The OpenIPMC project

Development of a portable FOSS IPMC software and design of an HW platform for its operation

André Cascadan, Luigi Calligaris, Bruno Casu (UNESP NCC), Luis Ardila, Oliver Sander (KIT IPE), Lucas Arruda Ramalho (UNEMAT)

16th CERN xTCA Interest Group Meeting
Elements of the project and their aim

● IPMC software (OpenIPMC)
  ○ Open Source SW → No license problems (e.g. students), flexibility
  ○ Multiplatform → Freedom to choose MCU, future-proofing (arch.)
  ○ PICMG-compliant → Compatibility (so far restricted to the HEP use case)
  ○ See our presentation in 15th xTCA IG (https://indico.cern.ch/event/897461/)

● IPMC DIMM hardware module (OpenIPMC-HW)
  ○ Open Source HW → Future-proofing (any group can re-use the design)
  ○ LAPP pinning → Compatible with CERN IPMC layout v4
  ○ Rather simple PCB → Can be fabricated by many PBC manufacturers
  ○ Well-supported MCU

● Firmware for the DIMM module (OpenIPMC-FW)
  ○ Open Source SW → No license problems (e.g. students), flexibility
  ○ Eclipse-based SDK → Popular, easy to use and install, supports Linux
  ○ Support for different ATCA boards
Brief recap on the OpenIPMC software
OpenIPMC

● IPMC software implementing 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

● 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
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
How OpenIPMC interfaces to the hardware

- 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..)
- Note that other FreeRTOS tasks (not shown in pic) can run aside of the OpenIPMC stack

---

**MCU**

- **FreeRTOS runtime**
- **OpenIPMC**
- **HAL**
- **Board-Specific Controls**
- **Drivers**
- **SoC Hardware**

- Interface to PICMG-required hardware. (e.g. I2C IPMB-0, Blue LED, Handle, …) Needs **always** to be defined in code.
- Drivers supplied by the MCU manufacturer (or written by the user)

Interface to resources specific to the ATCA board being targeted (e.g. Zynq module boot, FPGA configuration, main power enable, …)
OpenIPMC-HW
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 a 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)

[Full documentation on ST site](https://www.st.com/en/microcontrollers-microprocessors/stm32h745-755.html#documentation)
STM32H745XI peripherals

- Feature-rich microcontroller
  - Plenty of peripherals to play with

- Total 1MiB of RAM, most of it still free
  - Leaves space for future software upgrades

- The Cortex-M4 core is still not used
  - It can be used to run bare metal code
    - Bit-banging as 5th I2C channel (sensor master)
    - Bit-banging as JTAG master on the AMC GPIOs

- This MCU seems decently future-proof
OpenIPMC-HW layout: schematic
AMC IO expanders

- CERN/LAPP IPMC specifies a set of 10 control signals for each AMC mezzanine
  - Up to 9 mezzanines are supported → **90 GPIOs needed!!!**
- We use six Microchip MCP23S17 16-bit I/O expanders, controlled via a single SPI bus at 10 MHz
  - Only one SPI bus + shared Chip Select (CS) to control all the expanders
    - Expander address in SPI protocol header, address is set via package pull-ups /downs
    - One GPIO from MCU used to catch the interrupt signals from the expanders (open collector mode)
- Dedicated driver was developed to control all Expanders in a transparent way
Ethernet PHY

- Micrel/Microchip KSZ8091MNX
  - Same as other IPMCs
    - Compatibility with tx/rx bias scheme used in existing in ATCA boards
    - Well tested PHY, known to be reliable
  - 10/100 Mbit with auto MDI/MDI-X
  - MII interface @ 25 MHz
    - Separate 25 MHz crystal for the PHY on the IPMC
  - MDC/MDIO for configuration
  - Other features we don’t use
    - TDR for fault detection & ranging, ...
USB interface

● Intended for use in two functions
  ○ Quick access to the IPMC Command Line Interface
  ○ Firmware update via a simple USB cable

● Command Line Interface via USB VCP driver
  ○ VCP is part of the USB standard
  ○ Supported natively by ST development tools

● Hardware update via USB DFU protocol
  ○ DFU is part of the USB standard
  ○ Supported natively by the ROM bootloader in the MCU
3.3V power OR-ing switch

- We want to make possible to program the module in-hand via USB
  - Two Possible Power Sources and risk of reverse powering

- Source hierarchy
  - DIMM Edge connector (3.3V) → primary source
  - On-board USB (5V) + 3.3V LDO → secondary source
  - Source conflict resolution → use a COTS ORing switch for USB applications

- Texas Instruments TPS2115A
  - Automatic power ORing with 2 inputs
  - 3x3mm SON-8 package

- Analog Devices ADM7172
  - 3.3V LDO, max 2A
  - 8-LFCSP package
External flash

- The MCU supports external QSPI flash

- We added one to the DIMM to
  - Store configuration
  - Aid in firmware upgrade binary gymnastics
  - Store a failsafe “golden image”

- Our chosen part is Winbond W74M01GVZEIG
  - WSON-8 package
  - 1 Gbit = 128 MiB with 4kiB erasable blocks
  - Cheap (7.33 USD)
  - Currently severely impacted by chip shortage
  - Pin-compatible alternative parts from Micron
    - MT25QL01GBBB1EW9 (but costs twice as much)
OpenIPMC-HW layout: picture

- Micro-USB B
- SPI IO expanders (Microchip MCP23S17)
- USB ESD TVS
- Ethernet PHY (Microchip KSZ8091MNX)
- PHY 25 MHz crystal
- PHY led
- SPI IO expanders (Microchip MCP23S17)
- 3.3V rail ORing power switch (TI TPS2115ADRBR)
- MCU decoupling and thermal vias
- IPMB I2C isolation buffer (Linear LTC4300-1)
- Vusb to 3.3 V LDO
- SWD and UART0 lead holes
- MCU reset
- MCU HSE 25 MHz crystal
- MCU LSE 32 kHz crystal
- MCU
- NOR Flash (W25Q01VZIEIM)
- 18.3 mm height allows to fit in ATCA crate using a vertical DIMM slot

JEDEC MO-244 connector
Breakout Board used for development & programming

- IPMB-0 channels
- Current measurement jumper
- Power cord and LDO
- AMC [1 to 6] IOs
- JEDEC MO-244 socket
- AMC [7 to 9] IOs
- IPM IOs
- USER IOs

The breakout board exposes (almost) all the pins of the DIMM connector on 100-mil headers for test/development purposes. We plan to make a small update adding the breakout for 12V_EN

STLink-V3MINI JTAG/SWD debugger
Cost: around 10 USD

The breakout board is very inexpensive:
- 10 PCB + shipping = 26 USD
- Dimm connector = 17 USD
- The other components are cheaper
OpenIPMC-HW on ATCA boards
Test setups

- OpenIPMC-HW is currently being tested in 3 ATCA boards
  - Pulsar-IIb, at SPRACE (São Paulo)
  - Serenity, at KIT (Germany) and CERN
  - Apollo, at Boston University

For each target board, the OpenIPMC-FW code is forked and adapted to the specific hardware. We took care to make this process easy for users.
Sensor readings

Sensor reading test: Shelf Manager CLI is printing the sensor readings of Serenity @ KIT.
An OpenIPMC-HW was left reading out sensors on the PIM400 of its hosting Serenity board at TIF. The values were fed into the TIF Carbon server and plotted with Grafana. Data readout was stable.
Polaris PICMG standards compliance tests

- Verification of compliance with PICMG standard requires many tests
  - Needs an automated system allowing to perform tests in batches

- We are using an ATCA compliance testing sw by Polaris Networks
  - Kindly provided by the CERN EP-ESE group at bldg 14. Thanks!

- OpenIPMC-HW was put to test with the CERN Polaris setup
  - The results (below) have been fundamental in orienting our improvements
  - We will keep to use this tool to orient our improvements

<table>
<thead>
<tr>
<th>PASSED</th>
<th>FAILED</th>
<th>SKIPPED</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>56 (56%)</td>
<td>17</td>
<td>26</td>
<td>99</td>
</tr>
</tbody>
</table>
Summary and outlook

○ Following the development of OpenIPMC, we designed an IPMC module
  ○ Mini-DIMM form factor, pin-, function- and mechanically-compatible w/ CERN IPMC
  ○ Based on a powerful STM32H745 microcontroller
  ○ Board design and firmware released under open source license

○ We now develop for three target boards: Pulsar-IIb, Serenity and Apollo
  ○ Firmware can be customized for each target board rather easily
  ○ Tests for the implemented IPMI and non IPMI functions have been successful

○ Testing for compliance with the Polaris Network tester is very encouraging
  ○ 56% of the tests successful, most of failures due to still missing features (multi records,..)
  ○ We are aiming to smooth these issues soon

○ We are looking into extending features to include HPM.1, HPM.2 and HPM.3 standards
Repositories on Gitlab

OpenIPMC (IPMC software)
• [gitlab.com/openipmc/openipmc](https://gitlab.com/openipmc/openipmc)

OpenIPMC-FW (DIMM firmware)
• [gitlab.com/openipmc/openipmc-fw](https://gitlab.com/openipmc/openipmc-fw)

OpenIPMC-HW (DIMM board design)

Breakout baseboard
Questions?
Backup Slides
Command Line Interface: Telnet & UART

- CLI: allows extra control and debug capabilities beyond IPMI protocol (via Telnet or UART)
- Telnet: allows remote connection to IPMC or to any device on board if associated to a UART port

**Telnet**

```
Welcome to minicom 2.7
OPTIONS: 18n
Port /dev/ttyACM0, 19:50:56
Press CTRL-A Z for help on special keys
```

(The CLI can r/w simultaneously over two separate channels)

```
>> info
OpenIPMC-HW
Firmware commit: db7e6708
Target Board: OpenIPMC-HW
IPMB-0 Addr: 0x8c
>>
>> debug-ipmi
IPMB-0 rcvd: 8c 10 64 20 34 2d 03 7c
Get Sensor Reading
IPMB-0 sent: 20 14 cc 8c 34 2d 00 2a c0 00 00 29
IPMB-0 rcvd: 8c 10 64 20 38 2d 03 78
Get Sensor Reading
IPMB-0 sent: 20 14 cc 8c 38 2d 00 2b c0 00 00 24
Key ESC
msg: Command abort
>>
```

**DIMM-UART0**

(via Minicom)
OpenIPMC-HW DIMM connections

Note: among the GPIOs some pins can be configured as UARTs, following the SoC Interest group layout.
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 I²C
  - 6x SPI
  - 8x UART
  - Ethernet MAC
  - USB host/device/OTG
  - Quad-SPI

- **Others**
  - LCD-TFT
  - JPEG Codec
  - ADCs
  - DACs
  - OpAmps
  - Graphical Accelerator
Changes compared to previous presentations

- Platform is now called **OpenIPMC-HW**
- Switched MCU from **STM32H745II** (176 balls 0.65mm pitch) to **STM32H745XI** (240 balls 0.8mm pitch)
  - It’s an elephant, but it seems to still fit on available board space, and is easier to solder
- Removed iCE40 FPGA from the design (as suggested by Peter Wittich)
  - Makes the hardware design and signal routing easier
  - Removes the necessity of programming a second device in the board
STM32H745XI Microcontroller: SMPS x LDO

- **Vcore power sources**
  - Integrated step-down SMPS and LDO to be chosen or combined
  - Each power scheme requires a different off-chip extra circuitry

- **Options available in OpenIPMC-HW (by add/remove components)**
  - LDO only: low efficiency
  - SMPS only: high efficiency
  - SMPS supplying LDO: good efficiency & maximum CPU speeds

![Diagram of Vcore power sources](image)

- **TESTED SUCCESSFULLY AND CURRENTLY IN USE**
- **1.2V for 400MHz CPU speed**
Polaris test @ CERN 06/05/2021 page 1/6
Polaris test @ CERN 06/05/2021 page 3/6

- **Req. 3.749 Temperature Sensors Thresholds**
  - Passed 05-06-2021 15:18:34... Validated that all temperature sensors should have appropriate levels set for minor, major, and critical thresholds.

- **Req. 3.750 Temperature Sensor Hysteresis Commands**
  - Skipped 05-06-2021 15:18:34... Skip cause: All the temperature sensors are skipped.

- **Req. 3.751 Temperature Sensor Default Hysteresis Values**
  - Passed 05-06-2021 15:18:33... Validated that all temperature sensors provide default hysteresis values.

- **Req. 3.752 Critical Temperature Alert Condition**
  - Skipped 05-06-2021 15:18:33... Skip cause: No Fan Geography record found in the Shelf FRU information.

- **Req. 3.753 Telco Alarm Commands**
  - Skipped 05-06-2021 15:18:33... Skip cause: IPCMC 0x1A does not implement any FRU with Telco Alarm functionality.

- **Req. 3.754 Telco Alarm Cutoff Function**
  - Skipped 05-06-2021 15:18:33... Skip cause: IPCMC 0x1A does not implement any FRU with Telco Alarm functionality.

- **Req. 3.755 Telco Alarm Event Message**
  - Skipped 05-06-2021 15:18:41... Skip cause: IPCMC 0x1A does not implement any FRU with Telco Alarm functionality.

- **Req. 3.756 Telco Alarm Event Message**
  - Skipped 05-06-2021 15:18:32... Skip cause: IPCMC 0x1A does not implement any FRU with Telco Alarm functionality.

- **Req. 3.757 Telco Alarm Event Message**
  - Skipped 05-06-2021 15:18:32... Skip cause: IPCMC 0x1A does not implement any FRU with Telco Alarm functionality.

- **Req. 3.758 Telco Alarm Event Message**
  - Skipped 05-06-2021 15:18:32... Skip cause: IPCMC 0x1A does not implement any FRU with Telco Alarm functionality.

- **Req. 3.769 Telco Alarm Major Reset Input**
  - Passed 05-06-2021 15:18:43... Validated that Telco Manager and IPMC Controller functions are supported as defined in Table 3-99.

- **Req. 3.770 Telco Alarm Major Reset Input**
  - Passed 05-06-2021 15:18:44... Validated that IPMI Command number functions and requirements are supported as defined in Table 3-100.

- **Req. 3.771 Telco Alarm Major Reset Input**
  - Failed 05-06-2021 15:18:40... Error: IPCMC 0x1A does not support Event Receiver command.

- **Req. 3.772 Shelf Manager and IPMC Functionality**
  - Failed 05-06-2021 15:18:40... Failed to validate IPCMC 0x1A’s Entity ID assignments.

- **Req. 3.773 Command Number Assignment**
  - Failed 05-06-2021 15:18:40... Failed to validate IPCMC 0x1A’s Entity ID assignments.

- **Req. 3.774 Command Number Assignment**
  - Failed 05-06-2021 15:18:40... Failed to validate IPCMC 0x1A’s Entity ID assignments.

- **Req. 3.775 Entity ID Assignment**
  - Failed 05-06-2021 15:18:40... Failed to validate IPCMC 0x1A’s Entity ID assignments.

- **Req. 3.776 Entity ID Assignment**
  - Failed 05-06-2021 15:18:40... Failed to validate IPCMC 0x1A’s Entity ID assignments.

- **Board ID-Extending**
  - Passed 05-06-2021 15:18:51... Validated that PICMC Entity ID assignments shall be supported as defined in Table 3-104; “PICMC Entity ID assignments.”

- **Watchdog Timer Commands**
  - Skipped 05-06-2021 15:18:52... Skip cause: No support for Watchdog Timer commands.

- **Watchdog Timer Commands Support**
  - Skipped 05-06-2021 15:18:52... Skip cause: No support for Watchdog Timer commands.
Polaris test @ CERN 06/05/2021 page 4/6
Polaris test @ CERN 06/05/2021 page 5/6
OpenIPMC Internal Concept

Physical link layer
- Multi-master I²C
- Use 2 PS I²C controllers

Buffers the messages
- Collects income IPMI messages (Requests and Responses)
- Manage the transmission over the IPMB channels (arbitration)

Abstracts away the IPMI transport layer
- Manages sequence #, destination and checksums
- Associates responses to requests
- Retries and timeouts
- Accepts multiple internal requests (from different tasks)
- Call the specific functions to solve external requests

“Application layer”
- Hot Swap operation
- Power Negotiation
- Sensor Records
- Sensor Readings
- Other IPMI functions
  - user-implemented
OpenIPMC on ESP32 (Espressif Systems, CN)

- Quite powerful & flexible uC
  - 240 MHz Xtensa LX6 dual core
  - FPU, Big INTs & Crypto
  - WiFi, BT, SPI, I2C, UART…
  - FreeRTOS support

- Cheap Linux-supported boards
  - CP2102 USBtoUART converter
  - Boards sell for 5$

- 3.3 V device (same as IPMB)

- Development software
  - Arduino IDE, PlatformIO or **esp-idf**

- Very different arch w.r.t a Zynq US+
  - Good exercise on portability/
IPMC firmware & board for Pulsar-2b

- SPRACE collab with Fermilab (2014-2016)
  - AM+FPGA L1 Track Finder
  - One contribution was the IPMC for the Pulsar-2b
  - MCU: NXP LPC1700 (ARM Cortex-M3)
  - RTOS: ARM Keil RTX (proprietary compiler)
  - IPMC worked well and reliably, but...

- Non-generic implementation
  - Minimum set of required IPMI commands
    - Hot Swap and sensor readings
  - Other Features (not IPMI/PICMG)
    - TCP/IP & Xilinx Virtual Cable (XVC) for FPGA debug

- Rather rigid code base
  - Hard coded variables
  - Difficult to customize and port
  - Single task for all IPMI functions

- Redesign from scratch for ZynqMP
  - Pulsar-2b IPMC was the inspiration
  - Led to OpenIPMC
First target platform for ZynqMP development

**Trenz + Serenity setup (KIT)**

- Serenity ATCA card (Imperial College)
- Trenz Elektronik TE0803 module
  - Zynq US+ ZU4EG SoC
- Trenz Adapter board (KIT)
  - Interface TE0803 to COM Express slot
  - Additional IPMC features
  - (I2C buffers, Eth Phy, EEPROM, SDCard…)
  - Interface to DIMM adapter
- DDR3 Mini-DIMM Adapter (KIT)
  - Fits into CERN IPMC-compatible slot
  - Access to IPMC backplane signals
Fixes to ESP32 Integrated Development Framework

- I2C multi-master with variable size I2C msgs is required for IPMB bus communication
  - End of message is signalled by a stop bit
- The official esp-idf I2C driver was not supporting variable size msgs correctly
  - The driver expected a message size to be specified in advance
- We modified the driver* and now the slave read function correctly returns if receiving a stop bit

*The github esp-idf repository was forked. A merge request is still in progress
This layer allows OpenIPMC to run on the platform, and is (ideally) the only part that needs to be adapted.

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.
Xilinx ZynqMP SoC as unified mgmt module

- **Needs of ATCA boards for LHC experiments**
  - IPMC → board management & monitoring
  - Linux → higher-level functions (e.g. calibration)
  - Xilinx ZynqMP SoCs can satisfy both roles using one unit

- **Zynq Ultrascale+ MPSoC**
  - Two processor domains: Application PU and Real-time PU
  - Xilinx FPGA programmable logic (good 4 sys integration)
  - Plethora of peripherals (PCIe, ETH, I2C, UART, USB, …)

- **Power domain partitioning**
  - Low PD (ARM-R5 RPU) → IPMC (standalone/RTOS)
  - Full PD (ARM-A53 APU) → Slow Control (Linux)
  - PL PD (FPGA) → partitioned between IPMC and Linux uses

- **Pros and cons of tighter IPMC/Linux integration**
  - Simple IPMC/Linux communication through mem registers
  - Very flexible implementation
  - Can be optimized for reduced board area occupation
  - Complex gymnastics between the two systems
Architectures and boards that run OpenIPMC (so far)
1) Ultra96 + Pulsar-2b
We began OpenIPMC on AVNET Ultra96

- [link](https://www.96boards.org/product/ultra96/)
- Plenty of tutorials & Vivado support
- Excellent price (249$) allows buying more boards
  - More boards can be used by developers

Ultra96 uses Zynq Ultrascale+ ZU3EG

- Same family as ZynqMP Mgmt. Module
- APU → 4 x Cortex A-53
- RPU → 2 x Cortex R-5
- PL → Kintex US+ - like FPGA fabric
Using generic development boards in the ATCA shelf

- Dev board (e.g. Ultra96)

- Pulsar-2b board exposes signals to DIMM slot
  - IPMB-A and -B buses
  - Pulsar-2b LEDs
  - Pulsar-2b main power enable (not used so far)
  - Pulsar-2b local I2C for sensors (not used so far)

- Mini-DIMM adapter
  - Connects Pulsar-2b DIMM slot to Ultra96
  - Translates 1.8 V (Ultra96) ↔ 3.3V (ATCA)
  - Design and manufacture by Luis Ardila (KIT)

- Comtel CO6 ATCA chassis
  - Full-mesh, 6 slots horizontal
  - 2 PigeonPoint ShelfManagers (redundant)
Ultra96 + Pulsar-2b in the shelf

Digilent Analog Discovery USB o’scope as I2C logic analyzer

Monitoring IPMB-A with TeK o’scope
2) Trenz module + Serenity
OpenIPMC tests on Trenz + Serenity setup @ KIT

- From Ultra96, OpenIPMC code was successfully ported to Trenz+Serenity setup at KIT
  - Adapting HAL and Board-specific ctrls
  - All changes in one file

- All changes relays into the ipmc_ios.c file

- Hot-Swap operation successfully tested on Serenity board

- Since no real sensor are currently being read in this hardware
OpenIPMC tests on Trenz + Serenity setup @ KIT

Activation Status

# clia fru -v 96
Pigeon Point Shelf Manager Command Line Interpreter
96: FRU # 0
  Entity: (0x00, 0x1)
  Hot Swap State: M4 (Active), Previous: M3 (Activation In Process), Last State Change Cause: Normal State Change (0x0)
  Device ID String: "Trenz-Serenity"
  Site Type: 0x00, Site Number: 02
  Current Power Level: 0x02, Maximum Power Level: 0x02, Current Power Allocation: 100.0 Watts

FRU Information  (testing data from example code)

# clia fruinfo 96 0
Pigeon Point Shelf Manager Command Line Interpreter
96: FRU # 0, FRU Info
Common Header:  Format Version = 1
Board Info Area:
  Version = 1
  Language Code = 25
  Mfg Date/Time = Oct 1 00:00:00 2019 (12490560 minutes since 1990)
  Board Manufacturer = SPSPACE - KIT
  Board Product Name = OpenIPMC @ Trenz-Serenity
  Board Serial Number = 180180981-18099
  Board Part Number = AA00Y99
  FRU Programmer File ID = 01

Sensor Reading  (testing data from example code)

# clia sensordata 96 3
Pigeon Point Shelf Manager Command Line Interpreter
96: LUN: 0, Sensor # 3 ("FPGA TEMP")
  Type: Threshold (0x01), "Temperature" (0x01)
  Belongs to entity (0xa0, 0x60)
  Status: 0xc0
    All event messages enabled from this sensor
    Sensor scanning enabled
    Initial update completed
  Raw data: 50 (0x32)
  Processed data: 50.00000000 degrees C
  Current State Mask: 0x00
3) ESP32 + Pulsar-2B
Porting OpenIPMC to ESP32

- ESP32 microcontroller (see backup slides)
  - Very different from a Zynq US+

- Questions answered by this exercise
  - Architecture independency
    - Trivial, thanks to C and FreeROTS
  - Ease of integration on a different SoC
    - OpenIPMC needs I2C peripheral
    - Many SoCs have 2 or more
  - Effort needed to port OpenIPMC
    - Mainly IO/HAL interface bindings
    - Fixes needed in ESP32 IDF (see backup)
    - Porting took just 3 person-weeks :-)

- Overall the exercise was a success
- Repo: gitlab.com/openipmc/ipmc-esp32
OpenIPMC tests on ESP32

- IPMBus communication works
- ShM happily accepts the FRU
- Activation/deactivation are triggered using an ‘improvised’ Handle Switch
- Activation time significantly longer than in Ultra96
  - Likely due to ESP32 drivers
4) STM32 H745 + Pulsar-2B
ST Microelectronics STM32H745

- Powerful industrial control-oriented MCU
  - 480 MHz ARM Cortex-M7 main CPU
  - 240 MHz ARM Cortex-M4 aux CPU

- Plenty of peripherals
  - 4 x hardware I2C, 6 x hardware SPI
  - 4 USART + 4 UART + 1 LPUART
  - Up to 168 GPIO

- Moderate current consumption
  - 600 mA absolute max / 80-200 mA typ current

- Free development environment & compiler
  - STM32CubeIDE (gcc in the back-end)
  - Compatible with Linux, OpenOCD and GDB
  - ST provides a FreeRTOS distribution for STM32
ST Microelectronics NUCLEO

- For development we use the ST [NUCLEO-H745ZI-Q](#) devboard
  - STM32H745 in LQFP-144 package (same silicon, less pins than the TFBGA-240+25)
  - Easy to get from distributors and CERN stores, cheap (around 23 CHF)
Porting OpenIPMC to STM32

- Porting was similar to ZYNQ and ESP32, thanks to FreeRTOS being supported on STM32
  - We wrote a new OpenIPMC HAL to interface with the STM32 drivers

- Porting OpenIPMC core functions (IPMB-0) took just 4 person-weeks
  - Usual show-stopper was the I2C driver implementation of I2C multi-master mode
    - Relatively painless fix, similar to the Zynq case

- Testing: IPMI communication works properly on STM32

STM32 responding to shelf manager with 200 μs latency (timeout 300 ms)

NUCLEO board mounted onto the Pulsar-2b
5) Serenity-A2577 + ZynqMP Mezzanine
OpenIPMC Ported to the ZynqMP R5 Cores

- FMC+ management module with ZynqUS+ device
- OpenIPMC ported to the R5 cores
- CentOS 7 based root filesystem + petalinux kernel runs on the A53 cores
- Upstream OpenIPMC software in use via submodule in the Zynq R5 firmware framework.
- new PIM400 sensors working

Virtex Ultrascale+ VU9P / VU13P A2577
120 / 128 GTY

DAQ
4 @ 25 Gb/s

TFP
20/24 @ 25 Gb/s

DET
36 LpGBT

TCDS
C2C to ZynqMP

SAMTEC Firefly Optics