The OpenIPMC project

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

André Cascadan (UNESP - São Paulo)

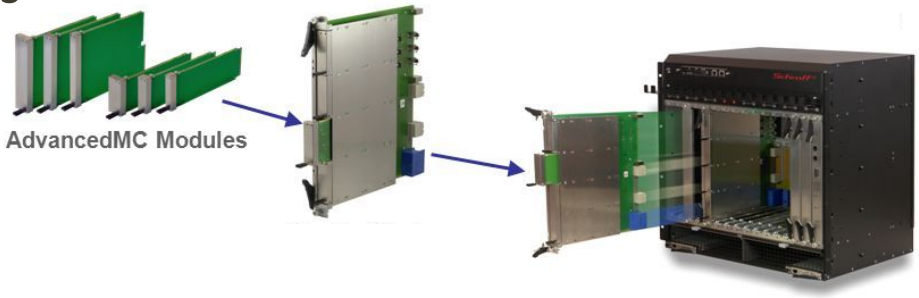
ICTP, SAIFR and UNESP School on Systems-on-Chip, Embedded Microcontrollers and their Applications in Research and Industry

21-Oct-2021

The ATCA

Advanced Telecommunication Computing Architecture (ATCA)

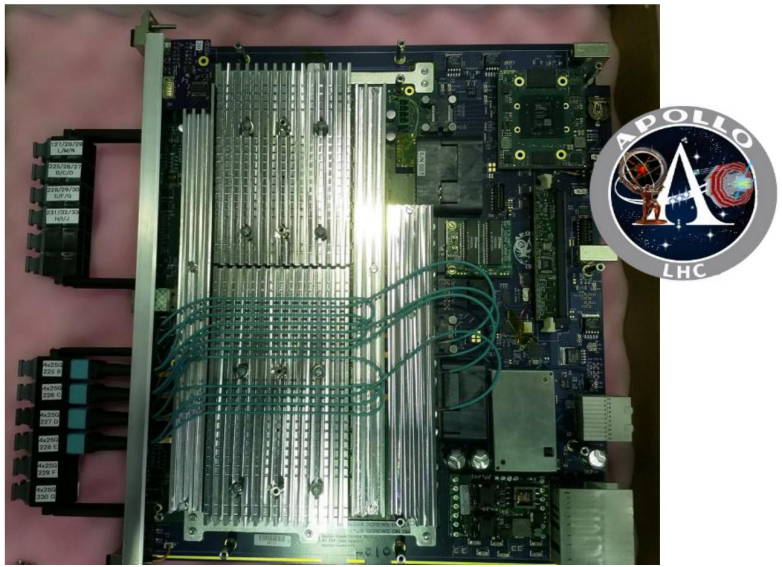
- Industrial computing standard
  - Standardized by PICMG
- Reliability
  - System monitoring & management via network
  - Power and management redundancy
- High power available to devices
  - Forced air cooling by the use of fans
  - More than 10 kW / cabinet possible
- High speed backplane connections
  - Precise clock synchronization (10s of ps jitter)
  - High-speed communication (16-20 Gbit)
- Modularity
  - Each ATCA board can still support Mezzanines
    - **Advanced Mezzanine Cards - AMC**



ATCA boards in High Energy Physics

- HEP employes ATCA boards to build its custom processing electronics
  - Mainly intended to host **powerful FPGAs** and large amount of **optical connections**

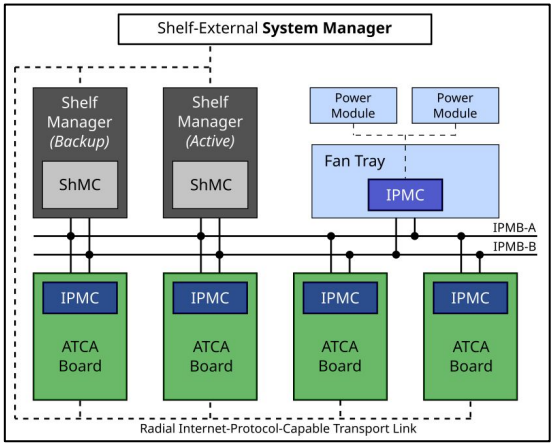
Serenity board (Imperial College London)



Apollo board (Boston University)

ATCA and its management system

- ATCA shelf like a LEGO set
  - Mechanical frame + interconnection backplane
  - Replaceable components (FRUs)
    - Boards, power supplies, boards, shelf managers, ...
- Shelf Manager Controller (ShMC)
  - Orchestrates turning on/off the FRUs
  - Controls the power of the fans (cooling loop)
  - Polls to each FRU to monitor its conditions
  - Provides a shelf management interface via UDP/IP
- Intelligent Platform Management Controller (IPMC)
  - One for each FRU (including the boards)
  - Manage board startup and shutdown in graceful way
  - Reports the status of the board to the ShMC
  - Agrees with the ShMC changes in the power state

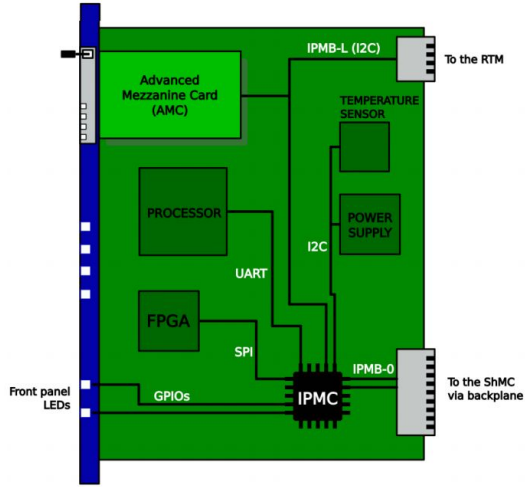


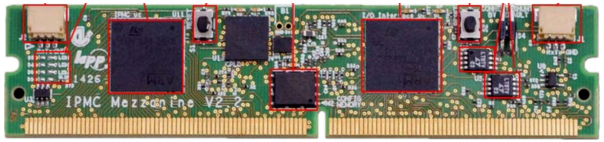
The ShMC is an orchestra director, doesn't know how to play an instrument

The IPMC follows the ShMC and knows to play its own specific instrument

IPMCs for the ATCA boards

- Now, consider ATCA boards designed by the user
  - e.g. the Serenity or the Apollo boards for the CMS tracker
- Standard communication protocol with ShMC
  - IPMI: Intelligent Platform Management Interface over IPMB
  - IPMB: Intelligent Platform Management Bus, based on I2C
- 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 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)
- A variety of IPMCs were already designed on this standard
  - LAPP IPMC
  - Fermilab IPMC
  - CERN IPMC





LAPP IPMC

FNAL IPMC

CERN IPMC

Characteristics of the project

- IPMC software (OpenIPMC)
  - Open Source SW → No license problems (e.g. students), flexibility
  - Multiplatform → Freedom to choose MCU and host hardware
  - PICMG-compliant → Implements IPMI to perform IPMC operations according to PICMG
- IPMC DIMM hardware module (OpenIPMC-HW)
  - Open Source HW → Any group can re-use the hardware design
  - LAPP pinning → Largely adopted by ATCA boards in LHC
  - Rather simple PCB → Can be fabricated by many PCB manufacturers and assemblers
  - 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

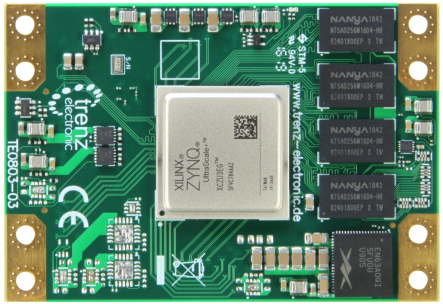
Brief intro on the OpenIPMC software

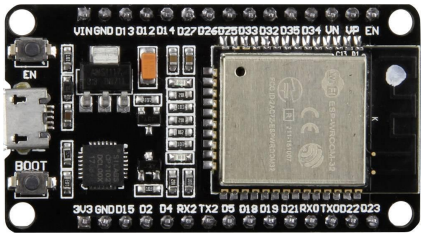
OpenIPMC software

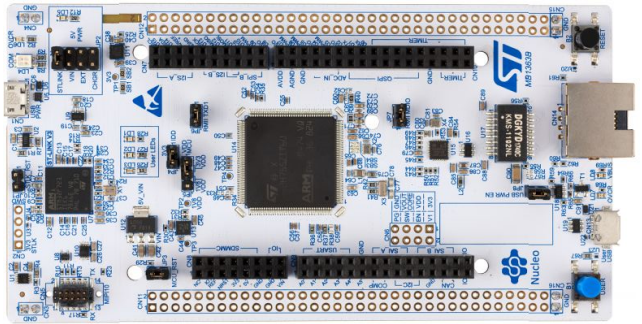
- IPMC software implementing PICMG-compliant IPMI functions
  - Power negotiation and hot-swap (M-states, handle, etc.)
  - Instantiate board sensor records, declare them to ShM, read-out and publish data
  - Focus on simplicity: optional functions can be added to the project by the user
- Platform-independent 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 (Cortex M7)

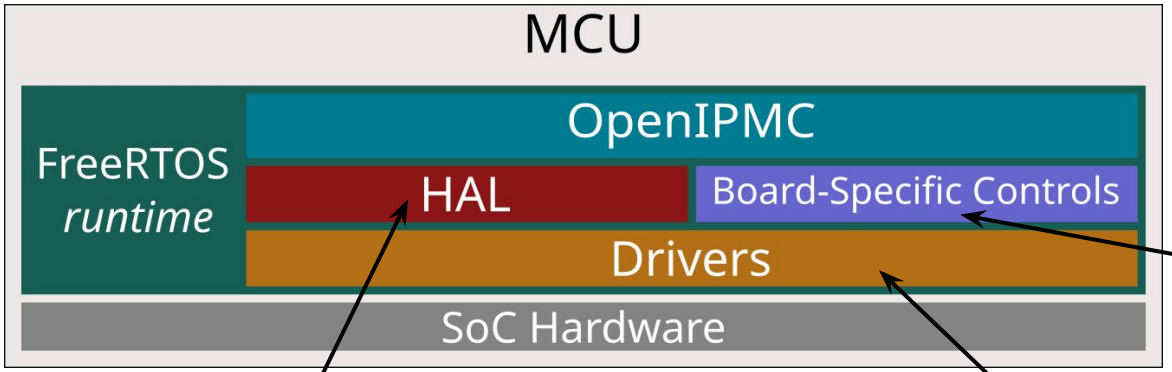






How OpenIPMC interfaces to the hardware

- 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



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

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)

HAL example: *Blue LED*

- PICMG states that all ATCA board must have a *Blue LED* in its faceplate
- Tells user about the activation state
  - OFF: board is active
  - Short blink: board is shutting down (“Please, do not pull the board out!”)
  - ON: board is inactive (“Board can be removed safetly”)
  - Long blink: board is activating
- *Blue LED* is controlled by OpenIPMC software
  - Its presence does not depend on any board design choice (ARM, ATMEL, ...)
  - So, how to control this led?

HAL example: *Blue LED*

```c
void ipmc_blue_led_blink_task( void )
{

    int        blue_led_mode = BLUE_LED_OFF;
    TickType_t block_mode    = 0;

    // Waits for the peripherals to be available
    while ( ipmc_ios_ready() != pdTRUE )
        vTaskDelay( pdMS_TO_TICKS(100) );

    queue_blue_led_mode = xQueueCreate( 3, sizeof( int ) );


    for(;;)
    {

        xQueueReceive(queue_blue_led_mode, &blue_led_mode, block_mode);

        switch( blue_led_mode )
        {
            case BLUE_LED_SHORT_BLINK:

                // When blinking, task do not block
                block_mode = 0;

                // Blink
                ipmc_ios_blue_led(0);
                vTaskDelay( pdMS_TO_TICKS(450) );

                ipmc_ios_blue_led(1);
                vTaskDelay( pdMS_TO_TICKS(100) );

                ipmc_ios_blue_led(0);
                vTaskDelay( pdMS_TO_TICKS(450) );
                break;
```

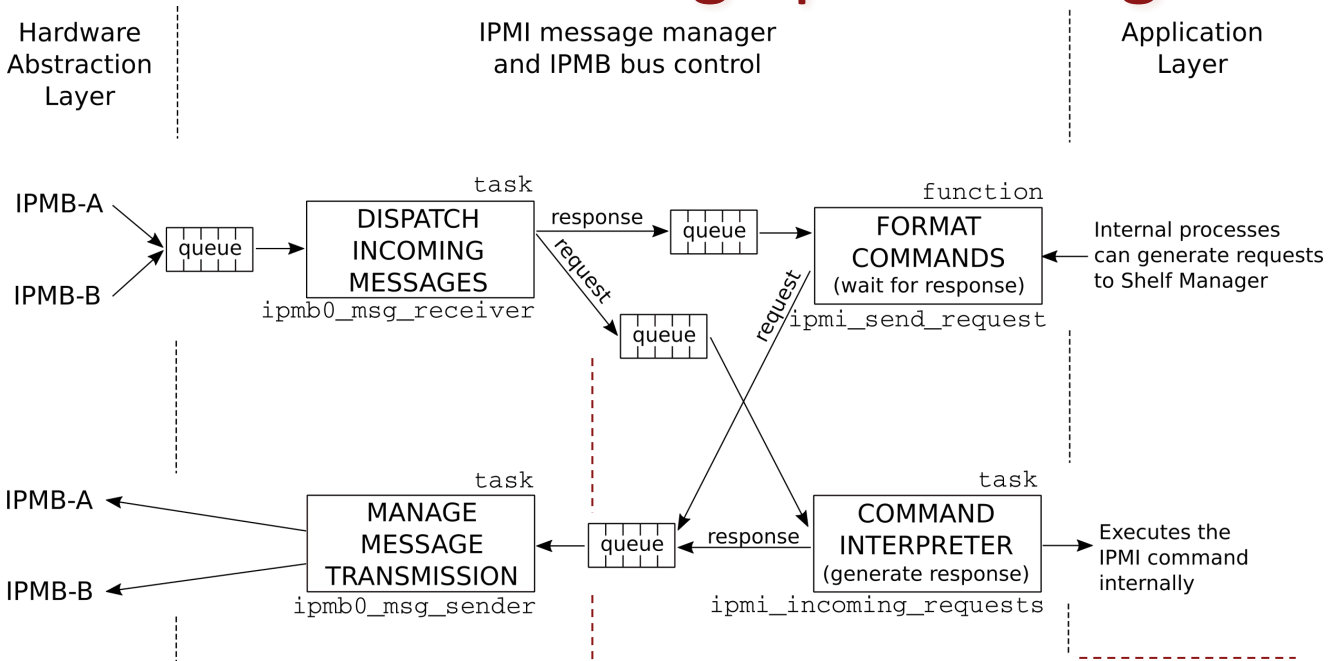
Blue LED task in OpenIPMC-SW

- OpenIPMC-SW has a task to control the *Blue LED*
  - `ipmc_ios_blue_led()` must make the led ON or OFF
  - `ipmc_ios_blue_led()` is left <u>undefined</u> in the SW project
- Adding OpenIPMC-SW into to STM32 project
  - The developer includes OpenIPMC-SW into the FW project
  - `ipmc_ios_blue_led()` is then defined using the specific API
  - No need to modify any line OpenIPMC-SW project

```c
/*
 * Control the Blue Led
 */
void ipmc_ios_blue_led(int blue_led_state)
{
    if (blue_led_state == 0) // LED OFF
        HAL_GPIO_WritePin( FP_LED_BLUE_GPIO_Port, FP_LED_BLUE_Pin, GPIO_PIN_RESET );
    else // LED ON
        HAL_GPIO_WritePin( FP_LED_BLUE_GPIO_Port, FP_LED_BLUE_Pin, GPIO_PIN_SET );
}
```

OpenIPMC-SW port for STM32

Multitask IPMI message processing



- Hot Swap operation
- Power Negotiation
- Sensor Records
- Sensor Readings
- Other IPMI functions
  - user-implemented

Physical link layer
- Multi-master I^2C
- Use 2 PS I^2C ctrlers

Buffers the messages
- Collects incoming 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

OpenIPMC-HW

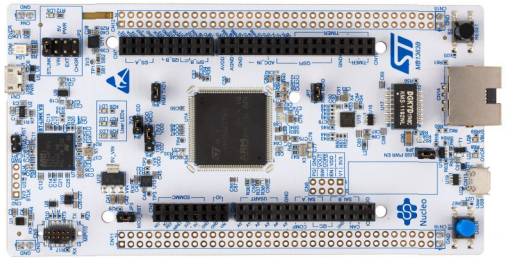
Choice of the microcontroller

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 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



STM32 NUCLEO-H745ZI-Q

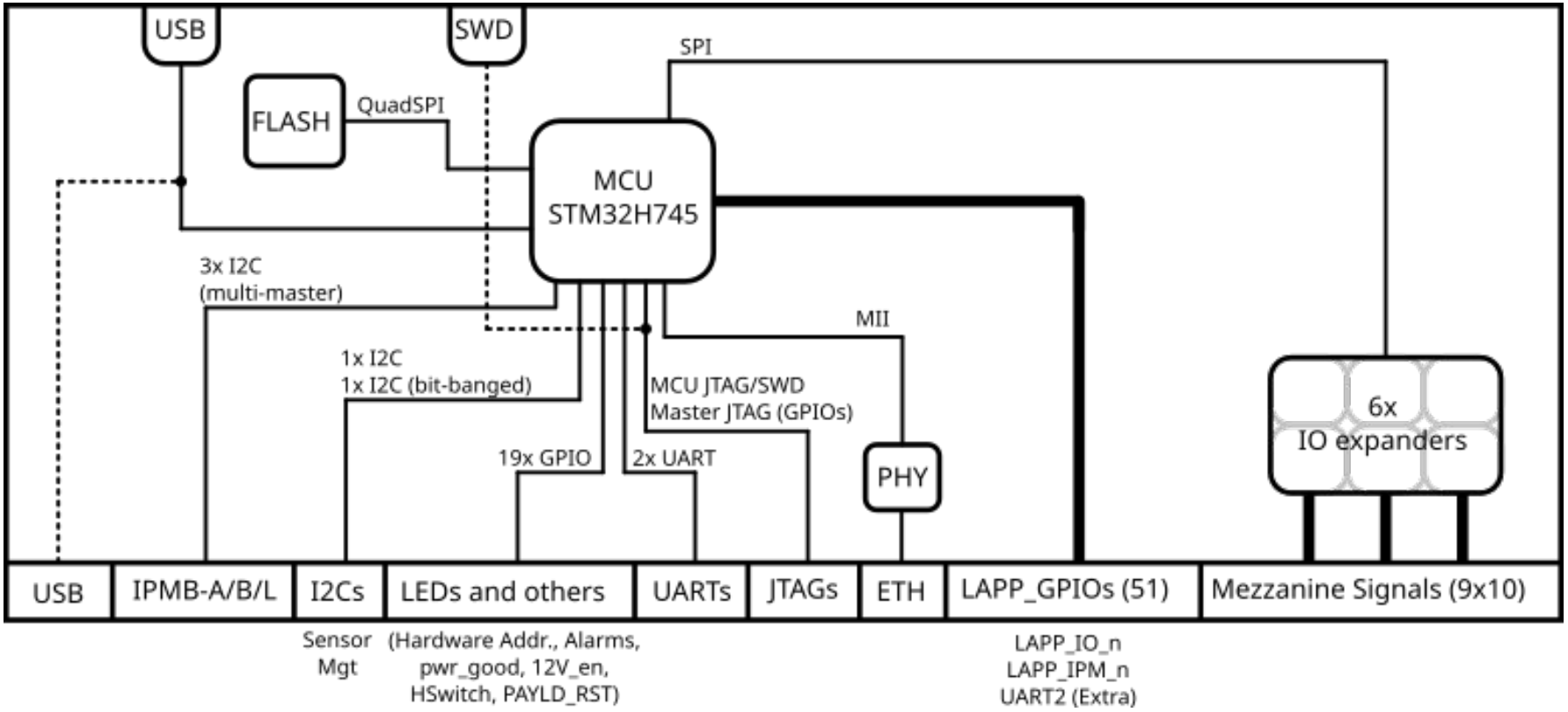
What we get in addition

- 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)

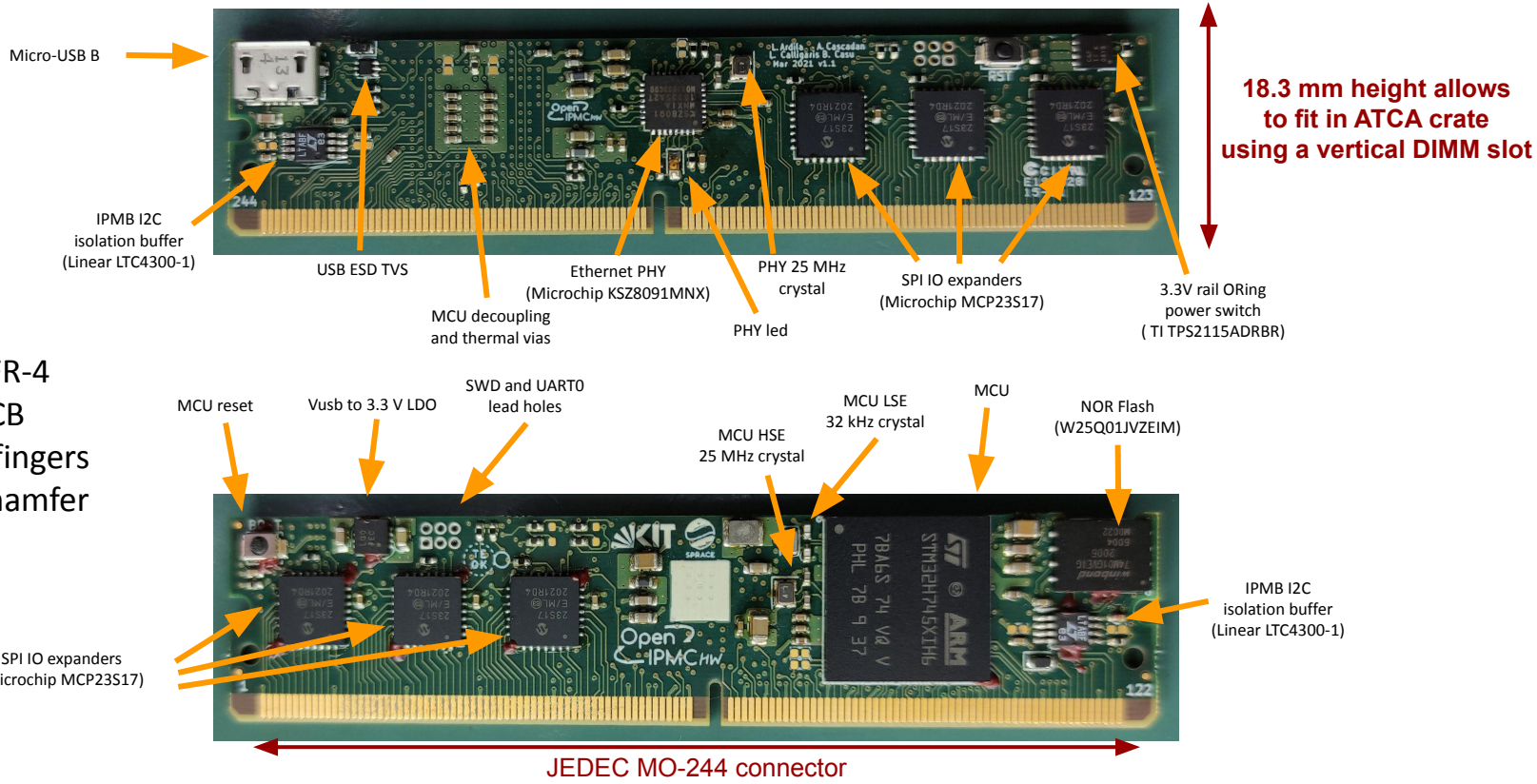
Full documentation on ST site

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

OpenIPMC-HW layout: schematic

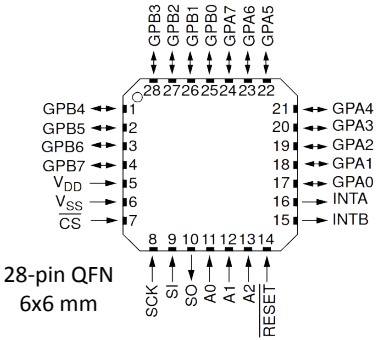


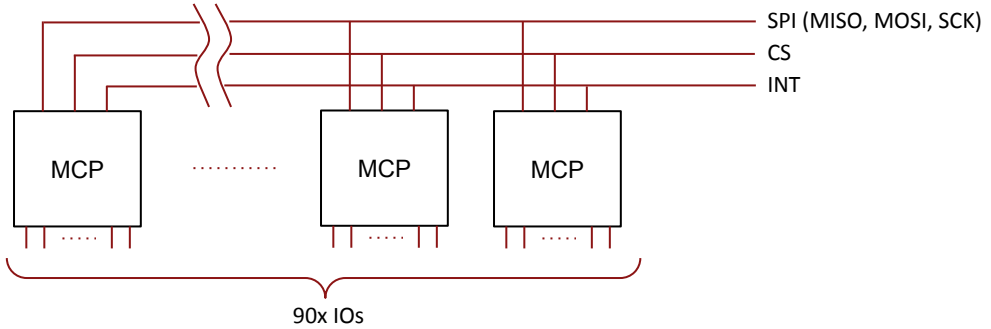
OpenIPMC-HW layout: picture



AMC IO expanders

- 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





Writing a driver for the IO expanders

- The driver allows to control all the 90 pins as it was a typical MCU GPIO
- Internally, the driver uses the SPI HAL provided by STM32CubeIDE to control all devices
  - Each device is selected by its address, sent into SPI stream (in 1st byte)
  - Each device has a set of registers to be written/read via SPI (address in 2nd, and data in 3rd byte)

How the driver interface looks like...

```
// Set pin 5 as output
amc_gpios_set_pin_direction( 5, OUT );

// Turn on pull-up on pin 8
amc_gpios_set_pin_pullup( 8, ON );

// Set pin 20 as HIGH
amc_gpios_write_pin( 20, 1 );

// Read pin 60
pin_60 = amc_gpios_read_pin( 60 );
```

How the IO expander wants it...

TABLE 3-3: SUMMARY OF REGISTERS ASSOCIATED WITH THE GPIO PORTS (BANK = 0)

Register Name	Address (hex)	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	POR/RST value
IODIRA	00	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IODIRB	01	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IPOLA	02	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0	0000 0000
IPOLB	03	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0	0000 0000
GPINTENA	04	GPINT7	GPINT6	GPINT5	GPINT4	GPINT3	GPINT2	GPINT1	GPINT0	0000 0000
GPINTENB	05	GPINT7	GPINT6	GPINT5	GPINT4	GPINT3	GPINT2	GPINT1	GPINT0	0000 0000
GPPUA	0C	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0	0000 0000
GPPUB	0D	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0	0000 0000
GPIOA	12	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
GPIOB	13	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
OLATA	14	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000
OLATB	15	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000

IO expanders driver code

```c
void amc_gpios_write_pin( uint8_t amc_pin, uint8_t pin_value )
{
    uint8_t reg_addr;

    // Protection against bad value
    if(amc_pin >= 90) return;

    // Calculate register addr GPIOA or GPIOB. Base addr: 0x12
    reg_addr = 0x12 + pin_map[amc_pin].port;

    set_expander_register_bit( pin_map[amc_pin].device, reg_addr, pin_map[amc_pin].pin, pin_value );
}
```

```c
static void set_expander_register_bit( uint8_t device_addr, uint8_t reg_addr, uint8_t bit_posic, uint8_t bit_value )
{
    uint8_t val;

    WAIT_FOR_MUTEX();

    // Get current value
    READ_1_REG( device_addr, reg_addr, val);

    val = val &~(0x01<<bit_posic); // The target bit is cleared
    if( bit_value != 0 )
        val |= ( 0x01<<bit_posic );

    // Write new value
    WRITE_1_REG( device_addr, reg_addr, val);

    GIVE_MUTEX();
}
```

IO expanders driver code

```
// Writes ONE register into an Expander device
#define WRITE_1_REG( DEVICE_ADDR, REG_ADDR, VAL)                                    \
    {                                                                               \
        while(spi_status != IDLE) { asm("nop"); }                                   \
        spi_status = BUSY;                                                          \
        spi_data_out[0] = 0x40 | (DEVICE_ADDR<<1);                                  \
        spi_data_out[1] = REG_ADDR;                                                 \
        spi_data_out[2] = VAL;                                                      \
        SCB_CleanDCache_by_Addr((uint32_t*)spi_data_out, 6*sizeof(uint8_t));        \
        HAL_SPI_TransmitReceive_DMA(&hspi_amc, spi_data_out, spi_data_in, 3);       \
    }

// Reads ONE register from an Expander device
#define READ_1_REG( DEVICE_ADDR, REG_ADDR, VAL )                                    \
    {                                                                               \
        while(spi_status != IDLE) { asm("nop"); }                                   \
        spi_status = BUSY;                                                          \
        spi_data_out[0] = 0x41 | (DEVICE_ADDR<<1);                                  \
        spi_data_out[1] = REG_ADDR;                                                 \
        spi_data_out[2] = 0  ;                                                      \
        SCB_CleanDCache_by_Addr((uint32_t*)spi_data_out, 6*sizeof(uint8_t));        \
        HAL_SPI_TransmitReceive_DMA(&hspi_amc, spi_data_out, spi_data_in, 3);       \
        while(spi_status != IDLE) { asm("nop"); }                                   \
        SCB_InvalidateDCache_by_Addr((uint32_t*)spi_data_in, 6*sizeof(uint8_t));    \
        VAL = spi_data_in[2];                                                       \
    }
```

...and finally,
the STM32
SPI HAL interface!

External flash

- The MCU supports external QSPI flash
  - QSPI → uses four pins for data → FASTER!
- We added one to the DIMM to
  - Store configuration, file system
  - Aid in firmware upgrade binary gymnastics
  - Store a failsafe “golden image” or backup
- Our chosen part is Winbond W74M01GVZEIG
  - NAND Flash
  - WSON-8 package
  - 1 Gbit = 128 MiB with 128kiB erasable blocks
  - Cheap (7.33 USD or equivalent W25N01GVZEIG for 3.27 USD)

External flash: read example

- All the read/write operations are commands over QSPI bus
- Example: read operation
  - Memory is divided in pages of 2048 bytes each
  - In a page, data are addressed by their column address
  - Read sequence:
    i. Send *Page Data Read* with the page address: the page content is fetched to the flash internal buffer
    ii. Send *Read* with the column address to transfer the data from flash chip to the MCU
      - Transfer starts from the byte in the referred column address

Address Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SpiFlash (up to 128M-Bit)	X	X	X	X	X	X	X	X	64KB Block Addr (256 Blocks)								Page Address (256 Pages)								Byte Address (0-255 Byte)							
SpiFlash (up to 32G-Bit)	64KB Block Address																Page Address (256 Pages)								Byte Address (0-255 Byte)							
Serial NAND (1G-Bit)	X	X	X	X	**Page Address (PA) [15:0]**																**Column Address (CA) [11:0]**											
	X	X	X	X	128KB Block Addr (1024 Blocks)												Page Addr (64 Pages)				Ext	Byte Address (0-2047 Byte)										

Commands	OpCode	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9
Page Data Read	13h	Dummy	PA15-8	PA7-0					
Read	03h	CA15-8	CA7-0	Dummy	D7-0	D7-0	D7-0	D7-0	D7-0

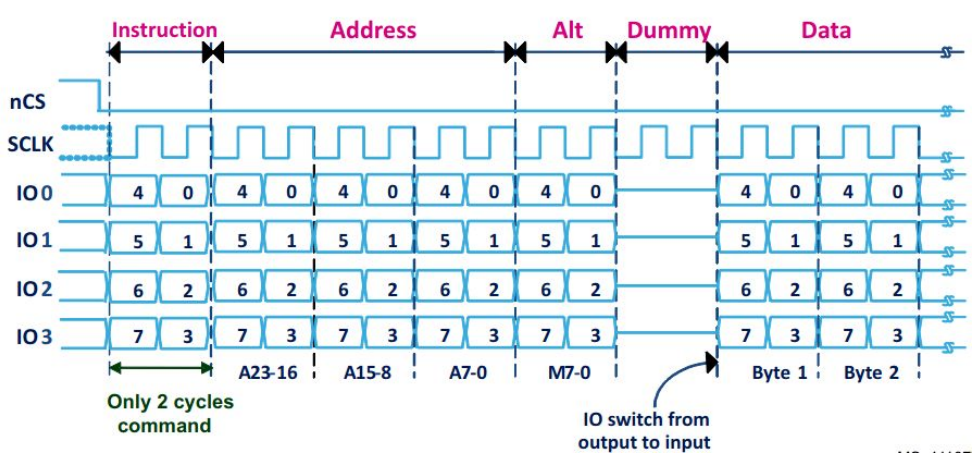
• • • Master clocks until transfer finish

Writing a driver for the External flash

```
void w25n01gv_page_data_read( uint16_t page )
{
    QSPI_CommandTypeDef cmd;

    INIT_CMD_WITH_DEFAULTS(cmd);

    cmd.Instruction = 0x13; // Page Data Read
    cmd.AddressMode = QSPI_ADDRESS_1_LINE;
    cmd.AddressSize = QSPI_ADDRESS_24_BITS; // 1 dummy byte + 2 address bytes
    cmd.Address     = page;

    HAL_QSPI_Command  ( &hqspi, &cmd,  100 );
}
void w25n01gv_read( uint16_t column, uint32_t len, uint8_t* data )
{
    QSPI_CommandTypeDef cmd;

    INIT_CMD_WITH_DEFAULTS(cmd);

    cmd.Instruction = 0x03; // Read
    cmd.AddressMode = QSPI_ADDRESS_1_LINE;
    cmd.AddressSize = QSPI_ADDRESS_16_BITS;
    cmd.Address     = column;
    cmd.DummyCycles = 8;
    cmd.DataMode    = QSPI_DATA_1_LINE;
    cmd.NbData      = len;

    HAL_QSPI_Command ( &hqspi, &cmd, 100 );
    HAL_QSPI_Receive ( &hqspi, data,100 );
}
```

Load the internal flash buffer with the requested page

Receive the page content into **`data`** array



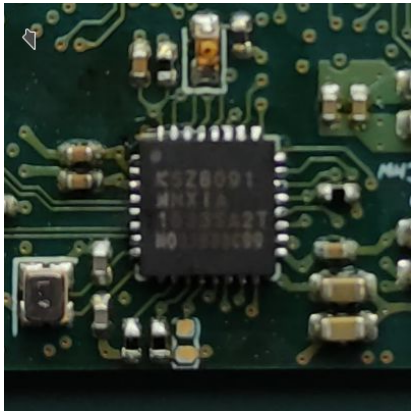
Using driver for the External flash

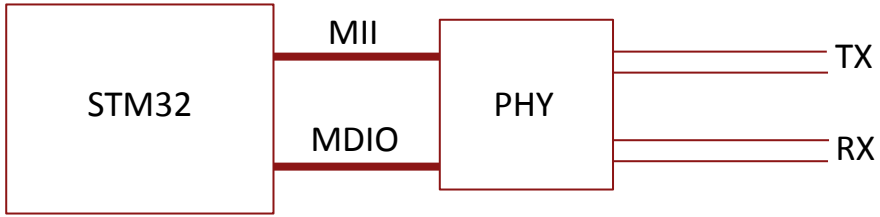
```
void ext_flash_read(uint32_t addr, uint32_t len, uint8_t* data)
{
    uint32_t read_index = 0;
    uint32_t bytes_to_read;

    while( read_index < len )
    {
        // Load page
        w25n01gv_page_data_read( (addr+read_index)/PAGE_SIZE );
        while( w25n01gv_is_busy() )
            vTaskDelay( pdMS_TO_TICKS(10) );

        bytes_to_read = PAGE_SIZE - ((addr+read_index)%PAGE_SIZE);
        if( (read_index + bytes_to_read) >= len )
            bytes_to_read = len - read_index;

        w25n01gv_fast_read_quad( (addr+read_index)%PAGE_SIZE, bytes_to_read, &data[read_index]);
        read_index += bytes_to_read;
    }
}
```

Reads any amount of data from any address

Ethernet

- LwIP over FreeRTOS
  - Flexibility for custom management
    - IPMI is slow and may not cover all the desired remote operations
  - NTP, FTP, TFTP, Telnet, XVC...
- PHY: Micrel/Microchip KSZ8091MNX
  - 10/100 Mbit with auto MDI/MDI-X
  - Media Independent Interface (MII) @ 25 MHz
  - Management Data Input/Output (MDIO)





USB interface

- Most STM32 MCUs support USB 2.0 natively!
- We intend for use in two functions
  - Easy access to the IPMC Command Line Interface
  - DFU firmware update via a simple USB cable
- Very helpful for pre-configuration

Command line interface

Available via
UART, USB and Telnet

```
cascadan@novaipiranga:~$
cascadan@novaipiranga:~$ telnet 192.168.0.12
Trying 192.168.0.12...
Connected to 192.168.0.12.
Escape character is '^]'.

>>
>> help
Count command: 6
[] - mandatory argument
<> - optional argument
| - choice between arguments

~          - reset cpu
-----------------------------------------------------------------

info       - Print information about this Bootloader.
-----------------------------------------------------------------

run        - Run Firmware
-----------------------------------------------------------------

load-bin   - Load binary from a TFTP server.
               ex: load-bin 192 168 0 1
               This client always load file named fimware.bin
-----------------------------------------------------------------

erase-firmware - Erase Firmware
-----------------------------------------------------------------
>>
>>
```

`help`: List of commands

```
>>
>> erase-firmware
SectorsErased

>>
>>
>>
>> load-bin 192 168 0 1

>> ..............................................
 206932 bytes written into Flash

>>
>> runConnection closed by foreign host.
cascadan@novaipiranga:~$ telnet 192.168.0.12
Trying 192.168.0.12...
Connected to 192.168.0.12.
Escape character is '^]'.

>>
>> info

OpenIPMC-HW
Firmware commit: 8cf13090

Target Board: some-ATCA-board
IPMB-0 Addr: 0xee

>>
```

Update sequence and new telnet connection
to the OpenIPMC-FW running

Firmware upgrade

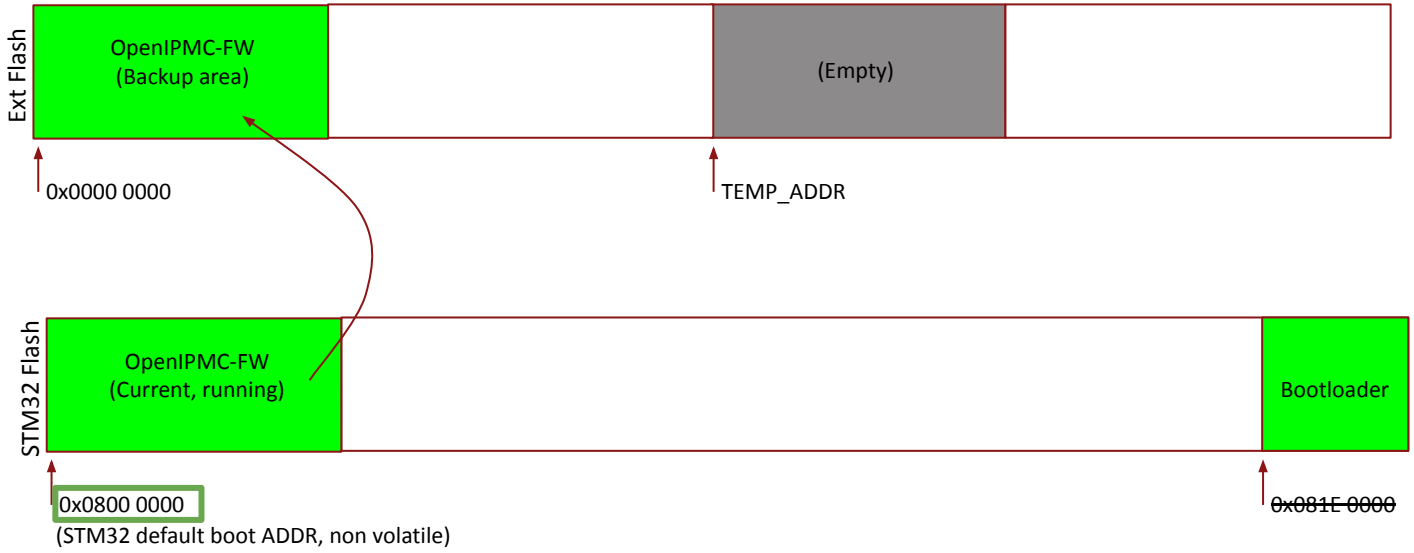
- Many ATCA crates...
  - ...with many ATCA boards...
    - … running in inaccessible sites (eg: CMS cavern)
      - Firmware need to replace itself by its own reliably!
- FW Image upload
  - Network: FTP, TFTP ...
  - IPMI: PICMG specifies dedicated commands (HPM.1)
- Backup
  - The current version need be kept in the device
  - Manual rollback: if user note some misbehaviour
  - Auto rollback: if new FW is severely fails

FW Upgrade steps

1. OpenIPMC-FW make a copy of itself into the external Flash

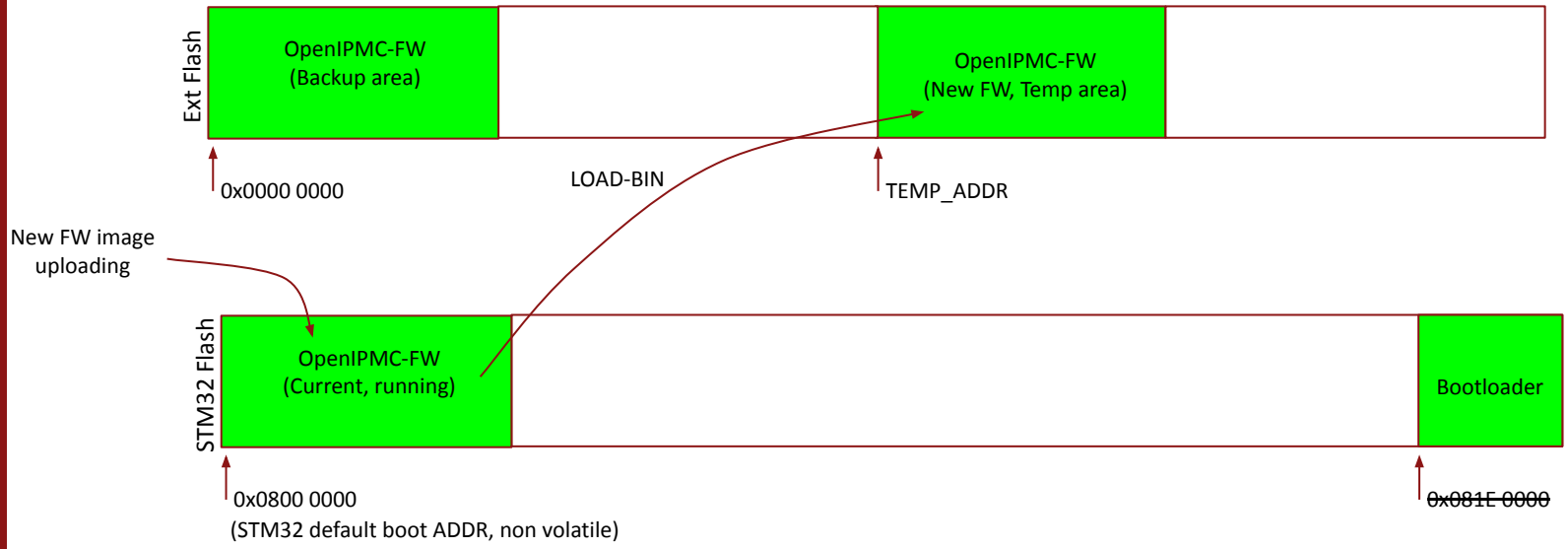
Boot address Option Bytes

Name	Value		Description
BOOT_CM7_ADD0	Value 0x800	Address 0x08000000	Define the boot address for Cortex-M7 when BOOT0=0
BOOT_CM7_ADD1	Value 0x1ff0	Address 0x1ff00000	Define the boot address for Cortex-M7 when BOOT0=1
BOOT_CM4_ADD0	Value 0x810	Address 0x08100000	Define the boot address for Cortex-M4 when BOOT0=0



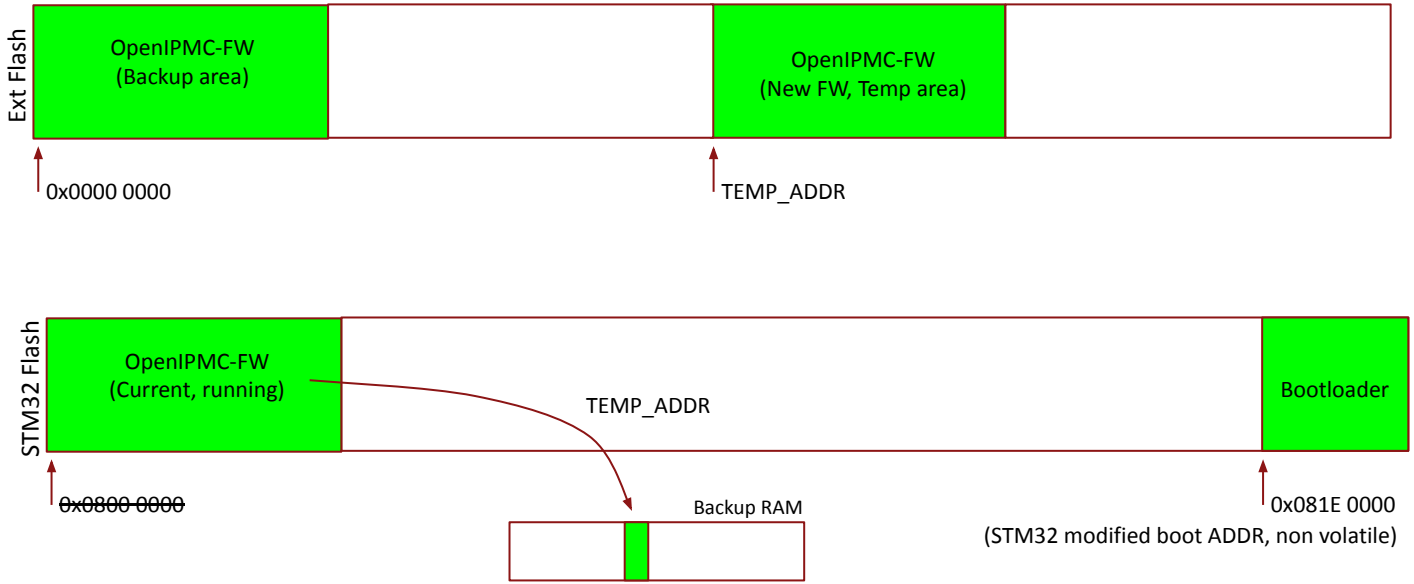
FW Upgrade steps

2. OpenIPMC-FW loads the new version into the TEMP area in External Flash (uploaded via IPMI or Network)



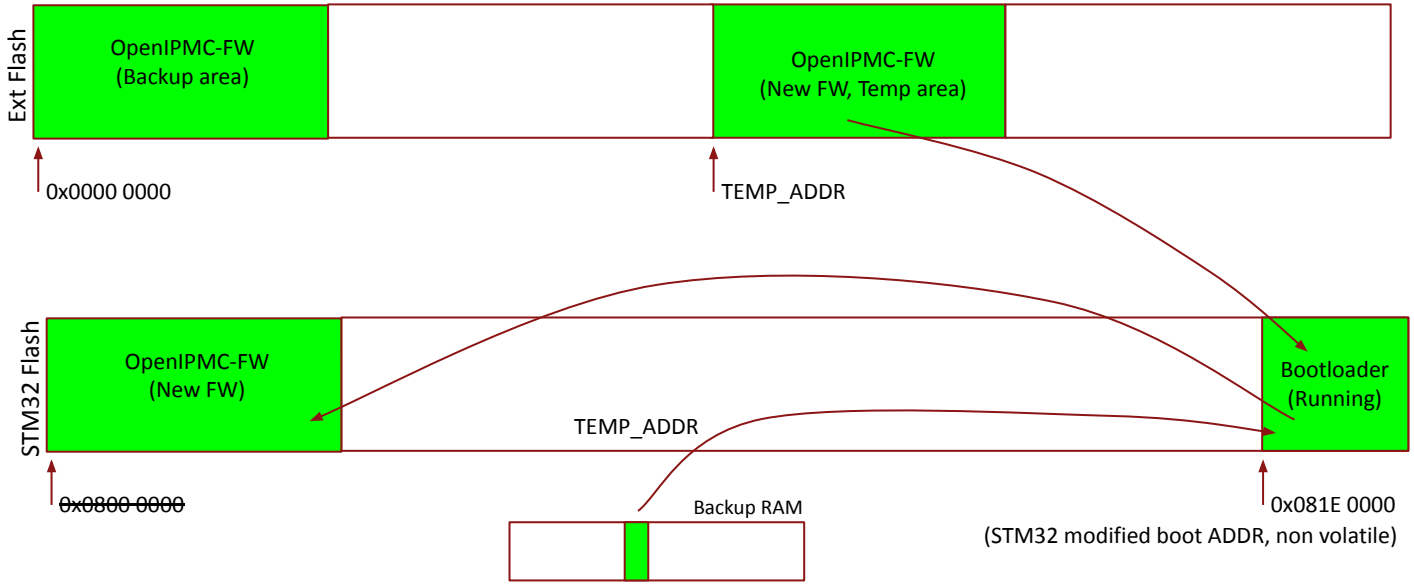
FW Upgrade steps

3. OpenIPMC-FW writes the position of the new firmware for Bootloader
4. Change boot address to Bootloader region and reboot.

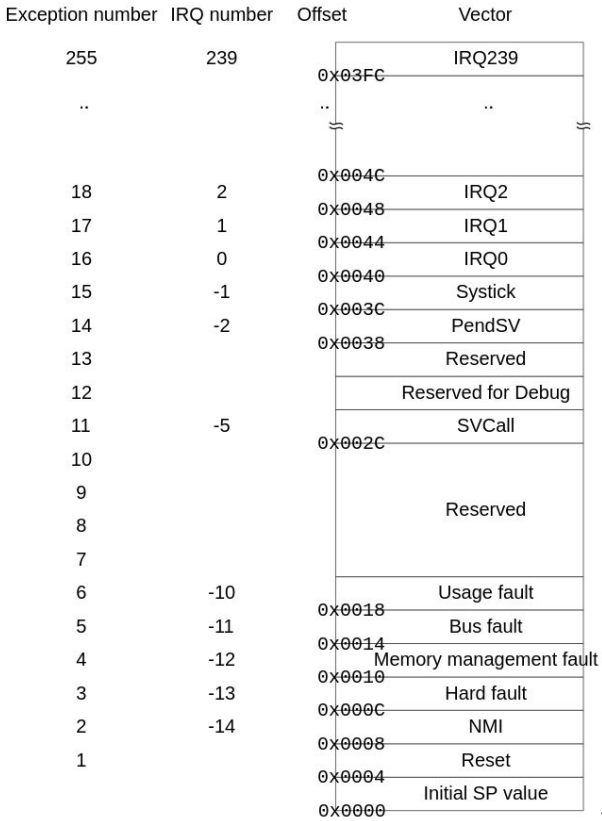


FW Upgrade steps

5. Bootloader reads the address
6. Bootloader starts and copies the new FW from TEMP to the RUN area
7. Bootloader jumps to new FW



Considerations about ARM boot



- Boot address is not the address where the program starts
  - It is actually the address of the *Vector table*
- Vector table
  - Initial stack pointer - The address where stack starts
  - Reset address - The address where the program actually starts (first instruction)
  - Addresses for all exceptions (faults and interrupts)

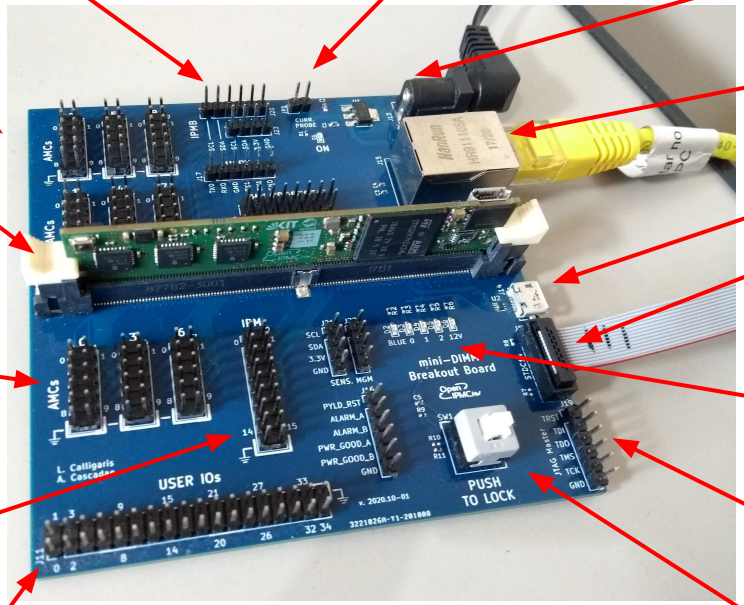
A FW can “call” another FW as it was a function

```
void (*jump_to_firmware)(void);
uint32_t BootAddr;

// Set the address of the entry point to the firmware
BootAddr = 0x08000000;

// Set up the jump address to firmware being launched
jump_to_firmware = (void (*)(void)) (*((uint32_t *) (BootAddr + 4)));

//Set the main stack pointer to the firmware being launched
__set_MSP(*(uint32_t *)BootAddr);

//Call the function to jump to firmware being launched
jump_to_firmware();
```

This process usually require all the MCU resource previously reset to its default state

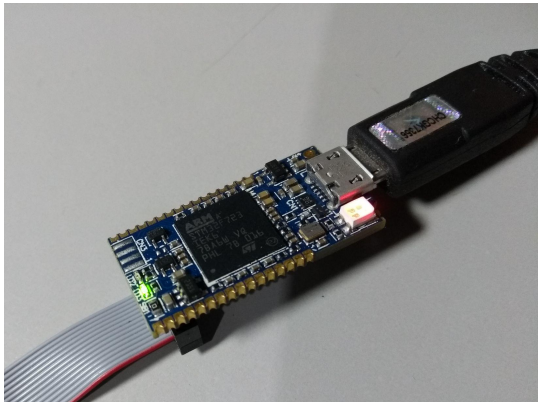
Test and dev platforms

Breakout Board used for development & programming



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



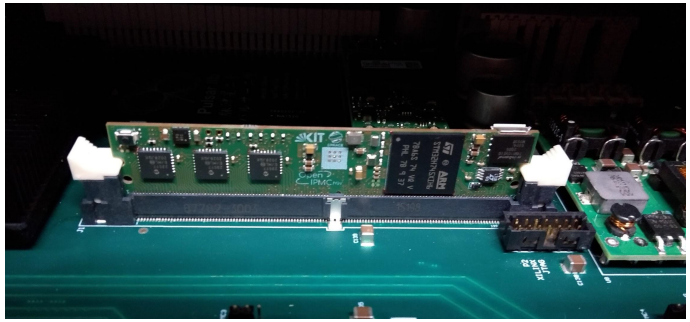
STLink-V3MINI JTAG/SWD debugger
Cost: around 10 USD

The breakout board is very inexpensive:

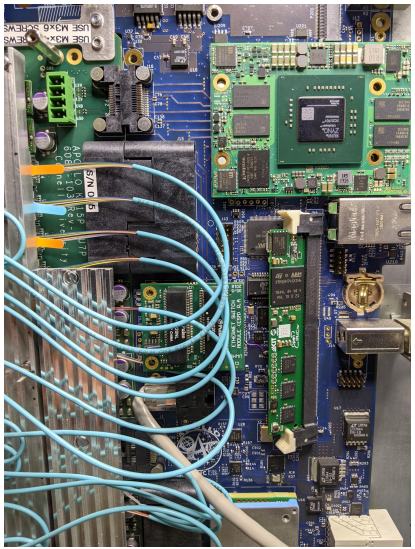
- 10 PCB + shipping = 26 USD
- Dimm connector = 17 USD
- The other components are cheaper

Test in ATCA boards

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



Pulsar-IIb @ SPRACE



Apollo @ BU

Serenity @ KIT

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
```

Current on PIM400

Voltage on -48 line (Channels A and B)

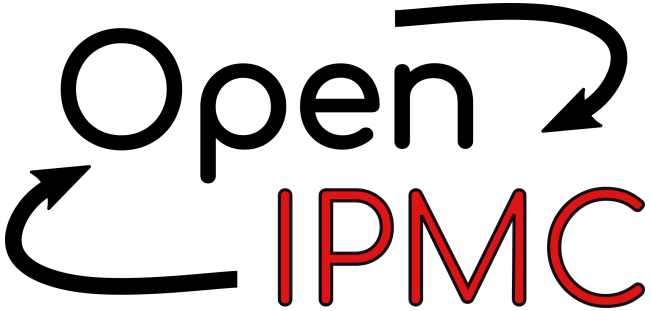
```
8c: LUN: 0, Sensor # 4 ("CURRENT PIM400")
    Type: Threshold (0x01), "Current" (0x03)
    Belongs to entity (0xa0, 0x60): FRU # 0
    Status: 0xc0
        All event messages enabled from this sensor
        Sensor scanning enabled
        Initial update completed
    Raw data: 3 (0x03)
    Processed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

#
```

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

Summary

- OpenIPMC: the importance of abstraction
  - Resolves IPMI communication and board management according to PICMG
  - standard C, FreeRTOS and no other special dependency
  - Access hardware IOs without knowing how they work

- OpenIPMC in STM32H745 chip
  - IO expander MCP23S17 via SPI (driver exemplified)
  - 128MB W74M01GVZEIG Flash memory via QSPI (driver exemplified)
  - Ethernet PHY KSZ8091MNX via MII
  - Device Firmware Update (DFU) and Virtual COM port (VCP) via USB
  - LwIP: TCP, UDP, TFTP, Telnet …
  - Command Line Interpreter via UART
  - Firmware Upgrade and Bootloader



Questions?

OpenIPMC-HW on ATCA boards

Sensor readout test at CMS TIF

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

PASSED	FAILED	SKIPPED	TOTAL
56 (56%)	17	26	99

Repositories on Gitlab

OpenIPMC (IPMC software)

- gitlab.com/openipmc/openipmc

OpenIPMC-FW (DIMM firmware)

- gitlab.com/openipmc/openipmc-fw

OpenIPMC-HW (DIMM board design)

- gitlab.com/openipmc/openipmc-hw

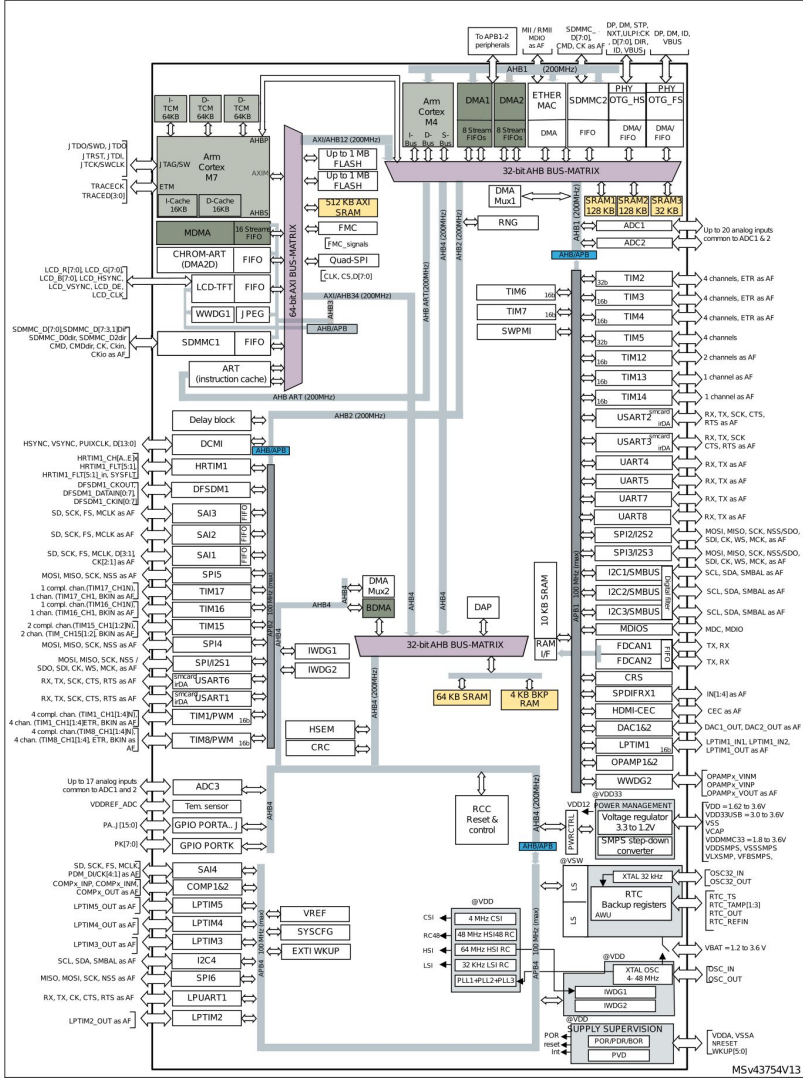
Breakout baseboard

- gitlab.com/openipmc/openipmc-hw_debug-base

Backup Slides

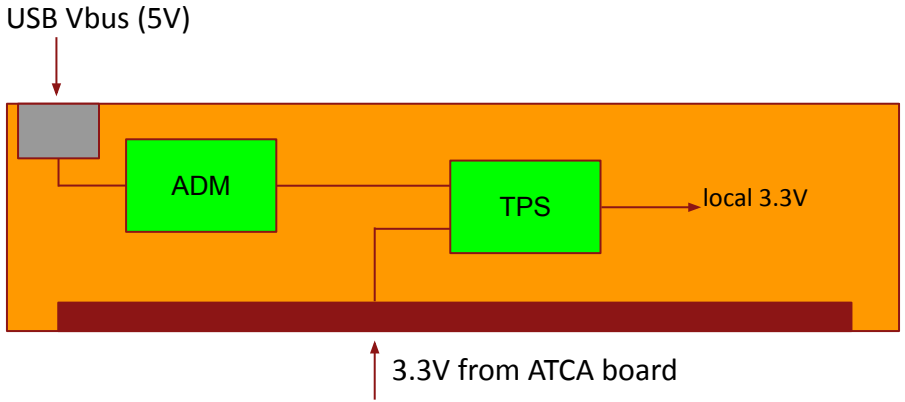
STM32H745XI peripherals

- 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



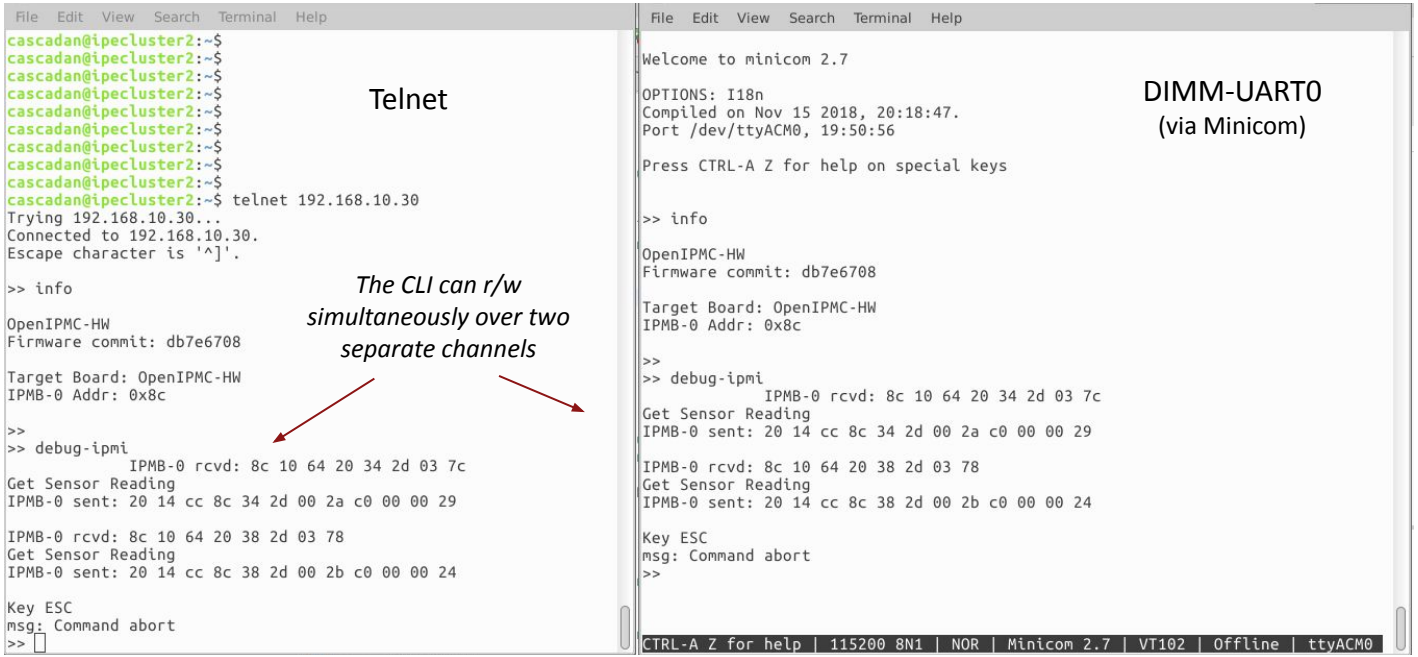
3.3V power OR-ing switch

- 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

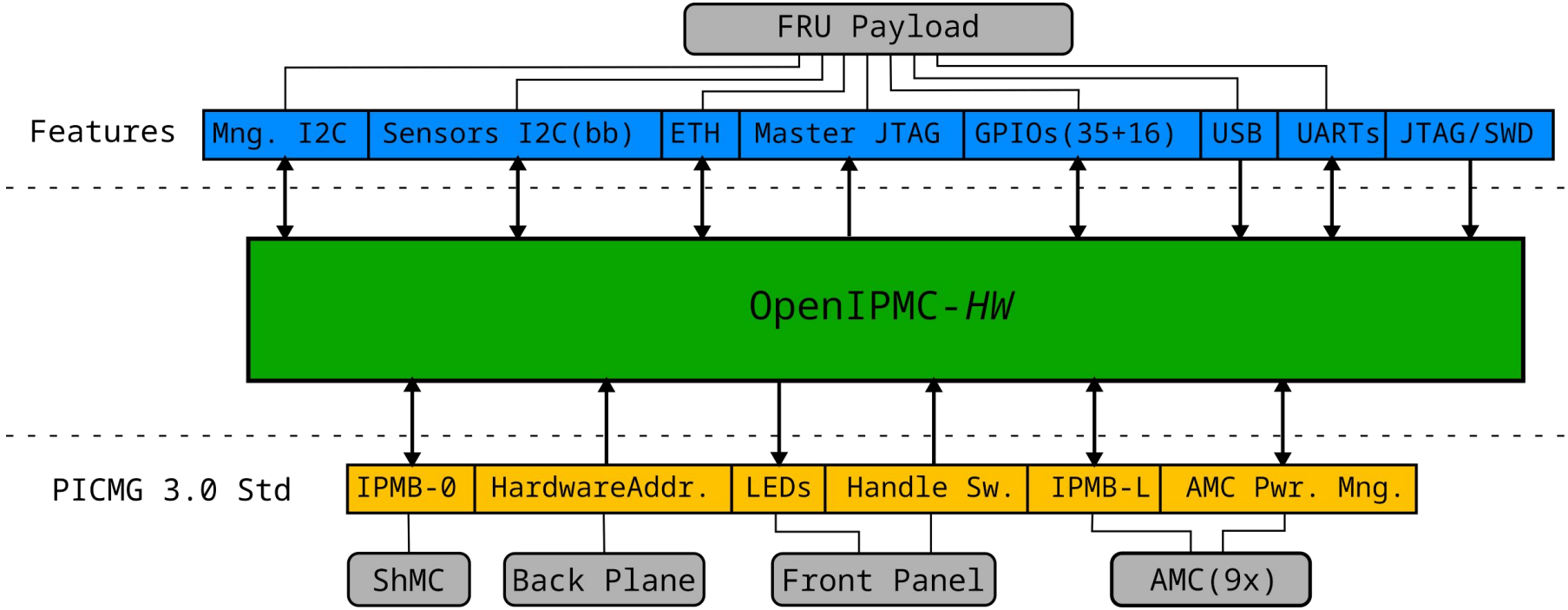


Command Line Interface: Telnet & UART

- 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



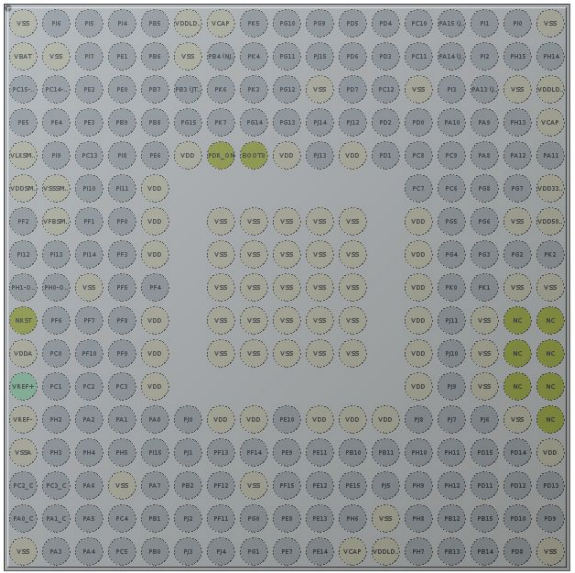
OpenIPMC-HW DIMM connections



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

STM32H745XI Microcontroller: specs

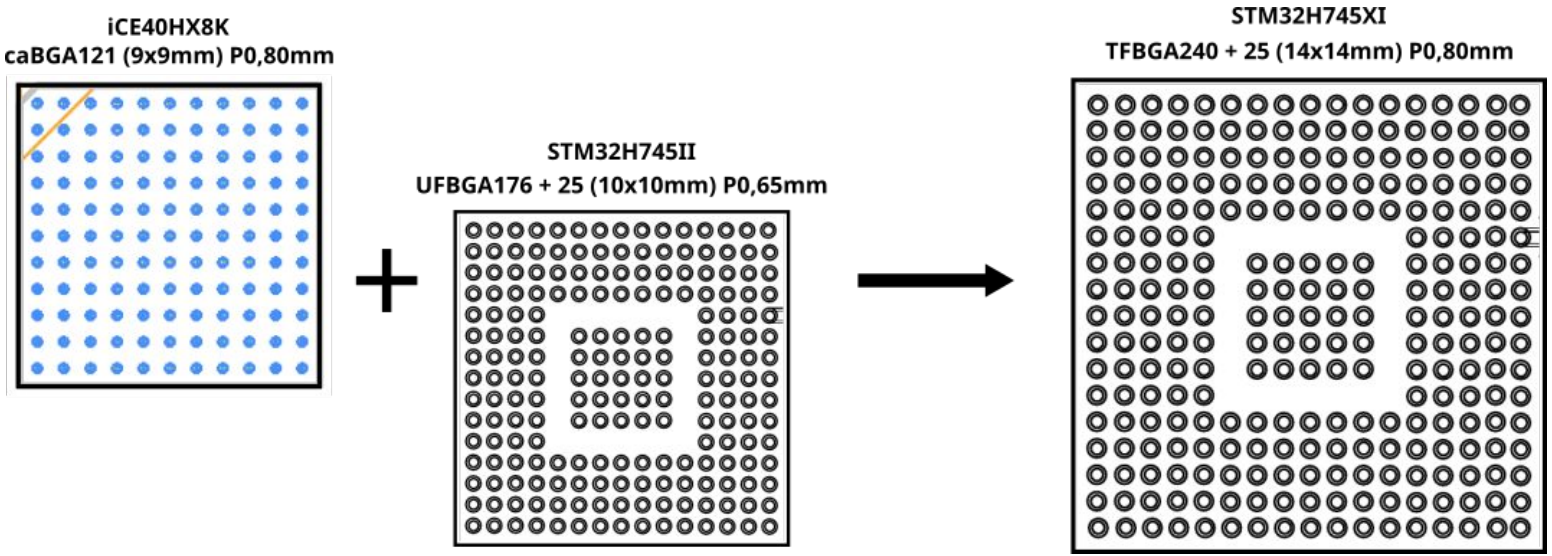
- Cores
  - 1x ARM Cortex-M7 (480 MHz max)
  - 1x ARM Cortex-M4 (240 MHz max)
- Package
  - 265-TFBGA
  - 14x14mm
  - 0.8mm pitch
- Memory
  - 2x 1 Mbyte Flash
  - 64 I + 128 D Kbytes TCM (M7 only)
  - 864 Kbytes SRAM
- Power
  - Input 1.62 to 3.6 V
  - Integrated SMPS+LDO
- IOs
  - 168x GPIOs
  - 4x I^2C
  - 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

Changes compared to previous presentations

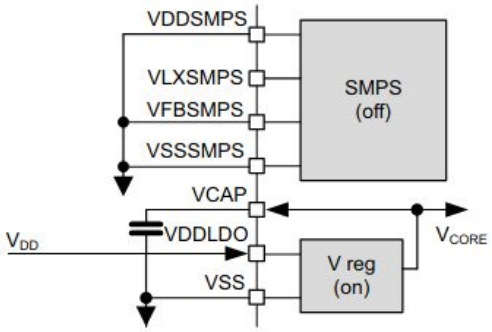
- Platform is now called **OpenIPMC-*HW***
- Switched MCU from **STM32H745*II*** (176 balls 0.65mm pitch) to **STM32H745*XI*** (240 balls 0.8mm pitch)
  - It's an elephant, but it seems to still fit on available board space, and **is easier to solder**
- Removed iCE40 FPGA from the design (as suggested by Peter Wittich)
  - Makes the hardware design and signal routing easier
  - Removes the necessity of programming a second device in the board



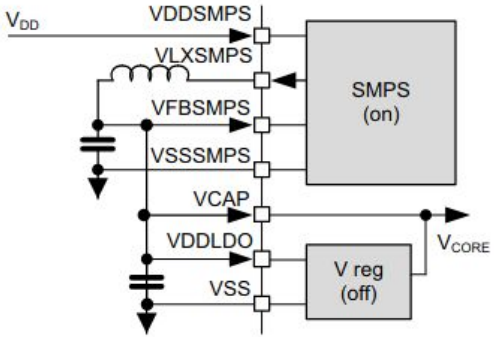
STM32H745XI Microcontroller: SMPS x LDO

- 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

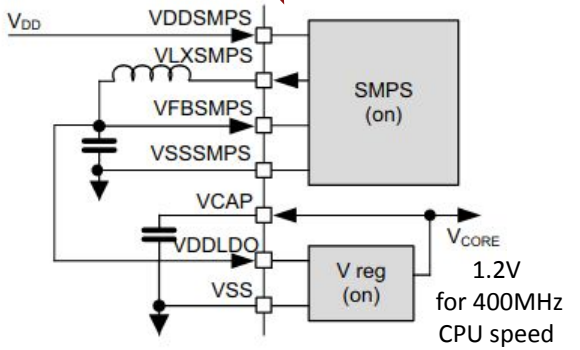
TESTED SUCCESSFULLY AND CURRENTLY IN USE



LDO only



SMPS only



SMPS supplying LDO

Polaris test @ CERN 06/05/2021 page 1/6

Req 3.1 IPMI Request and Response Format	Failed	05-06-2021 15:12:44.... Multiple Errors Occured. Last Error: Get Event Receiver failed with Completion Code 0xC1.
Req 3.2 NetFn Used by ATCA Defined Commands Request	Passed	05-06-2021 15:12:44.... Validated that AdvancedTCA defined commands use a NetFn of 2Ch in the request.
Req 3.3 NetFn Used by ATCA Defined Commands Response	Passed	05-06-2021 15:12:46.... Validated that responses to AdvancedTCA defined commands use a NetFn of 2Dh.
Req 3.4 PICMG Identifier Used by ATCA Defined Commands	Passed	05-06-2021 15:12:46.... Validated that the first byte of the Data Bytes area of both the ATCA request and response are the PICMG identifier of 00h.
Req 3.5 Reserved Bits in all Commands	Passed	05-06-2021 15:12:47.... Validated that reserved bits and fields in commands (request messages) and responses are written as 0.
Req 3.13 Invalid Completion Code for Not Applicable Commands	Passed	05-06-2021 15:12:48.... Validated that IPM Controller returns a Completion Code of Invalid Command(C1h) to the IPMI commands that are listed as not applicable (NA) in the ...
Req 3.15 Completion Code Returned by IPM Controller	Passed	05-06-2021 15:12:49.... Validated that the IPMC 0x8A returns the Completion Codes as described in the Completion Codes section of the IMPI Specification...
Req 3.17 Checking Invalid FRU Device ID in Request Data	Passed	05-06-2021 15:12:50.... Validated that all commands that take a FRU Device ID exceeding the maximum value as request Data, IPMC returns a 'parameter Out of Range(C9h)' c...
Req 3.19 Checking FRU Device ID of Absent FRU in Request Data		
Req 3.21 Board Containing Multiple Physical Slots		
Req 3.26 Event Receiver and LUN	Failed	05-06-2021 15:12:51.... Error: No support for Get Event Receiver command. (Completion Code = 0xC1)
Req 3.28 Event Generation in Response to Set Event Receiver Command	Failed	05-06-2021 15:12:52.... Error: No support for Get Event Receiver command. (Completion Code = 0xC1)
Req 3.34 Entity of the Version Change Sensor	Skipped	05-06-2021 15:12:54.... Skip cause: Carrier Manager (0x8A) does not implement Version Change Sensor
Req 3.35 Entity of the Version Change Sensor		
Req 3.46 Hardware to IPMB 0 Address Transformation	Passed	05-06-2021 15:12:54.... Validated that each IPM Controller uses a validated Hardware Address shifted left one bit as the upper seven bits of the IPMB address and assign 0 as bit...
Req 3.62 Physical Slot Number of Front Board		
Req 3.74 Managed FRU Represented by IPM Controller	Passed	05-06-2021 15:12:55.... Validated that for each IPMC, every FRU Device ID from 0 to Max FRU Device ID corresponds to a Managed FRU represented by that IPM Controller.
Req 3.82 Support of Get PICMG Properties Command for IPM Controller	Failed	05-06-2021 15:12:57.... Max FRU Device ID in the Get PICMG Properties response is 0.
Req 3.83 Get Address Info Support for IPM Controller	Passed	05-06-2021 15:12:59.... Validated that each IPM Controller supports the 'GetAddressInfo' command as defined in Table 3-13, ""Get Address Info" command (on IPM Controller...
Req 3.88 Site Type in Get Address Info Command	Passed	05-06-2021 15:12:59.... Validated that an IPM Controller returns Site Type values as defined in Table 3-9, 'Site type values' in Get Address Info response.
Req 3.101 FRU State Support	Passed	05-06-2021 15:13:00.... Validated that an Intelligent FRU supports, on behalf of itself and all the other FRUs it represents, the FRU states
Req 3.103 Set FRU Activation Policy Command	Failed	05-06-2021 15:14:06.... Error: Timeout waiting for transition of 0x8A FRU #0, M4->M5 (in range 5->12)
Req 3.138 Payload Power after Completion of Deactivation Process	Passed	05-06-2021 15:14:06.... Validated the Payload Power when the IPMC with all its Managed FRUs have transitioned to M1 state...
Req 3.157 Transition to M0 State for FRU Device ID 0		
Req 3.184 Implementaion of Cold Reset Command for IPM Controller		
Req 3.188 Implementation of Warm Reset Command for IPM Controller	Passed	05-06-2021 15:14:18.... Validated that an IPM Controller (0x8A) does not implement the Warm Reset command.
Req 3.189 Operational State of IPMC and Payload Due to Warm Reset Command	Skipped	05-06-2021 15:14:18.... Skip cause: Warm Reset command is not supported by IPMC: 0x8a (CC 0xc1).
Req 3.251 Support of Get FRU LED Properties Command for IPMC	Passed	05-06-2021 15:14:19.... Validated that an IPM Controller implements the "Get FRU LED Properties" command.
Req 3.253 Support of Get LED Color Capabilities Command for IPMC	Passed	05-06-2021 15:14:20.... Validated that an IPM Controller implements the Get LED Color Capabilities command
Req 3.254 Completion Code CCh in Response to Get LED Color Capabilities Command	Passed	05-06-2021 15:14:21.... Validated that if an LED is not present or is not under the control of the IPM Controller, the IPM Controller returns the "Invalid data field in Request (CCh...
Req 3.257 Hardware Restriction Bit in Response to Get FRU LED State Command		
Req 3.258 Invalid Completion Code in Current State Due to Dissatisfy Hardware Restriction		
Req 3.264 RTM as Managed FRU		
Req 3.266 FRU Hotswap Sensor for RTM		
Req 3.274 RTM FRU Information		
Req 3.275 Message Bridging to Intelligent Sub FRU		
Req 3.276 IPMI Message Channel 7		
Req 3.277 Get Address Info from Channel 7		
Req 3.350 Sensor Device Commands	Passed	05-06-2021 15:14:24.... Verified that IPM Controller (0x8A) implements the mandatory Sensor Device commands.
Req 3.351 Get Device SDR	Passed	05-06-2021 15:14:24.... Verified that IPM Controller (0x8A) implements the mandatory Get Device SDR and Get Device SDR Info as per IPMI1.5 Errata 310.
Req 3.353 Reserve Device SDR Repository		
Req 3.354 FRU Device Locator Record	Passed	05-06-2021 15:14:25.... Verified that IPM Controller 0x8A contains FRU Device Locator Records for each FRU that is represented by the IPM Controller except FRU Device ID 0.
Req 3.355 MC Device Locator Record	Passed	05-06-2021 15:14:26.... Verified that the IPMC 0x8A contains a Management Device Controller Record that described the IPMC and FRU Device ID 0...
REQ 3.356 Sensors of IPMC and all its Managed FRU is Described in Sensor Data Record	Passed	05-06-2021 15:14:28.... Validated that each IPMC maintains a Sensor Data Record for every sensor that it wants to report to the Shelf Manager for every sensor on every present...
Req 3.357 IPMC SDR Merging for Non Intelligent FRU		
Req 3.358 IPMC SDR Removing for Non Intelligent FRU		
Req 3.359 IPMC Implements Physical IPMB0 Sensor	Passed	05-06-2021 15:14:28.... Validated that each IPM Controller implements a physical IPMB-0 sensor.
Req 3.379 Entity ID and Entity Instance of Front Board		
Req 3.380 Entity ID Entity Instance of RTM		
Req 3.385 Intelligent FRU Entity		
Req 3.390 Containment Relationship of Entity in Device SDRs	Skipped	05-06-2021 15:14:29.... Skip cause: Device Entity Association record is not found.
Req 3.391 Top Level Containment	Skipped	05-06-2021 15:14:30.... Skip cause: Device Entity Association record is not found.
Req 3.392 Entity of Non FRU Components		
Req 3.400 IPMC as Event Generator	Failed	05-06-2021 15:14:33.... Error: IPMC 0x8A does not support Get Event Receiver Command.
Req 3.403 IPM Controller Sends All Events to Event Receiver	Failed	05-06-2021 15:14:33.... Error: No support for Get Event Receiver command for the IPM Controller 0x8a. (Completion Code = 0xC1)
Req 3.405 FRU Inventory Device Command Support for IPM Controller	Passed	05-06-2021 15:14:34.... Validated that each IPM Controller supports the FRU Inventory Device commands
Req 3.406 FRU Information Is Available without Payload Power	Passed	05-06-2021 15:14:38.... Validated that FRU Information is available even when main power is not applied to the unit's Payload function.

Polaris test @ CERN 06/05/2021 page 2/6

Req 3.407 Primary FRU at FRU Device ID 0	Failed	05-06-2021 15:14:38....	Error: Device Capabilities field of Management Controller does not match with Additional Device Support of the response of Get Device Id Command.
Req 3.408 Contiguous FRU Device ID	Passed	05-06-2021 15:14:39....	Validated that FRU Device IDs are contiguous.
Req 3.409 Entity Responsible for Updating Checksums in FRU Information	Passed	05-06-2021 15:14:41....	Validated that the entity updating the FRU Information is responsible for updating all appropriate checksums as well.
Req 3.413 Population of Predefined Fields of FRU Information			
Req 3.414 Multi Records in Multi Record Information Area	Failed	05-06-2021 15:14:41....	Multirecord Info Area is not present in Carrier (0x8A) FRU Information
Req 3.415 Presence of Multirecord of IPMC Implementing Shelf FRU Information			
Req 3.416 Implementation of Board Info Area			
Req 3.418 Implementation of Chassis Info Area	Passed	05-06-2021 15:14:42....	Validated that the IPM Controller not supporting the Shelf FRU Information populates the Chassis Info Area Starting Offset in the Common Header with...
Req 3.419 Implementation of Chassis Info Area for IPMC Implementing Shelf FRU Information			
Req 3.420 Board Point to Point Connectivity Record	Failed	05-06-2021 15:14:43....	Multirecord Area is not present in Carrier (0x8A) FRU Information
Req 3.421 Product Info Area for Non Zero FRUs	Passed	05-06-2021 15:14:45....	Validated that each IPM Controller that represents one or more FRUs with a non-zero FRU Device ID provides a Product Info Area associated with the co...
Req 3.422 Product Info Area Identify Distinct FRU Types	Passed	05-06-2021 15:14:45....	Validated that for any two or more otherwise identical FRUs with visually distinct Face Plates, the Product Info Area fields distinguish and identify the di...
Req 3.466 E-Keying Entry in Board FRU Information	Skipped	05-06-2021 15:14:48....	Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.467 Get and Set Port State	Skipped	05-06-2021 15:14:48....	Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.487 Board Channel in Board FRU Information			
Req 3.488 Separate Link Descriptor for every Protocol in a Channel			
Req 3.491 Multi Channel Links in Board FRU Information	Skipped	05-06-2021 15:14:49....	Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.492 Link Descriptor in Board FRU Information			
Req 3.493 Link Designator in Board FRU Information			
Req 3.494 Link Type Values in Board FRU Information	Skipped	05-06-2021 15:14:50....	Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.495 Link Type Extension Values in Board FRU Information	Skipped	05-06-2021 15:14:51....	Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.496 Link Type in Range F0 to FE in Board FRU Information	Skipped	05-06-2021 15:14:52....	Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.497 Combined OEM GUID Table in Board Point to Point Connectivity Record	Skipped	05-06-2021 15:14:53....	Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.498 OEM GUID in Board Point to Point Connectivity Record	Skipped	05-06-2021 15:14:54....	Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.503 Data Format of Set Port State Command	Skipped	05-06-2021 15:14:55....	Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.504 Data Format of Get Port State Command	Skipped	05-06-2021 15:14:56....	Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.603 IPMB 0 Sensors for IPMC	Passed	05-06-2021 15:14:57....	Validated the physical IPMB-0 sensors are implemented by each IPM Controller and used to monitor the state of the IPMBs.
Req 3.604 Get Sensor Reading for Physical IPMB 0 Sensor	Passed	05-06-2021 15:14:58....	Validated the data format of "Get Sensor Reading" command (physical IPMB-0 sensor)
Req 3.607 Get IPMB Link Info Command for IPMC Not Connecting to IPMB-0 on a Radial Basis	Failed	05-06-2021 15:14:59....	Get IPMB Link Info command returns 0xC1 instead of 0x00 as Completion Code.
Req 3.608 C9h Completion Code for Get IPMB Link Info	Failed	05-06-2021 15:15:00....	Get IPMB Link Info command returns 0xC1 instead of 0xC9 as Completion Code.
Req 3.609 Physical IPMB 0 Status Change Event Message	Failed	05-06-2021 15:15:32....	Error: Timeout waiting for IPMB-0 Status Change event (in range 39->39)
Req 3.610 Set IPMB State Command Support	Passed	05-06-2021 15:15:32....	Validated the data format for the Set IPMB State command.
Req 3.611 IPMC Refuses Set IPMB State Causing for Isolation	Passed	05-06-2021 15:15:34....	Validated that an IPM Controller refuses any "Set IPMB State" command that would cause it to become isolated from both buses.
Req 3.612 IPMC Operation With Both Buses			
Req 3.613 Invalid Command C1h Completion Code for Set IPMB State Command	Passed	05-06-2021 15:15:34....	Validated that an IPMC that does not connect to IPMB-0 on a radial basis and receives a radial-topology-specific "Set IPMB State" request returns an "In...
Req 3.614 Set IPMB State Command in a Radial Topology	Skipped	05-06-2021 15:15:35....	Skip cause: No IPMB0 Link Mapping Record found
Req 3.627 Status of the FRU Local IPMB Segment			
Req 3.668 Power Command Support	Passed	05-06-2021 15:15:49....	Validated that IPMC supports the Power Commands for every FRU that it supports.
Req 3.669 Compute Power Properties Command Support			
Req 3.674 Power Draws through Get Power Level Command	Passed	05-06-2021 15:15:49....	Validated the Minimum and Maximum Power Draws through Get Power Level command.
Req 3.678 Power Level Value while FRU is in Steady State Power Draw Level	Passed	05-06-2021 15:15:53....	Validated Power Level Values in Steady State Power Draw Levels.
Req 3.679 Power Level in early Power Draw Levels	Failed	05-06-2021 15:16:03....	Error: Power Drawn is not same as previously granted via the "Set Power Level" command (index: 0)
Req 3.680 Compute Power Properties Locks Different Power Levels	Passed	05-06-2021 15:16:27....	Validated that the IPMC locks "Desired steady state power draw levels" and "Desired early levels" after it receives "Compute Power Properties" comman...
Req 3.681 Compute Power Properties Locks Power Draws	Passed	05-06-2021 15:16:50....	Validated that the IPMC 0x8A locks the Power Draw Array Size and the Values after it receives "Compute Power Properties" command.
Req 3.682 Power Draw Adjustment Due to Set Power Level Command	Passed	05-06-2021 15:17:27....	Validated the Power Draw Adjustment Due to the Set Power Level command.
Req 3.683 Power Draw Levels for Different Power Types	Passed	05-06-2021 15:17:28....	Validated that the Power Draws is not greater for the "Desired steady state draw levels" than the Power Draws for the "Desired early levels" values.
Req 3.684 FRU Power up by Set Power Level Command	Passed	05-06-2021 15:17:51....	Validated that the IPMC 0x8A does not Power up any of its FRUs untill it receives a Set Power Level command with a power level greater than 0 from the...
Req 3.685 Power Management of IPMC	Passed	05-06-2021 15:17:57....	Validated the Power Management of IPMC via Get and Set Power Level commands.
Req 3.736 Temperature Sensor	Passed	05-06-2021 15:17:57....	Validated that all Boards and other Intelligent FRUs support at least one temperature sensor and its corresponding sensor record (SDR).
Req 3.739 Readability of Temperature Sensor	Passed	05-06-2021 15:17:58....	Validated that even if represented by a "virtual sensor", each temperature sensor is individually readable.
Req 3.741 Power Supply Sensor	Passed	05-06-2021 15:17:59....	Validated that all Boards and other Intelligent FRUs support at least one power supply sensor monitoring the status of the power Feed fuses.
Req 3.742 Sensor Data Record for Power Supply Sensor	Passed	05-06-2021 15:18:00....	Validated that the Sensor Data Record shall be provided for each power supply sensor.
Req 3.744 Temperature Event Message	Passed	05-06-2021 15:18:11....	Validated that Temperature Event message have the data format defined in Table 3-92, "Temperature event message."
Req 3.745 Assert Deassert Temperature Events	Passed	05-06-2021 15:18:21....	Validated that IPM Controllers indicate when they have reached a minor temperature alert condition by asserting a "07h = Upper Non-critical (minor) - ...
Req 3.746 Assert Deassert Temperature Events Critical	Skipped	05-06-2021 15:18:21....	Skip cause: All the temperature sensors are skipped
Req 3.747 Assert Deassert Temperature Events Upper Non Recoverable	Skipped	05-06-2021 15:18:22....	Skip cause: All the temperature sensors are skipped
Req 3.748 Temperature Sensors Threshold Commands	Failed	05-06-2021 15:18:23....	Error: Get Sensor Threshold failed for sensor 3. (Completion Code = 0xC1)

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 15:18:35.... Validated that all temperature sensors provide default hysteresis values.
Req 3.752 Critical Temperature Alert Condition	Skipped	05-06-2021 15:18:38.... Skip cause: No Fan Geography record found in the Shelf FRU Information.
Req 3.763 Telco Alarm Commands	Skipped	05-06-2021 15:18:38.... Skip cause: IPMC 0x8A does not implement any FRU with Telco Alarm functionality.
Req 3.764 Invalid Data Field in Request for Telco Alarm Commands	Skipped	05-06-2021 15:18:39.... Skip cause: IPMC 0x8A does not implement any FRU with Telco Alarm functionality.
Req 3.766 Telco Alarm State	Skipped	05-06-2021 15:18:41.... Skip cause: IPMC 0x8A does not implement any FRU with Telco Alarm functionality.
Req 3.767 Telco Alarm Input Sensor	Skipped	05-06-2021 15:18:41.... Skip cause: IPMC 0x8A does not implement any FRU with Telco Alarm functionality.
Req 3.768 Telco Alarm Event Message	Skipped	05-06-2021 15:18:42.... Skip cause: IPMC 0x8A does not implement any FRU with Telco Alarm functionality.
Req 3.769 Telco Alarm Minor Reset Input		
Req 3.770 Telco Alarm Major Reset Input		
Req 3.771 Telco Alarm Cutoff Function		
Req 3.772 Shelf Manager and IPMC Functionality	Passed	05-06-2021 15:18:43.... Validated that Shelf Manager and IPM Controller functions are supported as defined in Table 3-99.
Req 3.773 Command Number Assignment	Passed	05-06-2021 15:18:44.... Validated that IPMI Command number functions and requirements are supported as defined in Table 3-100.
Req 3.776 Sensor Specific Event Reading Type	Failed	05-06-2021 15:18:46.... Error: IPMC 0x8A does support Event Receiver command
Req 3.777 Entity ID Assignment	Passed	05-06-2021 15:18:46.... Validated that PICMG® Entity ID assignments shall be supported as defined in Table 3-104, "PICMG Entity ID assignments."
SDR Requirements	Failed	05-06-2021 15:18:48.... Get Device SDR Info reports wrong number of sensors for LUN 0 (8 instead of 6)
Board E-Keying	Skipped	05-06-2021 15:18:51.... Skip cause: No Board Point-to-Point Connectivity Record found
Watchdog Timer Commands Support	Skipped	05-06-2021 15:18:52.... Skip cause: No support for Watchdog Timer commands

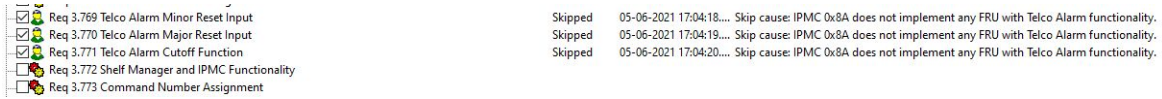
Polaris test @ CERN 06/05/2021 page 4/6

Req 3.1 IPMI Request and Response Format		
Req 3.2 NetFn Used by ATCA Defined Commands Request		
Req 3.3 NetFn Used by ATCA Defined Commands Response		
Req 3.4 PICMG Identifier Used by ATCA Defined Commands		
Req 3.5 Reserved Bits in all Commands		
Req 3.13 Invalid Completion Code for Not Applicable Commands		
Req 3.15 Completion Code Returned by IPM Controller		
Req 3.17 Checking Invalid FRU Device ID in Request Data		
Req 3.19 Checking FRU Device ID of Absent FRU in Request Data		
Req 3.21 Board Containing Multiple Physical Slots	Skipped	05-06-2021 16:40:56.... Skip cause: IPMC 0x8A does not span multiple slots.
Req 3.26 Event Receiver and LUN		
Req 3.28 Event Generation in Response to Set Event Receiver Command		
Req 3.34 Entity of the Version Change Sensor		
Req 3.35 Entity of the Version Change Sensor	Skipped	05-06-2021 16:41:07.... Skip cause: Carrier Manager (0x8A) does not implement any Version Change Sensor.
Req 3.46 Hardware to IPMB 0 Address Transformation		
Req 3.62 Physical Slot Number of Front Board	Passed	05-06-2021 16:41:19.... Validated that site number (7th byte of the response) is equal to Physical Slot Number of the front board.
Req 3.74 Managed FRU Represented by IPM Controller		
Req 3.82 Support of Get PICMG Properties Command for IPM Controller		
Req 3.83 Get Address Info Support for IPM Controller		
Req 3.88 Site Type in Get Address Info Command		
Req 3.101 FRU State Support		
Req 3.103 Set FRU Activation Policy Command		
Req 3.138 Payload Power after Completion of Deactivation Process		
Req 3.157 Transition to M0 State for FRU Device ID 0	Passed	05-06-2021 16:42:58.... Validated he transition to M0 State for the FRU #0 and all its managed FRUs.
Req 3.184 Implementaion of Cold Reset Command for IPM Controller	Passed	05-06-2021 16:43:20.... Validated that an IPM Controller implements the Cold Reset command.
Req 3.188 Implementation of Warm Reset Command for IPM Controller		
Req 3.189 Operational State of IPMC and Payload Due to Warm Reset Command		
Req 3.251 Support of Get FRU LED Properties Command for IPMC		
Req 3.253 Support of Get LED Color Capabilities Command for IPMC		
Req 3.254 Completion Code CCh in Response to Get LED Color Capabilities Command		
Req 3.257 Hardware Restriction Bit in Response to Get FRU LED State Command	Passed	05-06-2021 16:44:07.... Validated that "Hardware restriction" bit (bit 3) is set in the LED States byte of the response to the Get FRU LED State command.
Req 3.258 Invalid Completion Code in Current State Due to Dissatisfy Hardware Restriction	Passed	05-06-2021 16:44:15.... Validated that While the hardware restriction is not met, the IPMC responds with return code D5h.
Req 3.264 RTM as Managed FRU	Skipped	05-06-2021 16:44:43.... Skip cause: The IPMC (0x8A) does not implement RTM.
Req 3.266 FRU Hotswap Sensor for RTM	Skipped	05-06-2021 16:44:52.... Skip cause: The IPMC (0x8A) does not implement RTM.
Req 3.274 RTM FRU Information	Skipped	05-06-2021 16:44:58.... Skip cause: The IPMC (0x8A) does not implement RTM.
Req 3.275 Message Bridging to Intelligent Sub FRU	Skipped	05-06-2021 16:45:19.... Skip cause: The IPMC (0x8A) does not implement ant managed FRU.
Req 3.276 IPMI Message Channel 7	Skipped	05-06-2021 16:46:26.... Skip cause: The IPMC (0x8A) does not implement ant managed FRU.
Req 3.277 Get Address Info from Channel 7	Skipped	05-06-2021 16:46:09.... Skip cause: The IPMC (0x8A) does not implement any intelligent FRU.
Req 3.350 Sensor Device Commands		
Req 3.351 Get Device SDR		
Req 3.353 Reserve Device SDR Repository	Passed	05-06-2021 16:47:39.... Verified that an IPM Controller, having SDR with length more than 24, implements Reserve Device SDR Repository command...
Req 3.354 FRU Device Locator Record		
Req 3.355 MC Device Locator Record		
REQ 3.356 Sensors of IPMC and all its Managed FRU is Described in Sensor Data Record		
Req 3.357 IPMC SDR Merging for Non Intelligent FRU	Skipped	05-06-2021 16:48:15.... Skip cause: No non-intelligent FRU is available for the IPMC 0x8A.
Req 3.358 IPMC SDR Removing for Non Intelligent FRU	Skipped	05-06-2021 16:48:53.... Skip cause: No non-intelligent FRU is available for the IPMC 0x8A.
Req 3.359 IPMC Implements Physical IPMB0 Sensor		
Req 3.379 Entity ID and Entity Instance of Front Board	Skipped	05-06-2021 16:49:55.... Skip cause: Board containing the IPMC 0x8A is not a Multi Slot Board.
Req 3.380 Entity ID Entity Instance of RTM	Skipped	05-06-2021 16:50:53.... Skip cause: RTM is not implemented
Req 3.385 Intelligent FRU Entity	Passed	05-06-2021 16:51:37.... Validated that Entity in Management Controller Device Locator record for an IPM Controller is the Intelligent FRU's entity.
Req 3.390 Containment Relationship of Entity in Device SDRs		
Req 3.391 Top Level Containment		
Req 3.392 Entity of Non FRU Components	Skipped	05-06-2021 16:52:15.... Skip cause: Device Entity Association record is not found
Req 3.400 IPMC as Event Generator		
Req 3.403 IPM Controller Sends All Events to Event Receiver		
Req 3.405 FRU Inventory Device Command Support for IPM Controller		
Req 3.406 FRU Information Is Available without Payload Power		

Polaris test @ CERN 06/05/2021 page 5/6

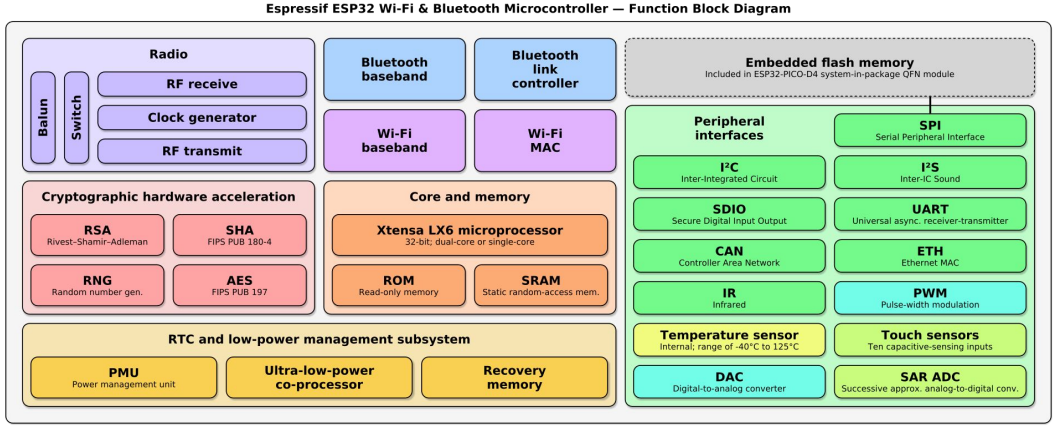
Req 3.413 Population of Predefined Fields of FRU Information	Passed	05-06-2021 16:55:06.... Validated that each IPM Controller populates all the predefined fields of the Board Info Area and Product Info Area associated with FRU Device ID 0, LUN.
Req 3.414 Multi Records in Multi Record Information Area		
Req 3.415 Presence of Multirecord of IPMC Implementing Shelf FRU Information	Failed	05-06-2021 16:58:59.... Error: Error while retrieving FRU #1 Information at 0x8A: Cannot Get FRU Inventory Area Info (FRU #1) cc=0xC9
Req 3.416 Implementation of Board Info Area	Passed	05-06-2021 16:56:55.... Validated that IPM Controllers implements valid data in all the predefined fields of the Board Info Area.
Req 3.418 Implementation of Chassis Info Area		
Req 3.419 Implementation of Chassis Info Area for IPMC Implementing Shelf FRU Information	Passed	05-06-2021 16:57:29.... Validated that each IPMC that is representing the Shelf FRU Information shall populate the Chassis Info Area Starting Offset with a valid offset to the Ch...
Req 3.420 Board Point to Point Connectivity Record		
Req 3.421 Product Info Area for Non Zero FRUs		
Req 3.422 Product Info Area Identify Distinct FRU Types		
Req 3.466 E-Keying Entry in Board FRU Information		
Req 3.467 Get and Set Port State		
Req 3.487 Board Channel in Board FRU Information	Skipped	05-06-2021 16:59:43.... Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.488 Separate Link Descriptor for every Protocol in a Channel	Skipped	05-06-2021 16:59:54.... Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.491 Multi Channel Links in Board FRU Information		
Req 3.492 Link Descriptor in Board FRU Information	Skipped	05-06-2021 17:00:04.... Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.493 Link Designator in Board FRU Information	Skipped	05-06-2021 17:00:14.... Skip cause: No Board Point To Point Connectivity Record (PICMG Record ID: 0x14) is found in Shelf FRU Information
Req 3.494 Link Type Values in Board FRU Information		
Req 3.495 Link Type Extension Values in Board FRU Information		
Req 3.496 Link Type in Range F0 to FE in Board FRU Information		
Req 3.497 Combined OEM GUID Table in Board Point to Point Connectivity Record		
Req 3.498 OEM GUID in Board Point to Point Connectivity Record		
Req 3.503 Data Format of Set Port State Command		
Req 3.504 Data Format of Get Port State Command		
Req 3.603 IPMB 0 Sensors for IPMC		
Req 3.604 Get Sensor Reading for Physical IPMB 0 Sensor		
Req 3.607 Get IPMB Link Info Command for IPMC Not Connecting to IPMB-0 on a Radial Basis		
Req 3.608 C9h Completion Code for Get IPMB Link Info		
Req 3.609 Physical IPMB 0 Status Change Event Message		
Req 3.610 Set IPMB State Command Support		
Req 3.611 IPMC Refuses Set IPMB State Causing for Isolation		
Req 3.612 IPMC Operation With Both Buses	Passed	05-06-2021 17:01:02.... Validated that each IPM Controller begins operation with both buses in Local Control state.
Req 3.613 Invalid Command C1h Completion Code for Set IPMB State Command		
Req 3.614 Set IPMB State Command in a Radial Topology		
Req 3.627 Status of the FRU Local IPMB Segment	Passed	05-06-2021 17:01:33.... Validated that reading the IPMB-0 sensor, returns the status of the FRU's local IPMB segment
Req 3.668 Power Command Support		
Req 3.669 Compute Power Properties Command Support	Passed	05-06-2021 17:01:50.... Validated the support for "Compute Power Properties command" for single slot boards, non-Board FRU and multi slot boards.
Req 3.674 Power Draws through Get Power Level Command		
Req 3.678 Power Level Value while FRU is in Steady State Power Draw Level		
Req 3.679 Power Level in early Power Draw Levels		

Polaris test @ CERN 06/05/2021 page 6/6



OpenIPMC on ESP32 (Espressif Systems, CN)

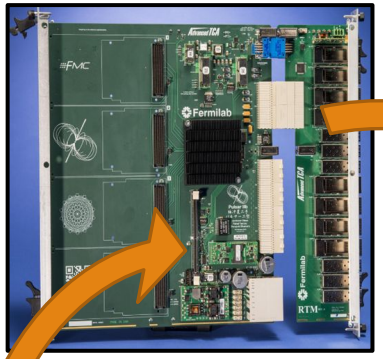
- Quite powerful & flexible uC
  - 240 MHz Xtensa LX6 dual core
  - FPU, Big INTs & Crypto
  - WiFi, BT, SPI, I2C, UART…
  - **FreeRTOS support**
- Cheap Linux-supported boards
  - CP2102 USBtoUART converter
  - Boards sell for 5$
- 3.3 V device (same as IPMB)
- Development software
  - Arduino IDE, PlatformIO or **esp-idf**
- Very different arch w.r.t a Zynq US+
  - **Good exercise on portability/**



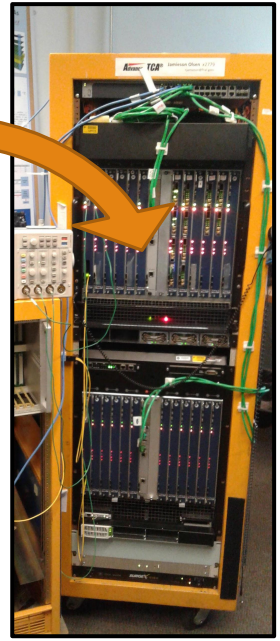
IPMC firmware & board for Pulsar-2b

- 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

Pulsar2b



Track Finder Demonstrator
ATCA crate @ Fermilab

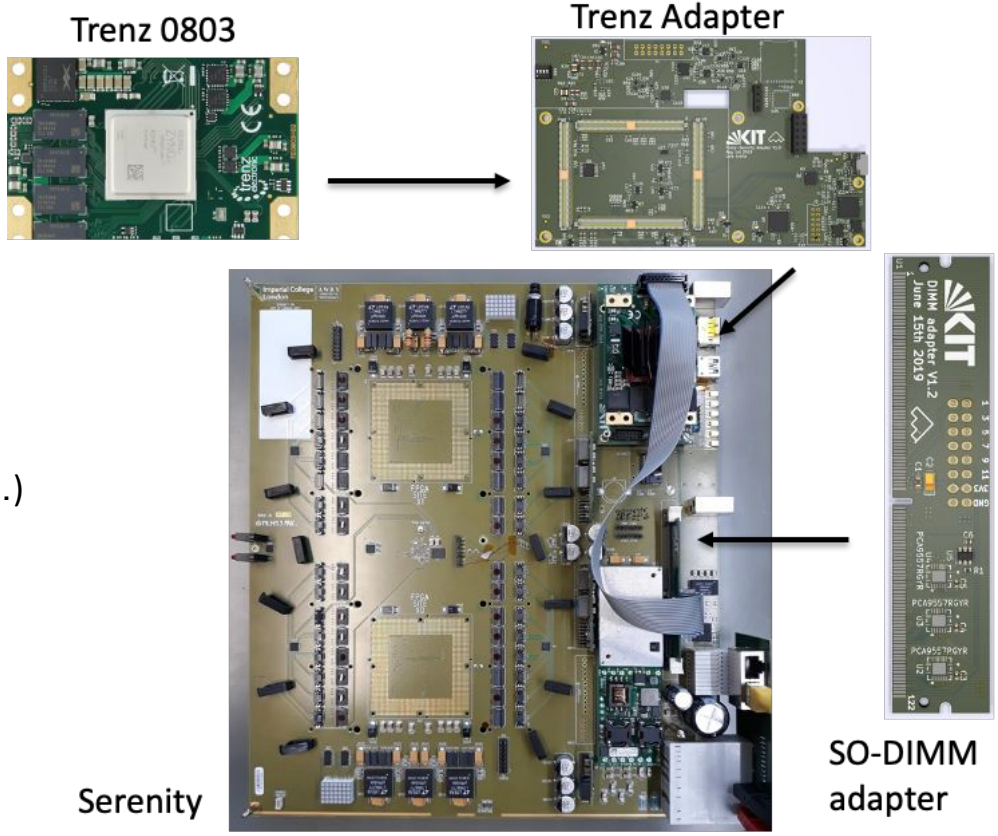


SPRACE/Fermilab IPMC

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

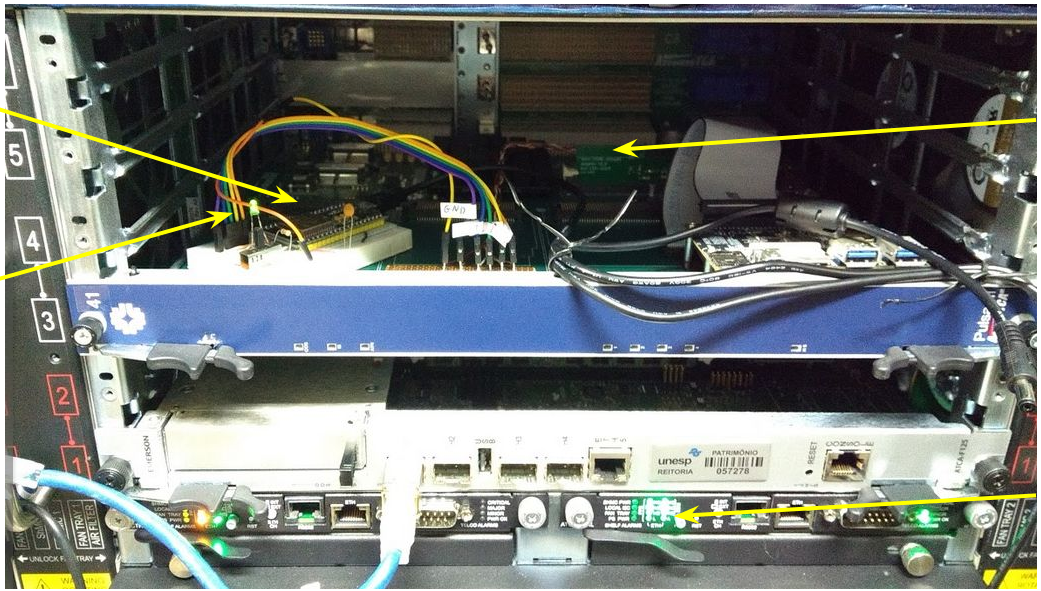


Fixes to ESP32 Integrated Development Framework

- 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

ESP32 w/ OpenIPMC

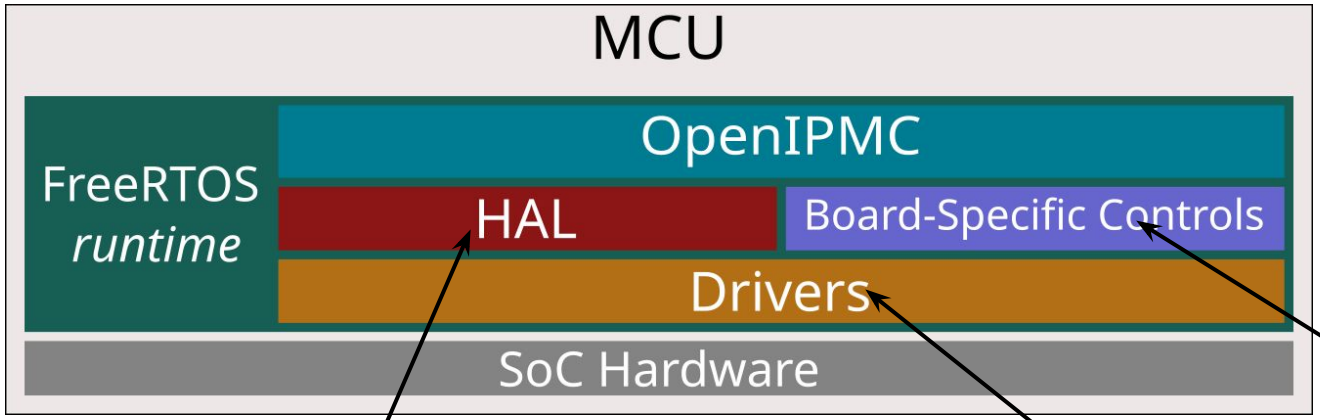
'Blue LED'



DIMM adapter giving access to the backplane

Shelf Manager

*The github esp-idf repository was forked. A merge request is still in progress



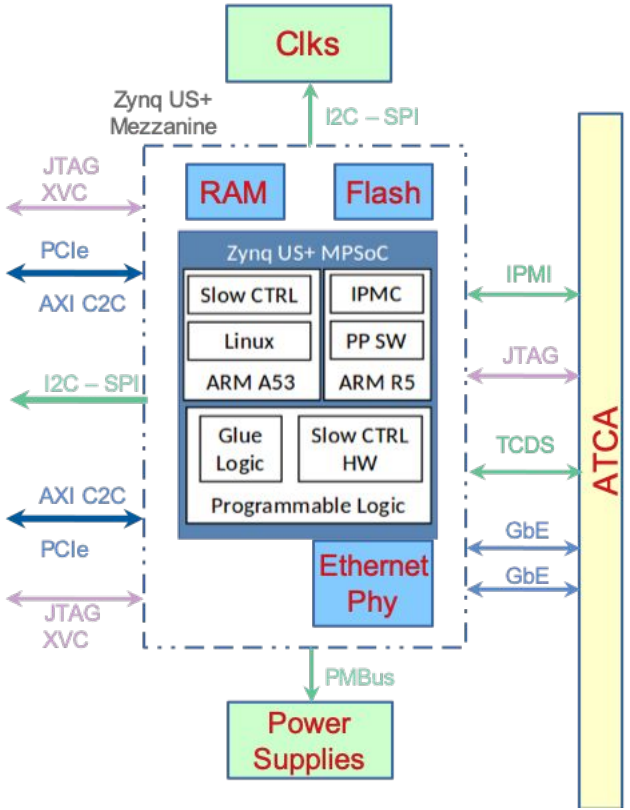
This layer allows OpenIPMC to run on the platform, and is (ideally) the only part that needs to be adapted

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

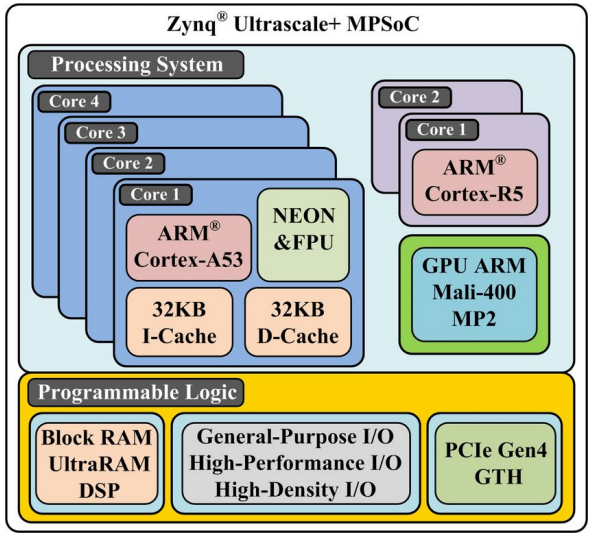
Xilinx ZynqMP SoC as unified mgmt module

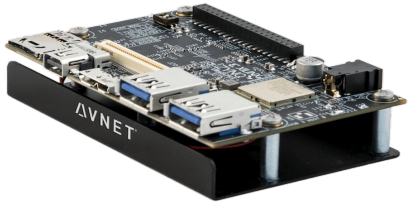
- **Needs of ATCA boards for LHC experiments**
  - IPMC → board management & monitoring
  - Linux → higher-level functions (e.g. calibration)
  - Xilinx ZynqMP SoCs can satisfy both roles using one unit
- **Zynq Ultrascale+ MPSoC**
  - Two processor domains: Application PU and Real-time PU
  - Xilinx 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



Development platform for ZynqMP: Ultra96

- We began OpenIPMC on AVNET Ultra96
  - 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



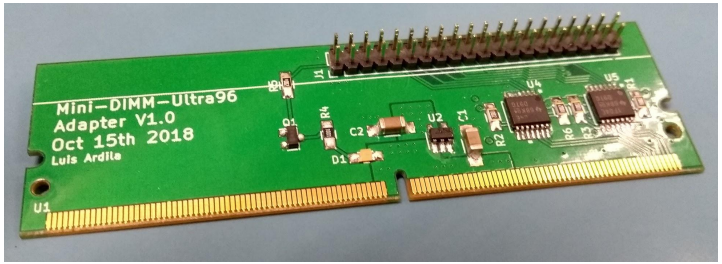


AVNET Ultra96 (Xilinx Zynq UltraScale+ ZU3EG)

Using generic development boards in the ATCA shelf

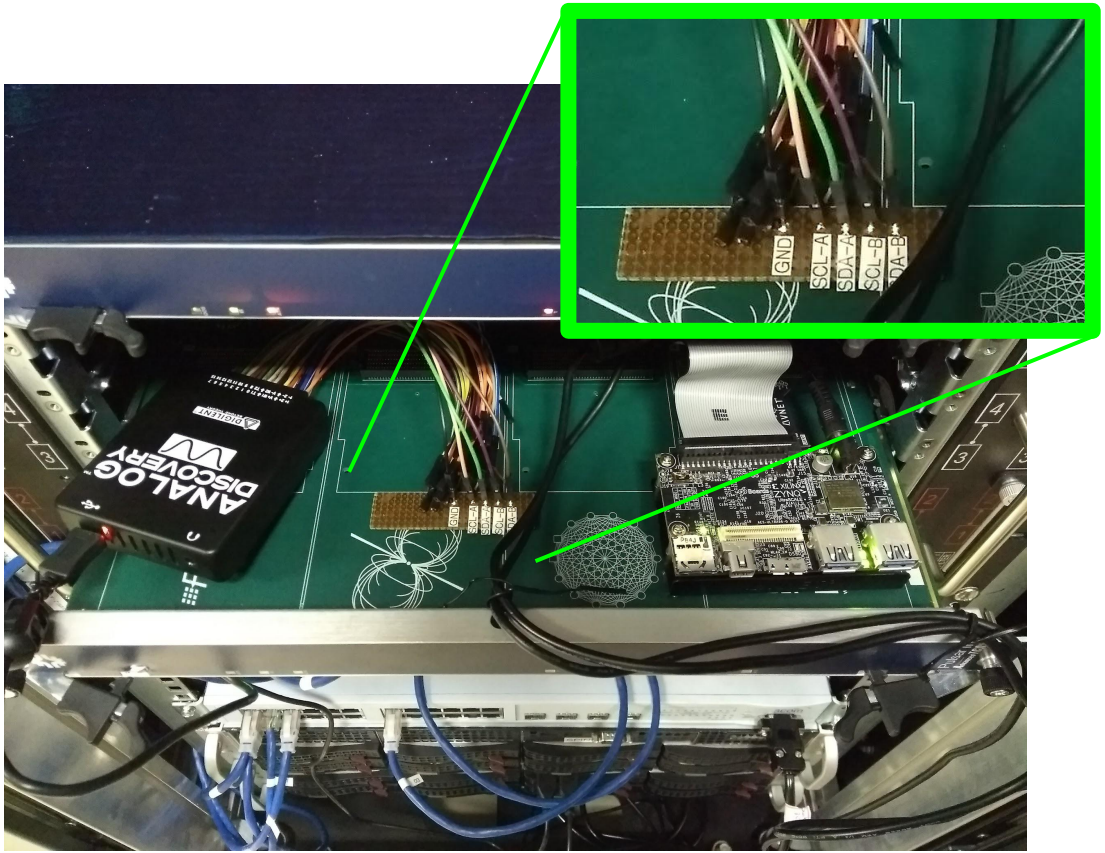
- Dev board (e.g. Ultra96)
- Pulsar-2b board exposes signals to DIMM slot
  - IPMB-A and -B buses
  - Pulsar-2b LEDs
  - Pulsar-2b main power enable (not used so far)
  - Pulsar-2b local I2C for sensors (not used so far)
- Mini-DIMM adapter
  - Connects Pulsar-2b DIMM slot to Ultra96
  - Translates 1.8 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 mated to the Pulsar-II through Mini-DIMM adapter



Mini-DIMM adapter

Ultra96 + Pulsar-2b in the shelf



Digilent Analog Discovery USB o'scope as I2C logic analyzer

Monitoring IPMB-A with TeK o'scope

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

Trenz-Serenity setup at KIT

OpenIPMC tests on Trenz + Serenity setup @ KIT

Activation Status

```
# clia fru -v 96

Pigeon Point Shelf Manager Command Line Interpreter

96: FRU # 0
    Entity: (0xb0, 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)

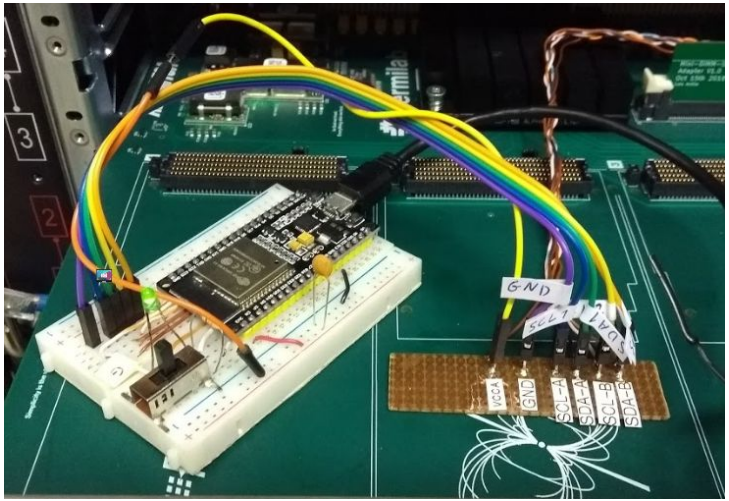
```
# clia fruinfo 96 0

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 1996)
    Board Manufacturer      = SPRACE - KIT
    Board Product Name      = OpenIPMC @ Trenz-Serenity
    Board Serial Number     = 189189981-18998
    Board Part Number       = AA00Y99
    FRU Programmer File ID  = 01
```

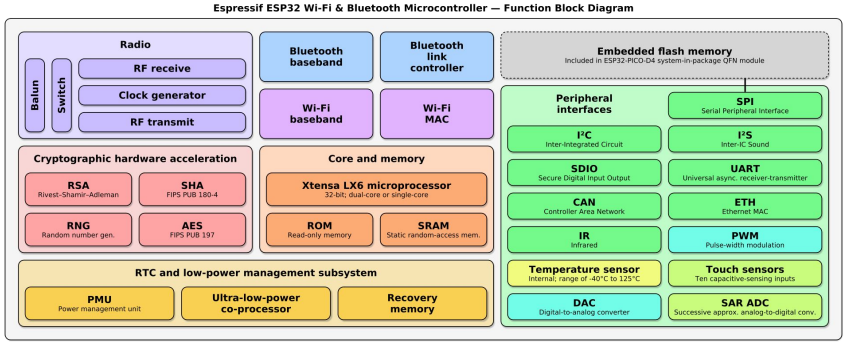
Sensor Reading (testing data from example code)

```
# 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: 50.000000 degrees C
    Current State Mask: 0x00
```

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

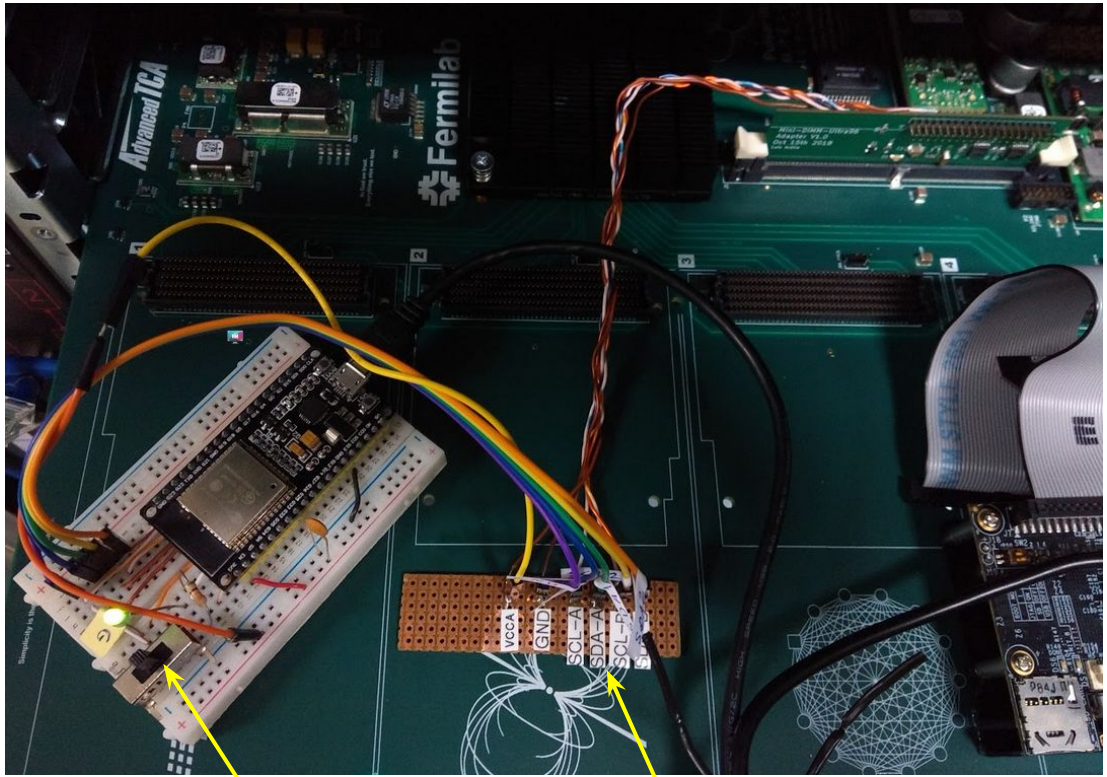




OpenIPMC tests on ESP32

- 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

```
86: LUN: 0, Sensor # 4 ("AIR TEMP")
    Type: Threshold (0x01), "Temperature" (0x01)
    Belongs to entity (0xb0, 0x60)
    Status: 0xc0
        All event messages enabled from this sensor
        Sensor scanning enabled
        Initial update completed
    Raw data: 32 (0x20)
    Processed data: 32.000000 degrees C
    Current State Mask: 0x00

86: LUN: 0, Sensor # 5 ("VCC1V0 VOUT")
    Type: Threshold (0x01), "Voltage" (0x02)
    Belongs to entity (0xb0, 0x60)
    Status: 0xc0
        All event messages enabled from this sensor
        Sensor scanning enabled
        Initial update completed
    Raw data: 22 (0x16)
    Processed data: 0.345400 Volts
    Current State Mask: 0x00

# clia fru 86

Pigeon Point Shelf Manager Command Line Interpreter

86: FRU # 0
    Entity: (0x0, 0x0)
    Hot Swap State: M1 (Inactive), Previous: M6 (Deactivation In Progress)
    Device ID String: " "
```



‘Handle Switch’ for tests

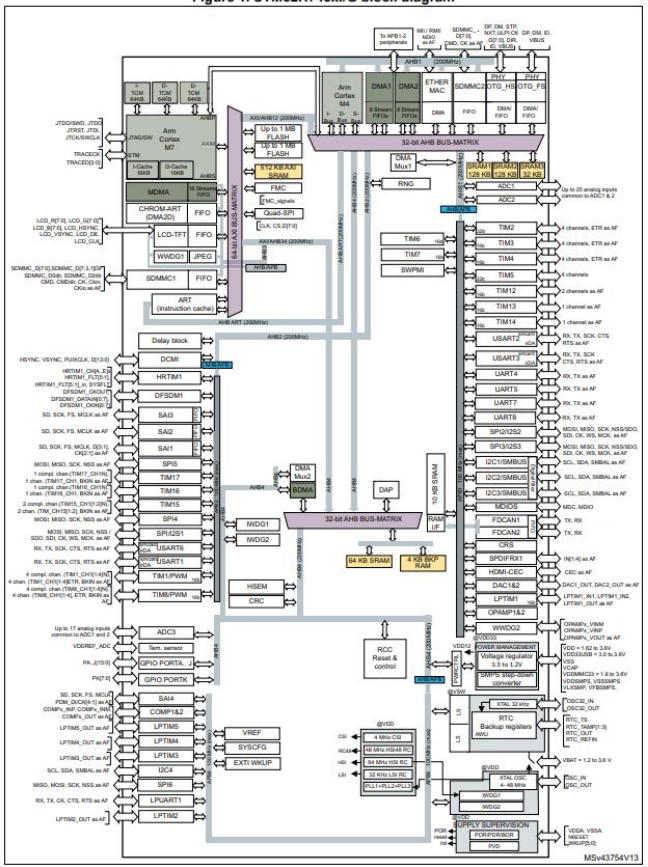
I2C lines connected to DIMM

4) STM32 H745 + Pulsar-2B

ST Microelectronics STM32H745

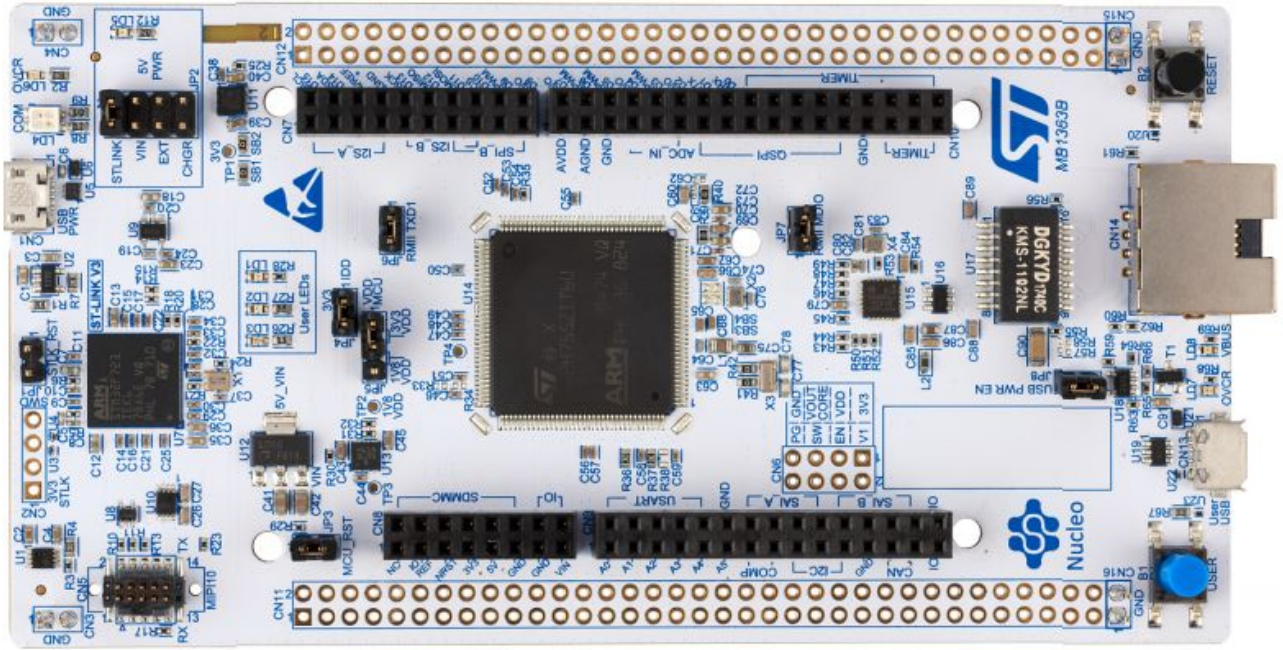
- Powerful industrial control-oriented MCU
  - 480 MHz ARM Cortex-M7 main CPU
  - 240 MHz ARM Cortex-M4 aux CPU
- Plenty of peripherals
  - 4 x hardware I2C, 6 x hardware SPI
  - 4 USART + 4 UART + 1 LPUART
  - Up to 168 GPIO
- Moderate current consumption
  - 600 mA absolute max / 80-200 mA typ current
- Free development environment & compiler
  - STM32CubeIDE (gcc in the back-end)
  - Compatible with Linux, OpenOCD and GDB
  - **ST provides a FreeRTOS distribution for STM32**

Figure 1. STM32H745xI/G block diagram



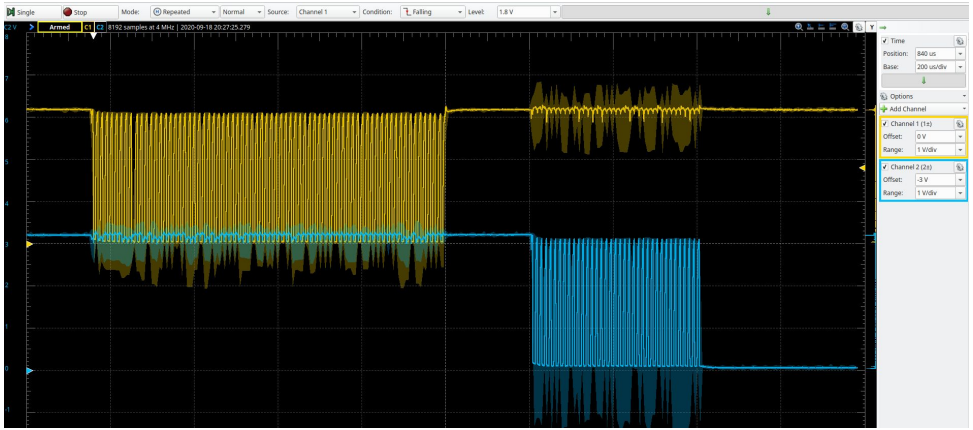
ST Microelectronics NUCLEO

- 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)

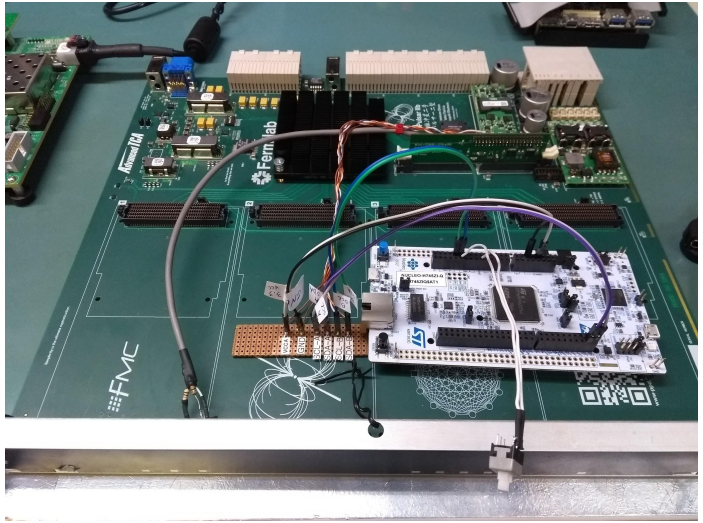


Porting OpenIPMC to STM32

- 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



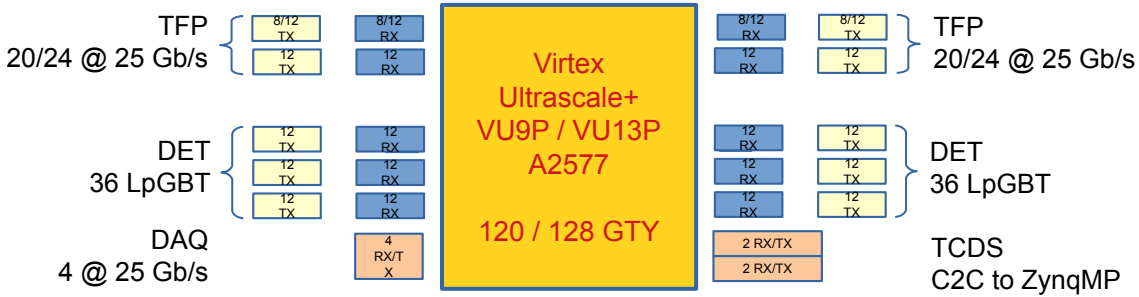
STM32 responding to shelf manager with 200 µs latency (timeout **300 ms**)



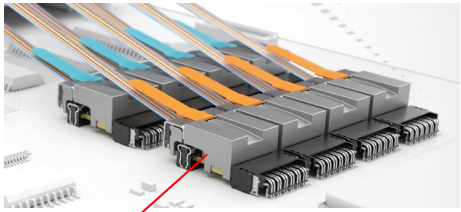
NUCLEO board mounted onto the Pulsar-2b

5) Serenity-A2577 + ZynqMP Mezzanine

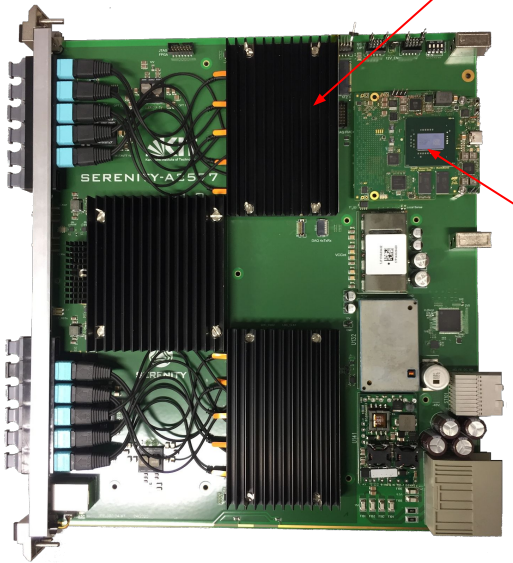
OpenIPMC Ported to the ZynqMP R5 Cores



SAMTEC Firefly Optics



- 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



Board Management Mezzanine