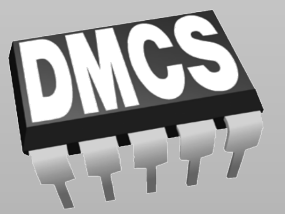


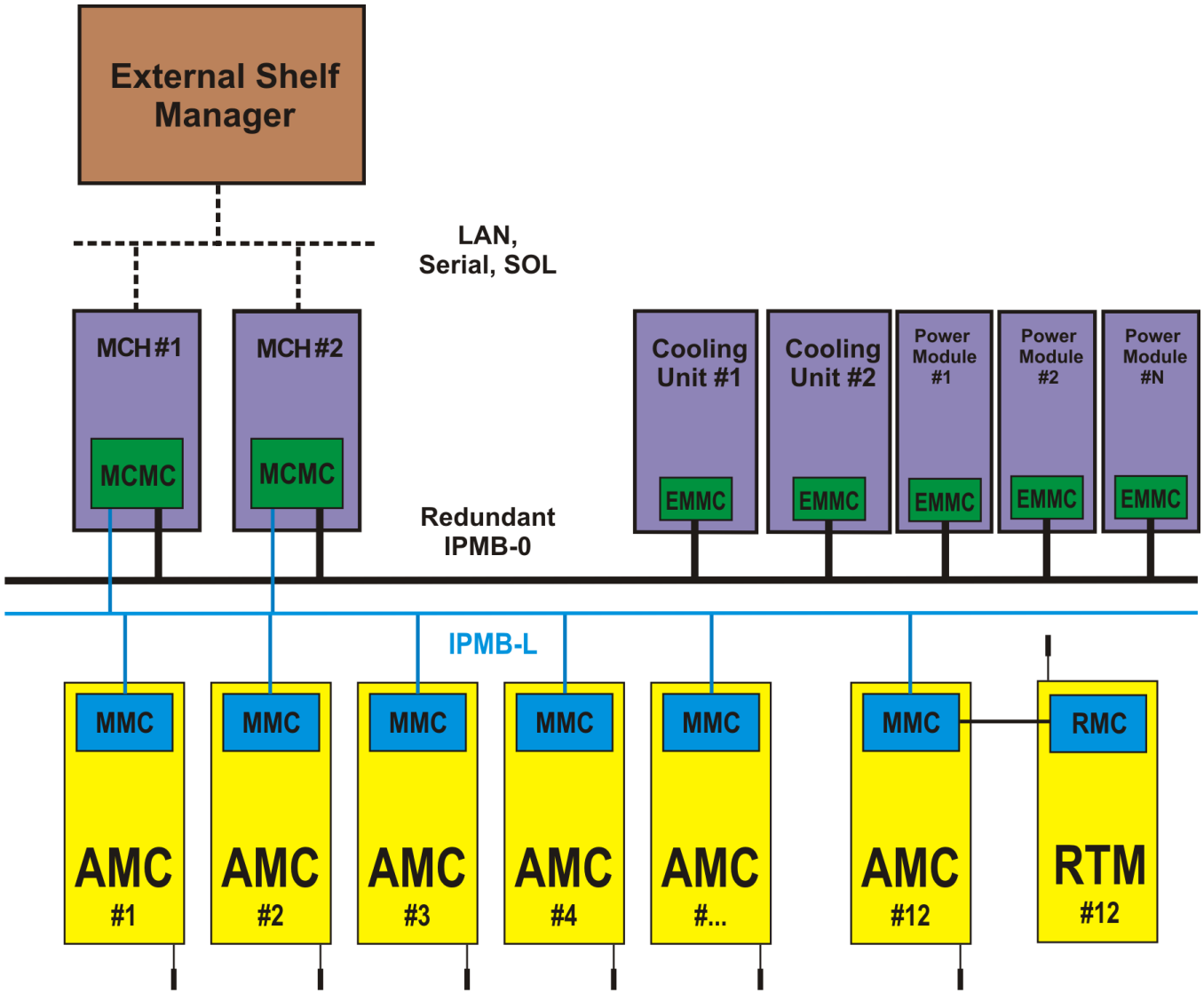
WU3 - IPMI EPICS Integration

Piotr Perek

Department of Microelectronics and Computer Science
Lodz University of Technology, Poland

Lund, 22 March 2019

MTCA Management

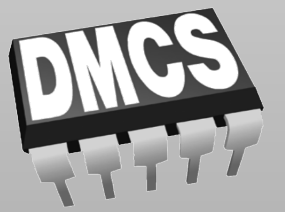


Department of Microelectronics
and Computer Science

IPMI

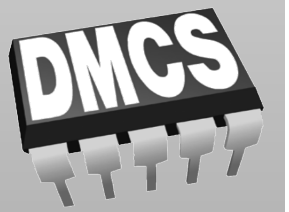
Intelligent Platform Management Interface (IPMI)

- standardized interface and protocol for an autonomous computer subsystem that provides management and monitoring capabilities independently of the host system's CPU, firmware (BIOS or UEFI) and operating system,
- published in 1998 and initially dedicated for out-of-band management of computer systems and monitoring of their operation,
- currently supported by more than 200 computer system vendors, such as Cisco, Dell, Hewlett-Packard, Intel, NEC Corporation, etc.,
- adapted by PICMG to be used as a management bus in the ATCA, AMC and MicroTCA standards.



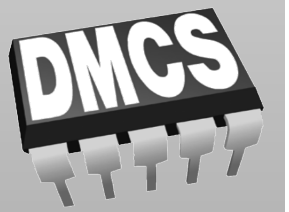
WU3 Scope

- Interface to read/write and monitor the MTCA system using EPICS
- Develop this into a scalable module that can be used for any number of MTCA modules
- Design this module in accordance to the ESS guidelines and to other designers in order to keep some level of homogeneity



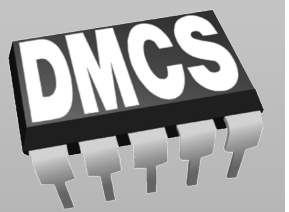
Functionality

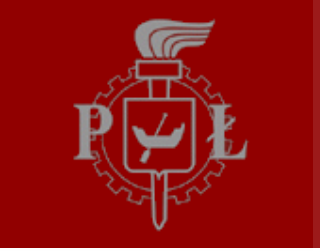
- Getting field replaceable unit (MCH, PM, CU, AMCs, backplane) info and device ID (could also be used for inventory purposes)
- Getting activation (hot-swap) states
- Getting sensor data record, sensor reading (including reading factors and hot swap handle) and provide human readable values
- Getting sensor thresholds
- Getting link capability and link status
- Setting activation (hot-swap) states (shutdown/remove or activate) FRU
- Setting sensor thresholds
- Receiving events (to be defined)
- Firmware remote update



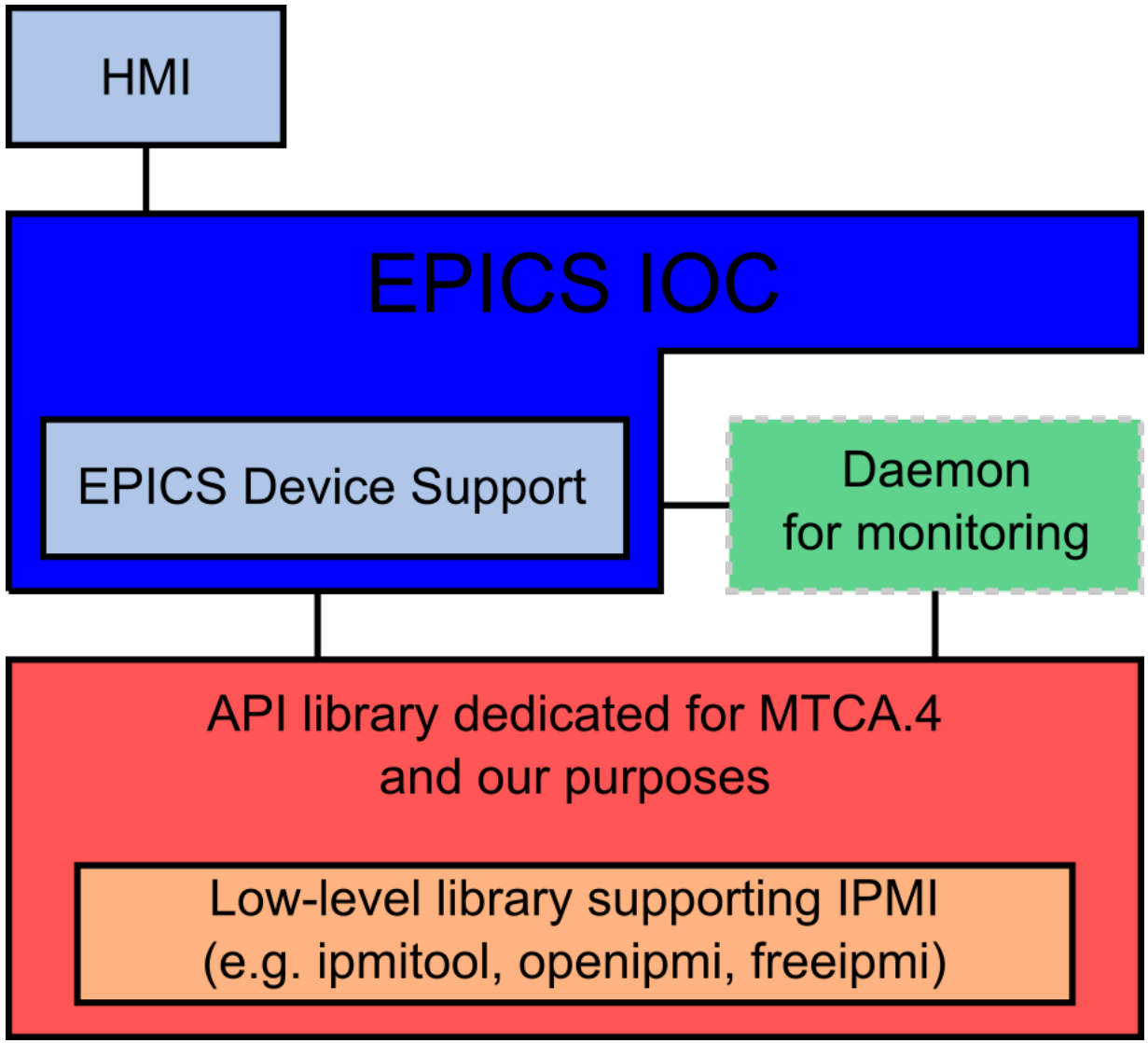
Assumptions

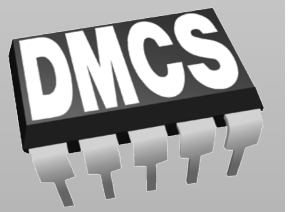
- We are providing the monitoring/control system for single MTCA chassis
  (1 chassis — 1 EPICS IOC)
- Complete EPICS monitoring system for the machine is out of scope of this contract
- Scalability — running separate EPICS IOC for every MTCA chassis in the accelerator
- Software should be fully compliant with NAT MCH — it will be used for development and tests
- Supported MicroTCA chassis types: 12 AMC (9U) and 6 AMC (3U)
- Supported modules: CCT AM 900/412, MRF EVR-300U, IFC14xx, SIS8300-KU, RTM Carrier AMC





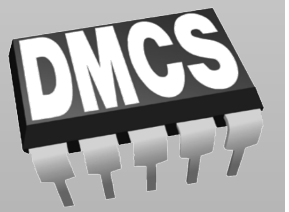
Initial Idea





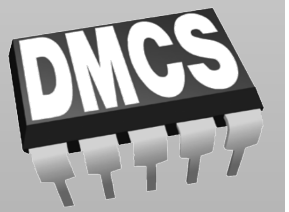
Available Solutions

- Low-level API:
  - ipmitool
    https://github.com/ipmitool/ipmitool
  - ipmiutil
    http://ipmiutil.sourceforge.net
    https://git.code.sf.net/p/ipmiutil/code-git
  - freeipmi
    https://www.gnu.org/software/freeipmi
    https://git.savannah.gnu.org/git/freeipmi.git
  - OpenIPMI
    http://openipmi.sourceforge.net
    https://git.code.sf.net/p/openipmi/code
  - OpenHPI
    http://openhpi.org
    https://github.com/open-hpi
- EPICS:
  - ipmiComm → e3-ipmiComm
    https://github.com/icshwi/e3-ipmiComm



Review of existing solutions

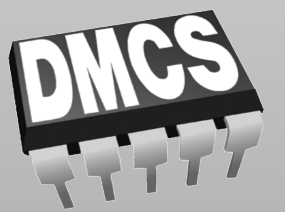
	ipmitool	ipmiutil	freeipmi	OpenIPMI	OpenHPI
License	BSD	BSD	GPL	GPL/LGPL	BSD
Last release	10/2016	09/2018	05/2018	04/2018	03/2018
Last commit	10/2018	09/2018	05/2018	10/2018	10/2018
Dynamic library	no	yes	yes	yes	yes



Review of existing solutions

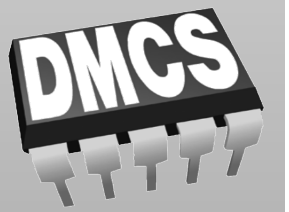
	ipmitool	ipmiutil*	freeipmi	OpenIPMI*	OpenHPI
FRU Listing	Yes	Yes	Yes	Yes	Yes
FRU Info	Yes	Yes	Yes	Yes	Yes
FRU M-State	Yes**	Yes	Yes	Yes	Yes
FRU Start/Stop	??	??	??	??	??
SDR reading	Yes	Yes***	Yes	Yes	Yes
Sensor reading	Yes	Yes***	Yes	Yes	Yes
Thr. reading	Yes	Yes***	Yes	Yes	Yes
Thr. writing	??	??	??	??	??
E-keying	??	??	??	??	??
Events	Yes	Yes	??	??	Yes
HPM.1	Yes****	Yes****	No****	No****	Yes****

* Bridging of IPMI messages seems to be not supported
** Significant overhead
*** All the information returned by single command - some filtering required
**** Information from documentation — not tested yet



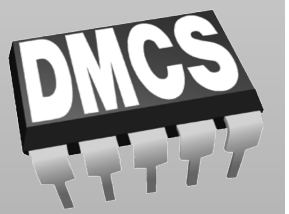
OpenHPI examples - FRU list


```
hpi_shell> ctrl
(001):Schroff uTCA CU:{S|RDR|INV|RST|FRU|CNT|RES}
(002):Schroff uTCA CU:{S|RDR|INV|RST|FRU|CNT|RES}
(003):AIES-MFMC:{S|RDR|INV|RST|FRU|CNT|RES}
(004):TAMC641:{S|RDR|INV|RST|FRU|CNT|RES}
(005):DMCS-PTPM:{S|RDR|INV|RST|FRU|CNT|RES}
(006):AIES-MPTM-1588:{S|RDR|INV|RST|FRU|CNT|RES}
(007):NAT-MCH-MCMC:{S|RDR|INV|RST|FRU|CNT|HS|RES}
(008):PM-AC1000:{S|RDR|INV|RST|FRU|CNT|RES}
(009):MCH-Clock:{S|RDR|INV|RST|FRU|CNT|RES}
(010):MCH-PCIe:{S|RDR|INV|RST|FRU|CNT|RES}
(011):PM-AC1000:{S|RDR|INV|RST|FRU|CNT|RES}
(012):NMCH-ShM:{S|RDR|ELOG|FRU|CNT|RES}
(013):Schroff uTCA CU:{S|RDR|INV|RST|FRU|CNT|RES}
(014):Schroff uTCA CU:{S|RDR|INV|RST|FRU|CNT|RES}
(015):AIES-MFMC:{S|RDR|INV|RST|FRU|CNT|RES}
(016):TAMC641:{S|RDR|INV|RST|FRU|CNT|RES}
(017):AIES-MPTM-1588:{S|RDR|INV|RST|FRU|CNT|RES}
(018):DMCS-PTPM:{S|RDR|INV|RST|FRU|CNT|RES}
(020):PM-AC1000:{S|RDR|INV|RST|FRU|CNT|RES}
(021):PM-AC1000:{S|RDR|INV|RST|FRU|CNT|RES}
(022):MCH-PCIe:{S|RDR|INV|RST|FRU|CNT|RES}
(023):MCH-Clock:{S|RDR|INV|RST|FRU|CNT|RES}
```



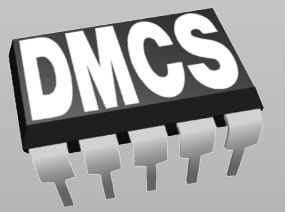
OpenHPI examples - FRU Info


```
inventory block ==> inv 005 011
Inventory: 11 Update count: 1 Read Only: TRUE Areas: 2
Area: 1 Type: OEM Read Only: TRUE Fields: 2
Field: 1 Type: CUSTOM Read Only: TRUE (BIN: 5A3100160028 (len=6))
Field: 2 Type: CUSTOM Read Only: TRUE (BIN:
5A31001900008002E0FFFFA498F300510000FC012F0000FD01230000FD01210000FD (len=34))
Area: 2 Type: PRODUCT_INFO Read Only: TRUE Fields: 7
Field: 1 Type: MANUFACTURER Read Only: TRUE (TEXT: ENGLISH: DMCS (len=4))
Field: 2 Type: PRODUCT_NAME Read Only: TRUE (TEXT: ENGLISH: PTPM (len=4))
Field: 3 Type: PART_NUMBER Read Only: TRUE (TEXT: ENGLISH: 1.0 (len=3))
Field: 4 Type: PRODUCT_VERSION Read Only: TRUE (TEXT: ENGLISH: 1.00 (len=4))
Field: 5 Type: SERIAL_NUMBER Read Only: TRUE (TEXT: ENGLISH: 000000631676 (len=12))
Field: 6 Type: ASSET_TAG Read Only: TRUE (TEXT: ENGLISH: -- (len=2))
Field: 7 Type: FILE_ID Read Only: TRUE (TEXT: ENGLISH: 21/12/2012 - P. Perek (len=21))
```



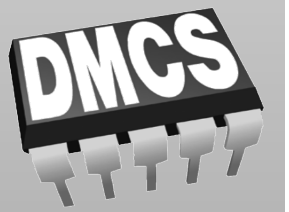
OpenHPI examples - Sensors List


```
(167): SENSOR_RDR ID=131239, Ctrl=0, EvtCtrl=RO, Tag=Hot Swap
(166): SENSOR_RDR ID=131238, Ctrl=1, EvtCtrl=WR, Tag=Ambient Temp 1
(165): SENSOR_RDR ID=131237, Ctrl=1, EvtCtrl=WR, Tag=Ambient Temp 2
(164): SENSOR_RDR ID=131236, Ctrl=1, EvtCtrl=WR, Tag=Ambient Temp 3
(163): SENSOR_RDR ID=131235, Ctrl=1, EvtCtrl=WR, Tag=VCC 1V2
(162): SENSOR_RDR ID=131234, Ctrl=1, EvtCtrl=WR, Tag=VCC 3V3
(161): SENSOR_RDR ID=131233, Ctrl=1, EvtCtrl=WR, Tag=VCC 12V PP
(145): SENSOR_RDR ID=131217, Ctrl=0, EvtCtrl=RO, Tag=Hot Swap
(144): SENSOR_RDR ID=131216, Ctrl=1, EvtCtrl=WR, Tag=TEMPERATURE
(143): SENSOR_RDR ID=131215, Ctrl=1, EvtCtrl=WR, Tag=VCC +12 VPP
(142): SENSOR_RDR ID=131214, Ctrl=1, EvtCtrl=WR, Tag=VCC +3.3 V MP
(141): SENSOR_RDR ID=131213, Ctrl=1, EvtCtrl=WR, Tag=VCC +1.2 V
(140): SENSOR_RDR ID=131212, Ctrl=1, EvtCtrl=WR, Tag=VCC +2.5 V
(139): SENSOR_RDR ID=131211, Ctrl=1, EvtCtrl=WR, Tag=VCC +3.3 V
(138): SENSOR_RDR ID=131210, Ctrl=1, EvtCtrl=RO, Tag=PCIE CABLE
(137): SENSOR_RDR ID=131209, Ctrl=1, EvtCtrl=RO, Tag=PCIE LANES
(136): SENSOR_RDR ID=131208, Ctrl=1, EvtCtrl=RO, Tag=PCIE GEN
```

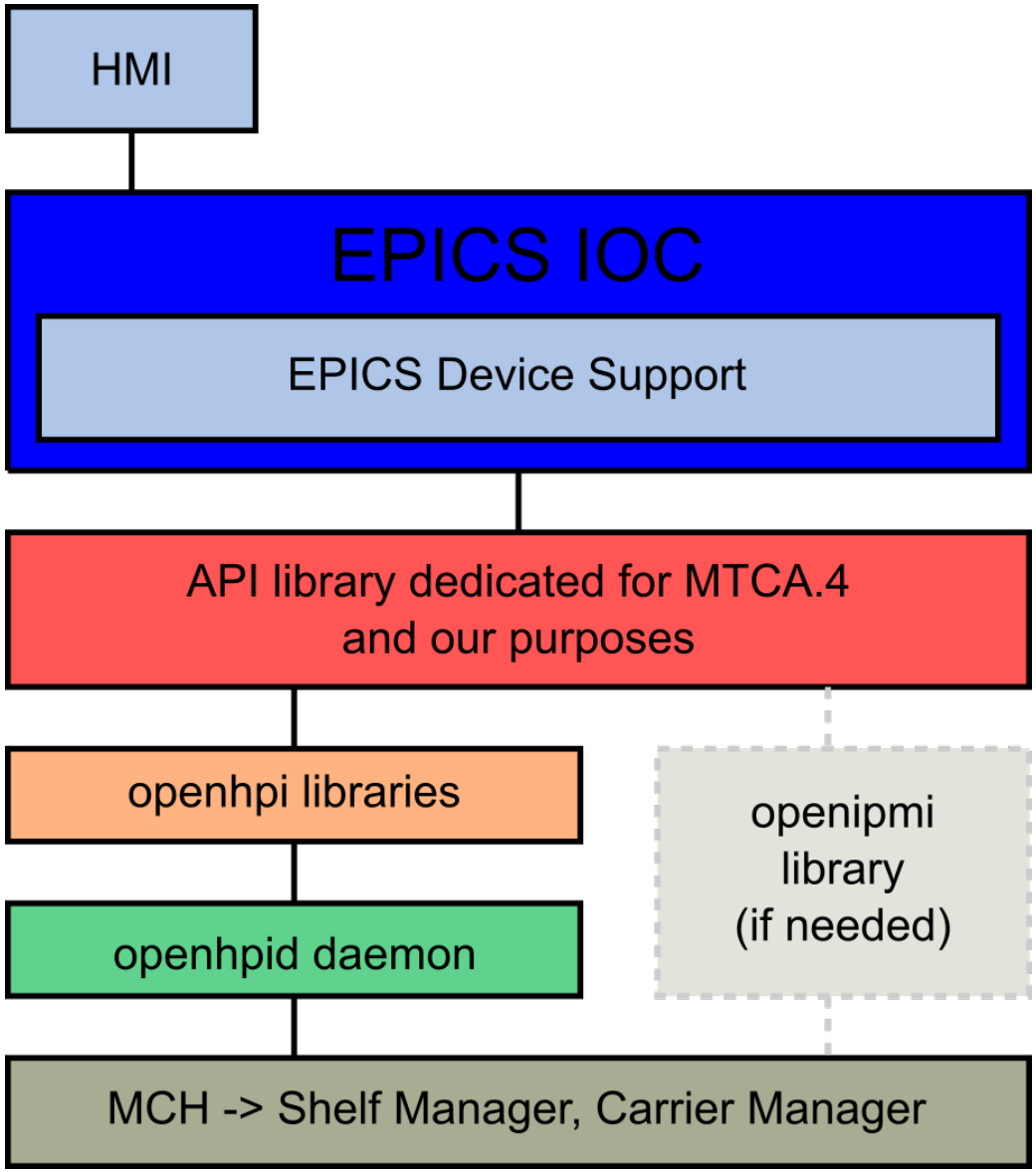


OpenHPI examples - Sensor Readout


```
Sensor(5/144) TEMPERATURE TEMPERATURE
Event states = 0x0
Reading Value = 33.000
Lower Minor Threshold(NA) 0.000
Lower Major Threshold(NA) -10.000
Lower Critical Threshold(NA) -20.000
Upper Minor Threshold(RW) 45.000
Upper Major Threshold(RW) 60.000
Upper Critical Threshold(RW) 70.000
Positive Threshold Hysteresis not supported.
Negative Threshold Hysteresis not supported.
```

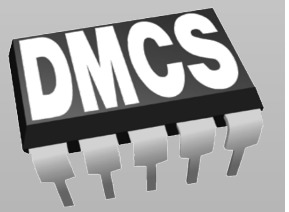


Proposed Software Architecture



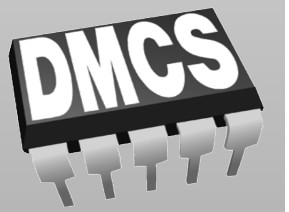
MTCA Example based on OpenHPI


```
 nat> show_fru
FRU Information:
---------------
FRU Device State Name
=============================================
0 MCH M4 NMCH-CM
3 mcmc1 M4 NAT-MCH-MCMC
6 AMC2 M4 DMCS-PTPM
7 AMC3 M4 TAMC641
9 AMC5 M4 AIES-MFMC
13 AMC9 M4 AIES-MPTM-1588
40 CU1 M4 Schroff uTCA CU
41 CU2 M4 Schroff uTCA CU
51 PM2 M4 PM-AC1000
53 PM4 M4 PM-AC1000
60 Clk1 M4 MCH-Clock
61 Hub1 M4 MCH-PCIe
64 RTM1 M4 AIES-MPCIE16
=============================================
```



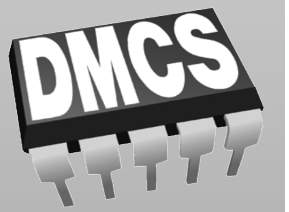
MTCA Example based on OpenHPI


```
 ==> RPT Entry ID: 0x3
Tag: DMCS-PTPM
Entity Path: {SYSTEM_CHASSIS,7}{SHELF_MANAGER,0}{AMC,2}
FRU ID: 6
==> RPT Entry ID: 0x1
Tag: Schroff uTCA CU
Entity Path: {SYSTEM_CHASSIS,7}{SHELF_MANAGER,0}{COOLING_UNIT,1}
FRU ID: 40
==> RPT Entry ID: 0x8
Tag: PM-AC1000
Entity Path: {SYSTEM_CHASSIS,7}{SHELF_MANAGER,0}{POWER_SUPPLY,4}
FRU ID: 53
==> RPT Entry ID: 0xC
Tag: NMCH-ShM
Entity Path: {SYSTEM_CHASSIS,7}{SHELF_MANAGER,0}{SWITCH_BLADE,0}
FRU ID: 0
```



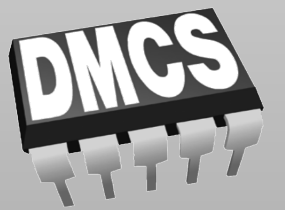
Last meeting - important question

- Can we monitor multiple MTCA crates with single CPU using OpenHPI?
  Yes, we can
  OpenHPI is flexible, we can:
  - run multiple daemons - one per chassis
  - run one daemon monitoring multiple chassis
  Can be decided later which version is better for us



Future plans

- Development of library dedicated for our purposes based on OpenHPI (and other libraries if really needed)
  - First step - support for all reading/monitoring features
  - Avoiding development of IPMI protocol from scratch
  - Providing some missing commands specific for MicroTCA.x
  - Check how to send configuration command using OpenHPI



Thank you for your attention