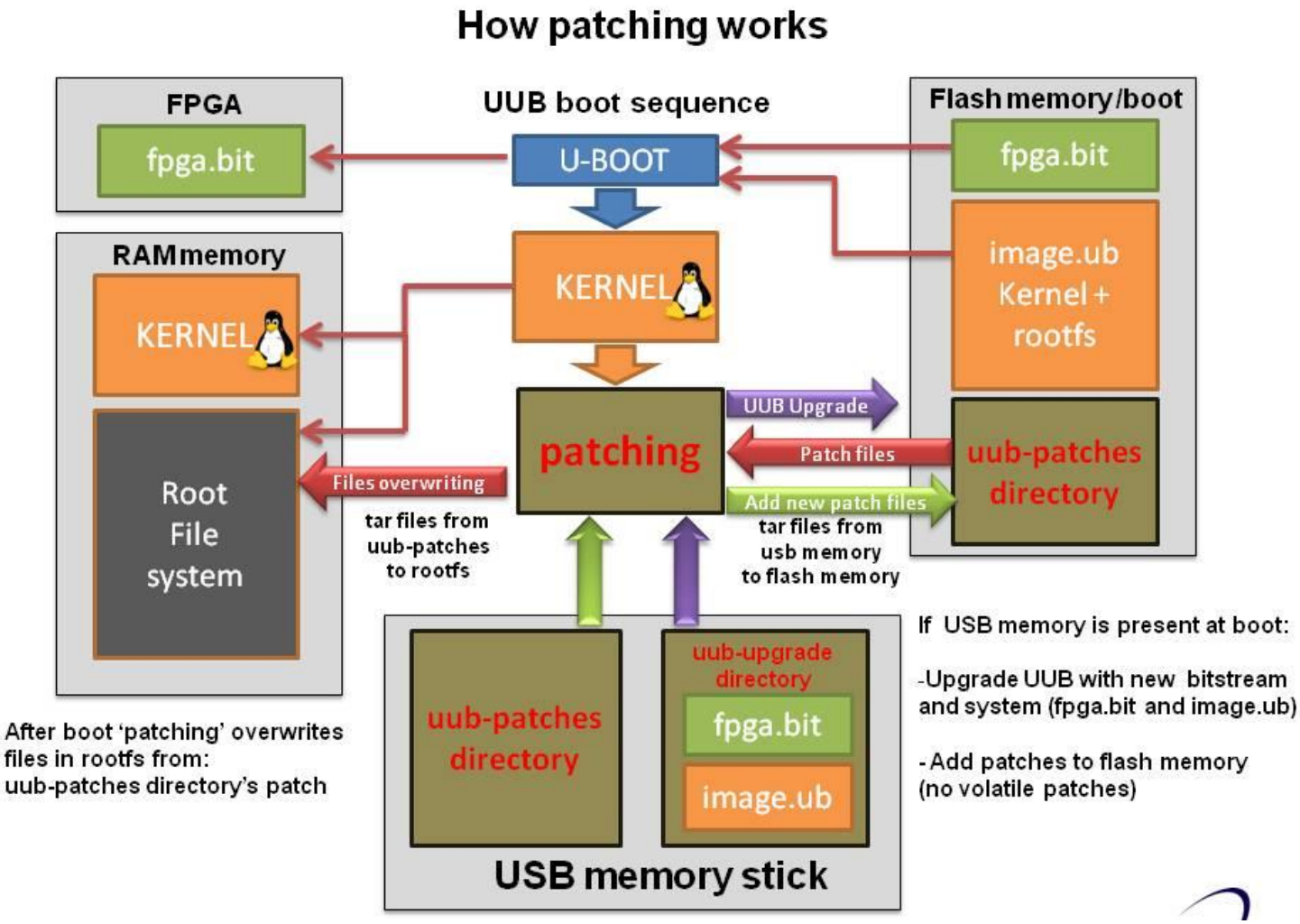
Solution:

The solution is to patching the root file system just after the kernel running.
"Patching" is the script's name which runs at any boot. It controls if into the volume /boot/uub-patches of the flash memory there are files for patching.
The patch file are compressed file type tar (.tgz) (lite to store and also to transfer by UUB's radio communication system).
To patch the system we need to create a compressed file and to save it in the flash memory in the folder **uub-pathes**

"Patching" the solution to add software and settings to the rootfs

"patching" runs after kernel start, it will:

1. write or overwrite files in rootfs from patches contents in flash memory (uub-patches folder)
2. overwrite files image.ub and bitstream from USB memory stick if connected (uub-upgrade folder)
3. if USB memory stick is connected "patching" writes or overwrites files in rootfs from patches contents (uub-patches folder). If patch's name starts with a "**P**", patch file will be stored in flash memory too (permanent patching).
4. if USB memory stick is connected "patching" write or overwrite files in rootfs from patches contents (uub-patches folder). If patch's name starts with "**_P**", patch file will be not stored to flash. (useful function to test software on UUB)

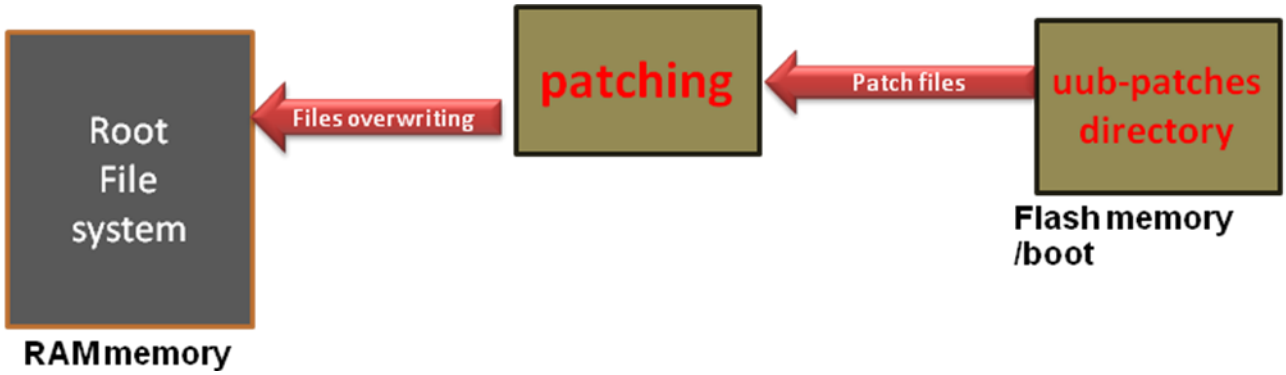


Repository's files for UUB:

- **UBOOT.bin** First file to transfer in UUB's flash memory. This file contents the image of u-boot only
- **uub.bin** Image of entire flash memory. (32Mb).
- **image.ub** Compressed image of operative system and RootFs
- **fpga.bit** bitstream file for FPGA

The patch

"Patch" is a compressed file which contains programs and scripts with path destination in root file system.
The patch's name must start with **"P"** followed by a sequence number from 00 to 99
In the flash memory the directory containing patch's files is **/boot/uub-patches**
In the USB memory stick the directory containing patch's files is **/usb/uub-patches**
"Patching" is the main script that runs, after Kernel starts, and it can write and overwrite files in rootfs



How build a no volatile patch into flash memory (/boot)

command to create a patchfrom UUB's command line:

Example:

1. Tranfer your application software in your workspace in file system (example: bin/hello.elf)

2. mount flash memory partition **/boot**:
   root@Auger-uub:~# **mountboot**

3. create tar file from /root:
   root@Auger-uub:~# **cd /**
   root@Auger-uub:~# **tar cvzf /boot/uub-patches/P01hello.tgz bin/hello.elf**

4. unmount partition /boot:
   root@Auger-uub:~# **umountboot**

How build a no volatile patch into USB memory stick

After boot, the patch file will be transferred to flash memory folder if connected
patch directory = uub-patches update directory = uub-update

Example:

1. Tranfer your application software in your workspace in file system (example: bin/hello.elf)
2. insert USB memory and mount it **:**
   root@Auger-uub:~# **mountusb**
3. create tar file from /root:
   root@Auger-uub:~# **cd /**
   root@Auger-uub:~# **tar cvzf /usb/uub-patches/P01hello.tgz bin/hello.elf**
4. unmount device:
   root@Auger-uub:~# **umountusb**

How build a volatile patch into USB memory stick

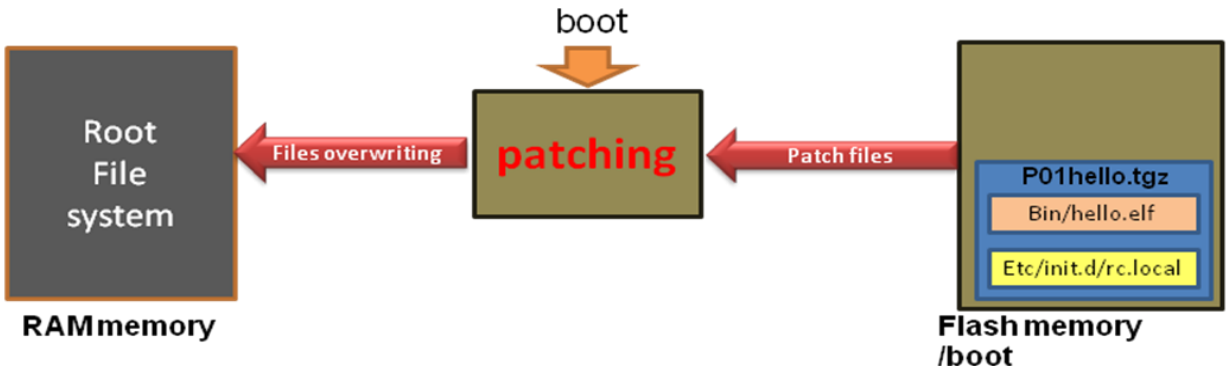
volatile patch is very useful during developing software

If patch's name starts with: **_P** the patch will be applied, but not stored in flash memory

no volatile patch name : **_P01name-patch.tgz**

How to build a patch to run automatically a program at boot

rc.local is a script file to edit to set an auto start process



1. edit and modify rc.local :
   root@Auger-uub:~# **vi etc/init.d/rc.local**

2. insert command a line to run your application (example hello.elf in flash memory)

   #!/bin/bash
   #Please put follow your application programs you want run at boot
   #example: name2run > /dev/null 2>/dev/null &

   hello.elf > /dev/null 2>/dev/null &
   echo "hello is running at boot!"

   exit and save from vi

3. build the patch file :
   root@Auger-uub:~# **cd /**
   root@Auger-uub:~# **mountboot**
   root@Auger-uub:~# **tar cvzf /usb/uub-patches/P01hello.tgz bin/hello.elf etc/init.d/rc.local**
   root@Auger-uub:~# **umountboot**

Reboot the UUB, hello.elf will run automatically after boot

Upgrading UUB software by radio