ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEBPP02

CENTRALIZED SYSTEM MANAGEMENT OF IPMI ENABLED PLATFORMS USING EPICS[*]

K. Vodopivec[†], Oak Ridge National Laboratory, Oak Ridge, TN, USA

Abstract

The Intelligent Platform Management Interface (IPMI) is a specification for computer hardware platform management and monitoring. The interface includes features for monitoring hardware sensors, such as fan rotational speed and component temperature, inventory discovery, event propagation, and logging. Additional features are available in PICMG compliant systems, including ATCA and Micro TCA. With IPMI support implemented in the hardware, all IPMI functionality is accessible without any host operating system involvement. In fact, IPMI can even be used to control remote host power management. With its wide breadth of support across many hardware vendors and the backing of a standardization committee, it is a compelling instrumentation choice for integration into control systems for large experimental physics projects. Integrating IPMI into the EPICS system provides the benefit of centralized monitoring, archiving and alarming within the facility control system. A new project has been started to enable this capability, by creating a native EPICS device driver built on the open-source FreeIPMI library for the remote host connection interface. The driver supports automatic system component discovery for creating EPICS database templates, including detailed device information from the Field Replaceable Unit interface, as well as sensor monitoring with remote threshold management, geographical PV addressing in PICMG based platforms and PICMG front panel lights readout.

INTRODUCTION

The Intelligent Platform Management Interface (IPMI) [1] is becoming a widely adopted technology, as supported by many hardware providers including Intel, Dell, Hewlett Packard, Cisco and others. More importantly, for the applications in large scale research facilities, the PICMG AdvancedTCA Base Specification [2] builds on IPMI with functionality specific to AdvancedTCA and MicroTCA platforms. With many VME components facing obsolescence issues, an increasing number of new applications at particle accelerator and synchrotron facilities are looking for alternatives, and settling for ATCA or MTCA platforms as the new standard for hardware based solutions. One of the reasons for this decision is also the availability of built-in native support for the IPMI standard, as this automatically furnishes the application with system health monitoring. Functionality that had previously been implemented on a case by case basis, and was often overlooked, is now part of every system and therefore can be used for more thorough monitoring and control of core system functions.

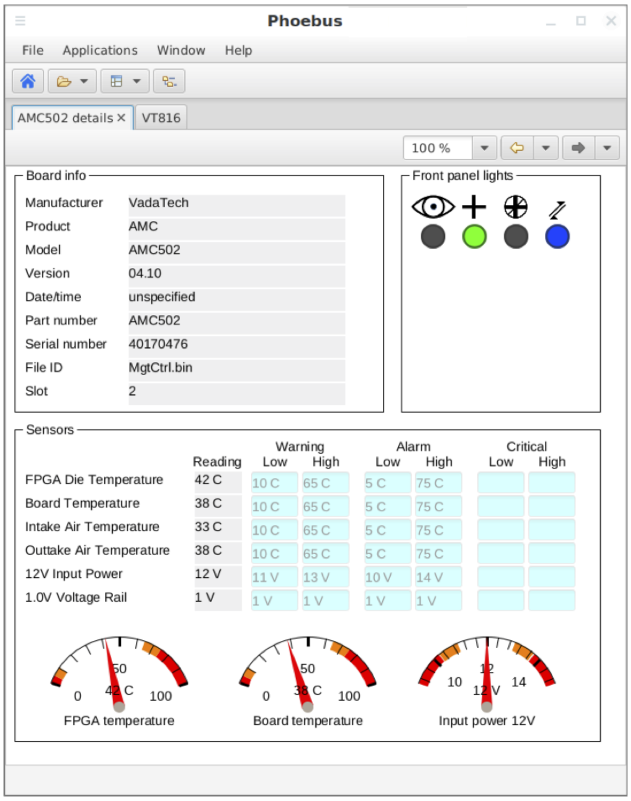
The IPMI standard provides interfaces to monitor embedded sensors such as temperature, voltage, current, fan speed and others, depending on the particular component implementation. Monitoring core sensors alone provides useful benefits for detecting component failures or potentially trying to prevent them. For example, a failed fan inside the chassis will immediately restrict the air flow and can rapidly cause the temperature of the components in the air flow path to increase, in turn causing the components and/or entire system to fail. By detecting the reduced fan speed and temperature increase, the failing fan can be identified and replaced. Alternatively, the speeds of other fans in the system can be increased to maintain adequate air flow until the failing fan can be replaced. Another notable feature of IPMI is its ability to read inventory information through the Field Replaceable Unit (FRU) mechanism. With the FRU interface, individual system components – including power supplies, cooling units, processing blades or expansion boards – can be addressed and queried for information such as the manufacturer name, model name or code, manufacturing date, serial number or several others. Having complete inventory information of the system and its individual components allows for detailed cataloging and traceability. A statistical analysis of hardware failures can be carried out in order to schedule preventive replacements of critical system components. Controlling low-level system functions is also available through IPMI. For example, it is possible to control a smart power supply and toggle power to the CPU, or in the case of ATCA and MTCA, the power distribution to individual expansion slots.

The Experimental Physics and Industrial Control System (EPICS) framework, on the other hand, provides infrastructure for distributed access and management of devices. Especially where already employed as the control system of choice, adding support for IPMI-capable devices greatly expands the repertoire of systems that EPICS can talk to or manage. Along with existing EPICS tools for real time status monitoring, alarm management, archiving data points for analyzing historic behaviour and others, IPMI-compliant systems can be easily integrated into the environment of experimental physics and industrial control systems. A new project called "*epicsipmi*" has been designed to provide the bridge between IPMI interfaces and the EPICS framework, enabling easy monitoring of system health functions in new

[*] This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan(http://energy.gov/downloads/doe-public-access-plan).

[†] vodopiveck@ornl.gov

project applications based on ATCA and MTCA and existing IPMI-capable hardware infrastructure.



Manufacturer	VadaTech
Product	AMC
Model	AMC502
Version	04.10
Date/time	unspecified
Part number	AMC502
Serial number	40170476
File ID	MgtCtrl.bin
Slot	2

		Warning		Alarm		Critical	
	Reading	Low	High	Low	High	Low	High
FPGA Die Temperature	42 C	10 C	65 C	5 C	75 C		
Board Temperature	38 C	10 C	65 C	5 C	75 C		
Intake Air Temperature	33 C	10 C	65 C	5 C	75 C		
Outtake Air Temperature	38 C	10 C	65 C	5 C	75 C		
12V Input Power	12 V	11 V	13 V	10 V	14 V		
1.0V Voltage Rail	1 V	1 V	1 V	1 V	1 V		

Figure 1: User interface displaying VadaTech AMC 502 information obtained from IPMI, including FRU, sensors and PICMG lights.

The *epicsipmi* system currently implements several IPMI features, including monitoring sensors, retrieving Field Replacable Unit (FRU) information, and displaying LED light status. Its modular design allows for easy future expansions. By leveraging the implementation of the IPMI protocol to the external library, it benefits from the wider community efforts beyond experimental physics. The project extends an underlying IPMI library where several beneficial features are missing; in particular it implements a subset of PICMG extensions. The *epicsipmi* repository includes out-of-the-box ready-to-use EPICS IOCs capable of connecting to multiple IPMI systems over Ethernet, discovering IPMI components and entities behind a given IPMI connection, and generating a database of EPICS records for connecting to corresponding IPMI entities. These records can then be applied to monitor sensors, their properties, inventory records and PICMG lights. Being part of an active IOC, these records are exported as EPICS PVs, available for subscription by interested clients such as control system user interfaces, alarm handlers and archivers. An example of user screen is show in Figure 1.

FREEIPMI LIBRARY

The IPMI specification quite extensively encompasses the communication and interfacing details that are most important from the software component interaction perspective, starting with establishing an encrypted TCP/IP connection, defining a binary communication protocol for exchanging messages, enabling direct and bridged addressing for subsystems, and many other actions. Several existing open source projects are available that already implement basic software interaction with IPMI systems, hence it would be redundant to start a new implementation from scratch, without at least evaluating the options. Therefore, in the search for an open-source, multi-platform project with a C library interface, we have considered three specific projects: ipmitool, OpenIPMI and FreeIPMI. We compared the implemented features and focused mostly on usability and the potential for custom extensions. The FreeIPMI library [3] was selected due to its complete standards-compliant implementation of the IPMI specification. It provides a rich and powerful Application Programming Interface (API) and supports a range of operating systems including Linux and MS Windows. From the provided source code, it builds both dynamic and static library files to be linked into the C/C++ application, and is available as an installable package on most popular Linux distributions. Two levels of APIs allow developers to select their proficiency with the IPMI specification. The comprehensive lower-level API fully implements the IPMI interfaces and closely matches the naming conventions from the IPMI specification, for easily mapping functionality from the specification to actual program functions. The higher-level API abstracts away some IPMI details in order to provide a somewhat easier-to-use interface.

The FreeIPMI library does not currently implement interfaces for PICMG extensions, due to the proprietary access of the PICMG standard. But given its modular design, *epicsipmi* is able to easily extend and fill in this missing functionality. The FreeIPMI library internally uses the FreeIpmi Interface Definition (abbreviated to “fiid”) structures to describe IPMI requests and responses. The fiids are used to instantiate request messages, send them to an IPMI target using the FreeIPMI transport functions, and decode the response messages when received. In order to bring front panel multi-color LED light status into EPICS records, *epicsipmi* implements PICMG-specific fiid definitions. These definitions are part of the *epicsipmi* code base, but they use the transport functions from FreeIPMI to actually communicate PICMG messages with the IPMI host. This extension not only adds the omitted functionality, but also shows the potential to easily extend FreeIPMI where such functionality is missing.

By using an existing library for realizing the IPMI connections, *epicsipmi* benefits in several areas. First, the generic aspects of our IPMI implementation project target a much larger audience than would a purely EPICS-specific implementation, which therefore inherently results in the project being better tested in many different operational environments, and with a variety of IPMI systems. Secondly, it is worth noting that, with its many powerful functions, IPMI systems are an attractive target for malicious attackers trying to gain access to remote hosts. The security-related details of establishing an encrypted connection and authenticating the user session must be carefully implemented, in order to support the strongest encryption available on remote systems, and to use the lowest required privilege level. The FreeIPMI library already handles these critical details very well.

EPICS DEVICE DRIVER

The *epicsipmi* project is designed to implement a new EPICS device driver, with no dependencies besides the EPICS base libraries and FreeIPMI. The device driver encapsulation manages encrypted network connections to IPMI systems, maps IPMI entities to EPICS records and provides diagnostic functions through an IOC console. The provided example IOC can be used directly, but linking the *epicsipmi* library into other custom IOCs is also possible.

For each remote IPMI system with a distinctive network address, an IPMI connection must be registered using the *ipmiConnect()* function in the IOC console. Only connections over a Ethernet network are supported (no local bus connections are supported at this time). After a connection has been registered, the corresponding FreeIPMI function is applied to establish the secure connection, given the interface type and authentication parameters. Connection security heavily depends on the selected interface type, and the use of *lanplus* is advised whenever the target system supports IPMI v2.0. After registering, the connection is managed automatically, and in the case of a disconnected socket it will be re-established. IPMI host implementations are commonly realized as standalone on-chip subsystems called Baseboard Management Controllers (BMCs) and they operate independently of the CPU or operating system running on the host. So in fact, the most benefit with IPMI in EPICS is realized when an EPICS IOC is running on a separate host and connects to monitor an IPMI-enabled system over the network. The recommended setup is to introduce a standalone "soft IOC" that serves multiple IPMI connections.

For each established connection, a thread and a queue are created to handle any requests that can potentially take significant time, due to the nature of the TCP/IP connection. In EPICS and asyn terminology, the IPMI connection is always blocking. In fact, the entire record processing mechanism is similar to the asyn's blocking request handler. The *epicsipmi* record processing takes advantage of the EPICS records' Process Active (PACT) mechanism to guarantee a single outstanding request per record. However, a particular record's request competes with other requests through a given connection, which is why *epicsipmi* serializes all record processing requests using one FIFO queue and one processing thread per connection. The oldest request in the queue is processed first. Based on the request, an appropriate IPMI request is created and sent through the established connection. The processing thread then waits for either a response or a timeout, whichever happens first. When received, the IPMI response is decoded and the available entity information is extracted from the response to populate the EPICS record counterparts. For example, the Sensor Database Record (SDR) response for an analog sensor includes the sensor value, measurement units, measured value precision, three pairs of low/high thresholds (only two pairs are connected to the EPICS record), and an alarm hysteresis factor - these fields correspond to "ai" record fields VAL, EGU, PREC, HIGH, LOW, HIHI, LOLO and HYST, respectively. The same record processing has proven successful in other EPICS device drivers and it works well for completion (put-callback) as well as posted requests.

In addition to mapping the IPMI entities into EPICS records, the established connection can also be used to discover IPMI entities on a given host. This is particularly useful during the initial commissioning of a new IPMI host. Triggered by the *ipmiScan()* function from the IOC console, the IPMI host is scanned for all IPMI entities that are supported by the *epicsipmi* system, including SDR, FRU, and PICMG lights. The results are printed to the IOC console output in a human friendly format, where it can be used for debugging and troubleshooting connection issues. Similar discovery results can also be saved in EPICS database format to a database file, using the *ipmiDumpDb()* function. The generated database file serves as a working template to be loaded by the IOC. The *ipmiDumpDb()* function chooses the most appropriate EPICS record types based on the IPMI entity introspection, and it pre-populates the record's metadata fields when available. Fields like description, valid operational ranges, thresholds for alarms and others are prepopulated in the generated EPICS record, for the developer to review and edit to create the final database. It is also possible to omit meta-data fields from the final database, in which case the *epicsipmi* device support will automatically include and manage them whenever the record is processed.

```
record(ai, ”Sys:LLRF1:CU130_40:CU:TEMP1") {
  field(SCAN, "1 second")
  field(DTYP, "ipmi")
  field(INP,  "@ipmi CONN1 SENSOR 65:1:0:2")
  field(DESC, "Cooling Unit 97 CU TEMP1")
  field(EGU,  "C")
  field(PREC, "2")
  field(LOLO, "10.000")
  field(LSV,  ”MAJOR")
  field(LOW,  "15.000")
  field(LSV,  "MINOR")
  field(HIGH, "65.000")
  field(HSV,  "MAJOR")
  field(HIHI, "75.000")
  field(HHSV, "MAJOR")
}
```

Figure 2: Example of generated EPICS record for cooling unit in MicroTCA chassis.

The records currently supported by *epicsipmi* are "ai" for monitoring analog sensors, "longin" for monitoring discrete sensors, "stringin" for obtaining inventory information from FRU, and "mbbi" for describing multi-colored light support. The record type is not tied to a particular IPMI entity type and the developer can choose to select a different record type for a given device, in which case *epicsimpi* will try to convert the value to be of that type. In order to select *epicsipmi* for device support, a given record must specify *ipmi* for the device type (DTYP) and provide a valid *epicsipmi*-specific address that connects the record to a specific IPMI entity, as shown in Figure 2. Courtesy of static IPMI addressing, the record addresses stay permanent for a given component. With ATCA and MTCA platforms, this means that each expansion slot is assigned an unique base address, which allows EPICS records to remain unchanged even in the case where the expansion board is replaced with another one of the same kind.

PICMG EXTENSIONS

Because FreeIPMI doesn't support any PICMG-specific extensions, *epicsipmi* implements that part of the PICMG AdvancedTCA specification that defines support for user visible panel lights. The specification distinguishes 4 specific system lights that are a standardized requirement for every ATCA-compliant component, and it allows for application-defined lights. In addition to its on/off state, each light can change colors and also define illumination duration intervals. Multi-colored lights are supported by *epicsipmi* through the 16 states of the "mbbi" record, while illumination duration is left for future expansions. In order to support PICMG-specific extensions, *epicsipmi* creates PICMG "fiid" messages that are communicated to IPMI hosts using common FreeIPMI mechanisms.

An important feature of the PICMG AdvancedTCA specification is to define the mapping between a component's address and its physical slot location within the expansion chassis. The expansion slot information is considered part of the geographical location, and is coupled with the location information for the actual chassis, through a manual inventory bookkeeping. It can be used to uniquely identify a particular board. This is very useful when a component needs to be physically located, for example in case of repair or replacement. Since the *epicsipmi* record addresses reflect actual corresponding IPMI addresses, the slot information is easily obtained through EPICS PVs.

CONCLUSION AND FUTURE WORK

The *epicsipmi* project has been tested with several IPMI systems, including enterprise grade PC servers from SuperMicro and Hewlett Packard, and several MicroTCA systems from VadaTech. Monitoring of system health functions through IPMI is being planned for monitoring MicroTCA applications like the Injection Kicker Waveform Monitor, the Machine Protection System controller and the Ring Low Level RF systems at the Spallation Neutron Source.

Future work plans include adding support for changing sensor thresholds, reading Sensor Event Log (SEL) information, handling pushed events through a record's I/O Intr scanning mechanism, and controlling some of the power management functions.

ACKNOWLEDGMENT

The author would like to thank Eric Breeding from Oak Ridge National Laboratory for identifying the need for this project, and for technical support throughout the development.

REFERENCES

[1] Intelligent Platform Management Interface specification, https://www.intel.com/content/www/us/en/products/docs/servers/ipmi/ipmi-home.html

[2] PICMG AdvancedTCA Base Specification, https://www.picmg.org/openstandards/advancedtca/

[3] FreeIPMI library, https://www.gnu.org/software/freeipmi/