Upgraded Unified Board

Operative System and firmware upgrading

User and developer manual

This document is a work in progress.
The most current version is kept in the github repository:

h

Written by Roberto Assiro

Manual version 2.0 – witten for UUB V2

INFN - SEZIONE DI LECCE
February 2018

Standard UUB system configuration

Default system setting parameters

- MAC address 00:0A:35:00:1E:53
- Eth0 dhcp enabled
- Eth0:0 192.168.168.168 - net mask 255.255.255.0 (fixed number for the field)
- System console speed **115200** baud (u-boot and petalinux)
- Login: **root**
- Password: **root**

To change standard configuration the patch must be applied

Standard file names

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
- image.ub Compressed image of the Petalinux System
- fpga.bit bitstream file for FPGA
- uub.bin Image of first 48Mb of flash memory. It includes uboot, image.ub and fpga

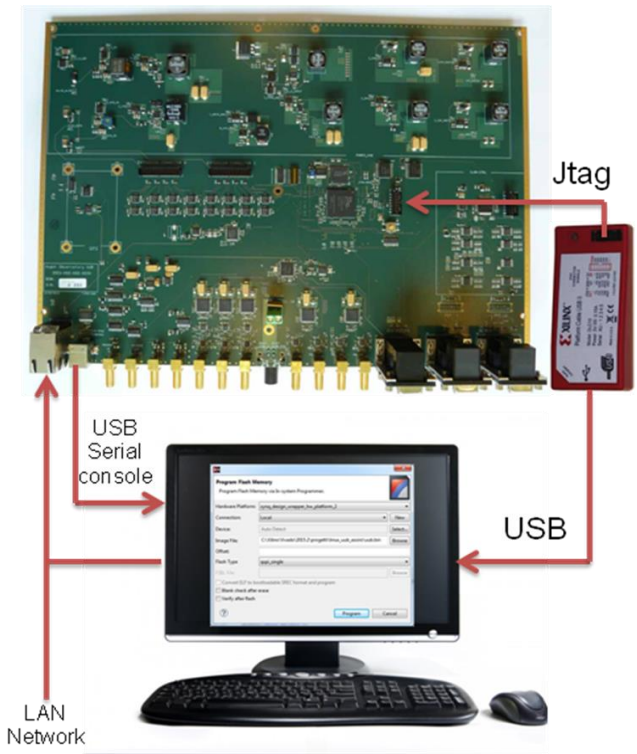
Implemented commands in Petalinux

- **mountboot** mounts itbs volume (system image folder) under /boot
- **umountboot** unmounts the volume
- **mountflash** mounts flash volume (80 Mb of flash memory) under /flash
- **umountflash** unmounts the volume
- **version (**or **ver)** shows version of system
- **recovery-boot** restore system image, bitstream and patch directory from **recovery volume to /boot**
- **recovery-update** update recovery image of system and bitstream with current into recovery volume
- **uub-update** command to update the UUB in SDECO lab only (IP 192.168.1.136)
- **makeflash** command to create the /flash volume in MTD3
- **daq on/off** command to stop (off) and run(on) the data acquisition on the UUB
- **watchdog-log** print watchdog log file from flash memory (history events)
- **image** replace petalinux image.ub from tftp server (no bitstream)

How to program a brand new UUB

store uub.bin to flash memory (Jtag only)

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 **uub.bin** as image file (49 Mb)
5. Click "Program" button to store the file into the flash memory. The programming phase requires 45 minutes. Restart UUB



step 1: store uboot.bin to flash (jtag + TFTP)

6. Step 1, 2, 3.
7. Select **uboot.bin** as image file (480 Kb)
8. Click "Program" button to store the file into the flash memory. The programming phase requires few minutes
9. 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

- **uub.bin** is the image file of flash memory (MTD1 and MTD2 partition - from address 0x0 to 0x3000000)
- Turn on the UUB and wait to run u-boot . After few seconds you will get the prompt command :

```
U-boot-UUB>
```

Please set TFTP server IP if needed set serverip 192.168.1.1 (example ip number)

run flash-uub is a custom command under u-boot to update the flash memory partitions

```
U-boot-UUB> run flash-uub  (tftp server required on the network. Please see page 13 to create TFTP service on your Linux machine)

Load address: 0x1000000
Loading: ##################################################################
         ##################################################################
         ##################################################################
         ##################################################################
         946.3 KiB/s
done
Bytes transferred = 33554432 (3000000 hex)

device 0 offset 0x0, size 0x3000000
18481152 bytes written, 15073280 bytes skipped in 59.551s, speed 578524 B/s
U-boot-UUB>
```

uub.bin is stored into the flash memory

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.

Create /flash volume in MTD3 partition (very important step)

- Turn on the UUB and login. Write a command **makeflash** and wait the response. When finished, reboot the UUB

Sequence to program a UUB's flash memory (not empty)

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

```
U-boot-UUB>
```

Erase two partitions area of flash memory by the follow commands:

sf probe
sf erase 0x0 0x3000000

1. Connect jtag programmer to UUB (Digilent or xilinx)
2. Launch SDK on computer and select "**Program flash**" from the Xilinx Tools menu
3. Select **uboot.bin** as image file (480 Kb)
4. Click "Program" button to store the file into the flash memory. The programming phase requires few minutes

Now the firts partition is programmed with u-boot

Restart the uub and wait u-boot prompt:

```
U-boot-UUB>
```

Please set TFTP server IP of your linux machine on the network: (to set TFTP service on your linux machine see page 13)

set serverip 192.168.1.1 (example ip number)

Download and store the file uub.bin into the flash:

```
U-boot-UUB> run flash-uub  (tftp server required on the network)
Load address: 0x1000000
Loading: ############################################################
#################################################################
#################################################################
#################################################################
946.3 KiB/s
done
Bytes transferred = 33554432 (3000000 hex)

device 0 offset 0x0, size 0x3000000
18481152 bytes written, 15073280 bytes skipped in 59.551s, speed 578524 B/s
U-boot-UUB>
```

Now two flash memory partitions are updated with a complete petalinux image .

uub.bin is stored into the flash memory

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 jtag programmer (long time)

Create /flash volume in MTD3 partition (very important step)

- Turn on the UUB and login. Write a command **makeflash** and wait the response. When finished, reboot the UUB

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 SDECO server machine 192.168.1.136).
- Command for specific server IP is **uub-update followed by IP number** of your tftp machine
  UUB will download and overwrite the image system and bitstream files on flash memory
- Check the new version number of operative system after reboot

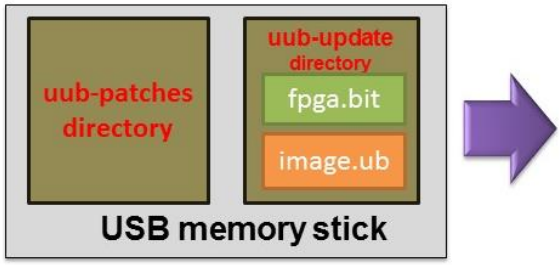
2.A - USB memory stick (very useful in the field):

- 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 to 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 :

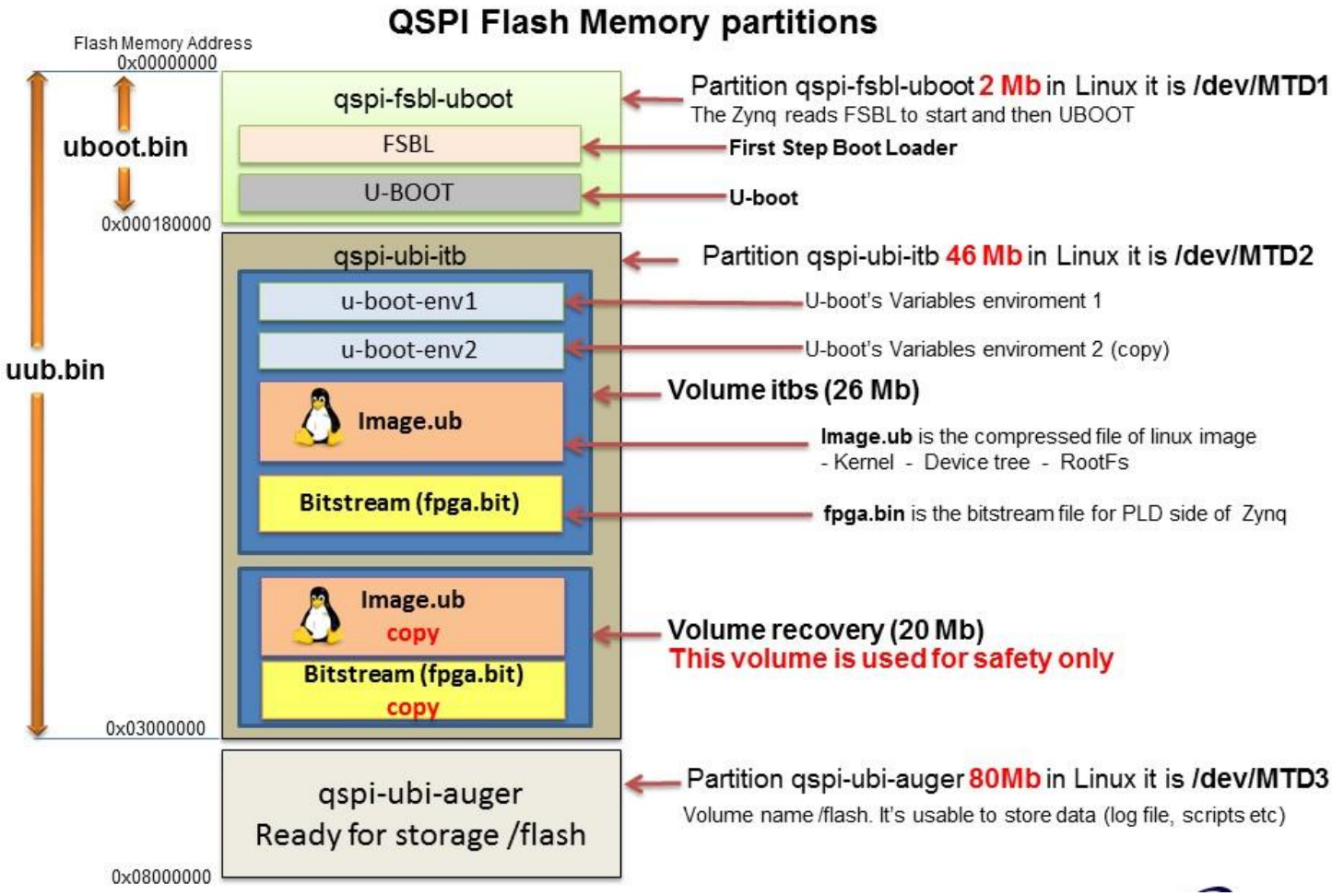
```
U-boot-UUB>

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

U-boot-UUB> saveenv       save the setting into the flash memory's environment

U-boot-UUB> 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's 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.



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?

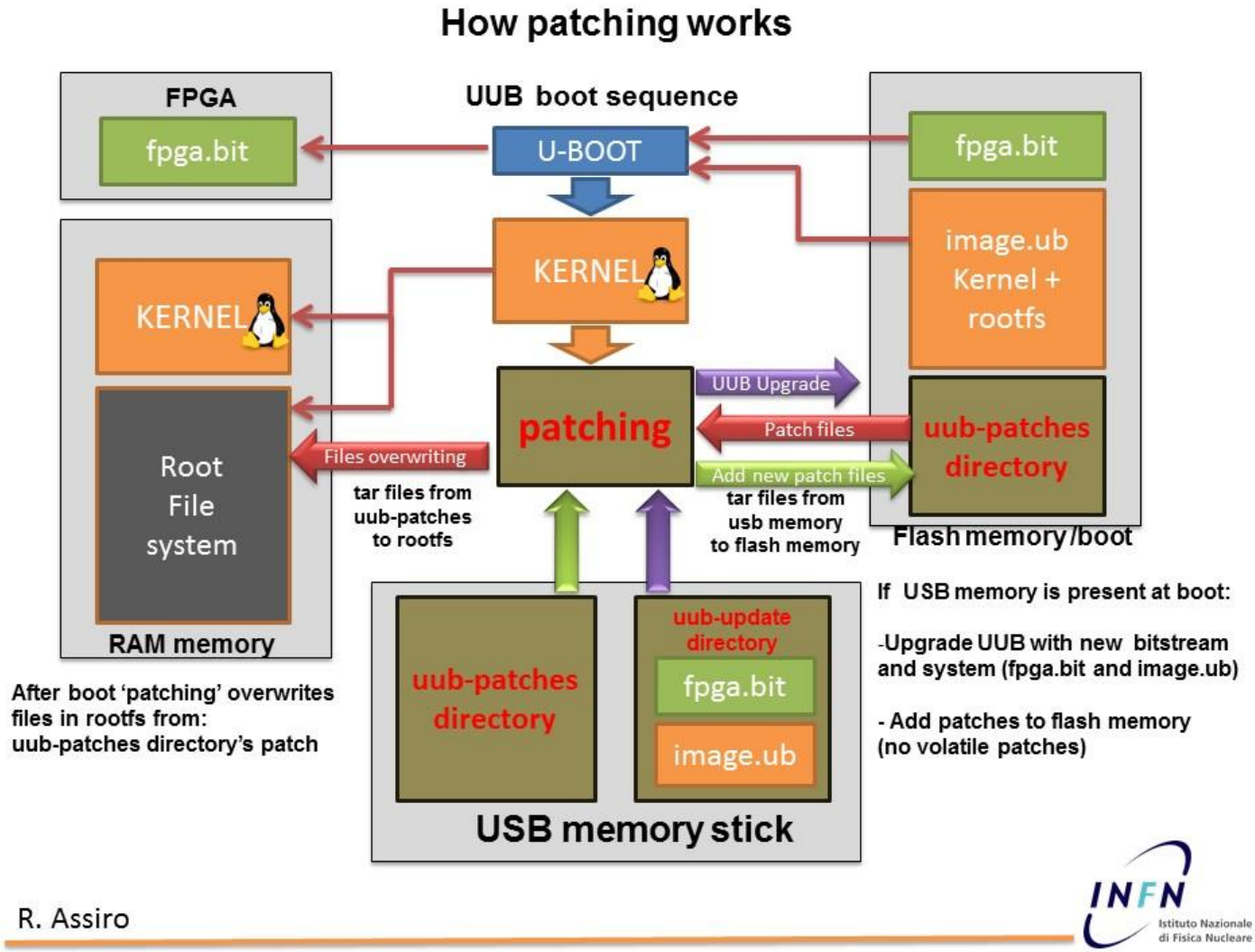
Solution:

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)

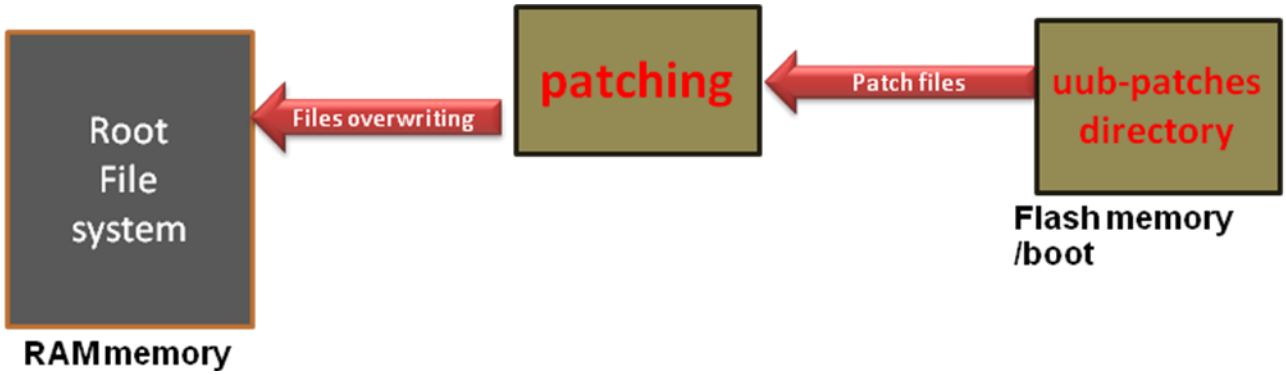


Repository's files for UUB:

- **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. (48Mb).
- **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 patch from 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

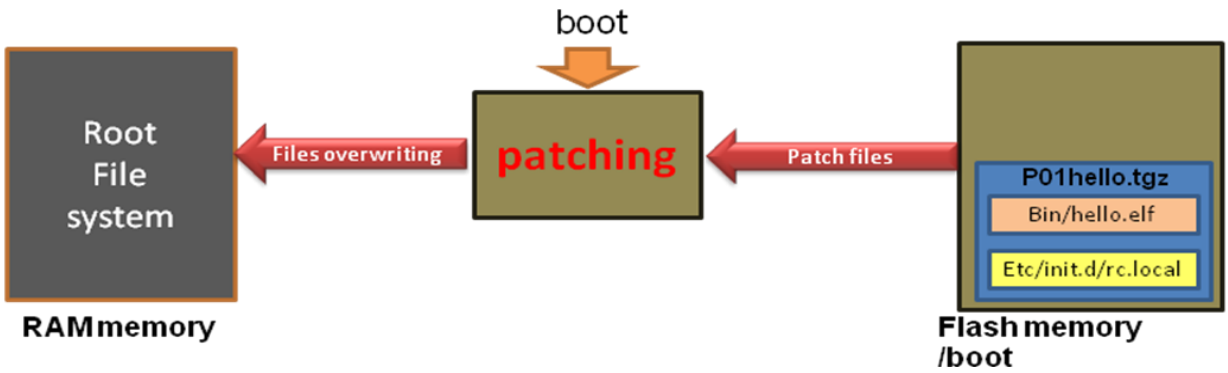
Volatile patch will not stored in flash memory

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

How to connect a laptop to the UUB by LAN in the field

Standard LAN configuration is DHCP (eth0) and alias address **192.168.168.168** (eth0:0)

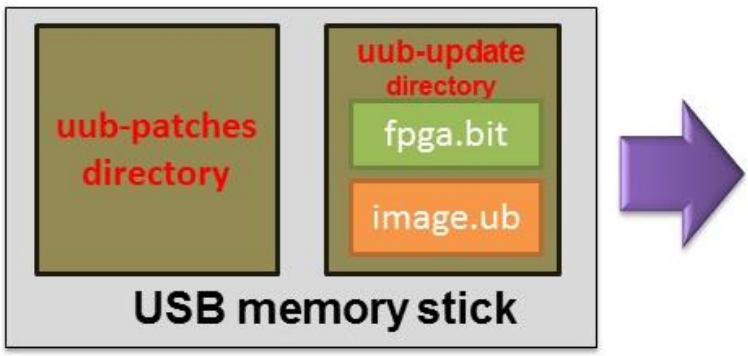
In the field is helful to use the direct connection LAN to LAN

Set your laptop LAN configuration as fixed IP number :

IP address : 192.168.168.1
IP mask : 255.255.255.0

Open your browser and connect to: 192.168.168.168

Upgrade UUB by memory stick (helpful in the field)



Directory: uub-patches and uub-update

uub-patches for tar files for patching

uub-update for image.ub (petalinux system) and fpga.bit (FPGA bitstream)

UUB update sequence:

- Turn off the uub
- Insert the usb memory stick on USB plug
- Turn on the UUB and wait the booting. The patches into the memory stick will be transferred into the flash memory and the system will be update (if update is present into the folder)

How to create a TFTP Server on your Ubuntu machine

Install and Setup

1. Install following packages.

```
sudo apt-get install xinetd tftpd tftp
```

3. Create /etc/xinetd.d/tftp and put this entry

```
service tftp
{
protocol        = udp
port            = 69
socket_type     = dgram
wait            = yes
user            = nobody
server          = /usr/sbin/in.tftpd
server_args     = /tftpboot
disable         = no
}
```

15. Create a folder /tftpboot this should match whatever you gave in server_args. mostly it will be tftpboot

```
sudo mkdir /tftpboot
sudo chmod -R 777 /tftpboot
sudo chown -R nobody /tftpboot
```

19. Restart the xinetd service.

newer systems:

```
sudo service xinetd restart
```

older systems:

```
sudo /etc/init.d/xinetd restart
```

Now our tftp server is up and running.

Testing our tftp server

5. Create a file named test with some content in /tftpboot path of the tftp server

Obtain the ip address of the tftp server using ifconfig command

6. Now in some other system follow the following steps.

```
tftp 192.168.1.2
tftp> get test
Sent 159 bytes in 0.0 seconds

tftp> quit

cat test

how to restart tftp service : sudo service tftpd-hpa restart
```

How to use recovery volume

In case of system booting problem (partition compromised), we might try to boot the UUB from the recovery volume.
From u-boot prompt command:

U-boot-UUB> **run recovery**

The system will start from recovery volume in the flash (recovery volume is a backup of first booting area called /boot)

Now is possible to copy bitstream and system image from **recovery** to the **/boot volume**:

root@Auger-uub:~# **recovery-boot**

Restart the UUB at the end of process

The reverse process is:

recovery-update **to** update recovery contents from /boot volume

UUB TEST TOOLS

Test by serial terminal:

LED DAC AD5694:

root@Auger-uub:/# **ad5694 <val ch0> <val ch1> <val ch2> <val ch3**

example : ad5694 200 200 200 200 to get 200 counts for each channels

ramp generator on each channels for 1 minute:

root@Auger-uub:/# **ad5694 5000** ramp on all channels for 1 minute

Clock generator DAC7551:

root@Auger-uub:/# **dac7551 <val >** val are counts 0 - 4095

example : dac7551 1024 to get 1024 counts on output

ramp generator on output for 1 minute:

root@Auger-uub:/# **dac7551 5000** ramp on output for 1 minute

RADIO RESET:

root@Auger-uub:/#**radio-reset** to get signal reset on radio connector