IPMC environment & configuration

Version 3.1 March 07th 2016

This document describes all the operations needed in order to program an IPMC.

It covers the programs installation, the hardware requirements, the ports configurations, the environment setup, CPLD programming, One Time Programming (OTP) of flash memory, firmware upload.

The setup is based on two machines, one holding the ICARE software release and one for accessing the IPMC board. In this document the Scientific Linux machine used for the ICARE release is called lappc-f106. The Windows 8.1 machine used for accessing the IPMC is called lappc-p509.

A chapter is also dedicated to upgrading the firmware using Ethernet.

1. Introduction

The IPMC has 2 micro-controllers called IOIF and IPMC. Each micro-controller has an internal flash memory for storing the firmware and a One Time Programming (OTP) memory for storing some permanent information.

There is also an external flash memory which is accessible by the two micro-controllers through a CPLD. This memory has been organized in 4 blocks: 2 for each micro-controller with one being used for a default factory firmware and the other for the current firmware. Some mechanism is implemented in order that if there is a problem with the current firmware not able to run, the micro-controller will revert after a few trials to the default factory code. A complete description is in the README file of the ICARE release.
The default factory firmware allows the upgrade of the firmware through Ethernet.
This has been implemented in order to have a safe upgrade mechanism of the firmware.

2. ICARE release and software tools installation

All the requirements and instructions for installing the software tools and the ICARE release on the Scientific Linux machine are described in the ICARE software stack that you can get here:

http://lappwiki.in2p3.fr/twiki/bin/view/AtlasLapp/Informatique

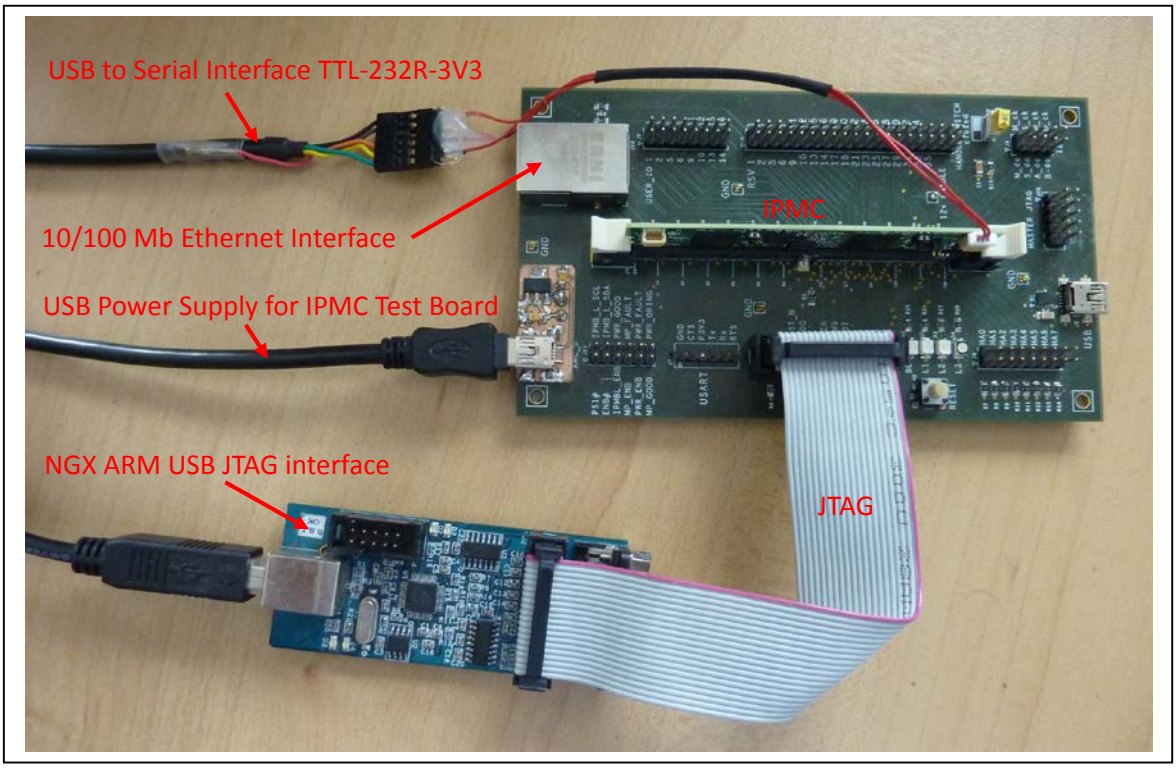
This will not be developed further here other than when it is needed to configure the IPMC.

3. Software and Hardware installation on Windows 8.1 PC

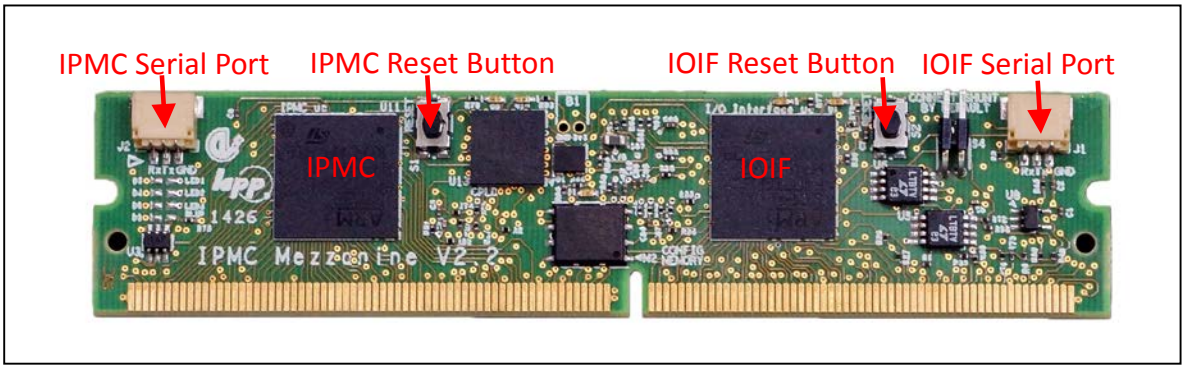
This chapter deals with the installation of the software, the configuration of the environment and the hardware required for communicating with the IPMC on the Windows 8.1 machine (in our case lappc-p509).

The PC communicates with the IPMC through the IPMC JTAG chain and through the serial port of each micro-controller on the IPMC. In our setup we are using an IPMC Test Board but this can be your own blade or any special setup that you would have produced, as long as you have access to the JTAG chain of the IPMC via a connector.

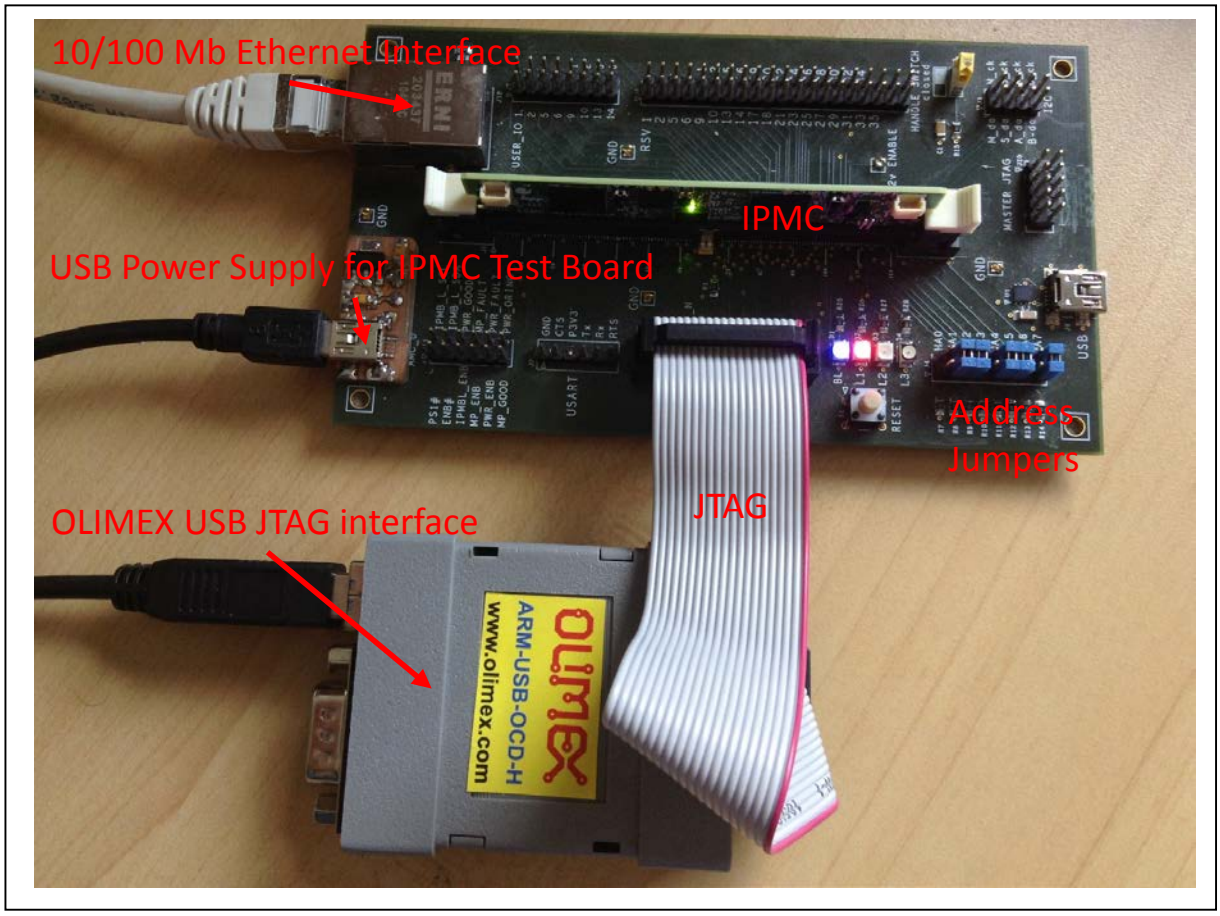
This is a picture of the setup with the different elements needed around the IPMC.



This is a picture of the IPMC with what you need to know in the next chapters.



Same setup with the OLIMEX interface instead of the NGX. Please also note that on this picture the jumpers have been set for having a proper geographical address



The IPMC Test Board is powered by a USB-A Male to mini USB-B Male cable. One possible reference is: http://fr.farnell.com/lumberg/2480-02/cable-usb-a-vers-mini-b-2m/dp/1308879

3.1. Openocd installation

Openocd provided in the release of ICARE is version 0.9.0. Depending on the version used, it may be better to get a later release from a web site where you can get the latest binaries for window.

http://www.freddiechopin.info/en/download/category/4-openocd

Latest tested by us release is 0.9.0.

Extract it into the directory of your choice or in C:\Program Files (x86). In this document we assume it has been copied in this directory.

In order to have the right configuration for the IPMC, you need the file ipmcv2_1.cfg that you will get in the ICARE release under \ICARE\contrib\openocd\windows\openocd-0.9.0\scripts\target and copy it in directory C:\Program Files (x86)\openocd-0.9.0\scripts\target. This operation is only needed if you didn't get openocd from the release.

A bash script is needed to start openocd.

Open a new file with your text editor preferably in a place easy to remember like a disk root directory (D:)

Copy this content into it

“C:\Program Files (x86)\openocd-0.9.0\bin-x64\openocd.exe” -f interface\ftdi\ngxtech.cfg -f target\ipmcv2_1.cfg
pause

You may want to use a different board, here is what the syntax would be for an Olimex interface:

"C:\Program Files (x86)\openocd-0.9.0\bin-x64\openocd.exe" -f interface\olimex-arm-usb-ocd-h.cfg -f target\ipmcv2_1.cfg

You can use the comment syntax ##... to have the different interfaces already included in your file

Save the file and rename it openocd.bat

3.2. Hypertrm software installation

In order to communicate with the serial port we need a terminal. This program is not part of the windows 8.1 release and need to be recovered from a PC running an older system like XP. You just need to copy the files hypertrm.exe and hypertrm.dll in a directory of your choice in example: C:\Program Files (x86)\Hyperterminal.

3.3. USB JTAG interface (NGX or OLIMEX)

This chapter describes the installation of the NGX ARM USB JTAG or OLIMEX interface on the PC lapppc-p509.

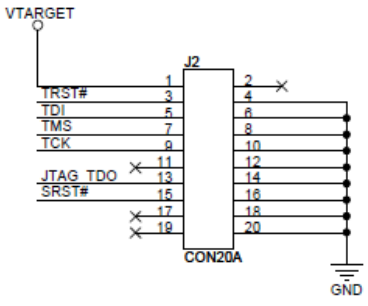
It is used to access the JTAG chain of the IPMC through which all the programming of the components (CPLD and flash) will be done.

You may need to manufacture a cable between the interface and the JTAG connector either on your blade or on our standalone IPMC Test Board. With the IPMC Test board a standard flat cable with 20 pins connectors will do the job. It may or not be provided with the kit you buy.

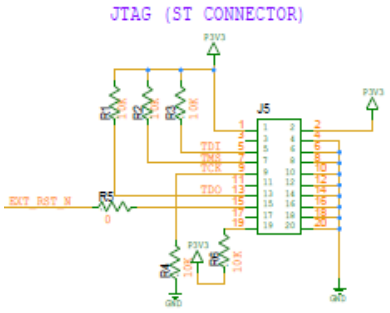
Both interfaces need a USB-A Male to USB-B Male cable which is not provided with the kit. This is one example of reference:

http://fr.farnell.com/molex/88732-9300/cable-usb-a-b-3-30m/dp/1201898

Signals on the NGX/OLIMEX ARM USB JTAG connector and on the IPMC Test Board JTAG connector are the following:



NGX/OLIMEX ARM USB JTAG Connector



IPMC Test Board JTAG Connector

3.3.1. NGX JTAG Port configuration

Connect the USB/JTAG adapter (NGX ARM USB JTAG from NGX Technologies) on a USB port of your Windows 8.1 PC.

Go to Device management (gestionnaire de périphériques)

Open USB Controllers (contrôleur de bus USB)

One should see 2 ports with the following names as long as the proper driver has not been installed.

The first port (A) is the JTAG port and the second one (B) is used to generate a virtual serial port which will map on a COMx port.

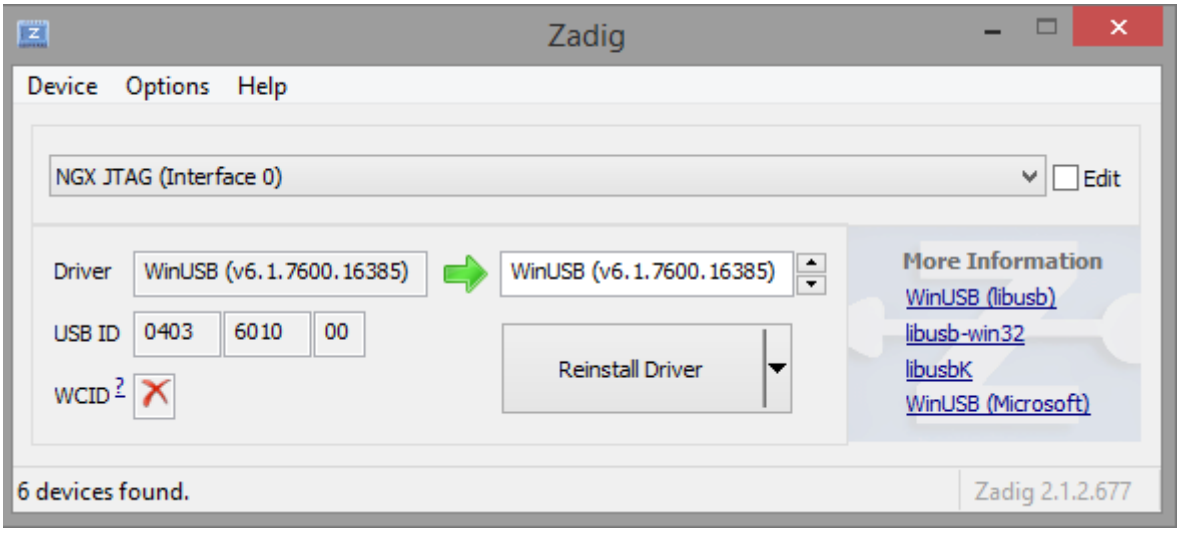
USB Serial Converter A

USB Serial Converter B

Install the correct driver for the JTAG port using the Zadig utility that you can download here:

http://zadig.akeo.ie/

Run the Zadig utility



Select NGX JTAG(Interface 0). If it doesn't appear go to menu Options List all Devices

In front of driver WinUSB(V6.17600.16385) select WinUSB (libusb) and click Reinstall Driver

After that operation the USB Serial Converter A port will have disappeared but you will find it in Périphériques Universal Serial Bus with the name NGX JTAG.

Connect the JTAG flat cable between the NGX JTAG connector and the IPMC Test Board JTAG connector.

3.3.2. OLIMEX JTAG Port configuration

Connect the USB/JTAG adapter (OLIMEX) on a USB port of your Windows 8.1 PC.

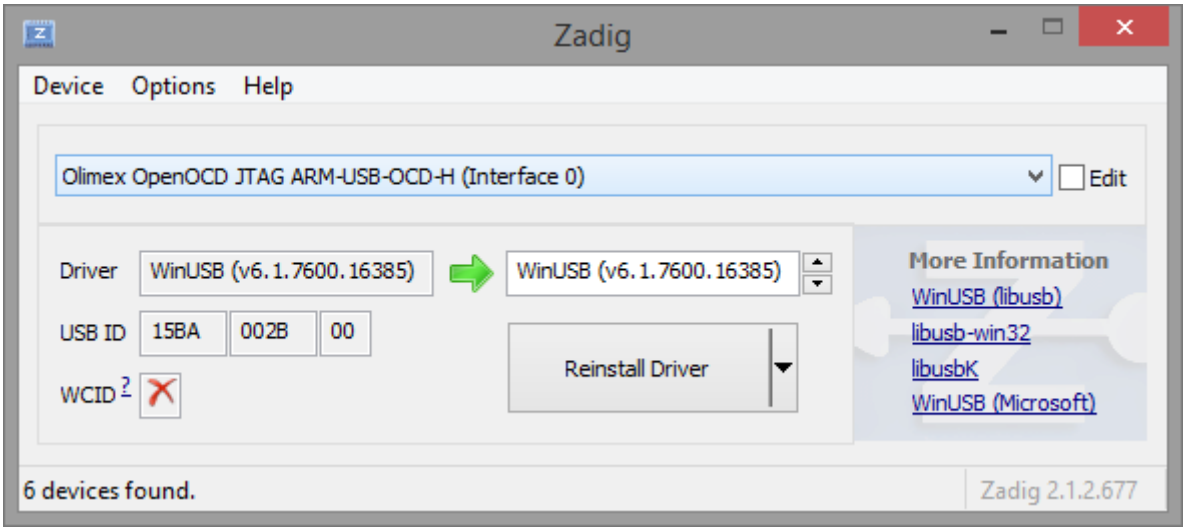
Go to Device management (gestionnaire de périphériques)

You should find it in other peripheries with the name Olimex OpenOCD JTAG ARM-USB-OCD-H

Install the correct driver for the JTAG port using the Zadig utility that you can download here:

http://zadig.akeo.ie/

Run the Zadig utility



Select Olimex OpenOCD JTAG ARM-USB-OCD-H (Interface 0). If it doesn't appear go to menu Options List all Devices

In front of driver WinUSB(V6.17600.16385) select WinUSB (v6.1.7600.16385) and click Reinstall Driver

Connect the JTAG flat cable between the OLIMEX JTAG connector and the IPMC Test Board JTAG connector.

3.4. USB to Serial port interfaces

This is used to communicate with the IPMC micro-controllers using their serial port. In this document we are using only one USB to Serial interface cable and move it from one micro-controller to the other. You can also use 2 USB to serial interface cables, each one connected to one micro-controller.

3.4.1. Hardware

The following equipment is needed and some cables need to be manufactured.

A USB RS232 cable, this version (version 1) having a flat one row connector at the end, reference TTL-232R-3V3

http://www.ftdichip.com/Products/Cables/USBTTLSerial.htm

http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_TTL-232R_CABLES.pdf

You can buy it from Farnell:

http://fr.farnell.com/ftdi/ttl-232r-3v3/cable-usb-vers-ttl-level-seri/dp/1329311

Or you can also use the same cable without the connector at the end (version 2) reference TTL-232R-3V3-WE

You can also buy it from Farnell:

http://fr.farnell.com/ftdi/ttl-232r-3v3-we/cable-usb-ttl-ser-conv-wire-end/dp/1740365

In order to connect it to the IPMC UART connector you need a small 3 pin connector with ref SHR-03V-S-B from manufacturer JST

http://www.jst-mfg.com/product/pdf/eng/eSH.pdf

also available from Farnell

http://fr.farnell.com/jst-japan-solderless-terminals/shr-03v-s-b/boitier-de-connecteur-3-voies/dp/1679109

with the proper wires reference SH3-SS5-28150 from manufacturer JST

also available from Farnell

http://fr.farnell.com/jst-japan-solderless-terminals/sh3-ss5-28150/fil-femelle-libre-150mm/dp/1679139

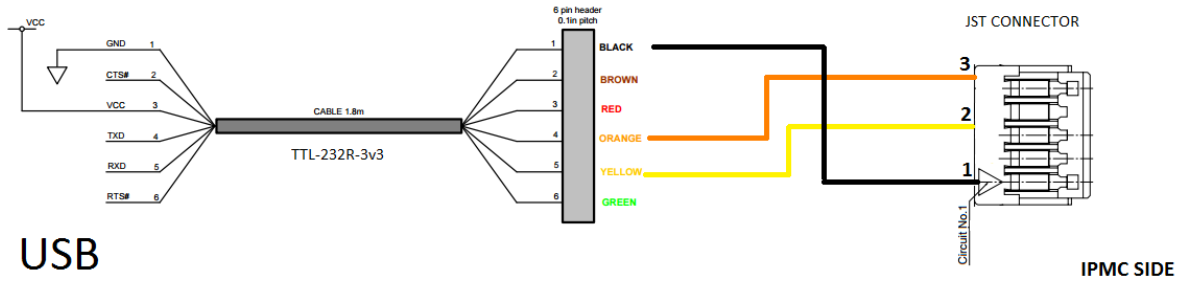
And an intermediate connector to solder the wires to for the cable with the connector at the end (version 1)

Reference 826646-6 from manufacturer TE CONNECTIVITY / AMP

Also available from Farnell

http://fr.farnell.com/te-connectivity-amp/826646-6/connecteur-6-voies-vertical-pas/dp/1772924

This is how you should do the cabling between the IPMC and both USB cables:



3.4.2. Serial Port configuration

Connect your cable USB to RS232 on your Windows 8.1 PC.

A new USB Serial Port appears. Right click on it and select properties. Go to thumb Port parameters

And set the parameters:

115200 bps
8 bits
No parity
1 Stop bit
No control flow

4. ICARE environment setup for your PC

In order to setup the ICARE environment for your PC (in this case lappc-p509)

Login to the Linux computer where the ICARE release is installed (in our case lappc-f106) with your standard account.

```
> source /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00admin/v0r5/cmt/setup.[c]sh
> mkdir ATLAS/ICARE_workarea
> cd ~/ATLAS/ICARE_workarea
> create_package.sh -a guy.perrot@cern.ch -n lappc-p509 -p Generic
```

You may want to use different directories for different project (Generic, PULSAR_IIb….)
You may know all the existing projects using the command

```
> create_package.sh –l
```

If you had already a directory from a previous version of ICARE, it is better to delete all its content

Verify in the file /ATLAS/ICARE_workarea/config/config/ioconfig.h that the handle switch uses the mode normally closed

Then compile the code:

> cd ~/ATLAS/ICARE_workarea/Generic/cmt
> make rebuild

to recompile everything and is needed the first time. After if you only want to compile your package (Generic in example) then use make bmc

5. Run Openocd

First power up and connect your setup

Start openocd:

On your Windows PC (in this case lappc-p509), Open a command window and type the following two commands:

Z:\> D:
D:\> openocd.bat

6. CPLD programming (by JTAG)

This operation needs to be done once, the first time you get a new unconfigured IPMC or for an update of the CPLD.

6.1. CPLD code

In order to program the CPLD, the code must be put at the same level as the .bat file (in example D:) on your PC.

You can get it from the Wiki page

http://lappwiki.in2p3.fr/twiki/bin/view/AtlasLapp/Informatique

The speed of the JTAG is defined inside the .svf file. By default it is set to 18 MHz.

FREQUENCY 1.80E+07 HZ;

It has to be set to 1 MHz in order to work properly with the NGX interface.

FREQUENCY 1.00E+06 HZ;

6.2. CPLD loading

Login onto the Linux machine where the ICARE environment is set (in our case lappc-f106) with your standard account.

> source /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/admin/v0r5/cmt/setup.[c]sh
> cd ~/ATLAS/ICARE_workarea/Generic/cmt
> make cpld_update

The CPLD is programmed either for the first time or with an updated version of the firmware.

If the micro-controllers on the IPMC are running you will have a problem loading the new CPLD firmware. To stop them, connect to the IPMC and halt them with the following procedure

> source /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/admin/v0r5/cmt/setup.[c]sh
> telnet lappc-p509 4444

```
Trying 134.158.99.203...
 Connected to lappc-p509.
 Escape character is '^]'.
 Open On-Chip Debugger
```

```
 > reset halt
Exit from telnet
```

7. IPMC microcontroller programming (JTAG)

7.1. OneTime Programming memory

Before using the IPMC a certain number of information must be stored in the One Time Programming (OTP) memory of the microcontrollers, like serial number of the IPMC, MAC address…

This is done only once on a new board just received after cabling or if the microcontrollers are replaced.

This requires 2 steps:

- First a special firmware to configure the OTP memory is uploaded via JTAG in the micro-controllers flash memory.
- Then using the serial port of each micro-controller, the OTP memory is updated.

7.1.1. Special firmware upload

Login onto the Linux machine where the ICARE environment is set (in our case lappc-f106) with your standard account.

In /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/otp/v0r1p3/cmt/ verify that the file requirements defines the correct PC; in our case lappc-p509

```
#-------------------------------------------------------------------------
------------

# FLASH programming

#-------------------------------------------------------------------------
------------

macro openocd_host "lappc-p509"


apply_pattern flash host=$(openocd_host) name=program_user_data
target_mcu=IPMC

apply_pattern flash host=$(openocd_host) name=program_user_data
target_mcu=IOIF


#-------------------------------------------------------------------------
------------
```

The code flash_program_user_data used for this operation is not compiled by default and need to be compiled once after the installation of the release.

```
> source /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/admin/v0r5/cmt/setup.[c]sh
> cd /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/otp/v0r1p3/cmt
> make tool
```

Check with telnet that we have the communication with the IPMC

> source /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/admin/v0r5/cmt/setup.[c]sh
> telnet lappc-p509 4444

```
Trying 134.158.99.203...
 Connected to lappc-p509.
 Escape character is '^]'.
 Open On-Chip Debugger
 > targets
    TargetName         Type       Endian TapName            State
 --  ------------------ ---------- ------ ------------------ -------
-----
  0* ipmc.cpu           cortex_m   little ipmc.cpu           halted
  1  ioif.cpu           cortex_m   little ioif.cpu           halted
```

Exit from telnet.

> source /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/admin/v0r5/cmt/setup.[c]sh
> cd /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/otp/v0r1p3/cmt
> make flash_program_user_data

The micro-controllers flash memory have been loaded with a special program to configure the OTP memory.

7.1.2. OneTime Programming memory programming

On your computer (in our case lappc-p509) connect the USB to Serial port cable and start a hyper terminal window:

Double click on the file hypertrm.exe in your hyperterminal directory.

Open a new connection, give it a name (in example Icare. It will be saved later on as Icare.ht for reuse) and set the parameter connecting using to COM3 then set the parameters of the port:

115200 bps
8 bits
No parity
1 Stop bit
No control flow

Do not forget also in parameters Configuration ASCII to check the box Reproduce locally the input characters.

You can save that configuration and then just use it next time.

Now you can store the information for the IPMC micro-controller.

Connect the other end of the USB to Serial port cable assembly on the IPMC Serial Port connector. Connect the console using the small Call Icon.

Push the IPMC reset button on the IPMC.

```
+------------------------------------------------------------+
 |                        _                                   |
 |                       | | __ _ _ __   _ __                 |
 |                       | |/ _` | '_ \| '_ \                |
```

```
 |                     | | (_| | |_) | |_) |                     |
 |                     |_|\__,_| .__/| .__/                      |
 |                             |_|   |_|                         |
 |                                                               |
 |              NON-VOLATILE USER DATA PROGRAMMING               |
 |                                                               |
 |                           =======                             |
 |                          |WARNING|                            |
 |                           =======                             |
 |                                                               |
 |   One-time programmable area, use with caution this tool.     |
 | You can abort programming at any time by resetting the MCU.   |
 +---------------------------------------------------------------+

 (GNU) gcc 4.7.0 [dbg] - Jan 8 2016, 14:42:09
 Copyright © 2013 LAPP/CNRS. All rights reserved.
   use> libopencm3 v0r3p2
   use> rcc v0r1p3
   use> gpio v1r3p4
   use> usart v0r1p2
   use> debug v0r2
   use> support v0r3
   use> otp v0r1p3


 INFO> IPMC MCU ID already programmed [0x193C].
INFO> Press <Enter> to programming PCB version '0x00020002':
```

0x193C means IPMC

The version for PCB is OK, just press enter.

Next step is for entering the serial number of the IPMC without the front 0 of the number

```
+---------------------------------------------------------------+
 |                      _                                        |
 |                     | | __ _ _ __  _ __                      |
 |                     | |/ _` | '_ \| '_ \                     |
 |                     | | (_| | |_) | |_) |                    |
 |                     |_|\__,_| .__/| .__/                     |
 |                             |_|   |_|                        |
 |                                                              |
 |              NON-VOLATILE USER DATA PROGRAMMING              |
 |                                                              |
 |                           =======                            |
 |                          |WARNING|                           |
 |                           =======                            |
 |                                                              |
 |   One-time programmable area, use with caution this tool.    |
 | You can abort programming at any time by resetting the MCU.  |
```

```
 +------------------------------------------------------------+

 (GNU) gcc 4.7.0 [dbg] - Jan 8 2016, 14:42:09
 Copyright © 2013 LAPP/CNRS. All rights reserved.
   use> libopencm3 v0r3p2
   use> rcc v0r1p3
   use> gpio v1r3p4
   use> usart v0r1p2
   use> debug v0r2
   use> support v0r3
   use> otp v0r1p3


 INFO> IPMC MCU ID already programmed [0x193C].
 INFO> PCB version already programmed [0x00020002].
 INFO> Press <Enter> to programming Serial Number '66/0x00000042':
 INFO> Start programming of the OTP data area...OK
 INFO> Check Serial Number...OK

INFO> **** Enter in an infinite loop ****
```

The IPMC micro-controller now has its OTP flash memory programmed. Next step is to do the same thing for the IOIF micro-controller.

Disconnect (End Icon) in the hypertrm console.

Move the cable connector on the IOIF Serial Port connector.

Connect (Call Icon) in the hypertrm console and push the IOIF reset button on the IPMC.

```
+------------------------------------------------------------+
 |                       _                                    |
 |                      | | __ _ _ __  _ __                   |
 |                      | |/ _` | '_ \| '_ \                  |
 |                      | | (_| | |_) | |_) |                 |
 |                      |_|\__,_| .__/| .__/                  |
 |                              |_|   |_|                     |
 |                                                            |
 |              NON-VOLATILE USER DATA PROGRAMMING            |
 |                                                            |
 |                           =======                          |
 |                          |WARNING|                         |
 |                           =======                          |
 |                                                            |
 |   One-time programmable area, use with caution this tool.  |
 | You can abort programming at any time by resetting the MCU. |
 +------------------------------------------------------------+

 (GNU) gcc 4.7.0 [dbg] - Jan 8 2016, 14:42:09
 Copyright © 2013 LAPP/CNRS. All rights reserved.
   use> libopencm3 v0r3p2
   use> rcc v0r1p3
   use> gpio v1r3p4
```

```
   use> usart v0r1p2
   use> debug v0r2
   use> support v0r3
   use> otp v0r1p3


INFO> Press <Enter> to programming IOIF MCU ID '0x101F':
```

0x101F means IOIF

Then continue with PCB version, Serial Number and MAC Address. The MAC address is derived from the Serial Number of the IPMC. Explanations are given in the Twiky there
http://lappwiki.in2p3.fr/twiki/pub/AtlasLapp/ATCA/IPMC_MAC_Address.txt

```
+-------------------------------------------------------------+
 |                       _                                     |
 |                      | | __ _ _ __  _ __                    |
 |                      | |/ _` | '_ \| '_ \                   |
 |                      | | (_| | |_) | |_) |                  |
 |                      |_|\__,_| .__/| .__/                   |
 |                              |_|   |_|                      |
 |                                                             |
 |              NON-VOLATILE USER DATA PROGRAMMING             |
 |                                                             |
 |                           =======                           |
 |                          |WARNING|                          |
 |                           =======                           |
 |                                                             |
 |   One-time programmable area, use with caution this tool.   |
 | You can abort programming at any time by resetting the MCU. |
 +-------------------------------------------------------------+

(GNU) gcc 4.7.0 [dbg] - Jan 8 2016, 14:42:09
 Copyright © 2013 LAPP/CNRS. All rights reserved.
   use> libopencm3 v0r3p2
   use> rcc v0r1p3
   use> gpio v1r3p4
   use> usart v0r1p2
   use> debug v0r2
   use> support v0r3
   use> otp v0r1p3


INFO> Press <Enter> to programming IOIF MCU ID '0x101F':
INFO> Start programming of the OTP data area...OK
INFO> Check MCU ID...OK
INFO> Press <Enter> to programming PCB version '0x00020002':
INFO> Start programming of the OTP data area...OK
INFO> Check PCB version...OK
INFO> Press <Enter> to programming Serial Number '66/0x00000042':
INFO> Start programming of the OTP data area...OK
INFO> Check Serial Number...OK
```

```
INFO> Press <Enter> to programming MAC address '0x000000228f024042':
INFO> Start programming of the OTP data area...OK
INFO> Check MAC address...OK

INFO> **** Enter in an infinite loop ****
```

The OTP flash memory zones for both micro-controllers are now up to date.

Next step will be to load the firmware in the flash memory of the micro-controllers.

7.2. Micro-controller Internal Flash Memory Programming by JTAG

Login onto the Linux machine where the ICARE environment is set (in our case lappc-f106) with your standard account.

In the shell:

We verify that we have access to the IPMC.

```
> source /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/admin/v0r5/cmt/setup.[c]sh
> cd ~/ATLAS/ICARE_workarea/Generic/cmt/
> source setup.[c]sh
> make info
```

```
============================
IPMC - OTP area information
============================
OpenOCD server: lappc-p509

MCU ID:          0x193c
PCB version:     v2.2
Serial Number:   113
--------------------------

============================
IOIF - OTP area information
============================
OpenOCD server: lappc-p509

MCU ID:          0x101f
PCB version:     v2.2
Serial Number:   113
MAC address:     00:22:8f:02:40:71
--------------------------
```

Micro-controllers flash memory can be programmed one at a time:

```
> cd ~/ATLAS/ICARE_workarea/Generic/cmt
> make bmc_IPMC_flash
> make bmc_IOIF_flash
```

Or both at the same time:

```
> cd ~/ATLAS/ICARE_workarea/Generic/cmt
> make flash_bmc

-----------------------------------------------------

OpenOCD server: lappc-p509

target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x61000000 pc: 0x0800104a msp: 0x2001ff28
background polling: on
TAP: ipmc.cpu (enabled)
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x61000000 pc: 0x0800104a msp: 0x2001ff28
Loading section .text, size 0x42730 lma 0x8000000
Loading section .ARM.exidx, size 0xd8 lma 0x8042730
Loading section .data, size 0x2db4 lma 0x8042808
Loading section .ARM.extab, size 0x30 lma 0x80455bc
Start address 0x8000c41, load size 284140
Transfer rate: 22 KB/sec, 11839 bytes/write.
requesting target halt and executing a soft reset
soft_reset_halt is deprecated, please use 'reset halt' instead.
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x080009ac msp: 0x20020000
-----------------------------------------------------

#CMT---> Info: Document bmc_IOIF_flash

-----------------------------------------------------
OpenOCD server: lappc-p509

background polling: on
TAP: ioif.cpu (enabled)
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000b44 msp: 0x20020000
Loading section .text, size 0x2f848 lma 0x8000000
Loading section .data, size 0x2c20 lma 0x802f848
Start address 0x8000c41, load size 205928
Transfer rate: 22 KB/sec, 12113 bytes/write.
requesting target halt and executing a soft reset
soft_reset_halt is deprecated, please use 'reset halt' instead.
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000c40 msp: 0x20020000
-----------------------------------------------------
```

After loading the code in the internal flash memory you need to run it, either using the command make run or resetting both controllers using their respective reset push button.

Note that for the programs to run correctly you need to have a correct geographical address set on the IPMC Test Board:
This address cannot be anything; it must be a valid address as defined by the norm and it includes a parity bit. In our setup we have jumpers on HA1, 2, 4, 5, 7

7.3. External Flash Programming (Default factory firmware)

In order to upload the default factory firmware inside the external flash memory, you need to use Ethernet.

You can use the Ethernet RJ45 connector on the IPMC Test Board to connect the IPMC to the network. The procedure needs the IP address attributed by sysadmin to the MAC address of the IPMC. This address must be fixed and not dynamic. In this example it is 134.158.98.182

You can obtain the IP address using the command make info described earlier.

```
#CMT---> Info: Document mcu_info

============================
IPMC - OTP area information
============================
OpenOCD server: lappc-p509

MCU ID:          0x193c
PCB version:     v2.2
Serial Number:   113
--------------------------


============================
IOIF - OTP area information
============================
OpenOCD server: lappc-p509

MCU ID:          0x101f
PCB version:     v2.2
Serial Number:   113
MAC address:     00:22:8f:02:40:71
IP address:      134.158.98.182
--------------------------
```

Beware that the number displayed will only be valid if the IOIF code with Ethernet access for the IOIF controller has been loaded in the flash.

The procedure for upgrading the firmware id detailed in the README file of the ICARE release.

http://lappwiki.in2p3.fr/twiki/bin/view/AtlasLapp/Informatique

And the commands to use for uploading the default factory firmware is

> source /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/admin/v0r5/cmt/setup.[c]sh
> cd ~/ATLAS/ICARE_workarea/Generic/cmt/
> source setup.[c]sh
> fwu –F -t IPMC -f ../arm-gcc47-dbg/bmc_IPMC.bin -n 134.158.98.182

> fwu –F -t IOIF -f ../arm-gcc47-dbg/bmc_IOIF.bin -n 134.158.98.182

The –F option tells that the firmware should be uploaded in the default factory part of the external flash memory.

For the time being there is a bug which prevents using 10MbE. Only 100MbE works.

7.4. Verification

7.4.1. Standalone test

To check that micro-controllers flash memory is properly loaded, use the serial port of the micro-controller with the hypertrm window and push the reset button. Repeat the operation for the second micro-controller. This is what you should see:

IPMC micro-controller:

```
+-----------------------------------------------------------+
|                  ___ ___   _   ___ ___                    |
|                 |_ _/ __| /_\ | _ \ __|                   |
|                  | | (__ / _ \|   / _|                    |
|                 |___\___/_/ \_\_|_\___|                   |
|                                                           |
|  Intelligent plateform management Controller softwARE     |
|                    Target MCU: IPMC                       |
|                                                           |
+-----------------------------------------------------------+
Binary image: bmc_IPMC.bin
(GNU) gcc 4.7.0 [dbg] - Jan 11 2016, 16:03:38
Copyright (c) 2012-2015, LAPP/CNRS. All rights reserved.
  use> resourceBroker v0r1
  use> queue v0r2
  use> libopencm3 v0r3p2
  use> otp v0r1p3
  use> rcc v0r1p3
  use> gpio v1r3p4
  use> usart v0r1p2
  use> taskQueue v0r2p2
  use> led v0r1p2
  use> module v0r2p1
  use> debug v0r2
  use> support v0r3
  use> ekeying v0r1p2
  use> lwip v1.4.0
  use> spi v0r4p1
  use> cpld v0r1
  use> main v0r1p1
  use> iopin v0r3
  use> config v0r2
  use> i2c v3r2p2
  use> sensors v0r2p2
  use> dataStorage v0r3
  use> ipmb v0r4
  use> messageQueue v1r2
```

```
  use> nsBroker v0r2
  use> imc v0r6p3
  use> channel v0r5
  use> messageDispatcher v0r2p2
  use> fwUpgrade v0r1p2
  use> ipmc v0r10
  use> Generic v0r1


Reset cause: NRST pin reset
hw_get_board_address(91)addr(49) OK
io_core_init(38)local(92)remote(20)
INFO> [IPMB-A] addr=0x49 base=0x40005800 channel ID=2 index=1
INFO> [IPMB-B] addr=0x49 base=0x40005c00 channel ID=3 index=2
[CH-IMC] try to send: >> 193c->101f Seq00 Cmd2e:05 [OK/IDLE]
(IMC-C) 00:00:04 >> 193c->101f Seq00 Cmd2e:05 [00 00 00 00 ed 01 00
00]
(IMC-P) 00:00:04 << 101f->193c Seq00 Cmd2f:05 [00 00 00 00 00 ed ...
49 21 01 4a
 41 01]
[CH-IMC] try to send: >> 193c->101f Seq00 Cmd2e:05 [OK/IDLE]
(IMC-C) 00:00:04 >> 193c->101f Seq00 Cmd2e:05 [ed 01 00 00 ed 01 00
00]
(IMC-P) 00:00:04 << 101f->193c Seq00 Cmd2f:05 [00 ed 01 00 00 ed ...
ff ff ff ff
 ff ff]
[CH-IMC] try to send: >> 193c->101f Seq01 Cmd2e:05 [OK/IDLE]
(IMC-C) 00:00:04 >> 193c->101f Seq01 Cmd2e:05 [da 03 00 00 ed 01 00
00]
(IMC-P) 00:00:04 << 101f->193c Seq01 Cmd2f:05 [00 da 03 00 00 ed ...
ff ff ff ff
 ff ff]
[CH-IMC] try to send: >> 193c->101f Seq02 Cmd2e:05 [OK/IDLE]
(IMC-C) 00:00:04 >> 193c->101f Seq02 Cmd2e:05 [c7 05 00 00 ed 01 00
00]
(IMC-P) 00:00:04 << 101f->193c Seq02 Cmd2f:05 [00 c7 05 00 00 ed ...
ff ff ff ff
 ff ff]
[CH-IMC] try to send: >> 193c->101f Seq03 Cmd2e:05 [OK/IDLE]
(IMC-C) 00:00:04 >> 193c->101f Seq03 Cmd2e:05 [b4 07 00 00 4c 00 00
00]
(IMC-P) 00:00:04 << 101f->193c Seq03 Cmd2f:05 [00 b4 07 00 00 4c ...
ff ff ff ff
 ff ff]
[CH-IMC] try to send: >> 193c->101f Seq04 Cmd2e:07 [OK/IDLE]
(IMC-C) 00:00:04 >> 193c->101f Seq04 Cmd2e:07 [00 08 00 00 ed 01 00
00]
(IMC-P) 00:00:04 << 101f->193c Seq04 Cmd2f:07 [00 00 08 00 00 ed ...
ff ff ff ff
 ff ff]
[CH-IMC] try to send: >> 193c->101f Seq05 Cmd2e:07 [OK/IDLE]
```

```
(IMC-C) 00:00:04 >> 193c->101f Seq05 Cmd2e:07 [ed 09 00 00 ed 01 00
00]
(IMC-P) 00:00:04 << 101f->193c Seq05 Cmd2f:07 [00 ed 09 00 00 ed ...
ff ff ff ff
 ff ff]
[CH-IMC] try to send: >> 193c->101f Seq06 Cmd2e:07 [OK/IDLE]
(IMC-C) 00:00:04 >> 193c->101f Seq06 Cmd2e:07 [da 0b 00 00 ed 01 00
00]
(IMC-P) 00:00:04 << 101f->193c Seq06 Cmd2f:07 [00 da 0b 00 00 ed ...
ff ff ff ff
 ff ff]
[CH-IMC] try to send: >> 193c->101f Seq07 Cmd2e:07 [OK/IDLE]
(IMC-C) 00:00:04 >> 193c->101f Seq07 Cmd2e:07 [c7 0d 00 00 ed 01 00
00]
(IMC-P) 00:00:04 << 101f->193c Seq07 Cmd2f:07 [00 c7 0d 00 00 ed ...
ff ff ff ff
 ff ff]
[CH-IMC] try to send: >> 193c->101f Seq08 Cmd2e:07 [OK/IDLE]
(IMC-C) 00:00:04 >> 193c->101f Seq08 Cmd2e:07 [b4 0f 00 00 4c 00 00
00]
(IMC-P) 00:00:04 << 101f->193c Seq08 Cmd2f:07 [00 b4 0f 00 00 4c ...
ff ff ff ff
 ff ff]
fru_state_check(1324)FRU state init
fru_m0(1041)#0
fru_set_power_level(1500)#0(M0):level(0)
fru_add(1038):fru(0)(A0.60) hw_addr(49) size(123)
fru_set_power_level(1500)#1(M0):level(0)
 fru(0) level(1) draw(80W)
 fru(1) level(1) draw(0W)
fru_add(1038):fru(1)(F2.60) hw_addr(49) size(431)
fru_handle_power(1362)#0(M0):Pwr(OFF)
```

IOIF micro-controller:

```
+-----------------------------------------------------+
|                  ___ ___   _   ___ ___              |
|                 |_ _/ __| /_\ | _ \ __|             |
|                  | | (__ / _ \|   / _|              |
|                 |___\___/_/ \_\_|_\___|             |
|                                                     |
|  Intelligent plateform management Controller softwARE |
|                   Target MCU: IOIF                  |
|                                                     |
+-----------------------------------------------------+
Binary image: bmc_IOIF.bin
(GNU) gcc 4.7.0 [dbg] - Jan 11 2016, 16:03:38
Copyright (c) 2012-2015, LAPP/CNRS. All rights reserved.
  use> resourceBroker v0r1
  use> queue v0r2
  use> libopencm3 v0r3p2
  use> otp v0r1p3
```

```
  use> rcc v0r1p3
  use> gpio v1r3p4
  use> usart v0r1p2
  use> taskQueue v0r2p2
  use> led v0r1p2
  use> module v0r2p1
  use> debug v0r2
  use> support v0r3
  use> ekeying v0r1p2
  use> lwip v1.4.0
  use> spi v0r4p1
  use> cpld v0r1
  use> main v0r1p1
  use> iopin v0r3
  use> config v0r2
  use> i2c v3r2p2
  use> sensors v0r2p2
  use> dataStorage v0r3
  use> ipmb v0r4
  use> messageQueue v1r2
  use> nsBroker v0r2
  use> imc v0r6p3
  use> channel v0r5
  use> messageDispatcher v0r2p2
  use> fwUpgrade v0r1p2
  use> ipmc v0r10
  use> Generic v0r1


Reset cause: NRST pin reset
INFO> [I2C_MGT] addr=0x70 base=0x40005400 channel ID=1 index=0
INFO> [I2C_SENSOR] addr=0x71 base=0x40005800 channel ID=2 index=1
INFO> [I2C_<USR|IPM_IO>] addr=0x72 base=0x40005c00 channel ID=3
index=2
DEBUG> [SensorsInit]: name="LTC4151 12V Voltage" channel=3
address=0x6a ID=44
DEBUG> [SensorsInit]: name="LTC4151 12V Current" channel=3
address=0x6a ID=43
DEBUG> [SensorsInit]: name="AD7414 Temp" channel=3 address=0x48
ID=42
INFO> ethernet_bsp_autonegotiate: 100 Mbit/s, FULL duplex
INFO> MAC address: 00:22:8F:02:40:71
INFO> Looking for DHCP server please wait...
INFO> IP address assigned by a DHCP server 134.158.98.182
INFO> IP: 134.158.98.182
INFO> Netmask: 255.255.252.0
INFO> Gateway: 134.158.96.1
INFO> [F/W upgrade] enable channel
INFO> [134.158.98.182:5555/tcp] TCP_ServerInit: service 'Firmware
Upgrade' is ac
tive
00:00:29
```

7.4.2. Shelf test

Put the IPMC on your blade or our carrier blade and connect to the shelf manager using the web interface. In our case http://lapp-sm10.in2p3.fr/

Use Board Information to see all blades present in the shelf.

8. Ethernet firmware upgrade

You can use the Ethernet RJ45 connector on the IPMC Test Board to connect the IPMC to the network.

The MAC address has been declared and you have a corresponding IP address which must be of the fixed type (not dynamic)

The procedure for upgrading the firmware using the IPMC IP address is detailed in the README file of the ICARE release.

http://lappwiki.in2p3.fr/twiki/bin/view/AtlasLapp/Informatique

9. To get information on the firmware in the IPMC

To get generic information:

```
> source /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/admin/v0r5/cmt/setup.[c]sh
> cd ~/ATLAS/ICARE_workarea/Generic/cmt/
> source setup.[c]sh
> make version
```

```
#CMT---> Info: Document mcu_version

==================================
IPMC - Version
==================================
OpenOCD server: lappc-p509

Release ver.: ICARE-00-01-00
Compilator  : (GNU) gcc 4.7.0 [dbg]
Binary image: bmc_IPMC.bin
Build ver.  : Jan 11 2016, 16:03:38
----------------------------------


==================================
IOIF - Version
==================================
OpenOCD server: lappc-p509

Release ver.: ICARE-00-01-00
Compilator  : (GNU) gcc 4.7.0 [dbg]
Binary image: bmc_IOIF.bin
Build ver.  : Jan 11 2016, 16:03:38
----------------------------------
```

To get more detailed information about the packages used:

```
> source /software_dev/atlas/project/ICARE/releases/ICARE-00-01-00/admin/v0r5/cmt/setup.[c]sh
> cd ~/ATLAS/ICARE_workarea/Generic/cmt/
> source setup.[c]sh
> make full_version
```

```
#CMT---> Info: Document mcu_full_version

==================================
IPMC - version
==================================
OpenOCD server: lappc-p509

Release ver.: ICARE-00-01-00
Compilator  : (GNU) gcc 4.7.0 [dbg]
Binary image: bmc_IPMC.bin
Build ver.  : Jan 11 2016, 16:03:38
Use packages:
            - resourceBroker v0r1
            - queue v0r2
            - libopencm3 v0r3p2
            - otp v0r1p3
            - rcc v0r1p3
            - gpio v1r3p4
            - usart v0r1p2
            - taskQueue v0r2p2
            - led v0r1p2
            - module v0r2p1
            - debug v0r2
            - support v0r3
            - ekeying v0r1p2
            - lwip v1.4.0
            - spi v0r4p1
            - cpld v0r1
            - main v0r1p1
            - iopin v0r3
            - config v0r2
            - i2c v3r2p2
            - sensors v0r2p2
            - dataStorage v0r3
            - ipmb v0r4
            - messageQueue v1r2
            - nsBroker v0r2
            - imc v0r6p3
            - channel v0r5
            - messageDispatcher v0r2p2
            - fwUpgrade v0r1p2
            - ipmc v0r10
            - Generic v0r1
```

```
----------------------------------


==================================
IOIF - version
==================================
OpenOCD server: lappc-p509

Release ver.: ICARE-00-01-00
Compilator  : (GNU) gcc 4.7.0 [dbg]
Binary image: bmc_IOIF.bin
Build ver.  : Jan 11 2016, 16:03:38
Use packages:
            - resourceBroker v0r1
            - queue v0r2
            - libopencm3 v0r3p2
            - otp v0r1p3
            - rcc v0r1p3
            - gpio v1r3p4
            - usart v0r1p2
            - taskQueue v0r2p2
            - led v0r1p2
            - module v0r2p1
            - debug v0r2
            - support v0r3
            - ekeying v0r1p2
            - lwip v1.4.0
            - spi v0r4p1
            - cpld v0r1
            - main v0r1p1
            - iopin v0r3
            - config v0r2
            - i2c v3r2p2
            - sensors v0r2p2
            - dataStorage v0r3
            - ipmb v0r4
            - messageQueue v1r2
            - nsBroker v0r2
            - imc v0r6p3
            - channel v0r5
            - messageDispatcher v0r2p2
            - fwUpgrade v0r1p2
            - ipmc v0r10
            - Generic v0r1
----------------------------------
```