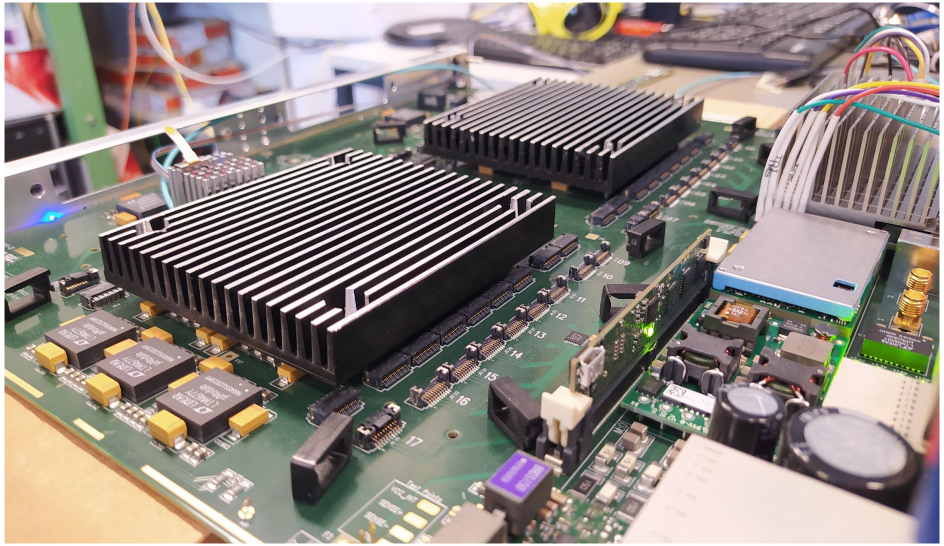
SPRACE

OpenIPMC-HW

An open-source mezzanine with a customizable firmware for electronic applications

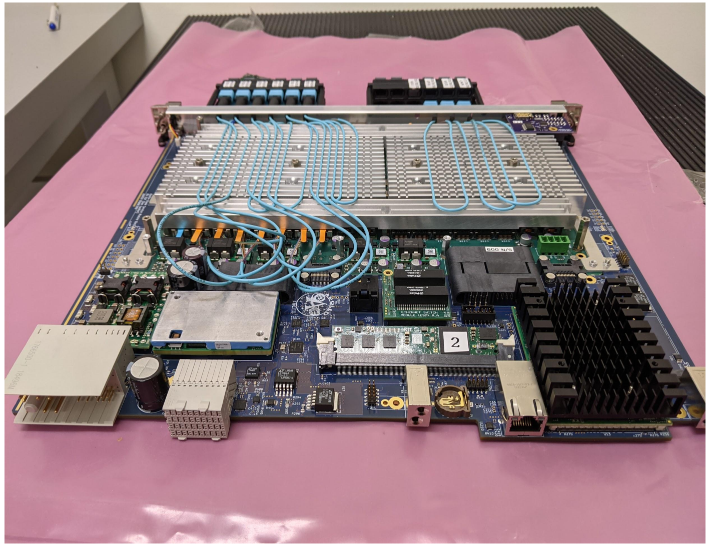
André Cascadan, Luigi Calligaris, Antonio Bassi (SPRACE - UNESP)

ATCA boards for CMS Phase-2 tracker



Serenity board (Imperial College London & KIT)

- Outer Tracker Data Trigger and Control (DTC)
- Total of 216 to be used
- ATCA standard, swappable FPGA mezzanine



Apollo board (Boston and Cornell Universities)

- Inner Tracker DTC and Track Finder
- Total of 28 (DTC) + 162 (TF) to be used
- ATCA standard, two-pieces board
  - Services + Optical/Processing

A note on naming

- **OpenIPMC(-SW)** → Software for IPMI function
- **OpenIPMC-HW** → IPMC mezzanine board
- **OpenIPMC-FW** → Firmware for the mezzanine

OpenIPMC-SW software

- PICMG-compliant IPMI functions
  - Power negotiation and hot-swap (M-states, handle, etc.)
  - Instantiate board sensor records, declare them to ShM, read-out and publish data
  - Focus on simplicity: optional functions can be added to the project by the user
- Portable, platform-independent design, written in C
  - Can quickly port the project to different architectures (e.g. ZynqMP, ESP32, STM32)
- Based on FreeRTOS operating system
  - Can run independent “tasks” in parallel. (w/ prioritization)
  - Flexible software development, thanks to task decoupling.
  - Supported by many SoC manufacturers. (TI, NXP, ST, Xilinx, Microsemi…)
- OpenIPMC is free and open source software
  - Can be easily customized to fit a new board, and modified to be debugged.
  - No need to sign NDAs for contributors, curious newcomers and students.

OpenIPMC-HW mezzanine

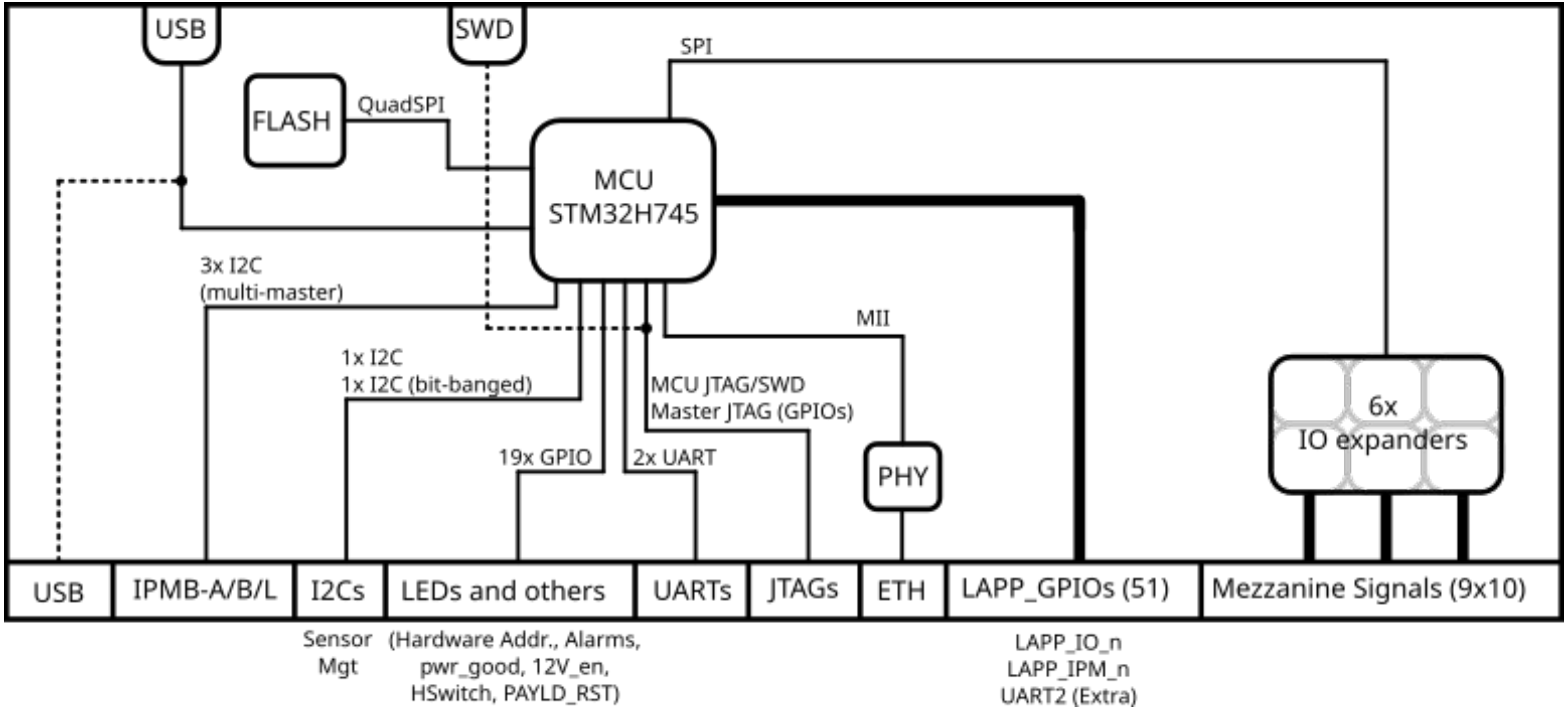
- Open hardware platform for IPMC software (like OpenIPMC)
  - Developed in collaboration with **Karlsruhe Institute of Technology (KIT).**
  - Complements OpenIPMC to realize a fully open IPMC platform.
  - Designed using the CERN-sponsored KiCAD EDA suite.

- Compatible with LAPP/CERN IPMC carrier board interface
  - **JEDEC MO-244 LP-DIMM** form factor: 82,0 x 18,3 mm.
  - Compatible with the new UART scheme (SoC debug via SOL + IPMC debug + SoC configuration).
  - 9 sets of AMC control pins + all remaining LAPP GPIOs routed to the DIMM connector.

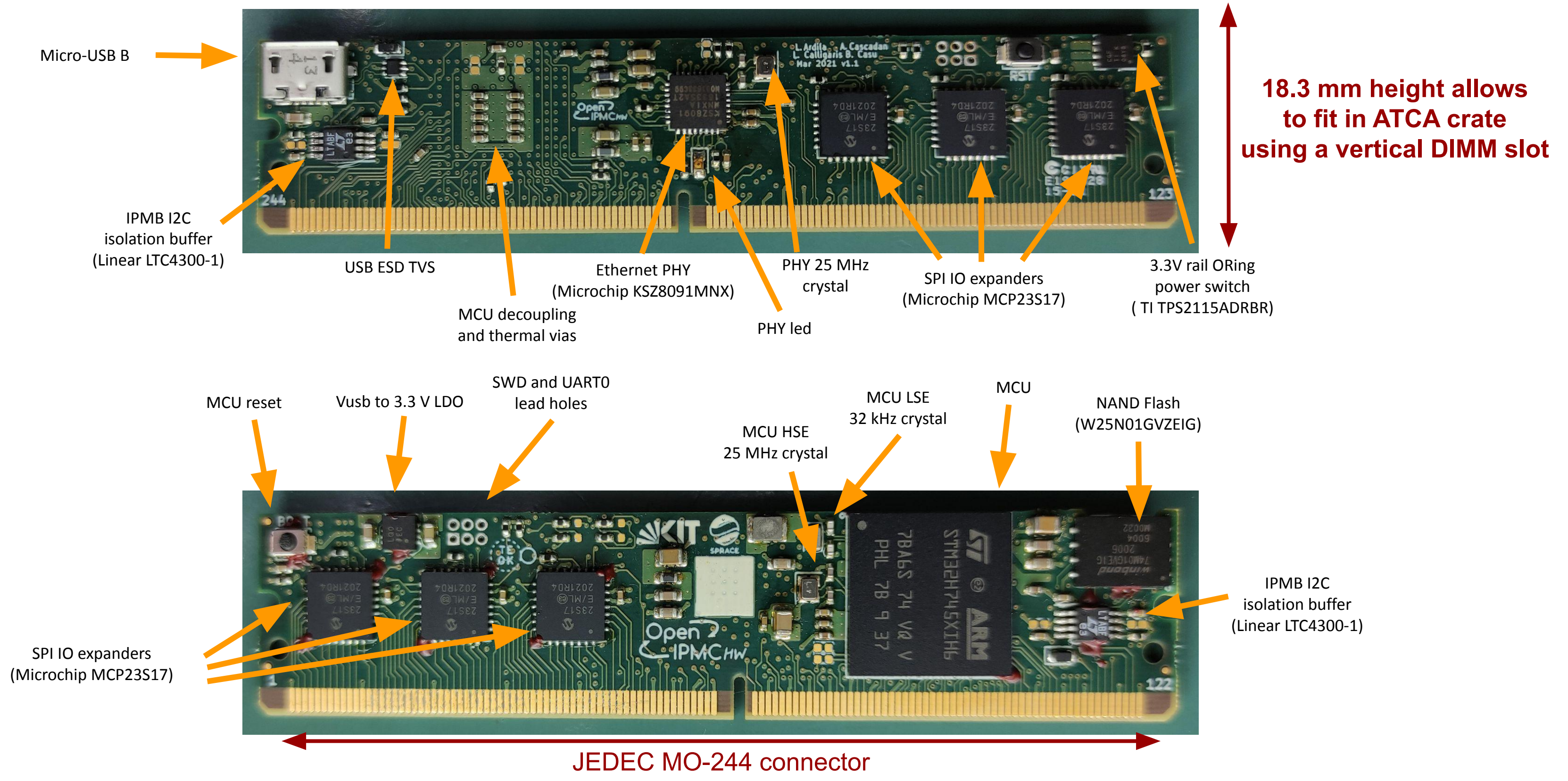
- Hardware components
  - Low-cost components, in current production, with long lifecycle.
  - STMicroelectronics **STM32H745** Microcontroller.
    - Extensive documentation & free toolchain/IDE software (STM32CubeIDE).

- Relatively simple manufacturing
  - 8 layers FR4, no buried/blind vias, 0.8 mm pitch BGA, needs hard gold edge fingers.

OpenIPMC-HW layout: schematic



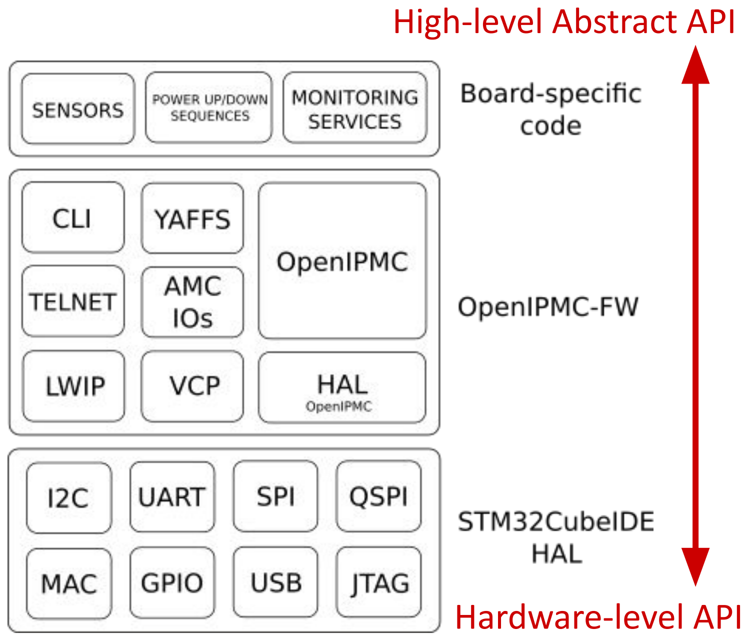
OpenIPMC-HW layout: picture

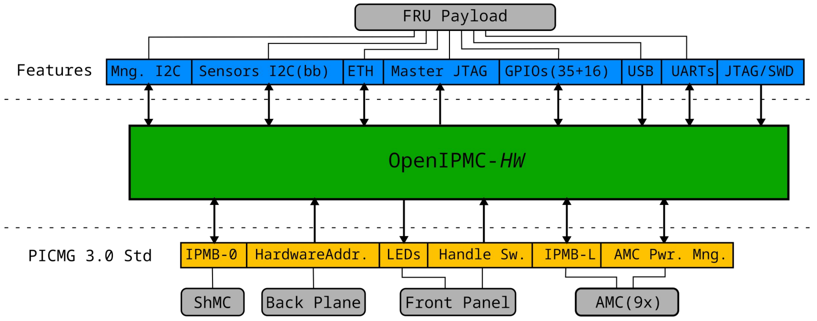


Micro-USB B
IPMB I2C isolation buffer (Linear LTC4300-1)
USB ESD TVS
MCU decoupling and thermal vias
Ethernet PHY (Microchip KSZ8091MNX)
PHY led
PHY 25 MHz crystal
SPI IO expanders (Microchip MCP23S17)
3.3V rail ORing power switch (TI TPS2115ADRBR)
18.3 mm height allows to fit in ATCA crate using a vertical DIMM slot
MCU reset
Vusb to 3.3 V LDO
SWD and UART0 lead holes
MCU HSE 25 MHz crystal
MCU LSE 32 kHz crystal
MCU
NAND Flash (W25N01GVZEIG)
IPMB I2C isolation buffer (Linear LTC4300-1)
SPI IO expanders (Microchip MCP23S17)
JEDEC MO-244 connector

OpenIPMC-FW

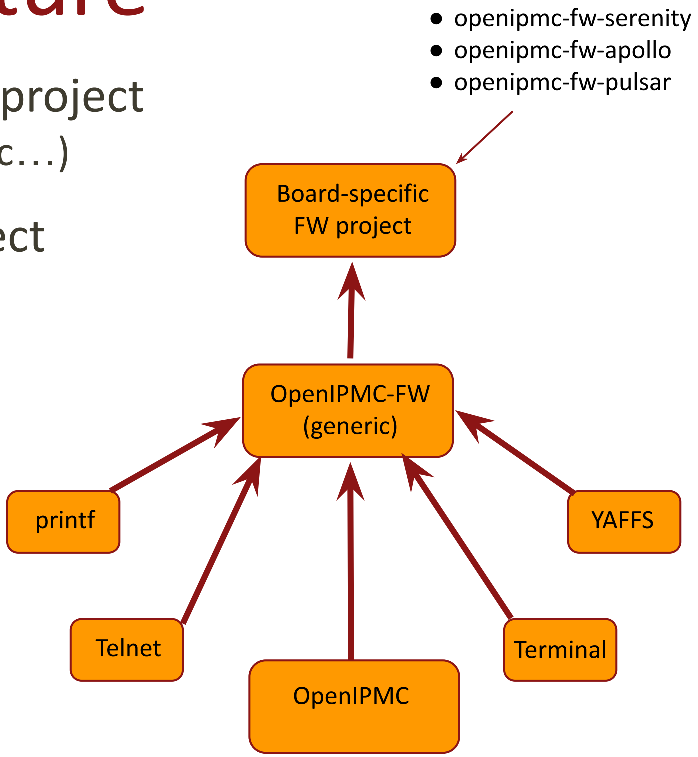
- Characteristics
  - Developed with **STM32CubeIDE** (supports Make as well).
  - Multitasking provided by FreeRTOS.
- Components
  - OpenIPMC-FW
    - Access to the resources available on the mezzanine
      - File system on flash memory, Network
    - Common services
      - CLI/printf via Telnet, UART and USB VCP.
      - Xilinx XVC (to be implemented).
    - Common interfaces to the ATCA board.
      - Common to all carriers: I2C, GPIOs, UART, JTAG.
    - **Common API**
  - OpenIPMC-FW board-specific code
    - High-level functions specific to each ATCA board
      - Sensors, boot sequence, monitoring policy…
    - Access to the ATCA board through the **common API**



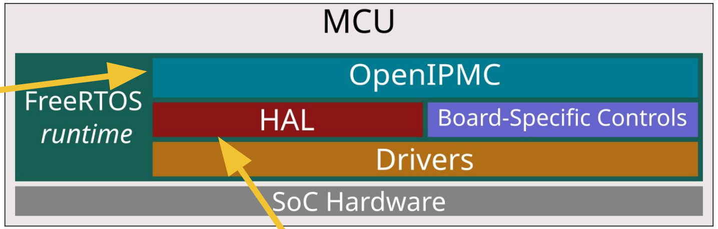


OpenIPMC-FW Repo Structure

- Each board has its own custom OpenIPMC-FW project
  - Accounts for what sits in the ATCA carrier (sensors, etc…)
- OpenIPMC-FW sits as a submodule of the project
  - Board-specific project adds OpenIPMC-FW into it
  - Clear code separation
  - Keep IPMC tools dev. independent from the board
  - Simplify code updates
  - Contains main makefile
- Example board-specific repo
  - OpenIPMC-FW-board-specific-demo
  - Basic example code
  - Reference makefile to integrate board-specific code
  - gitlab-ci.yml for automatic build
    - Generates binary and HPM.1 remote upgrade file



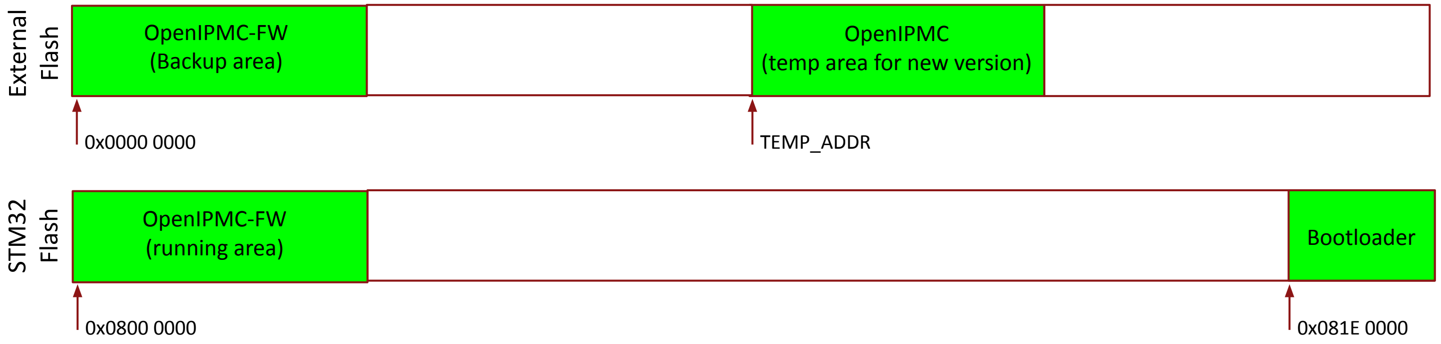
Implementation of HPM.1



- OpenIPMC (as a portable submodule)
  - Interprets the IPMI commands associated to HPM.1
  - Manages the *long-duration* commands
    - IPMI commands which take more the standard response time (eg: generate backup copy)
  - Use the HAL interface to call hardware-aware functions
    - Initiate backup copy, Upload FW Image Bytes, Activate FW, Rollback, etc.
- OpenIPMC-FW
  - Initializes OpenIPMC submodule with upgrade capabilities and current FW info
  - Implements the upgrade HAL callbacks, which involve operation over hardware
    - Read from internal STM32 Flash
    - Read/write over external Flash on DIMM
    - Control OpenIPMC-FW bootloader
    - System Reset, Control STM32 Option Bits (change boot address)
- Bootloader
  - A Bootloader was developed to help the image swap in the STM32 internal Flash (process described later)
  - It is a separate project available here
  - It must be installed at final portions of the STM32 Internal Flash

FW Upgrade steps

1. The current OpenIPMC-FW makes a copy of itself into the External Flash (backup)
2. The current FW receives the new version via IPMI and saves it into the External Flash (temp)
3. The current FW reboots changing the boot address to 0x081E 0000 (bootloader)
4. The bootloader runs, and copy the new FW to the 0x0800 0000 position
5. Bootloader jumps to the FW at 0x0800 0000


External Flash
OpenIPMC-FW (Backup area)
OpenIPMC (temp area for new version)
0x0000 0000
TEMP_ADDR
STM32 Flash
OpenIPMC-FW (running area)
Bootloader
0x0800 0000
0x081E 0000

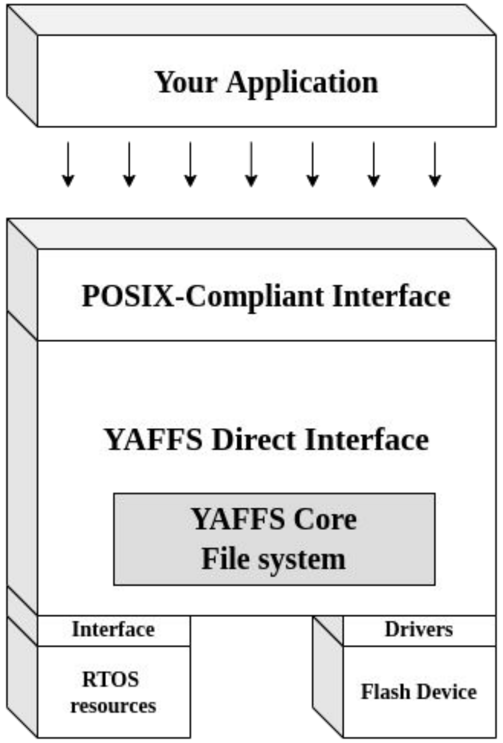
File system in OpenIPMC-HW

- Flash memory on **OpenIPMC-HW DIMM** card
  - 3V, 1G-bit NAND flash Winbond **W25N01GVZEIG** (or compatible **W74M01GVZEIG**).
  - Connected via **QuadSPI** bus to the MCU.
- YAFFS
  - “Yet Another Flash File System”
  - Designed specifically for NAND and NOR chips using wear-leveling techniques.
- Embedded robustness
  - Supports SLC as well as MLC flash devices.
  - Low run-time RAM footprint.

YAFFS-2 Integration Process

- YAFFS Direct Interface (YDI)
  - Provides a set of POSIX-compliant functions.
  - Ready to configure environment. System integrator needs to provide a few callbacks for RTOS interfacing and resource access.
  - Built-in debugging mechanism : YAFFS trace.
- Flash Device
  - For flash device operations to be executed, some driver callbacks must be defined.
  - Flash device driver functions must be provided.
  - Flexible partition control : YAFFS parameter tuning.
- Performance Evaluation
  - YAFFS was first implemented on a emulated flash device using a static RAM memory inside the MCU in order to debug and pinpoint possible integration errors.

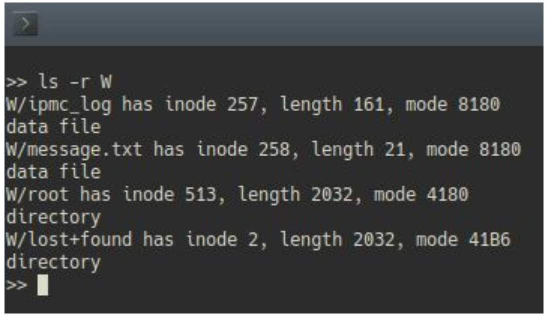
YAFFS Integration Building Blocks



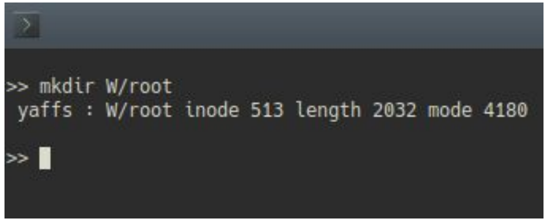
Adapted from https://yaffs.net/documents/yaffs-direct-interface

YAFFS-2 Integration into FW

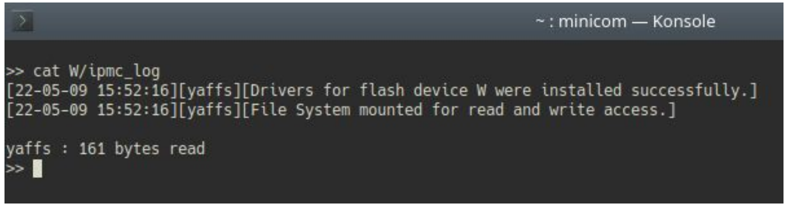
- UNIX-style Command Line Interface (CLI)
  - Running on **openIPMC-FW** terminal process.
  - mount, umount, ls, cd, mkdir, rm…
  - CLI implements POSIX-Compliant functions to manage file system.
- File System Utilities
  - YAFFS is currently operating on an onboard 128 MB (1G-bit) flash memory.
  - A logging task was implemented (ipmc_scribe).
- Preliminary tests
  - Robust against power failures.
  - Mounting time not perceivable from CLI.



```
>> ls -r W
W/ipmc_log has inode 257, length 161, mode 8180
data file
W/message.txt has inode 258, length 21, mode 8180
data file
W/root has inode 513, length 2032, mode 4180
directory
W/lost+found has inode 2, length 2032, mode 41B6
directory
>>
```



```
>> mkdir W/root
 yaffs : W/root inode 513 length 2032 mode 4180

>>
```

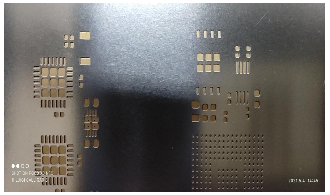
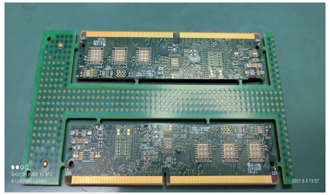
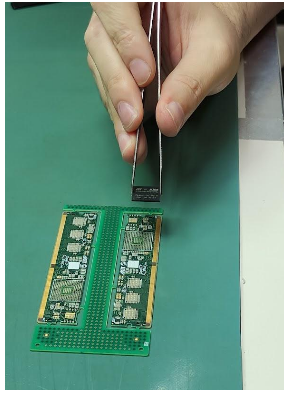
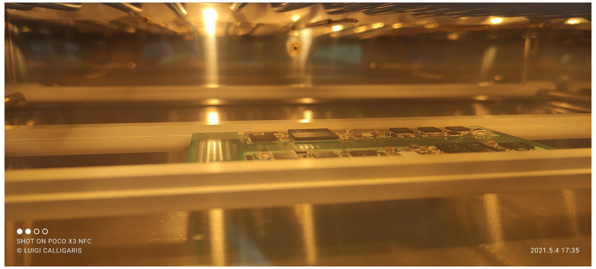


~: minicom — Konsole

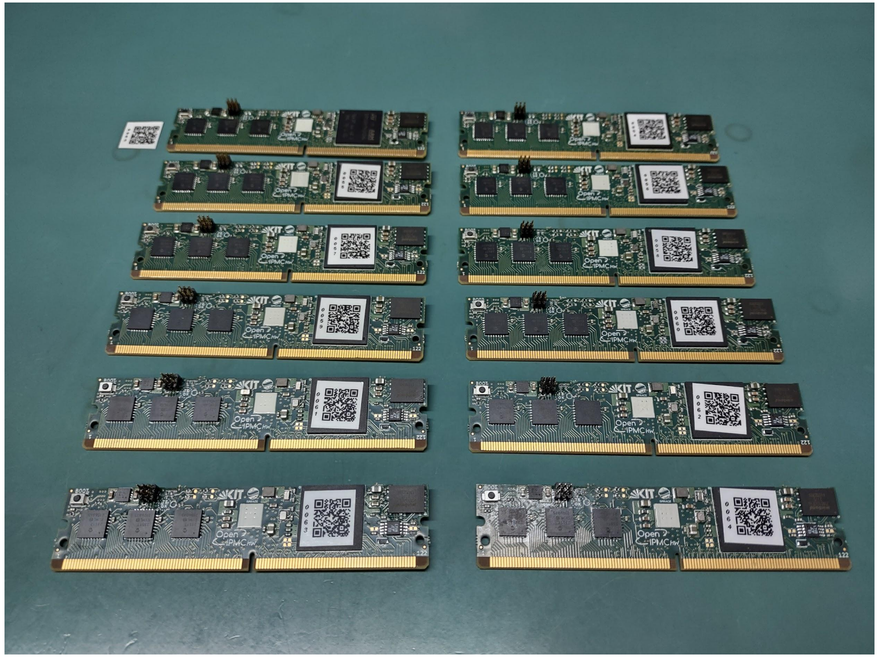
```
>> cat W/ipmc_log
[22-05-09 15:52:16][yaffs][Drivers for flash device W were installed successfully.]
[22-05-09 15:52:16][yaffs][File System mounted for read and write access.]

yaffs : 161 bytes read
>>
```

Production

- Prototype production by three institutes ⟶ test the “ease of production”
  - No significant issues, some lessons learned
  - Board v1.0 was followed by v1.1
- Karlsruhe Institute of Technology - KIT (Germany)
  - PCB outsourced, in-house assembly
  - 16 prototypes v1.0 nov/2020 + 50 boards v1.1 june/2021
- Boston University (USA)
  - PCB and assembly outsourced
  - 8 boards v1.1 jan/2021
- São Paulo
  - 20 boards planned to be produced in 2021
    - **Chip shortage** forced us to reduce to 12 boards v1.1 (dec/21)
    - PCB and assembly outsourced to Brazilian companies
  - Plan to produce in Brazil more boards 2022/2023
    - Chip shortage may continue to impact production 😒
    - TPS2115ADRBR is estimated to be available in Feb/2023

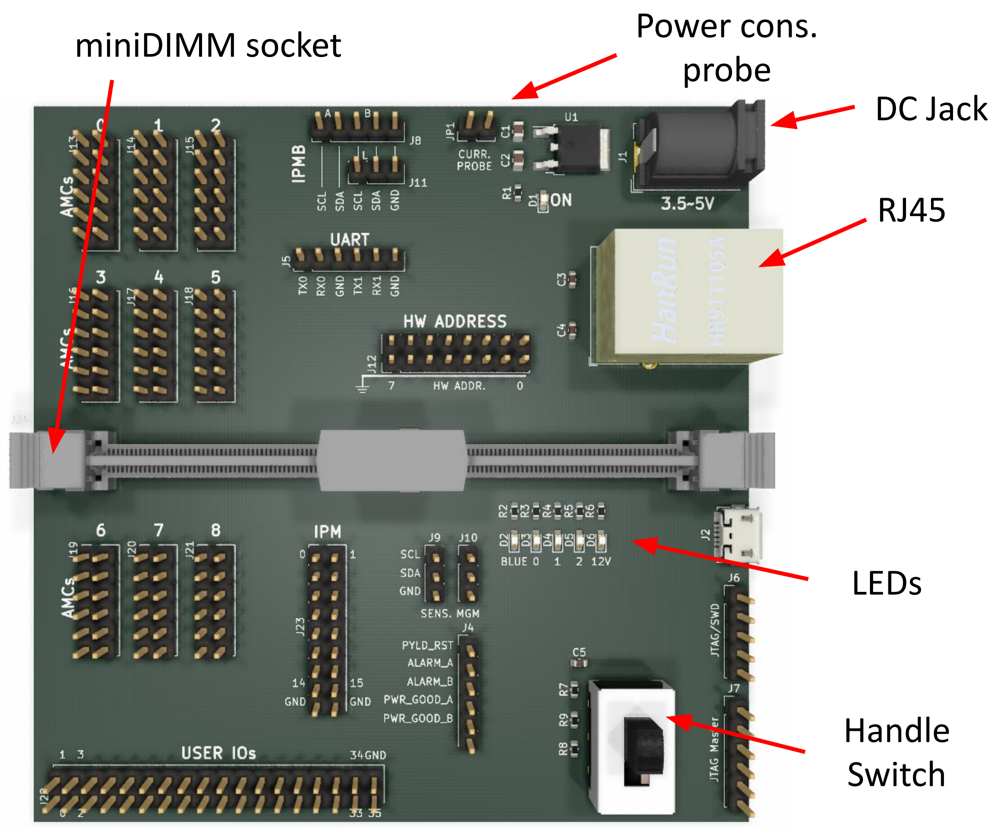
Production


Boards produced in Brazil (dec/2021)

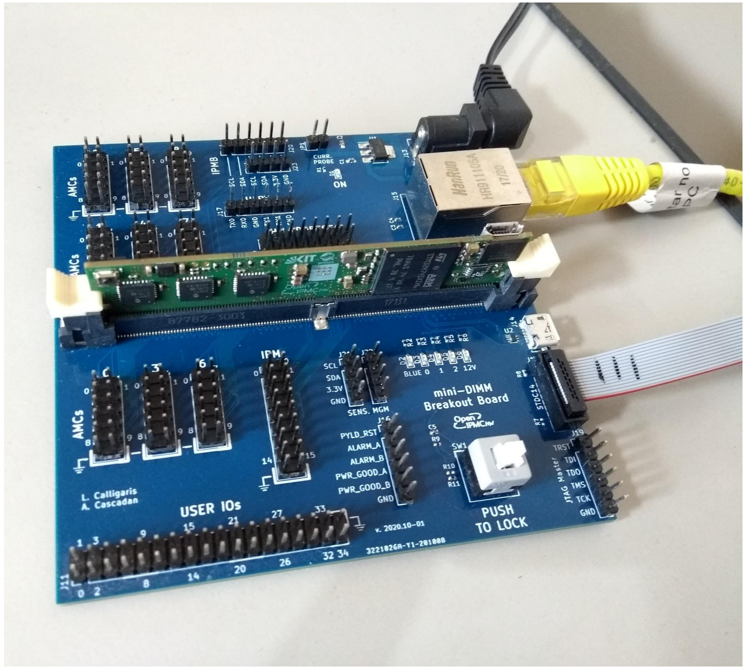
Boards produced in Germany (june/2021)

Dev tools: Breakout Board

- Useful for tests and basic developments
- **Very cheap** and simple to produce



3D model for the breakout board

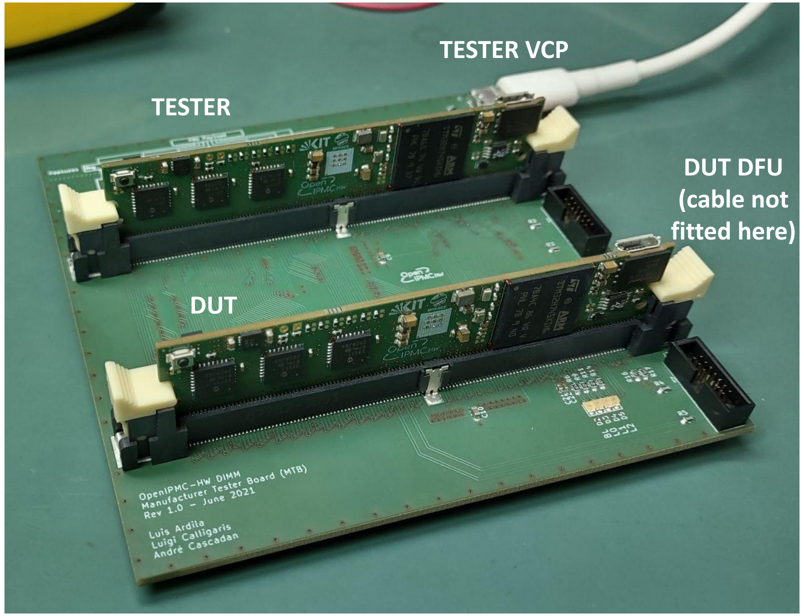


Manufactured PCBs for the breakout board

Test Tools: Manufacturing Test Board

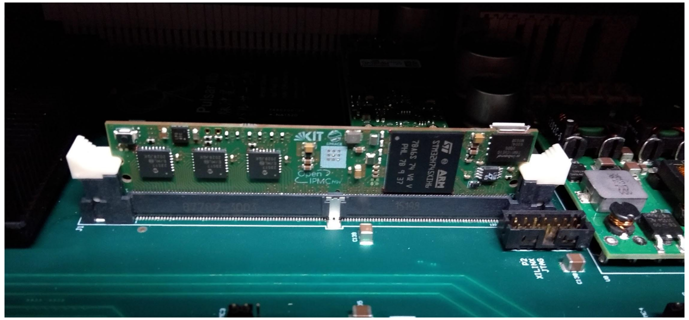
- Uses a healthy board to quickly test a lot of production
  - Checks all connections and board functionalities
  - One UART of both are used for communication

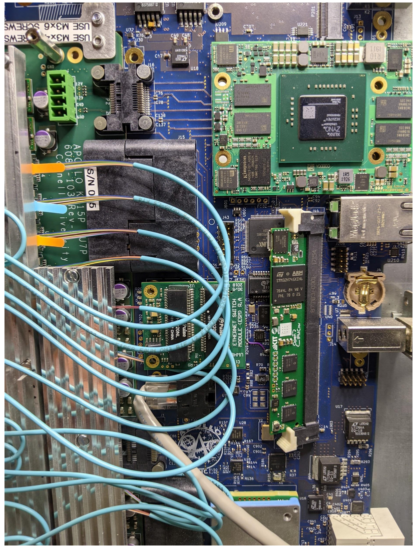
- Two complementary firmwares:
  - TESTER firmware
    - Controlled via PC console (USB VCP)
    - Sends test commands to DUT
    - Monitors test output
    - Prints results on PC console (USB VCP)
  - DUT (Device Under Test) firmware
    - Programmed into the MCU via USB DFU
    - Responds to commands from TESTER


TESTER VCP
TESTER
DUT DFU (cable not fitted here)
DUT
OpenIPMC-HW DIMM Manufacturer Tester Board (MTB) Rev 1.0 - June 2021
Luis Ardila
Luigi Calligaris
André Cascadan

Test setups

- OpenIPMC-HW currently being tested on 3 ATCA boards
  - Pulsar-IIb, at SPRACE (São Paulo).
  - Serenity, at KIT (Germany) and CERN TIF.
  - Apollo, at Boston University.
- Stress tests running for months
  - 6 Serenity and 7 PulsarIIb boards since May/21
  - Critical issues were being detected in meanwhile
    - Eg: IPMB collision avoidance mechanism
- Soon large-scale testing
  - Fill 2-4 crates with 14 OpenIPMCs each.


Pulsar-IIb @ SPRACE


Apollo @ BU

Serenity @ KIT

Summary and outlook

- The OpenIPMC project is now divided in three subprojects
  - OpenIPMC (aka OpenIPMC-SW) ⟶ Software providing IPMI behavior (Mozilla Public License 2.0)
  - OpenIPMC-HW ⟶ Mezzanine (CC BY-NC-SA 4)
  - OpenIPMC-FW ⟶ Firmware for the mezzanine (GNU GPL 3)
    - Common part
    - ATCA carrier-specific part
- Total 85 boards produced and being used in tests
  - Firmware customized for **Apollo**, **Serenity** and **Pulsar-IIb** so far
- New features being added
  - Integration of TFTP with YAFFS (network transfer of files in the flash) ⟶ ongoing
  - Xilinx Virtual Cable implementation ⟶ ongoing
  - DHCP discovery with Client ID (“option 61”) ⟶ planned
  - Hardware watchdog against FW freezes ⟶ planned

Thanks from the SPRACE team!

A pesquisa aqui apresentada recebe o suporte da FAPESP (processos 2018/18955-0 e 2019/18166-9)

BACKUP

Lead times

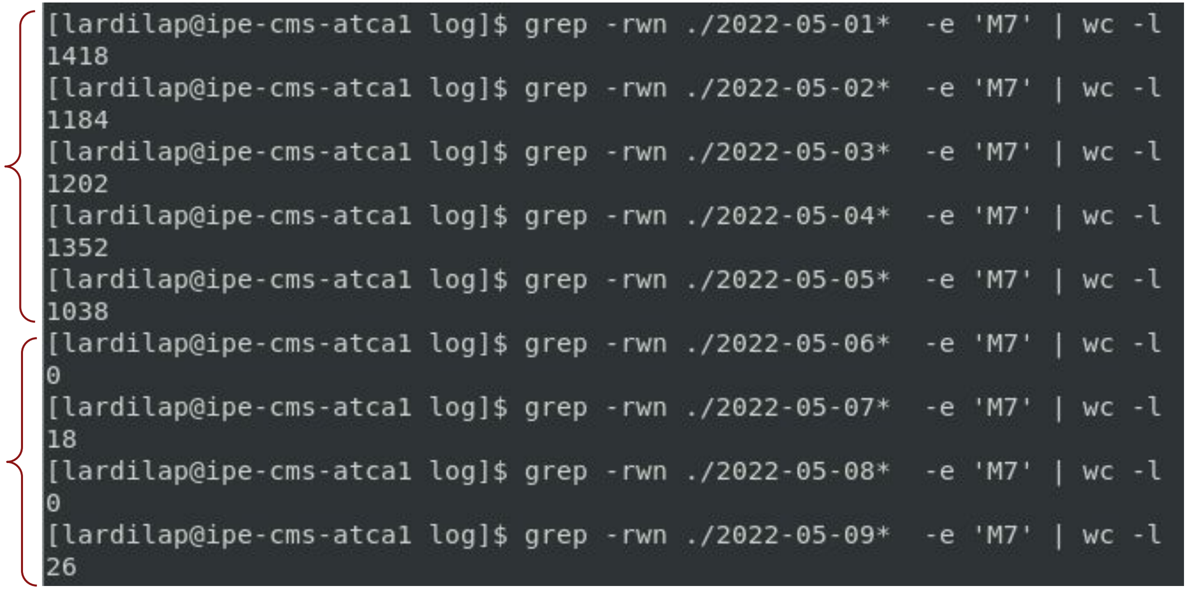
- Lead times for the missing silicon devices used by OpenIPMC-HW
- Updated on 9th-May-22
- MCU is the most critical: Lead Time from Newark is **June/23**

	Mouser	Digikey	Description
ADM7172ACPZ-3.3-R7	Jan/23	**June/22**	Voltage Regulator
USBLC6-2P6	**Sep/22**	Mar/23	ESD protection for USB
KSZ8091MNX	**Oct/22**	Feb/23	Ethernet PHY
MCP23S17-E/ML	Mar/23	**Jan/23**	GPIO expander
TPS2115ADRBR	May/23	**Feb/23**	Power OR'ing logic
STM32H745XIH6	N/A	N/A	Microcontroller

*Best lead times in bold

Collision avoidance

- Important IPMI reliability fix recently implemented
- Commited on 3rd May
- Tests in KIT started in 6th May on 8 Pulsar2b boards



Before

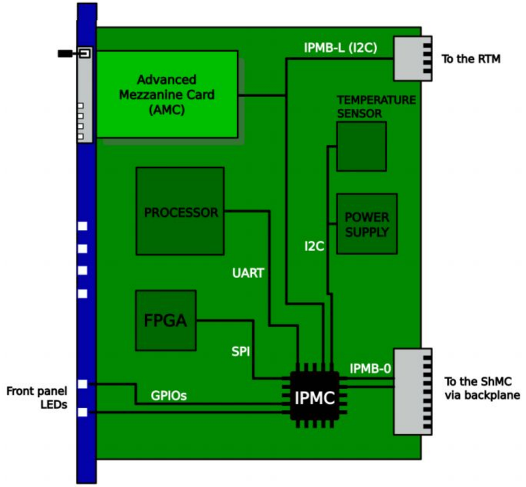
After

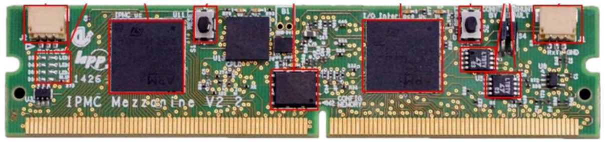
```
[lardilap@ipe-cms-atca1 log]$ grep -rwn ./2022-05-01*  -e 'M7' | wc -l
1418
[lardilap@ipe-cms-atca1 log]$ grep -rwn ./2022-05-02*  -e 'M7' | wc -l
1184
[lardilap@ipe-cms-atca1 log]$ grep -rwn ./2022-05-03*  -e 'M7' | wc -l
1202
[lardilap@ipe-cms-atca1 log]$ grep -rwn ./2022-05-04*  -e 'M7' | wc -l
1352
[lardilap@ipe-cms-atca1 log]$ grep -rwn ./2022-05-05*  -e 'M7' | wc -l
1038
[lardilap@ipe-cms-atca1 log]$ grep -rwn ./2022-05-06*  -e 'M7' | wc -l
0
[lardilap@ipe-cms-atca1 log]$ grep -rwn ./2022-05-07*  -e 'M7' | wc -l
18
[lardilap@ipe-cms-atca1 log]$ grep -rwn ./2022-05-08*  -e 'M7' | wc -l
0
[lardilap@ipe-cms-atca1 log]$ grep -rwn ./2022-05-09*  -e 'M7' | wc -l
26
```

Number of M4->M7 state events per day: before and after collision avoidance mechanism

Other IPMC alternatives

- Thought experiment: consider ATCA boards designed by an user
  - e.g. the Serenity or the Apollo boards for the CMS tracker
- The IPMC is **specific** to that board design
  - Different boards have different components
    - CPUs, FPGAs, hard disks, radio transceivers, …
    - The IPMC needs to know how to turn them on/off
    - The IPMC needs to know how to read temp/voltage/...
- The board designer chooses an IPMC for the board
  - This can be an IPMC designed by him...
  - ...or an IPMC designed by **someone else**
  - What counts is the **configuration** of the IPMC for that board
- LHC expts adopted an IPMC DIMM standard by LAPP (Annecy, FR)
  - LAPP IPMC module was quite complicated ⟶ abandoned
  - FNAL IPMC designed uniquely for Pulsar2b (**fw by SPRACE**)
  - CERN IPMC is the adaptation of a commercial product by PPS
    - **BIG** problems with license, NDAs, support...



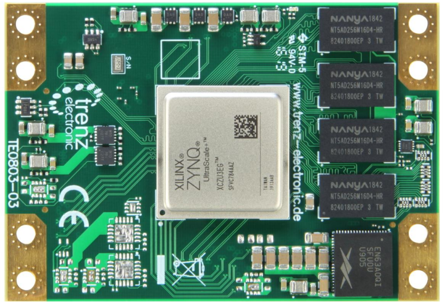
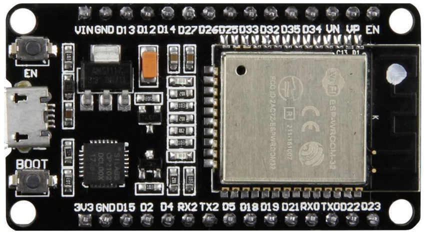
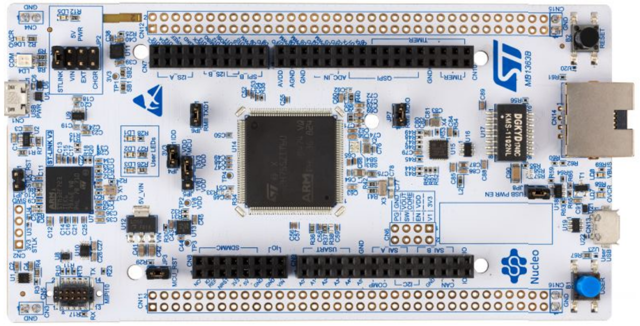

LAPP IPMC

FNAL IPMC

CERN IPMC

Evolution of OpenIPMC support on different devices

- First platform: Cortex-R5 cores on Zynq US+
  - IPMC (R5) and Linux (A53) running in the same device
  - Targeting the ATCA-ZynqMP management module by KIT (proposed for Serenity-A2577)
- Portability exercise: ESP32 microcontroller
  - Not a “serious” device, but very different arch from Zynq, cheap and very flexible
- First mainstream MCU: STM32 microcontroller
  - Successful porting opened the way to design of the DIMM module

Choice of the microcontroller

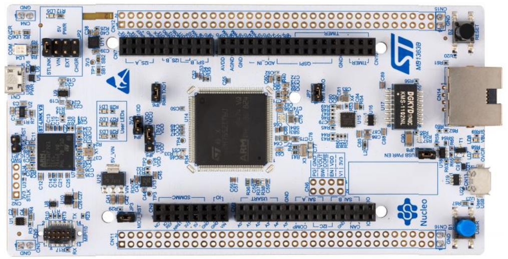
- The OpenIPMC software runs on top of FreeRTOS
  - Software shown to be easily portable on new MCUs (~3 wks)
  - Plenty of MCU manufacturers to choose from

- We chose **STM32H745XIH6** by STMicroelectronics

Number of I2C/SPI hardware peripherals	⟶ 4 / 6
Number of GPIOs/UART/USART	⟶ up to 168 / 4 / 4
Availability of an free toolchain	⟶ STM32CubeIDE
Availability of an evaluation board	⟶ NUCLEO-H745ZI-Q (cost: 23 CHF)
Our experience with other STM32 MCUs	⟶ STM32F103C8T6 (e.g. “Blue pill” board)
Performance margin for future upgrades	⟶ 480 MHz Cortex-M7+240 MHz Cortex-M4
Large SRAM/Flash memories	⟶ 1024 kiB / 2048 kiB
Expected reliability of the manufacturer	⟶ STMicroelectronics is a leader in MCUs
Cost	⟶ 17.45 $ per piece

- What we get in addition

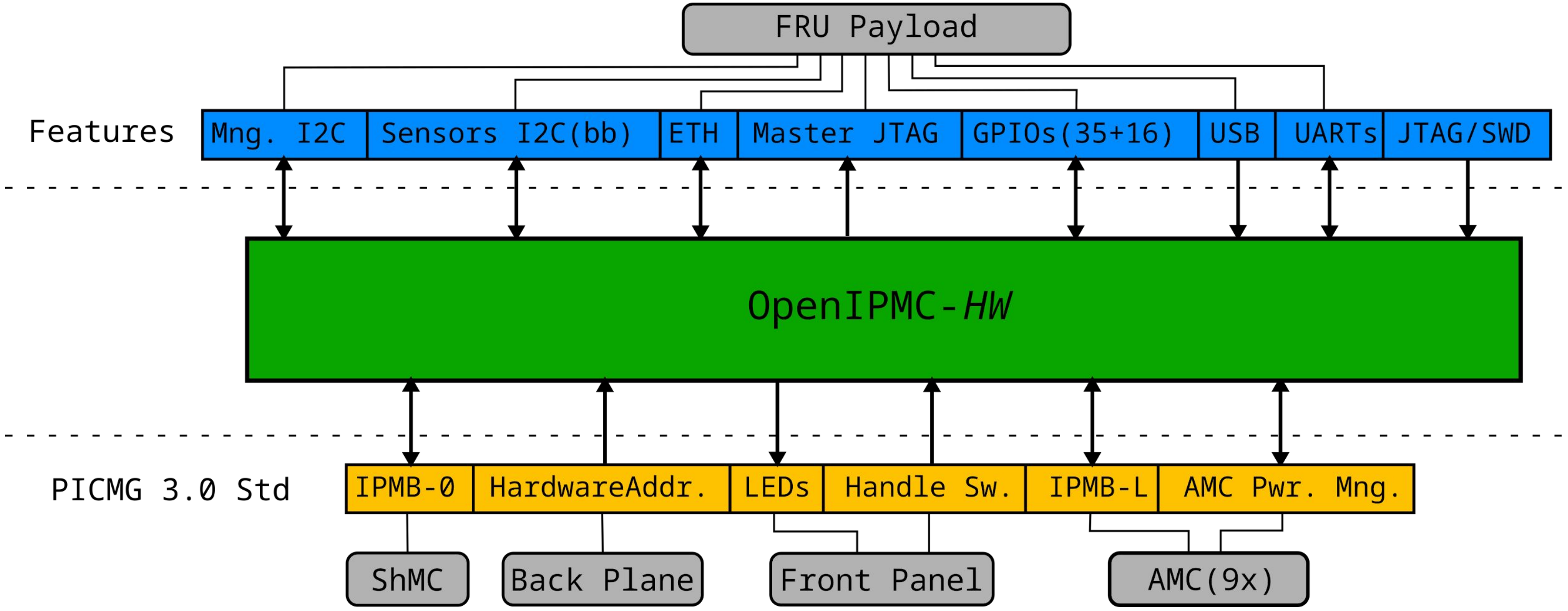
High speed USB device/host/OTG	⟶ USB programming & terminal
Efficient SMPS to power the core	⟶ better thermals
External memory support	⟶ store config/firmwares/etc
Lots of other features we will not use (e.g. HDMI driver)	


STM32 NUCLEO-H745ZI-Q

Full documentation on ST site

https://www.st.com/en/microcontrollers-microprocessors/stm32h745-755.html#documentation

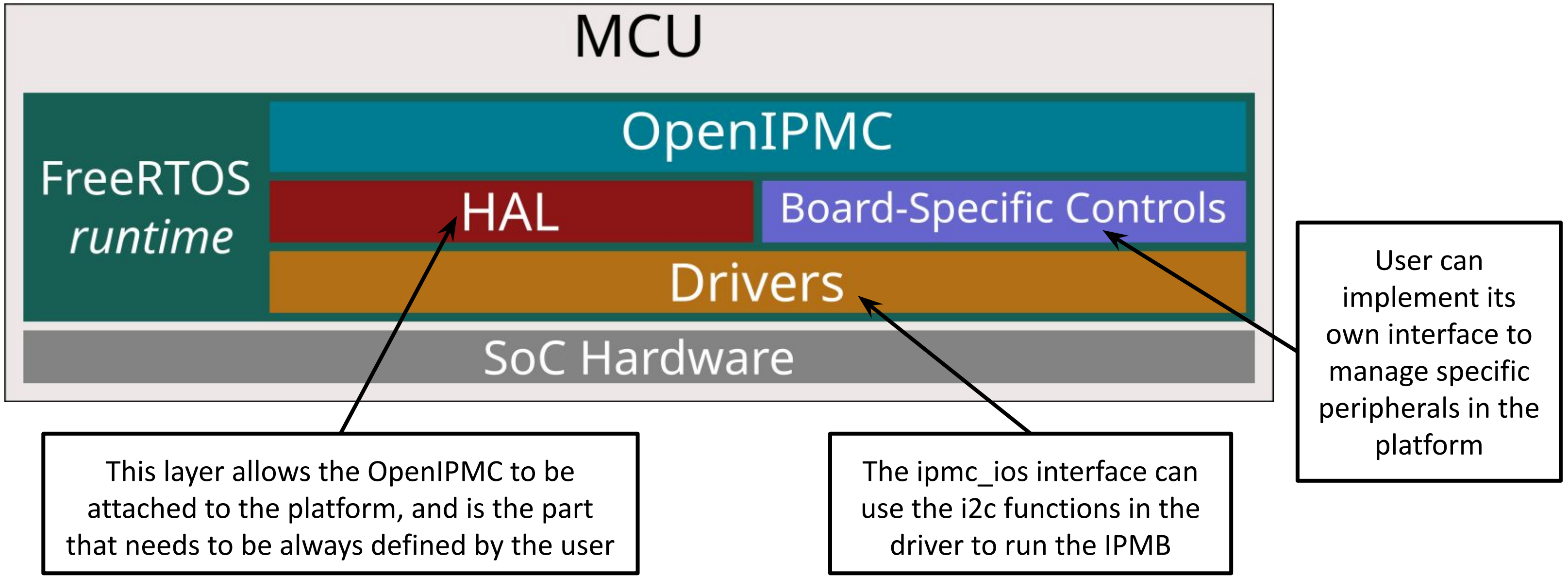
OpenIPMC-HW DIMM connections



Note: among the GPIOs some pins can be configured as UARTs, following the SoC Interest group layout

OpenIPMC runtime structure

- Two interfaces between OpenIPMC hardware-agnostic code and hardware drivers
  - **Hardware Abstraction Layer** → interface to hardware driver used for IPMI functions (IPMB, blue led..)
  - **Board-specific controls** → customize board-specific behavior (how to turn on power, read sensors..)
- Other tasks (not shown in picture) can run aside of the OpenIPMC stack



This layer allows the OpenIPMC to be attached to the platform, and is the part that needs to be always defined by the user

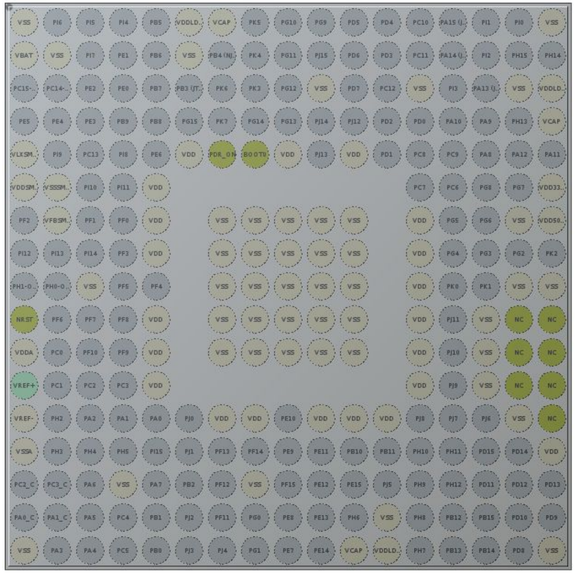
The ipmc_ios interface can use the i2c functions in the driver to run the IPMB

User can implement its own interface to manage specific peripherals in the platform

STM32H745XI Microcontroller: specs

- Cores
  - 1x ARM Cortex-M7 (480 MHz max)
  - 1x ARM Cortex-M4 (240 MHz max)
- Package
  - 265-TFBGA
  - 14x14mm
  - 0.8mm pitch
- Memory
  - 2x 1 Mbyte Flash
  - 64 I + 128 D Kbytes TCM (M7 only)
  - 864 Kbytes SRAM
- Power
  - Input 1.62 to 3.6 V
  - Integrated SMPS+LDO

- IOs
  - 168x GPIOs
  - 4x I^{2}C
  - 6x SPI
  - 8x UART
  - Ethernet MAC
  - USB host/device/OTG
  - Quad-SPI
- Others
  - LCD-TFT
  - JPEG Codec
  - ADCs
  - DACs
  - OpAmps
  - Graphical Accelerator

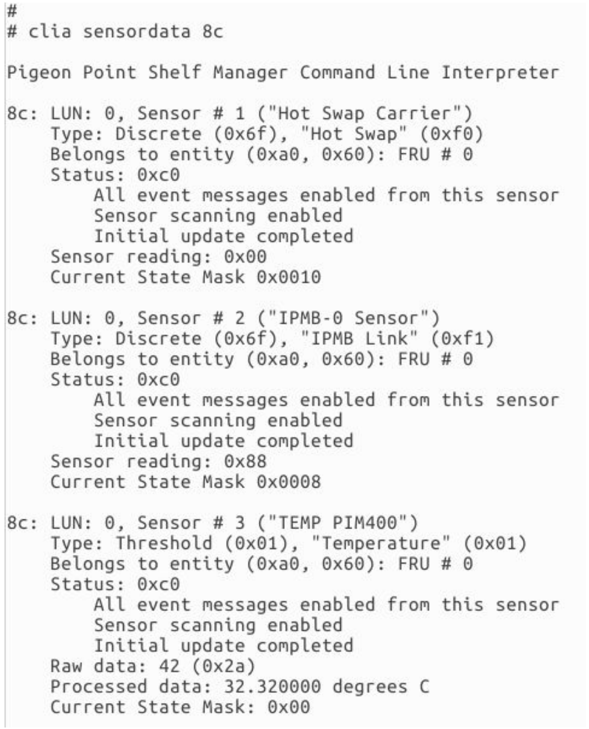

TFBGA 240+25

Sensor readings

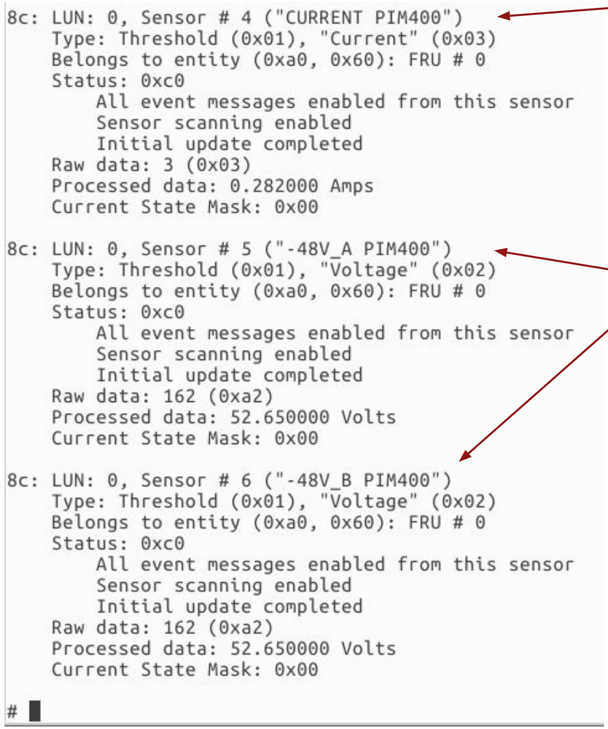
HotSwap Sensor

IPMB Sensor

Temperature form PIM400



```
#
# clia sensordata 8c

Pigeon Point Shelf Manager Command Line Interpreter

8c: LUN: 0, Sensor # 1 ("Hot Swap Carrier")
    Type: Discrete (0x6f), "Hot Swap" (0xf0)
    Belongs to entity (0xa0, 0x60): FRU # 0
    Status: 0xc0
        All event messages enabled from this sensor
        Sensor scanning enabled
        Initial update completed
    Sensor reading: 0x00
    Current State Mask 0x0010

8c: LUN: 0, Sensor # 2 ("IPMB-0 Sensor")
    Type: Discrete (0x6f), "IPMB Link" (0xf1)
    Belongs to entity (0xa0, 0x60): FRU # 0
    Status: 0xc0
        All event messages enabled from this sensor
        Sensor scanning enabled
        Initial update completed
    Sensor reading: 0x88
    Current State Mask 0x0008

8c: LUN: 0, Sensor # 3 ("TEMP PIM400")
    Type: Threshold (0x01), "Temperature" (0x01)
    Belongs to entity (0xa0, 0x60): FRU # 0
    Status: 0xc0
        All event messages enabled from this sensor
        Sensor scanning enabled
        Initial update completed
    Raw data: 42 (0x2a)
    Processed data: 32.320000 degrees C
    Current State Mask: 0x00
```



```
8c: LUN: 0, Sensor # 4 ("CURRENT PIM400")
    Type: Threshold (0x01), "Current" (0x03)
    Belongs to entity (0xa0, 0x60): FRU # 0
    Status: 0xc0
        All event messages enabled from this sensor
        Sensor scanning enabled
        Initial update completed
    Raw data: 3 (0x03)
    Processed data: 0.282000 Amps
    Current State Mask: 0x00

8c: LUN: 0, Sensor # 5 ("-48V_A PIM400")
    Type: Threshold (0x01), "Voltage" (0x02)
    Belongs to entity (0xa0, 0x60): FRU # 0
    Status: 0xc0
        All event messages enabled from this sensor
        Sensor scanning enabled
        Initial update completed
    Raw data: 162 (0xa2)
    Processed data: 52.650000 Volts
    Current State Mask: 0x00

8c: LUN: 0, Sensor # 6 ("-48V_B PIM400")
    Type: Threshold (0x01), "Voltage" (0x02)
    Belongs to entity (0xa0, 0x60): FRU # 0
    Status: 0xc0
        All event messages enabled from this sensor
        Sensor scanning enabled
        Initial update completed
    Raw data: 162 (0xa2)
    Processed data: 52.650000 Volts
    Current State Mask: 0x00

#
```

Current on PIM400

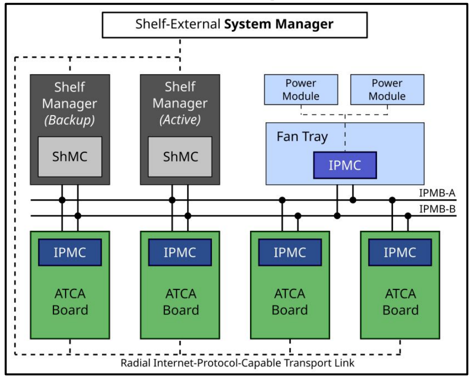
Voltage on -48 line (Channels A and B)

Sensor reading test: Shelf Manager CLI is printing the sensor readings of Serenity @ KIT.

The HPM.1 Specification

- Hardware Platform Management IPM Controller Firmware Upgrade Specification
  - Released by PICMG
  - Specifies a set of IPMI commands to upload a new FW or rollback to backup copy
  - Upgrade is remotely operated by an *Upgrade Agent* tool (eg: `ipmitool`, `ipmiutil`)
  - The *Upgrade Agent* reads the new image(s) from a standard `*.hpm` file
    - HPM.1 has multi-component support (firmware for each IPMC component)

- Upgrade is intermediated by Shelf Manager
  - Upgrade Agent sends IPMI commands to ShMM via Network (System Manager)
  - ShMM forwards the commands to the target IPMC via IPMB



The Upgrade Agent

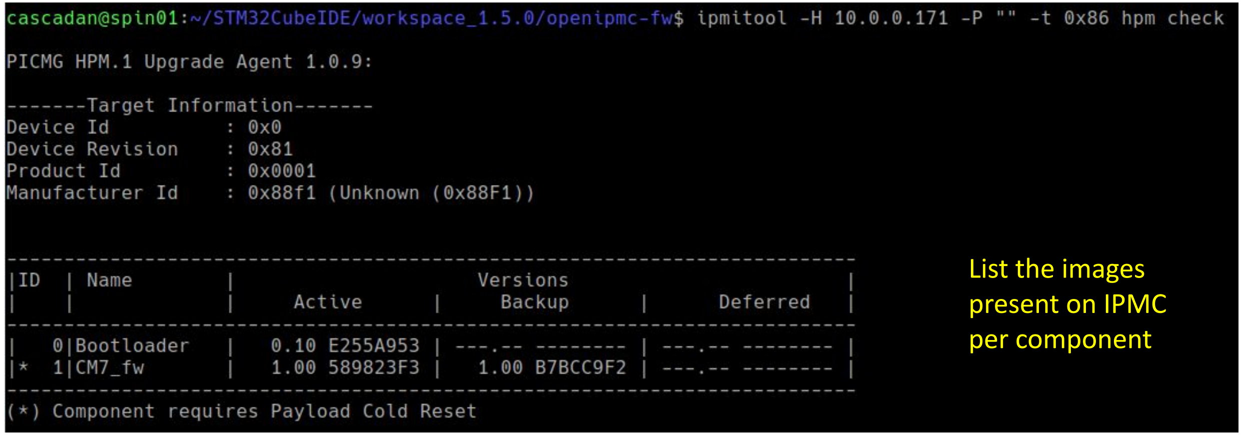
- Speaks with an IPMC remotely
  - Runs in PC, uses IPMI commands over Ethernet to communicate.

- Is guided by a standard Upgrade Image file (*.hpm)
  - One *header* describes the manufacturer, product, upgrade capabilities, etc.
  - Several *actions* guide the upgrade sequence of different components
    - Types of actions: Backup, Prepare, Upload (contains image)
    - Actions are executed in sequence as they appear in the file

- Checks consistency between running and new firmware before upgrade
  - Manufacturer ID and Product ID must match the expected ones
  - The new version can specify the oldest version from which an upgrade is allowed

- OpenIPMC-FW project provides a program to *pack* the compiled `.bin` into an `.hpm`
  - Version and Product information is extracted from the `.bin`
  - Instructions here

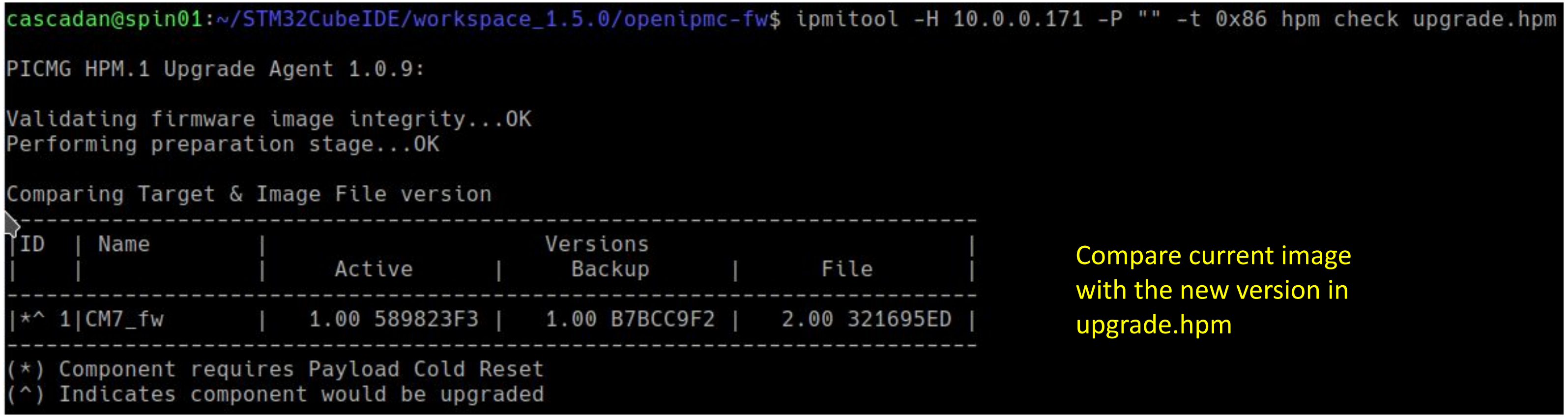
The Upgrade Agent - ipmitool example



```
cascadan@spin01:~/STM32CubeIDE/workspace_1.5.0/openipmc-fw$ ipmitool -H 10.0.0.171 -P "" -t 0x86 hpm check

PICMG HPM.1 Upgrade Agent 1.0.9:

-------Target Information-------
Device Id          : 0x0
Device Revision    : 0x81
Product Id         : 0x0001
Manufacturer Id    : 0x88f1 (Unknown (0x88F1))


--------------------------------------------------------------------------
|ID  | Name        |                     Versions                          |
|    |             |     Active       |     Backup       |      Deferred    |
--------------------------------------------------------------------------
|   0|Bootloader   |   0.10 E255A953 | ---.-- -------- | ---.-- -------- |
|*  1|CM7_fw       |   1.00 589823F3 |   1.00 B7BCC9F2 | ---.-- -------- |
--------------------------------------------------------------------------
(*) Component requires Payload Cold Reset
```

List the images present on IPMC per component

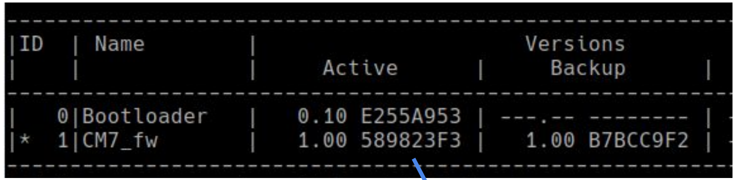
Note: Bootloader is listed as a separate component to allow an independent upgrade via HPM.1



```
cascadan@spin01:~/STM32CubeIDE/workspace_1.5.0/openipmc-fw$ ipmitool -H 10.0.0.171 -P "" -t 0x86 hpm check upgrade.hpm

PICMG HPM.1 Upgrade Agent 1.0.9:

Validating firmware image integrity...OK
Performing preparation stage...OK

Comparing Target & Image File version
--------------------------------------------------------------------------
|ID  | Name        |                     Versions                          |
|    |             |     Active       |     Backup       |      File        |
--------------------------------------------------------------------------
|*^ 1|CM7_fw       |   1.00 589823F3 |   1.00 B7BCC9F2 |   2.00 321695ED |
--------------------------------------------------------------------------
(*) Component requires Payload Cold Reset
(^) Indicates component would be upgraded
```

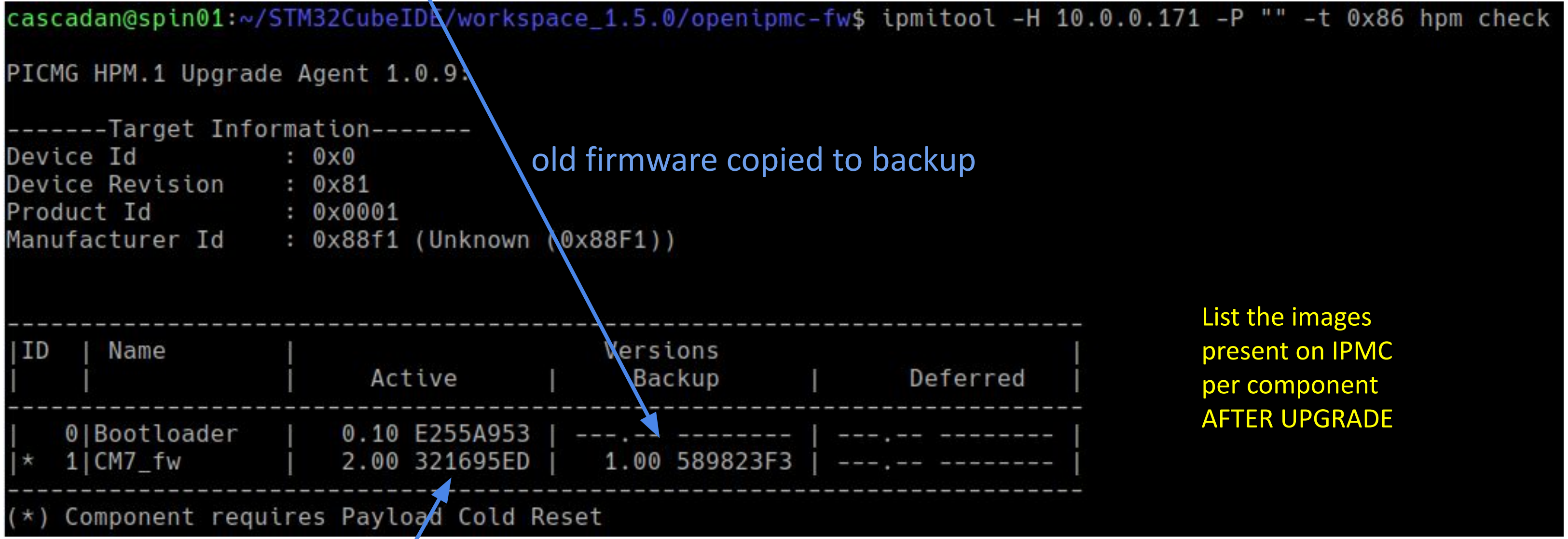
Compare current image with the new version in upgrade.hpm

The Upgrade Agent - ipmitool example



```
---------------------------------------------------------------
|ID  | Name           |              Versions
|    |                |     Active        |    Backup         |
---------------------------------------------------------------
|   0|Bootloader      |   0.10 E255A953 | ---.-- -------- |
|*  1|CM7_fw          |   1.00 589823F3 |   1.00 B7BCC9F2 |
---------------------------------------------------------------
```

(before upgrade)



```
cascadan@spin01:~/STM32CubeIDE/workspace_1.5.0/openipmc-fw$ ipmitool -H 10.0.0.171 -P "" -t 0x86 hpm check

PICMG HPM.1 Upgrade Agent 1.0.9:

-------Target Information-------
Device Id          : 0x0
Device Revision    : 0x81
Product Id         : 0x0001
Manufacturer Id    : 0x88f1 (Unknown (0x88F1))


-------------------------------------------------------------------------
|ID  | Name        |                  Versions                          |
|    |             |     Active      |     Backup      |     Deferred    |
-------------------------------------------------------------------------
|   0|Bootloader   |   0.10 E255A953 | ---.-- -------- | ---.-- -------- |
|*  1|CM7_fw       |   2.00 321695ED |   1.00 589823F3 | ---.-- -------- |
-------------------------------------------------------------------------
(*) Component requires Payload Cold Reset
```

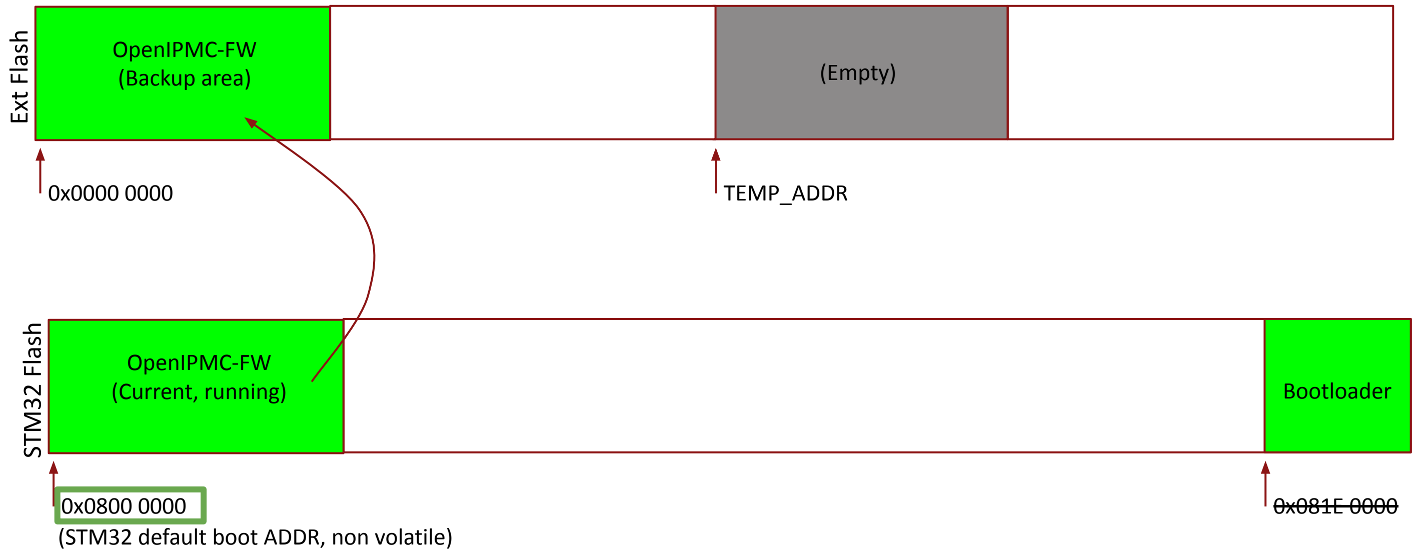
old firmware copied to backup

List the images present on IPMC per component AFTER UPGRADE

new firmware

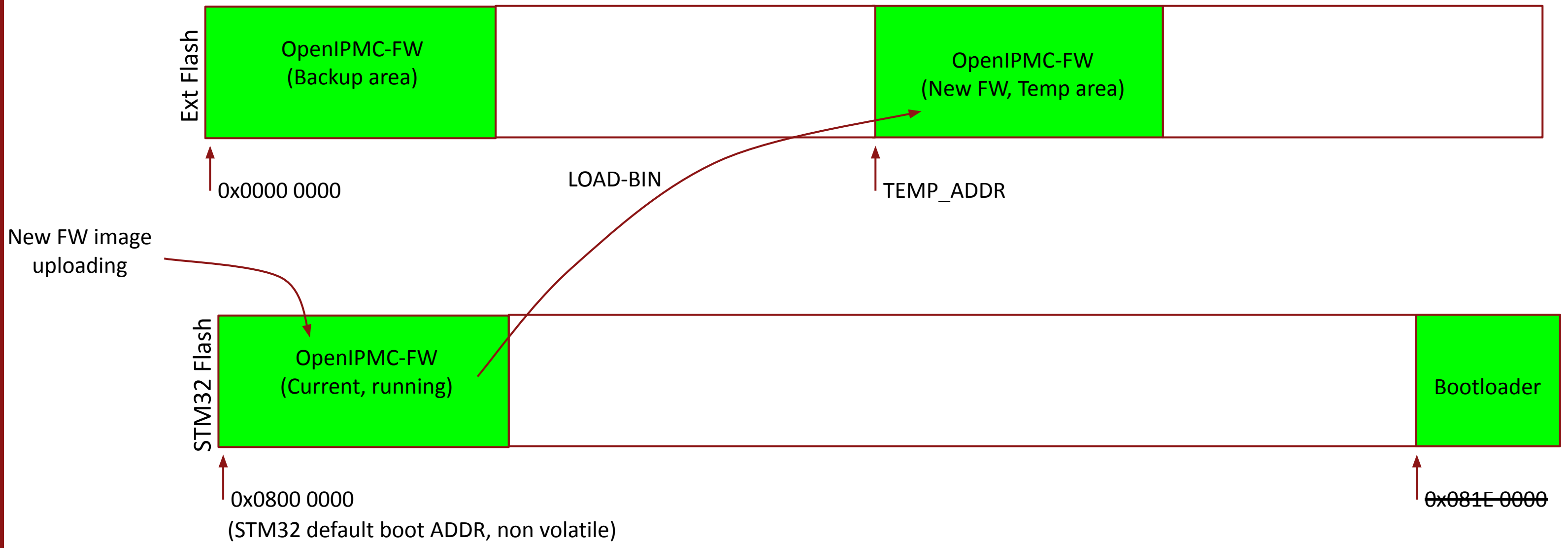
FW Upgrade steps

1. OpenIPMC-FW makes a copy of itself into the external Flash present on DIMM. This operation is triggered by a command from Upgrade Agent, following the *actions* present in the .hpm file



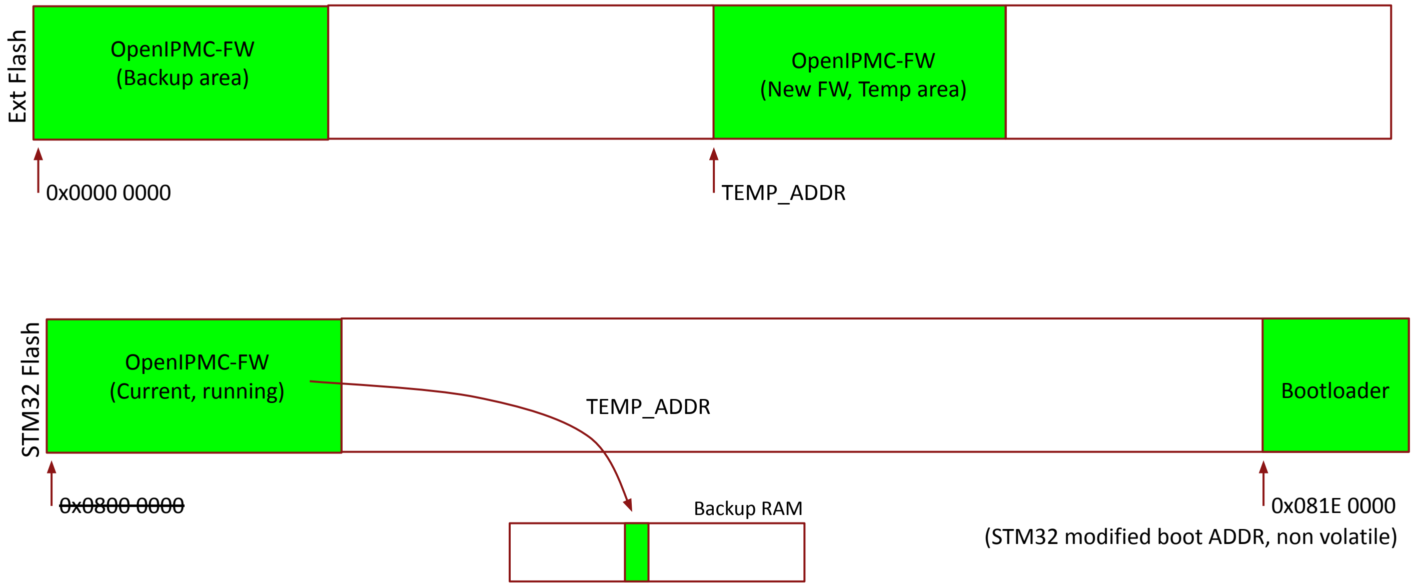
FW Upgrade steps

2. The Upgrade Agent uploads the new image, which is saved in a temporary area of the external Flash


Ext Flash
OpenIPMC-FW (Backup area)
OpenIPMC-FW (New FW, Temp area)
0x0000 0000
LOAD-BIN
TEMP_ADDR
New FW image uploading
STM32 Flash
OpenIPMC-FW (Current, running)
Bootloader
0x0800 0000 (STM32 default boot ADDR, non volatile)
~~0x081E 0000~~

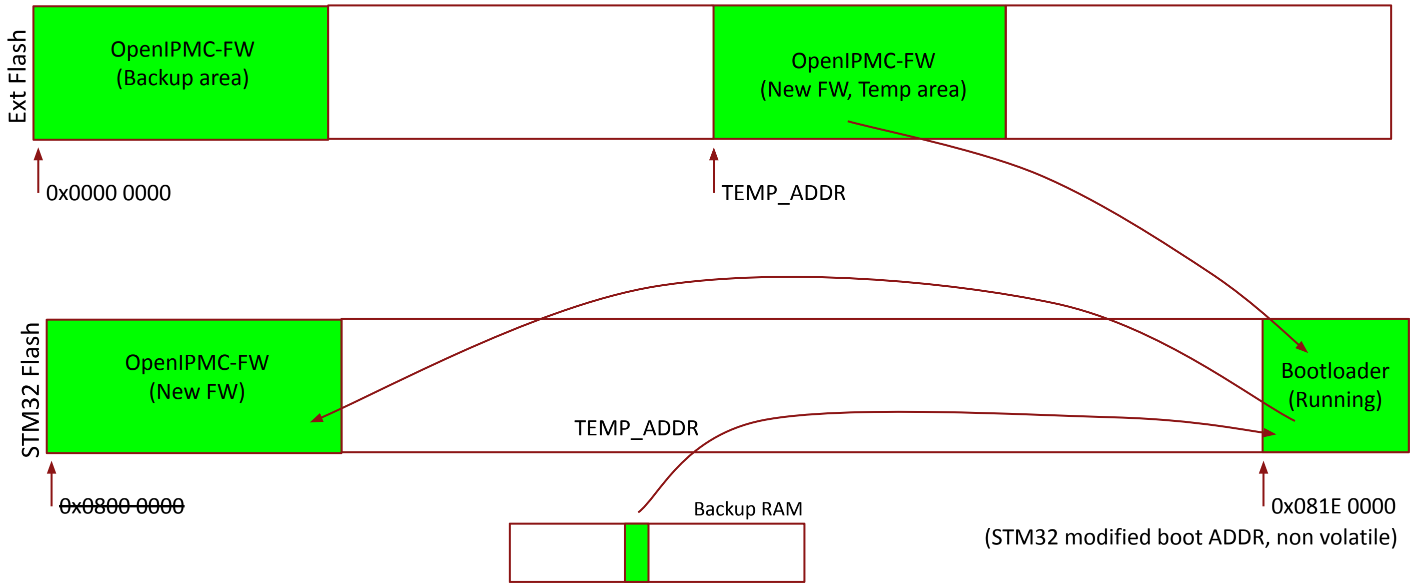
FW Upgrade steps

3. By receiving the *activate* command from the Agent, OpenIPMC-FW writes control data to the Bootloader informing the position of the new firmware
4. Set boot address in NV memory to Bootloader region and reboot
   a. (note: this is important just the first upgrade, as the bootloader is meant to always run on boot)


Ext Flash
OpenIPMC-FW (Backup area)
OpenIPMC-FW (New FW, Temp area)
0x0000 0000
TEMP_ADDR
STM32 Flash
OpenIPMC-FW (Current, running)
TEMP_ADDR
Bootloader
~~0x0800 0000~~
Backup RAM
0x081E 0000
(STM32 modified boot ADDR, non volatile)

FW Upgrade steps

5. Bootloader reads the control data
6. Bootloader copies the new FW from TEMP to the RUN area
7. Bootloader jumps to new FW



Firmware upgrade safety

- Upload with checksum
  - HPM.1 specifies a completion code to signalise a checksum failure during the upload step
  - The image is uploaded with a CRC32 appended to its end, so IPMC can check the integrity
  - The CRC32 is also saved to the TEMP area, so the bootloader can check the integrity

- Backup with checksum
  - The backup copy receives a CRC32 at the end, so the bootloader can check the integrity

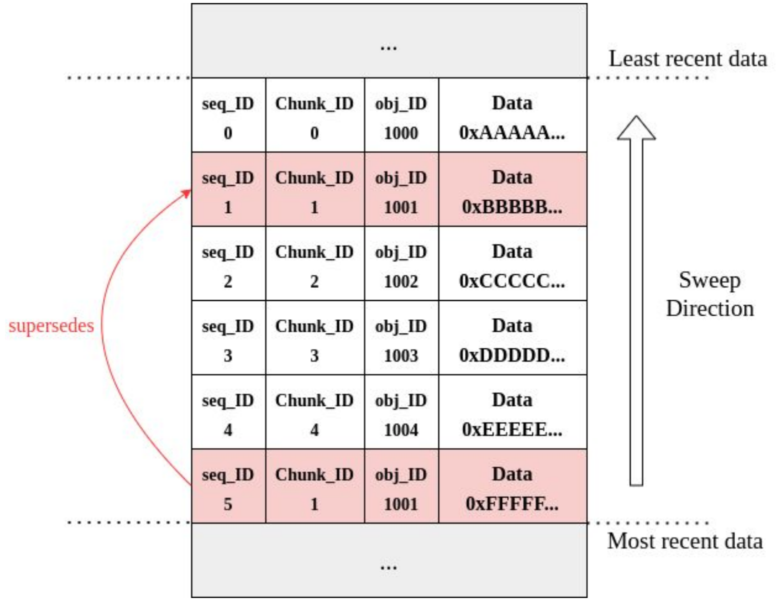
- Bootloader
  - Bootloader is a very simple firmware
    - Single task, no RTOS
    - Peripherals: QSPI (for Ext. Flash), CRC32 calculator, UART-TX (for printouts)
  - Safety mechanisms still to be implemented
    - Watchdog can be employed for an auto-rollback in case of failure in new firmware
    - In case of power loss during the copy, Bootloader is able to recognise the corrupted copy

File system in OpenIPMC-HW

- Flash memory on **OpenIPMC-HW DIMM** card
  - 3V, 1G-bit NAND flash Winbond **W25N01GVZEIG** (or compatible **W74M01GVZEIG**).
  - Connected via **QuadSPI** bus to the MCU.
- Benefits of a File System
  - Can be mounted as an USB mass storage device or accessed via TFTP.
  - Logs storage.
    - Abnormal sensor readings, error logs, etc.
  - Can be used to store IPMC configuration parameters
    - Sensor thresholds, IP address, etc.
- Choice of a file system
  - Needs to be robust against bad blocks (to be expected in flash storage).
  - Properly spread the data writes, to extend flash lifetime as much as possible.

YAFFS-2

- YAFFS
  - “Yet Another Flash File System”
  - Designed specifically for NAND and NOR chips using wear-leveling techniques.
- “True Log-structured”, i.e.
  - Sequential page writing within a block of memory.
  - No deletion markers used in flash memory.
    - File state is described by the last sequential update of the file content.
    - Uses page-level partial updates of file content.
- Embedded robustness
  - Supports SLC as well as MLC flash devices.
  - Low RAM footprint.



...
Least recent data
seq_ID 0 | Chunk_ID 0 | obj_ID 1000 | Data 0xAAAAA...
seq_ID 1 | Chunk_ID 1 | obj_ID 1001 | Data 0xBBBBB...
seq_ID 2 | Chunk_ID 2 | obj_ID 1002 | Data 0xCCCCC...
seq_ID 3 | Chunk_ID 3 | obj_ID 1003 | Data 0xDDDDD...
seq_ID 4 | Chunk_ID 4 | obj_ID 1004 | Data 0xEEEEE...
seq_ID 5 | Chunk_ID 1 | obj_ID 1001 | Data 0xFFFFF...
supersedes
Sweep Direction
Most recent data
...