INTERFACING EPICS TO THE WIDESPREAD PLATFORM MANAGEMENT INTERFACE IPMI

M. Ritzert[*], Heidelberg University, Germany

Abstract

The Intelligent Platform Management Interface (IPMI) is a standardized interface to management functionalities of computer systems. The data provided typically includes the readings of monitoring sensors, such as fan speeds, temperatures, power consumption, etc. It is provided not only by servers, but also by µTCA crates that are often used to host an experiment's control and readout system. Therefore, it is well suited to monitor the health of the hardware deployed in HEP experiments. In addition, the crates can be controlled via IPMI with functions such as triggering a reset, or configuring IP parameters. We present the design and functionality of an EPICS module to interface to IPMI that is based on ipmitool. It supports automatic scanning for IPMI sensors and filling the PV metadata (units, meaning of status words in mbbi records) from the IPMI sensor information. Most importantly, the IPMI-provided alarm thresholds are automatically placed in the PV for easy implementation of an alarm system to monitor IPMI hardware.

IPMI

The Intelligent Platform Management Interface (IPMI) is a common interface to monitoring and management data of servers, ATCA and µTCA crates and similar devices. Monitoring data is provided as analog or digital sensor readings. The sensor metadata provided includes several alarm thresholds that can be used in a monitoring system. Server management is performed via messages sent to the IPMI device from the control PC.

The structure of a typical IPMI system in an ATCA crate is shown in Fig. 1 The typical mode of access to IPMI systems in a distributed system landscape is via Ethernet LAN. The connection is received on the server side by the IPMI baseboard management controller (BMC) that operates independently of the host's CPU, often even when the host device is currently powered down. The BMC communicates via the intelligent platform management bus (IPMB) with other management controllers (MCs) using a modified I²C protocol. Each MC manages a set of sensors, and can be queried to enumerate and read these sensors in a standardized format. Automatic detection of the sensors available in a system is therefore possible, and all sensors are accessed in the same way; all details of the actual hardware access are hidden by the MC.

Since IPMI access is available also when the host device is powered off, it is also a useful tool to remotely manage compute farms. Servers or crates can be reset, started, or stopped without invoking the OS.

When custom hardware is built for IPMI-enabled platforms such as ATCA crates, the IPMI system can be extended by adding new MCs. The required connections to interface to the IPMB are provided on the backplane's connectors.

IPMITOOL-IOC

Ipmitool [1] has been chosen as the basis to develop an interface (input/output controller, IOC) from EPICS to IPMI systems. The main functionality of ipmitool is linked into a static library that is not installed into the target system. Therefore, the source code of ipmitool must be available at build time.

Since ipmitool is used for the communication, advanced features of IPMI such as authentication are easily implemented. Compatibility with a wide range of hardware is given; the systems examined during development include ATCA crates and 19" servers of several brands.

The IOC is written in C++, interfacing to the EPICS CA libraries and the ipmitool library written in plain C. Since data queries via IPMI are "slow" in the EPICS sense, the asynchronous PV support of EPICS is used. Typical scanning times for all sensors in a system could be in the order of a few seconds.

A custom logging library written in C++ is used to send textual log messages from the IOC to a STOMP server that can pass them on to JMS2RDB, so that they can eventually be displayed in the "Message History" view of CSS. This way, all deployed IOCs report their messages to a central place, and messages are less likely to be overlooked.

Usage of the Module

First, the connection to the IPMI host has to be established. From the IOC, this can be done by calling the new ipmiConnect function. It accepts parameters to define a numeric handle for the connection, the host name, and if required, the user name, password, and access level. Next, the device is scanned for sensors by calling ipmiScanDevice. The result of the scan is used to populate the meta data of the PVs that are loaded via the standard dbLoadRecords call.

Optionally, the result of the device scan can be dumped in EPICS database format to serve as a starting point when preparing the IOC to control the device.

The link addresses in the database use the AB_IO format "#Lw Ax Cy Sz @p", where w is the id of the link established with ipmiConnect, x is the IPMB target address, and y is the sensor id. z is reserved for future use (possibly for IPMB bridging addresses), as is p.

Automatic Sensor Detection

The IOC uses ipmitool's functions to enumerate the sensors in a system. It maintains a work list of IPMB addresses

[*] michael.ritzert@ziti.uni-heidelberg.de

Copyright © 2015 CC-BY-3.0 and by the respective authors

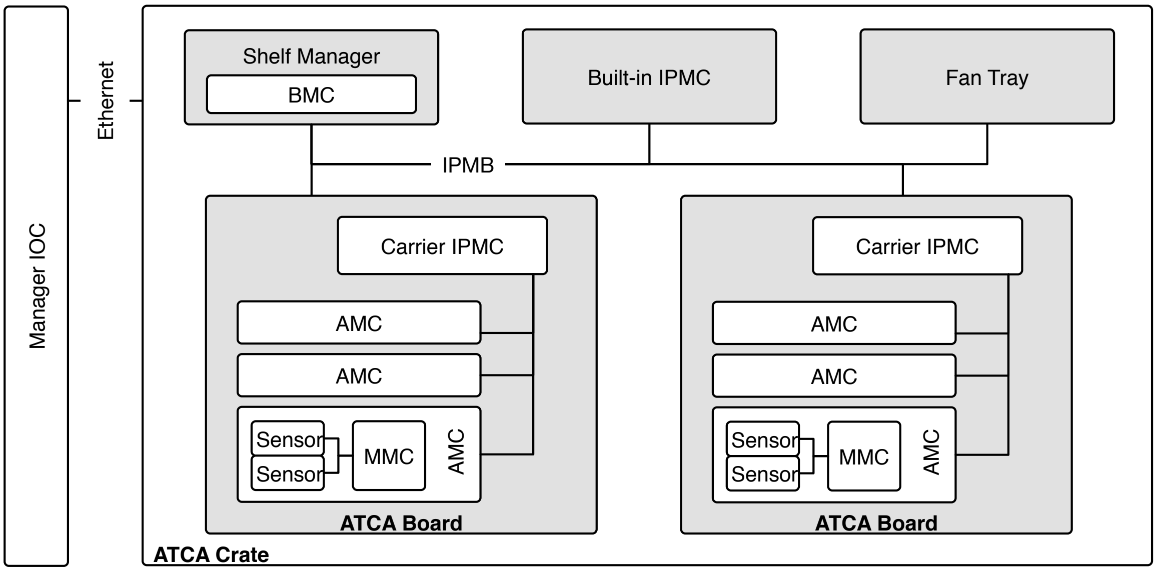


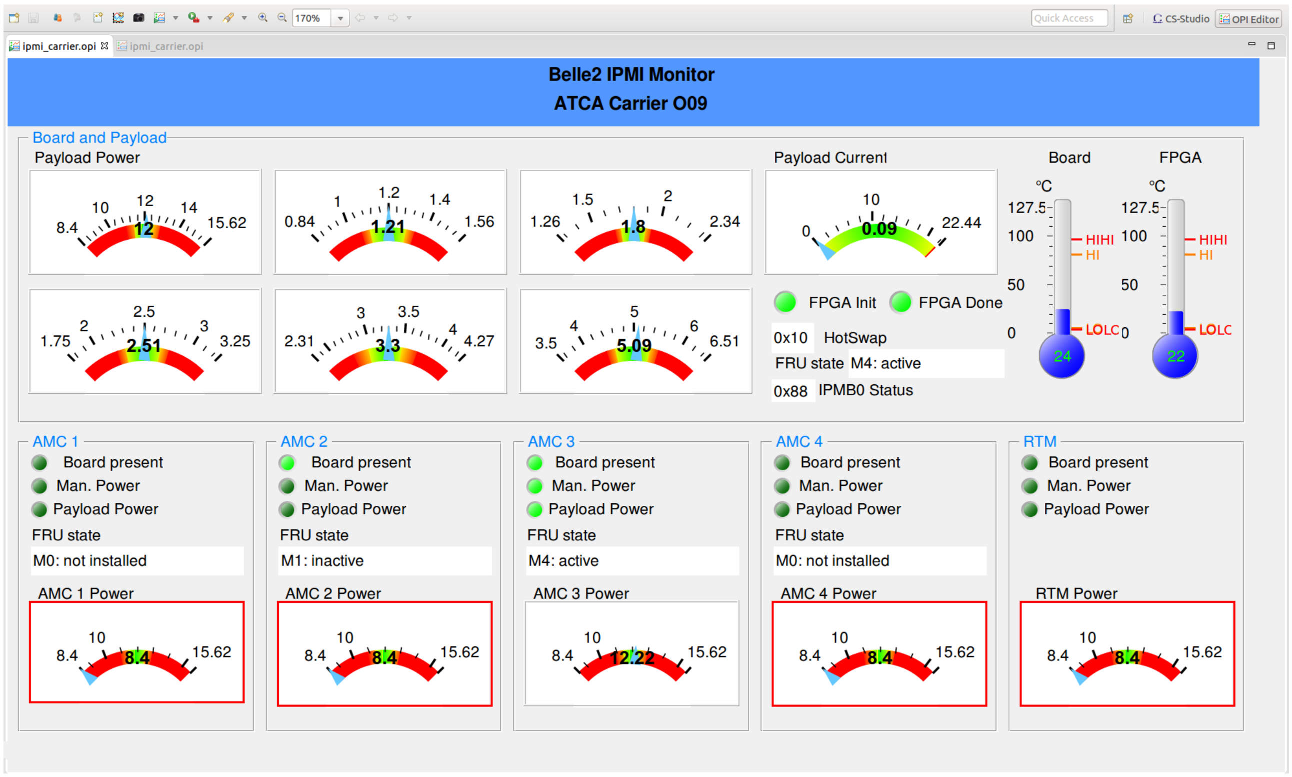
Figure 1: Typical IPMI system in an ATCA crate.

that are to be scanned. The list is amended, as new addresses are detected in the system via device locator records. In addition, the system is checked for the presence of the PICMG extension, that also provides access to a sensor bus.

The capability of IPMI to add hotplug devices at any time is not relevant for the intended application, and is not supported at the moment. It is planned to amend the code in the future.

EXAMPLE

The IOC is developed to be used with ATCA and μTCA crates housing readout and control hardware for the Belle II PXD subdetector. The health of the crates and custom add-on cards is monitored via EPICS. The information obtained via IPMI nicely complements health information read directly from the onboard FPGA. Figure 2 shows a CSS OPI



0x10 HotSwap
FRU state M4: active
0x88 IPMB0 Status

AMC 1 FRU state M0: not installed
AMC 2 FRU state M1: inactive
AMC 3 FRU state M4: active
AMC 4 FRU state M0: not installed

Figure 2: Control System Studio (CSS) displaying a GUI showing current IPMI readings from an ATCA board.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

screen that summarizes the state of one of the ATCA boards used in the PXD readout system. All data visible in this screen comes directly from the IPMI IOC. As can be seen, the analog readings are complemented with alarm thresholds automatically read from the crate. The alarms generated via these thresholds will be used to alert the operators to hardware malfunctions in the crates.

CONCLUSION AND OUTLOOK

An EPICS IOC to interface to IPMI-enabled devices has been written. A first stable version that supports automatic detection of sensors in a system and their readout is available. The next version of the IOC will support sending IPMI messages that are used to trigger actions in the system, e.g. a hard reset of a server. A future version will support accessing sensors that are dynamically added to the system. This involves processing IPMI events that are created when new hardware is added, and defining a new link naming scheme that takes into account that the sensor ids are no longer predictable in this case. Also planned for a later version is support for reading the IPMI event log.

ACKNOWLEDGEMENTS

This work has been supported by the German Federal Ministry of Education and Research (BMBF).

REFERENCES

[1] http://sourceforge.net/projects/ipmitool/.

Copyright © 2015 CC-BY-3.0 and by the respective authors