
European Coordination for Accelerator Research and Development

**PUBLICATION**

# Real-Time IPMI Protocol Analyzer

Kozak, T (TUL) *et al*

08 May 2011

The research leading to these results has received funding from the European Commission under the FP7 Research Infrastructures project EuCARD, grant agreement no. 227579.

This work is part of EuCARD Work Package **10: SC RF technology for higher intensity proton accelerators and higher energy electron linacs**.

The electronic version of this EuCARD Publication is available via the EuCARD web site <http://cern.ch/eucard> or on the CERN Document Server at the following URL : <http://cdsweb.cern.ch/record/1349299

# Real-Time IPMI Protocol Analyzer

T. Kozak, P. Predki, D. Makowski
Technical University of Lodz
Department of Microelectronics and Computer Science
Lodz, Poland

***Abstract*—The Advanced Telecommunications Computing Architecture (ATCA) is a modern platform, which gains popularity, not only in telecommunication, but also in others fields like High Energy Physics (HEP) experiments. Computing systems based on ATCA provide high performance and efficiency and are characterized by significant reliability, availability and serviceability. ATCA offers these features because of an integrated management system realized by the Intelligent Platform Management Interface (IPMI) implemented on dedicated Intelligent Platform Management Controller (IPMC).**

**IPMC is required on each ATCA board to fulfill the ATCA standard and is responsible for many vital procedures performed to support proper operation of ATCA system. It covers, among others, activation and deactivations of modules, monitoring of actual parameters or controlling fans. The commercially available IPMI implementations are expensive and often not suited to demands of specific ATCA applications and available hardware. Thus, many research centers and commercial companies decide to develop their own version of IPMC software. Despite precise IPMI specification, these implementations are often incompatible with each other, which leads to incorrect in-system behavior of devices equipped with IPMC from various vendors.**

**ATCA specifies the $I^2C$ protocol as a physical layer of IPMI. There are many devices able to monitor the $I^2C$ bus such as logic analyzers or specialized oscilloscopes. However, there is no available equipment capable of debugging IPMI as a higher level protocol. The article compares available methods of IPMI debugging and describes a custom made device prepared to monitor in real-time up to eight IPMI lines and analyze the IPMI protocol. Accessibility of this kind of equipment allows to discover errors and find the reasons of faulty behavior of the IPMC under development, greatly reduce the time to market factor and decrease costs of ATCA system development.**

***Index Terms*—Advanced Telecommunications Computing Architecture, Intelligent Platform Management Interface, protocol analyzer**

## I. Introduction

Thanks a high standard of reliability and availability the Advanced Telecommunications Computing Architecture (ATCA) has become more and more popular in many modern applications. Although intended mainly for telecommunications markets it has spread into other areas, including High Energy Physics (HEP) and other large-scale experiments [1]. Among the features that make this solution a good candidate in such applications are the redundancy of various management components as well as adaptation of the Intelligent Platform Management Interface (IPMI) [2].

### *A. IPMI as part of Advanced Telecommunications Computing Architecture*

IPMI has been developed by Intel as a set of common interfaces to a computer system which can be used by system administrators to monitor the state of the system and manage it [3]. It is completely independent of the underlying operating system and can even be implemented without one. Remote access to the monitored system is possible even if it has failed and IPMI allows to find out about the cause of the failure and to take appropriate actions leading to system recovery. At the top of the hierarchical structure of IPMI lies the Base Management Controller (BMC) to which all other controllers are connected. These satellite controllers, placed among various system modules, use the Intelligent Platform Management Bus (IPMB) to exchange messages with the BMC located in the same chassis.

The IPMI protocol is used in ATCA as the basis for communication between all the elements of the system [4]. Here, the Shelf Manager (ShM) takes the role of the BMC while the satellite controllers are the Intelligent Platform Management Controllers (IPMCs) situated on Field Replaceable Units (FRUs) such as ATCA carrier boards. The IPMB takes the form of a redundant $I^2C$ bus [5] (IPMB-0) and the chassis is called an ATCA Shelf, see TABLE I for comparison of these two naming conventions. Since the proposed IPMI protocol analyzer has been developed mainly with ATCA-based applications in mind, the system components' names will follow the ATCA nomenclature in the article.

TABLE I
Base IPMI vs. ATCA nomenclature

| Base IPMI | ATCA | AMC |
|---|---|---|
| Base Management Controller | Shelf Manager | — |
| Satellite Controller | Intelligent Platform Management Controller | Module Management Controller |
| IPMB | Redundant $I^2C$ IPMB-0 | $I^2C$ IPMB-L |
| Chassis | ATCA Shelf | — |

Another part of ATCA and MicroTCA ($\mu$TCA) systems are the Advanced Mezzanine Modules which are installed on carrier boards or directly in a shelf, respectively. They, too, have to be able to deal with the IPMI protocol and the component responsible for this is the Module Management Controller (MMC) which communicates with the IPMC on a carrier board over the IPMB-Local (IPMB-L) $I^2C$ bus. Each

TABLE II
IPMI MESSAGES STRUCTURE

| Byte No | Request | Response | |
|---|---|---|---|
| 1 | $I^2C$ Responder Address | 1 | $I^2C$ Requester Address |
| 2 | Network Function | 2 | Network Function |
| 3 | Header Checksum | 3 | Header Checksum |
| 4 | $I^2C$ Requester Address | 4 | $I^2C$ Responder Address |
| 5 | Command Sequence Number | 5 | Command Sequence Number |
| 6 | Command Code | 6 | Command Code |
| | | 7 | Completion Code |
| 7:N | Data | 8:N | Data |
| N+1 | Message Checksum | N+1 | Message Checksum |

AMC module requires an independent connection to the IPMB and carrier boards exist with up to four slots for these devices.

### *B. IPMI messaging*

IPMI is designed as a request-response protocol where messages are transferred from the IPMC to the ShM and back. The structures of both request and response messages are presented in TABLE II. The IPMI messages are divided into categories called Network Functions and each one consists of many commands. The majority of the Network Functions come from the base IPMI specification. However, ATCA introduced some additional commands that were specific to this kind of architecture and so another category was introduced. These extra commands are grouped in the PICMG Extension Network Function, getting its name from the collaboration that created ATCA - the PCI Industrial Computer Manufacturers Group (PICMG).

Each request in the IPMI communication needs to be followed by a response. The side initiating the exchange has to retry sending the original message if no response is received. This allows, for example, the ShM to know if a given FRU has become unresponsive or if one of the IPMB-0 buses has failed. What is more, even though the specification is explicit in the definitions of command implementations, some vendors do not completely comply with them which may cause errors in the message interchange. Information about the actual causes of failures in communication is not accessible from the level of the IPMI or ATCA and can only be obtained by monitoring the communication channels and analyzing the conversations between the modules in the system.

In addition to that, many research facilities decide to design their own ATCA-compliant boards which require the implementation of the IPMC. The realization of such a controller can be a daunting task as the number of mandatory commands that need to be taken care of exceeds one hundred with another hundred or more optional ones. This is the most time consuming stage of IPMC software development which also includes low-level drivers, interrupt service routines and other high-level application logic [6]. Knowing exactly where and when errors in communication arise and what their cause is can greatly improve the quality of the software and allow to develop it much faster.

## II. REQUIREMENTS AND AVAILABLE SOLUTIONS

There is a wide variety of devices capable of monitoring communication links between electronic equipment. Most of them are able to analyze common used standards like $I^2C$, SPI, UART or CAN. Other, more sophisticated ones, can monitor modern protocols such as USB, Gb Ethernet or PCI Express. ATCA specifies a $I^2C$ bus as the physical layer for IPMI. Therefore, only this protocol is of interest as far as the considerations of this article are concerned. The authors investigated solutions available on the market in order to find the device which could fulfill the main project requirements which include:

- Analyzing IPMI as a higher level protocol,
- Simultaneously monitoring a minimum of 5 $I^2C$ buses in real time:
  - Two IPMB-0 buses,
  - Three IPMB-L buses,
- Possibility of data storage for further analysis,
- Easy integration with the developed ATCA system is also desirable.

The following part of the paper will consist of a short evaluation of a few devices which may be used as $I^2C$ analyzers like oscilloscopes, logic analyzers and dedicated devices.

Modern oscilloscopes and logic analyzers give wide possibilities of electronic circuits monitoring. Moreover, these two kinds of equipment are often integrated in a single device. Leading manufacturers of measurement equipment such as Agilent Technologies, Tektronix or Hameg provide whole families of devices capable of monitoring the $I^2C$ protocol, for example the 16800 family from Agilent Technologies or the MSO70000 series from Tektronix. Modern analyzers may monitor signals with sampling rate up to 50 GS/s (e.g. MSO72004, DPO/DSA72004B), offer from several to a hundreds channels (e.g. 16806A). Despite its great potential this kind of measurement equipment is not a suitable solution because of many drawbacks. First of all, the analyzers are not able to monitor IPMI as a higher level protocol which dramatically increases the time needed to analyze the data. Moreover, the amount of gathered $I^2C$ frames is limited by the memory installed in the device. However, the most serious disadvantage are high costs of such equipment. Modern logic analyzers and oscilloscope presents possibilities which exceed the needs of simple $I^2C$ bus monitoring. Therefore, a set of devices utilized only as $I^2C$ bus analyzers is available on the market. The following solutions include Beagle I2C/SPI Protocol Analyzer [7], BusPro-I Bus Analyzer and Exerciser, CAS-1000-I2C/E Bus Analyzer and Exerciser [8], DV3100 and DV3400 from DigiView [9], Telos Tracii400 [10], Telos Connii MM 2.0 [11] and Bus Pirate [12], a project from the open source community. These devices present various capabilities, but all of them are able to monitor the $I^2C$ bus in transparent mode. It means that they do not influence the behavior of the monitored system. They are delivered with dedicated software which is used for visualization of $I^2C$ frames. Moreover, some of them, like the Beagle Protocol

Analyzer or Connii ML, offer a set of software libraries which allow to create a custom application. It should give the possibility to build an IPMI analyzer basing on the available, low cost hardware. Unfortunately, all of the presented devices are able to monitor only single $I^2C$ bus what is not sufficient. At least two IPMB-0 lines and three additional lines should be simultaneously monitored to provide a full view about IPMI communication in any ATCA system. Therefore, it was necessary to developed a custom analyzer, preferably as an ATCA board in order to simplify the connectivity.

## III. Proposed Solution

One of the major drawbacks of all the available solutions is that they do not support IPMI message analysis directly. The end-user is required to analyze the raw $I^2C$ frames in order to decode the IPMI requests and responses built from them. Another one is that these devices require additional connectors for integration with an ATCA system. A limitation in the number of buses is also a disadvantage.

The IPMI Protocol Analyzer deals with these problems. The developed PCB is the size of an ATCA Board and complies with all the specification requirements dealing with physical dimensions and connectivity of such a board. This means that the Analyzer can be inserted into any ATCA Shelf with no additional cabling and no extra work is necessary to monitor the communication over the IPMB-0 buses which the Analyzer taps into using a Zone 1 connector. What is more, up to eight $I^2C$ channels can be monitored independently, with buffers restoring the levels of each pair of data (SDA) and clock (SCL) signals. This allows the Analyzer to be used when developing not only IPMC for ATCA Boards but also Module Management Controllers (MMCs) for Advanced Mezzanine Cards (AMCs) that are typically installed in carrier boards as far as ATCA is concerned. Communication between the MMCs and the carrier board IPMC can be observed on the IPMB-Local buses.

All the data produced on the Protocol Analyzer board is transferred to a PC over a USB connection where it is collected by a dedicated software where the analysis of IPMI frames takes place. The application capabilities include detection of errors in communication, such as not acknowledged (NACK) $I^2C$ frames, filtering messages according to various criteria as well as storing the message exchange in a file on disk.

The IPMI Protocol Analyzer is also a cost efficient solution as it does not require advanced electronic components and production techniques. It should be possible to manufacture the whole design on a two-layer PCB provided a non-BGA FPGA is used.

### A. Hardware Architecture

The IPMI Protocol Analyzer board allows to monitor of up to eight independent $I^2C$ channels simultaneously which carry IPMI messages. The two IPMB-0 buses are available at the Zone 1 connector directly from the backplane while the six remaining buses require a link to be made to one of the two connectors present at the board. These additional channels

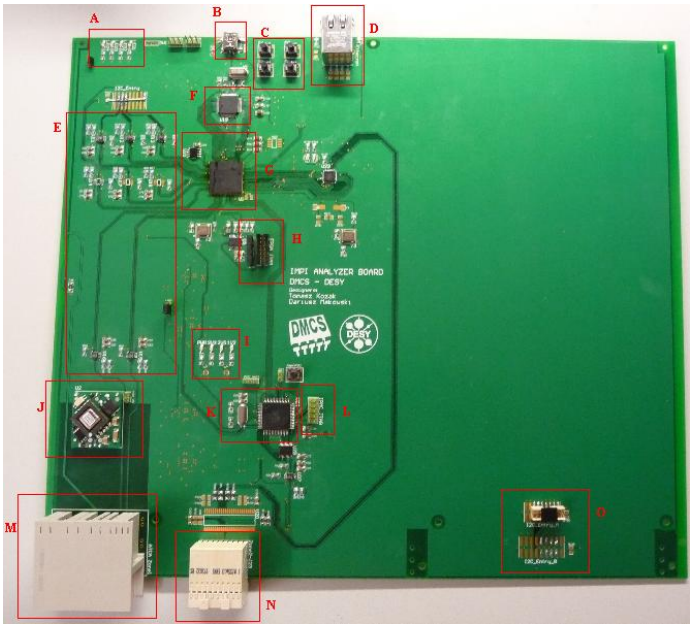


Fig. 1. Photograph of the IPMI Protocol Analyzer board

are useful when developing or diagnosing a system which uses AMC modules. The central part of the board is a Xilinx Spartan3 FPGA, where all the signal processing takes place. The algorithms and dataflow implemented in the device are explained in the next subsection. A short description of the vital components of the Analyzer board, as indicated in the photograph shown in Fig. 1, is presented below. The letters in parentheses correspond to the location of the components on the board.

The heart of the device is an XC3S200 FPGA chip from Xilinx Spartan 3 family (A). All of the $I^2C$ buses are connected to the FPGA, where they are collected into packets containing whole IPMI messages and then are sent to the PC application. The current firmware revision of the Analyzer logic covers almost 50% of available resources. The Analyzer PCB is designed to provide Ethernet connectivity, but the currently used FPGA's resources are not sufficient to implement the IP stack. Thus, in the next version of the device an FPGA with greater resources e.g. XC3S1000 is recommended. In order to establish an Ethernet connection via Zone 2 a simplified version of IPMC must be present on the board to cover Electronic Keying. Therefore, an Atmel ATmega 128 microcontroller (B) was applied. Moreover, it is also responsible for power management on the board. It is the first device to be powered up after the main power has been applied. It then proceeds with enabling other LDO regulators which provide power, among others, to the FPGA. All these converters are located on the back side of the IPMI Protocol Analyzer board and include 5 V, 3.3 V, 2.5 V and 1.8 V regulators. Every power supply is equipped with an additional LED indicator (C) to confirm its proper operation. All $I^2C$ buses are buffered via a set of LTC4300 devices (D). They allow to regenerate the level of the signals on the lines and can be switch on or off

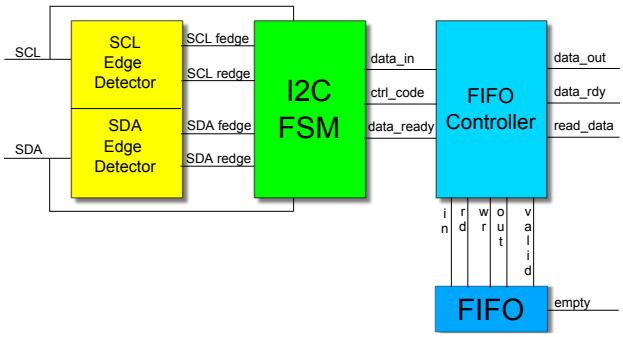


Fig. 2. 'I$^2$C Data Gatherer' module

by FPGA logic to choose whether a given I$^2$C bus should be monitored by the IPMI Protocol Analyzer or not. The USB communication standard was chosen to provide user friendly communication links between the Analyzer and an external computer, where analyzing software is run. Therefore, the PCB is equipped with a USB connector (E) and an FTDI FT2232HL (F) integrated circuit. It features two UART controllers which present themselves as virtual COM ports to the PC to which the USB cable is connected. The I$^2$C connectivity is provided by two external connectors (G). Both of these allow the IPMI Analyzer to listen to I$^2$C buses not available at the ATCA backplane, most notably the IPMB-L buses for IPMC $\Leftrightarrow$ MMC communication on ATCA carrier boards. The same signals are present at both connectors which only differ in pitch one of them being 2 mm and the other 2.54 mm. The board is also equipped with set of the LED indicators (H) and push buttons (I) to control and visualize the state of the Analyzer.

### B. FPGA Firmware

The FPGA firmware is designed in a modular way that enables instantiation of up to N I$^2$C monitoring modules, where N is the maximum number of buses supported by the underlying hardware (eight in this case). For different hardware architectures, if more than eight buses were available, it would not be a problem to monitor all of them, provided there were enough FPGA resources.

Each pair of SDA and SCL signals for a given bus are inputs to an 'I$^2$C Data Gatherer' module (Fig. 2) which consists of units detecting falling and rising edges on the inputs, an 'I$^2$C Finite State Machine' and a 'FIFO controller' transferring full IPMI frames to and from a Xilinx FIFO IP Core. Each 'I$^2$C Data Gatherer' module is identified by a unique 8-bit ID number chosen in an arbitrary way. This ID number, along with an additional byte marking the Start or Stop bit, is appended to the beginning and the end of the whole message, respectively, in order for the PC application to be able to distinguish between frames coming from different buses. For example, a message appearing on the IPMB-A bus has the following general form:

'A' | 's' | address | byte1 | ... | byten | 'A' | 'p'

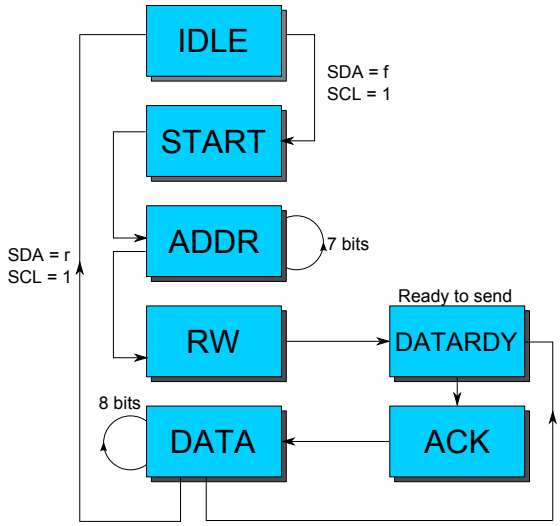


Fig. 3. 'I$^2$C Finite State Machine' state transition diagram

- 'A' - ID number of the IPMB-A bus,
- 's' - special character indicating Start Bit,
- address - address of the slave device on the I$^2$C bus,
- byte1 ... byten - data bytes,
- 'p' - special character indicating Stop Bit.

Other special characters include 'n' for Not Acknowledge event and 'r' for Repeated Start event on the I$^2$C bus. Also, if a data byte is equal to the bus ID number an additional byte equal to the bus ID number is added before this data byte, which acts as an escape character.

The 'I$^2$C Finite State Machine', see Fig. 3, requires six input signals in order to properly monitor the bus at the lowest level, that is forming raw I$^2$C frames from the Start to the Stop bit. These signals are SDA, SCL and rising and falling edge indicators of both. The states the FSM goes through during data acquisition are the following:

- *IDLE* - The machine enters this state on reset and waits for the falling edge of the SDA signal while the SCL signal is high, which marks the first step of the Start Bit on the I$^2$C bus,
- *START* - The machine waits for the falling edge of the SCL signal following the previous falling edge on the SDA signal, completing the Start Bit on the I$^2$C bus,
- *ADDR* - The I$^2$C bus address of the target device is read in this state. Seven values, corresponding to seven address bits, are read from the SDA bus on the rising edge of the SCL signal,
- *RW* - The Read/Write bit is read from the SDA bus on the falling edge of the SCL signal. A '1' indicates that the master reads from the slave while a '0' indicates that the master writes to the slave. In case of IPMI messaging the master transmitter mode is used exclusively, so this value should always be '0',
- *DATARDY* - In this state the signal indicating that a valid byte of data is present at the output of the 'I$^2$C FSM' module is asserted,

- *ACK* - After every eight bits an acknowledge bit should be asserted by the device receiving the data (slave for R/W equal to '0' and master for R/W equal to '1') by pulling the SDA line low. The state of the SDA line is checked in this state,
- *DATA* - Eight data bits are collected in this state in a similar manner as in the *ADDR* state. However, if a rising edge of the SDA line is found when the SCL signal is high a Stop Bit is detected on the $I^2C$ bus indicating end of transmission.

It is important to emphasize that the signal monitoring modules do not interfere with the communication taking place on the bus. The IPMI Protocol Analyzer is transparent as far as $I^2C$ is concerned, it does not have an address assigned and can not act as neither master nor slave in any kind of $I^2C$ conversation.

The signals coming out of the '$I^2C$ FSM' are forwarded to the 'FIFO Controller' which puts the data into a FIFO queue and makes it available to the 'Decision Module' which reads one message at a time and outputs it to the PC over the USB cable. Each module responsible for monitoring one of the buses has its own FIFO which can hold up to 1024 IPMI messages with a maximum length of 32 bytes.

The 'Decision Module' receives signals from 'FIFO Controllers' of all the monitored $I^2C$ buses and so it is important that all of them have equal access to the analyzer output. In order to meet this requirement the 'Decision Module' acts as a multiplexer switching between 'FIFO Controllers' in a round robin fashion. It sends one whole IPMI message at a time from one $I^2C$ bus and moves to the next one. If a message is available there, it is sent. Otherwise, the next $I^2C$ bus is checked and so on. The data bytes that move through the 'Decision Module' are not transparent to it because it uses the Stop Bit sequence (bus ID + 'p') to know when one IPMI message is finished before accessing another 'FIFO Controller'. Thus, it is vital that a failure on one of the buses does not halt the entire Analyzer. For this reason a timeout counter has been implemented and if neither a data byte nor a Stop Bit sequence is detected for more than 100 us after the transmission has started, the faulty bus is skipped.

### C. PC Software for IPMI Frame Analysis

The PC application has been written in Perl scripting language using Tkx as the Graphical User Interface (GUI) library [13]. It consists of two parts - one part collects the data from the serial port and writes it to a temporary file on the disk while the second one, with a GUI, enables the user to analyze the obtained IPMI messages. The GUI part of the software has an auto-refresh option that allows the data to be analyzed in real-time. The frame list is updated every one second. However, it is possible to record the transmission and analyze it at a later time, offline. (Fig. 4).

The application is equipped with several functions which make the frame analysis easier and more user-friendly. The search option uses regular expressions to find a message according to user input. Flexible filtering mechanisms have been implemented. The user can filter out messages according to the IPMI command Net Function, requester address, responder address or $I^2C$ bus. It also can show or hide Not Acknowledged frames. All these options enable fast access to only those messages that are of interest to the user (Fig. 5).

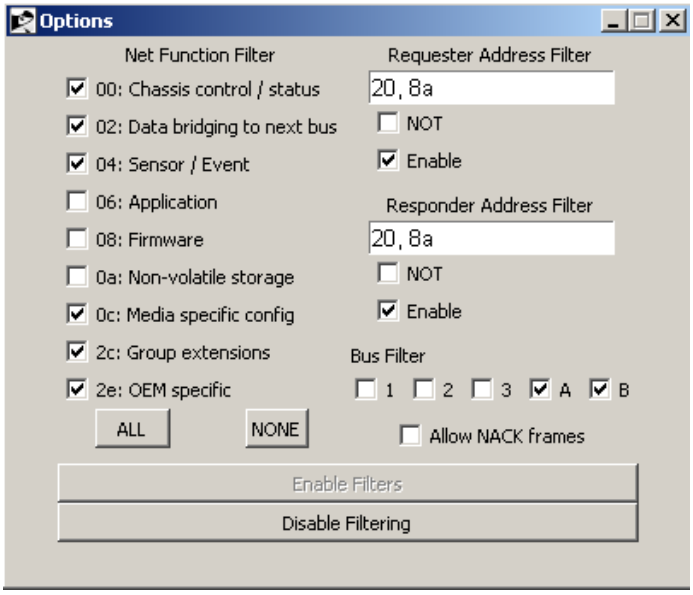


Fig. 5. IPMI Protocol Analyzer filtering window

## IV. Test Results

Tests of the IPMI Protocol Analyzer have been carried out at Deutsches Elektrontn-Synchrotron (DESY) laboratories using a Schroff 13U, 14 slot ATCA Shelf [14]. It has been populated with the Analyzer board, the Carrier Board for LLRF Control System and a RadiSys Promentum ATCA-1200 ATCA Managed Quad-AMC Carrier Blade [15]. A Pigeon ShMM-500R Shelf Manager [16] has been used and the AMC modules that have been installed in the carrier boards have been TEWS TAMC900 [17], Emerson PrAMC-6210 [18] and NAT NAMC-8560-8E1 [19]. These devices have been present in the Shelf in various combinations while the IPMI Protocol Analyzer gathered and analyzed IPMI frames from five channels - two IPMB-0 and three IPMB-L.

The periods of time where the most data is transferred over the $I^2C$ buses is during the activation and deactivation of FRU devices [20]. In case of AMC module activation and deactivation messages are transmitted from the AMC's MMC to the carrier board's IPMC and further down to the ShM. Responses to these messages have to travel all the way back to the MMC. This means that, if all the AMC modules on one carrier board are being activated at once, all the communication channels are active during these transmissions. This constitutes the hardest working conditions for the IPMI Protocol Analyzer as it needs to be able to gather and send to the PC frames from up to five channels simultaneously. The Analyzer had no problems dealing with such a workload and no data has been lost. This has been verified by checking

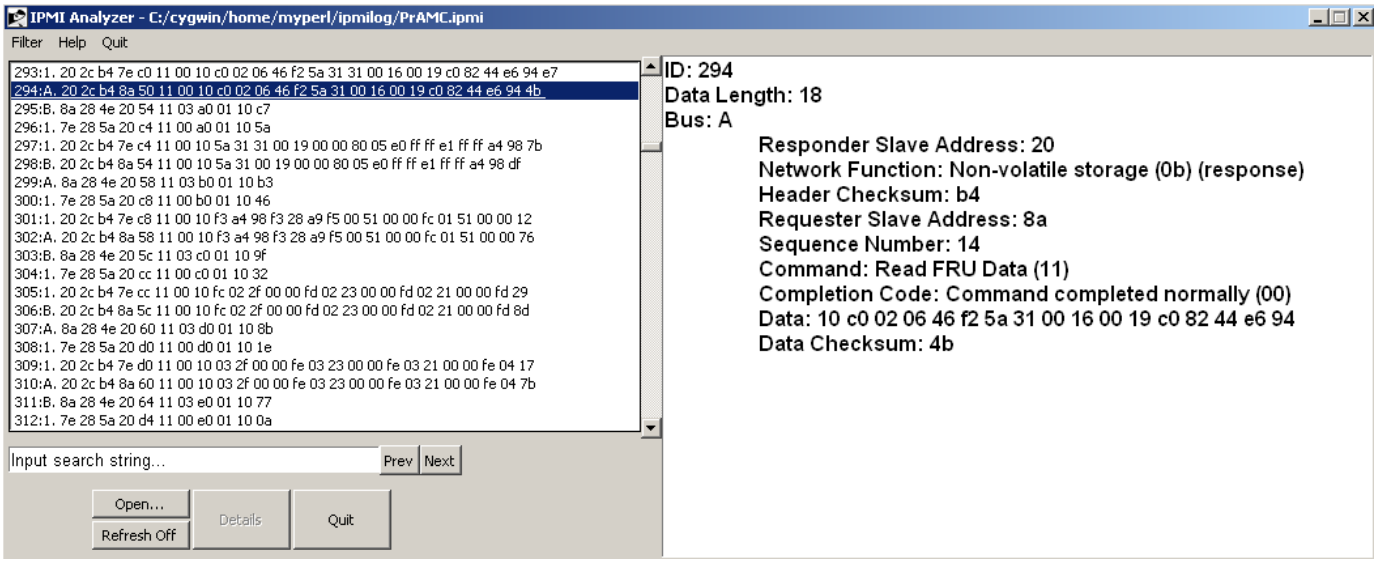


Fig. 4. Main IPMI Protocol Analyzer application window

the sequence numbers of the IPMI messages received at the PC. The sequence number is a 6-bit variable and so it goes from 00h to 3Fh and then is reset to 00h again. Making sure that no gaps appear in the communication and that the overall activation or deactivation procedure is carried out according to the specification proves that the IPMI Analyzer is capable of handling such situations.

While verifying the proper operation of the IPMI Analyzer several inconsistencies in operation of certain devices that have been monitored have been found. First of all, the TAMC900 AMC (Serial Number 9369462) module does not comply with the IPMI requirement of resending requests if no response is received. During the activation (deactivation) process AMC modules send several requests to the carrier board. If, for some reason such as physical failure, such a request from the TEWS TAMC900 does not reach the target carrier board or a response does not reach the TEWS device, the whole activation (deactivation) is halted as no retries are carried out. The carrier board does not have the possibility to restart the process since the AMC module is the initiating side. The only solution to this problem is restarting the AMC by power cycling the device. As an additional note, it is worth mentioning that the TEWS module is responsive and it is possible to, for example, retrieve the information concerning the state of the on-board sensors. Only the activation (deactivation) logic is corrupt in such situations.

Another issue has been encountered with the NAT NAMC-8560-8E1. Although this module has been activated and deactivated by the Carrier Board for LLRF Control System, the activation process has constantly failed on the RadiSys carrier board. The problem turned out to be with the FRU Inventory which is read by the carrier board during the activation process. In one of the fields of one of the FRU records a constant value of 'Manufacturer ID', given by the specification, is required. Although this value does not influence the overall operation of the AMC, it is verified by the RadiSys carrier board. In case of the NAT device the byte order of this field is reversed which the RadiSys board detects as an invalid value and halts the activation process. The content of the frame in question is presented in Fig. 6 as compared with a proper byte ordering of 'Manufacturer ID' value for an Emerson AMC module.

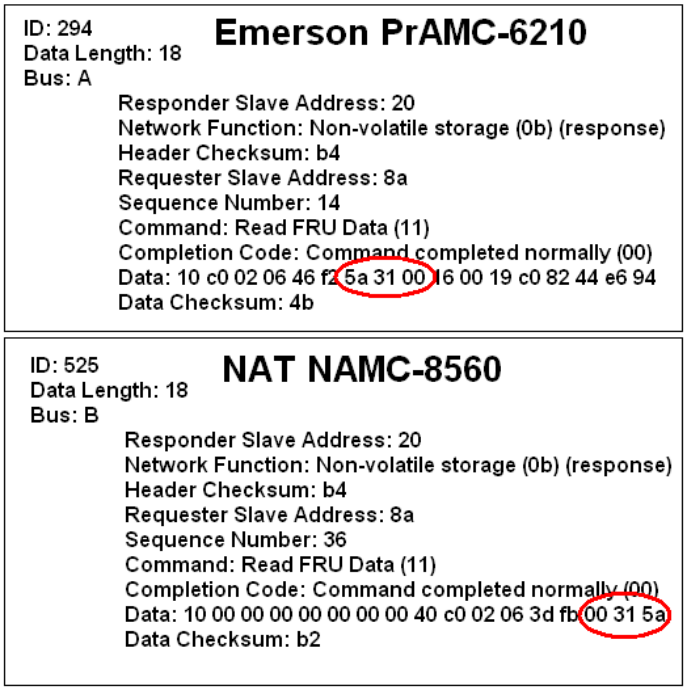


Fig. 6. ATCA standard implementation inconsistencies. The top figure shows the correct byte ordering for the PrAMC-6210 module. The bottom figure shows the incorrect byte ordering for the NAMC-8560 module.

## V. Conclusions

IPMI messaging is a crucial part of every ATCA-based system. The redundancy of the IPMB-0 bus is one of the reasons the message flow between the ATCA boards and the ShM is very stable which contributes to the overall reliability and availability of the ATCA solution. However, such a redundancy will not guarantee proper operation of the system if the possible problems arise from the implementation of the standard in commercially-available and in-house devices such as ATCA carrier boards or AMC modules [21]. Therefore, it is very important to identify these issues as quickly as possible during development of IPMCs' and MMCs' new firmware and to do so with as little effort as possible in order to focus on other features of the devices.

Some equipment is available that can be used for this purpose. Oscilloscopes and Logic Analyzers can monitor the $I^2C$ signals at a very low level which is not convenient for frames consisting of several dozens of bytes and message exchange consisting of hundreds of such frames, for example during activation or deactivation. Other devices, built specifically for monitoring and analyzing serial protocols, including $I^2C$, can be more helpful but still require a lot of attention in order to form IPMI messages from the gathered raw $I^2C$ data.

A new solution has been presented that fills that void in the ATCA development environment. The IPMI Protocol Analyzer is a perfect choice for such an application. It can be easily integrated in any ATCA Shelf, it does not require any additional power supply and it is possible to monitor and analyze all the IPMI messaging channels. Additionally, the PC software has been developed in the Perl scripting language which makes this a portable solution that can be run on any operating system provided that the computer is equipped with a USB slot. A standard 10/100 Mbps Ethernet connection will be sufficient in the future revisions of the Analyzer while all the data acquisition and storage will be done remotely on a server connected to the board. The ability to filter and search for specific messages in software as well as disabling any of the $I^2C$ channels in hardware allows the developer to focus only on these parts of the standard implementation that are the most important at any given time. The IPMI Protocol Analyzer proved to be very helpful during development of IPMC and MMC firmware of modules for LLRF control system at DESY, namely the Carrier Board for LLRF and the AMC_B universal communication module [22].

What is more, the IPMI Protocol Analyzer can also be used in situations where commercially-available devices seem to be operating incorrectly. In spite of strict requirements of the standard, not all of the equipment vendors follow them which may lead to instable or erroneous activity. Information about such occurrences can be fed back to the manufacturer which can greatly increase the overall quality of all ATCA-compliant devices.

## Acknowledgments

"The research leading to these results has received funding from the European Commission under the EuCARD FP7 Research Infrastructures grant agreement no. 227579. The authors are scholarship holders of the project entitled "Innovative education ..." supported by European Social Fund."

## References

[1] S. Simrock, L. Butkowski, M. Grecki, T. Jezynski, W. Koprek, G. Jablonski, W. Jalmuzna, D. Makowski, A. Piotrowski, and K. Czuba, "Evaluation of an ATCA based LLRF system at FLASH," in *Mixed Design of Integrated Circuits Systems, 2009. MIXDES '09. MIXDES-16th International Conference*, 25-27 2009, pp. 111 –114.

[2] A. Karlsson and B. Martin, "ATCA: its performance and application for real time systems," *Nuclear Science, IEEE Transactions on*, vol. 53, no. 3, pp. 688 – 693, june 2006.

[3] IPMI v2.0 rev. 1.0 specification. [Online]. Available: http://www.intel.com/design/servers/ipmi/spec.htm/

[4] PICMG 3.0 AdvancedTCA Base R3.0. [Online]. Available: http://www.picmg.org/v2internal/specifications.htm/

[5] The I2C-bus specification, version 2.1. [Online]. Available: www.nxp.com/documents/other/39340011.pdf/

[6] A. Zawada, D. Makowski, T. Jezynski, S. Simrock, and A. Napieralski, "ATCA Carrier Board with IPMI supervisory circuit," in *Mixed Design of Integrated Circuits and Systems, 2008. MIXDES 2008. 15th International Conference on*, 19-21 2008, pp. 101 –105.

[7] Beagle Protocol Analyzers Data Sheet v3.06. [Online]. Available: http://www.totalphase.com/download/pdf/beagle-v3.06.pdf

[8] CAS-1000-I2C/E Product Webpage. [Online]. Available: http://www.corelis.com/products-bus-analyzers/Bus_Analyzer_I2C_CAS-1000-I2C-E.htm

[9] DigiView DV3400 Product Webpage. [Online]. Available: http://www.tech-tools.com/dv_dv3400.htm

[10] Telos Tracii 400 Product Webpage. [Online]. Available: http://www.telos.de/tracii400/

[11] Telos Connii MM 2.0 Product Webpage. [Online]. Available: http://www.telos.de/conniimm20/

[12] Bus Pirate Project Webpage. [Online]. Available: http://dangerousprototypes.com/bus-pirate-manual/

[13] Comprehensive Perl Archive Network: Tkx. [Online]. Available: http://search.cpan.org/dist/Tkx/

[14] Schroff 12 U, 14 slot ATCA shelf. [Online]. Available: http://www.schroff.pl/internet/html_pl/index.html

[15] Radisys Promentum ATCA-1200 Quad-AMC Carrier Blade datasheet. [Online]. Available: http://www.radisys.com/Products/ATCA/Carrier-Blades/Promentum-ATCA-1200.html

[16] Pigeon Point Products Webage. [Online]. Available: http://www.pigeonpoint.com/products.html

[17] TAMC900 8 Channel 105 MSps 14 Bit AD Converter. [Online]. Available: http://www.powerbridge.de/daten_e/AMC-IO-Modules/TAMC900/tamc900.htm

[18] MPC8641D PowerPC Processor Based AMC Module. [Online]. Available: http://www.emerson.com/sites/Network_Power/en-US/Products/Product_Detail/Product1/Pages/EmbCompPrAMC-6210.aspx

[19] NAT NAMC-8560-xE1/T1/J1 Datasheet. [Online]. Available: www.nateurope.com/data_sheets/NAMC-8560-E1_ds.pdf

[20] P. Predki and D. Makowski, "Hot-plug based activation and deactivation of ATCA FRU devices," in *Mixed Design of Integrated Circuits Systems, 2009. MIXDES '09. MIXDES-16th International Conference*, 25-27 2009, pp. 119 –122.

[21] J. Lang, M. Liu, Q. Wang, W. Kuehn, Z. Liu, and H. Xu, "Intelligent Platform Management Controller for ATCA Compute Nodes," in *Real Time Conference, 2009. RT '09. 16th IEEE-NPSS*, 10-15 2009, pp. 35 –37.

[22] D. Makowski, A. Piotrowski, and A. Napieralski, "Universal communication module based on AMC standard," in *Mixed Design of Integrated Circuits and Systems, 2008. MIXDES 2008. 15th International Conference on*, 19-21 2008, pp. 139 –143.