ATCA-BASED BEAM LINE DATA SOFTWARE FOR SLAC'S LCLS-II TIMING SYSTEM*

D. Alnajjar†, M. P. Donadio, K. Kim, M. Weaver
SLAC National Accelerator Laboratory, Menlo Park, USA

Content from this work may be used under the terms of the CC BY 4.0 licence (© 2023). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Abstract

Among the several acquisition services available with SLAC's high beam rate accelerator, all of which are contemplated in the acquisition service EPICS support package, resides the new ATCA Beam Line Data (BLD) service, which runs on top of SLAC's common platform software and firmware, and communicates with several high-performance systems (i.e. MPS, BPM, LLRF, timing, etc.) in LCLS, running on a 7-slot Advanced Telecommunications Computing Architecture (ATCA) crate. Once linked with an ATCA EPICS IOC and with the proper commands called in the IOC shell, it initializes the BLD FPGA logic and the upper software stack and makes PVs available allowing the control of the BLD data acquisition rates, and the starting of the BLD data acquisition. This service permits the forwarding of acquired data to configured IP addresses and ports in the format of multicast network packets. Up to four BLD rates can be configured simultaneously, each accessible at its configured IP destination, with a maximum rate of 1 MHz. Users interested in acquiring any of the four BLD rates will need to register in the corresponding IP destination to receive a copy of the multicast packet on their respective receiver software. BLD has allowed data to be transmitted over multicast packets for over a decade now at SLAC, but always at a maximum rate of 120 Hz. The present work focuses on bringing this service to the high beam rate high-performance systems using ATCAs, allowing the reuse of so many of those legacy inhouse-developed client software infrastructures.

INTRODUCTION

The 7-slot Advanced Telecommunications Computing Architecture (ATCA) crate is used for numerous high-performance systems (HPS) at SLAC, such as the bunch charge monitor[1], bunch length monitor[1], beam position monitor[2], low-level radio frequency[3], machine protection system[4], timing system, and a few others. In all of these sub-systems, raw data is acquired, processed, timestamped, and transmitted upstream to a server where it is analyzed and exported to the network through EPICS[5].

SLAC has a set of 4 services, called Acquisition Services, used to organize timestamped data in different ways. Each way has its own use case with its set of requirements from users. Beam Synchronous Acquisition (BSA)[6] is one example of an Acquisition Service. The other ones are Beamline Data (BLD), Beam Synchronous Scalar Service (BSSS), and Beam Synchronous Acquisition Service (BSAS).

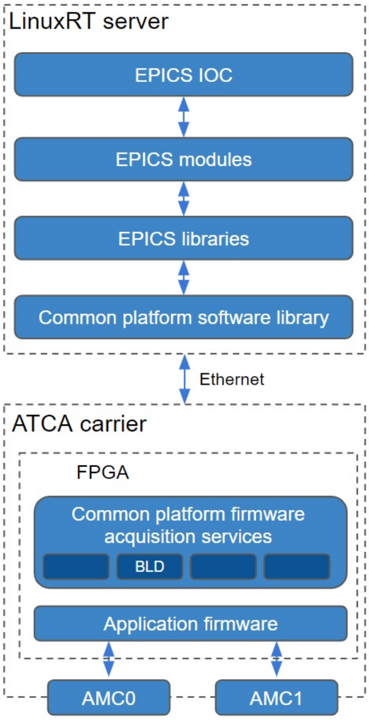


Figure 1: High-performance system SW/HW overview.

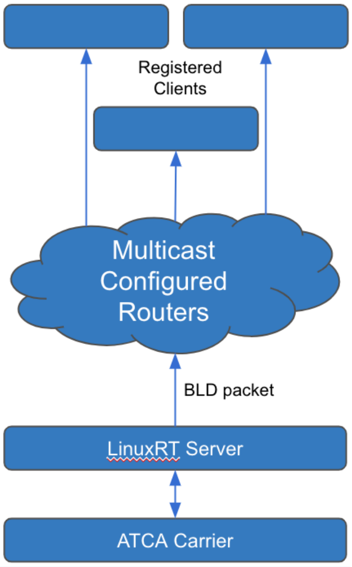


Figure 2: BLD message flow overview.

BLD runs on top of SLAC's common platform software and firmware[7] and permits the forwarding of acquired

[*] Work supported by the U.S. Department of Energy under contract number DE-AC02-76SF00515
[†] dnajjar@slac.stanford.edu

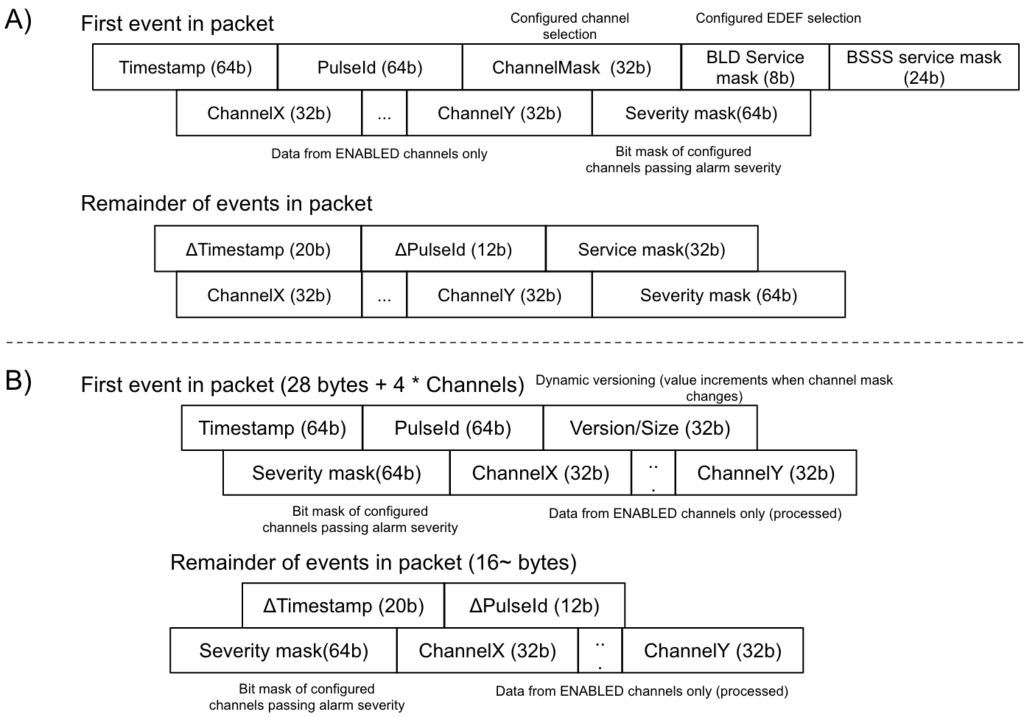

A)
First event in packet
Configured channel selection
Configured EDEF selection
Timestamp (64b)
PulseId (64b)
ChannelMask (32b)
BLD Service mask (8b)
BSSS service mask (24b)
ChannelX (32b)
...
ChannelY (32b)
Severity mask(64b)
Data from ENABLED channels only
Bit mask of configured channels passing alarm severity
Remainder of events in packet
ΔTimestamp (20b)
ΔPulseId (12b)
Service mask(32b)
ChannelX (32b)
...
ChannelY (32b)
Severity mask (64b)
B)
First event in packet (28 bytes + 4 * Channels)
Dynamic versioning (value increments when channel mask changes)
Timestamp (64b)
PulseId (64b)
Version/Size (32b)
Severity mask(64b)
ChannelX (32b)
...
ChannelY (32b)
Bit mask of configured channels passing alarm severity
Data from ENABLED channels only (processed)
Remainder of events in packet (16~ bytes)
ΔTimestamp (20b)
ΔPulseId (12b)
Severity mask(64b)
ChannelX (32b)
...
ChannelY (32b)
Bit mask of configured channels passing alarm severity
Data from ENABLED channels only (processed)

Figure 3: BLD data packet structure from FPGA firmware to software (A) and from software to the multicast network clients (B).

data to configured IP addresses and ports in the format of multicast network packets. BLD has allowed data to be transmitted over multicast packets for over a decade now at SLAC, but always at a maximum rate of 120 Hz. Coping with the needs of the new superconducting accelerator, which runs at a maximum rate of 1 MHz, more severe performance requirements were imposed deeming the previous implementation ineffective. With this in mind, the ATCA-based BLD service was partially accelerated in FPGA, and the software was refactored to accommodate those needs. In this paper, we will discuss the BLD Acquisition Service software package for the new ATCA-based high-performance system.

HPS OVERVIEW

A general overview of the high-performance systems is shown in Fig. 1.

The application firmware, residing in the FPGA, processes data from its input source. One source of input can be an Advanced Mezzanine Card[4]. Once ready to be sent upstream, the processed data is forwarded to the common platform firmware[7] Acquisition Services module. Communication between the Linux server and the ATCA is also established through the standard common platform software. Each acquisition service in firmware communicates to its respective software component (Acquisition Service software package) and transmits the processed data upstream. Once obtained, the Acquisition Service software exposes this data in its specific format. All high-performance systems have the BLD software acquisition service package linked as an EPICS Asyn Driver[8].

BLD EPICS ASYN DRIVER

Adding the BLD EPICS Asyn Driver support package to an IOC is simple and straightforward. The BLD support package works for all high-performance systems that have the BLD firmware module implemented in its FPGAs. The support package initializes the BLD firmware and instantiates the upper software stack providing a set of PVs that access the API to control the hardware and to operate the BLD service.

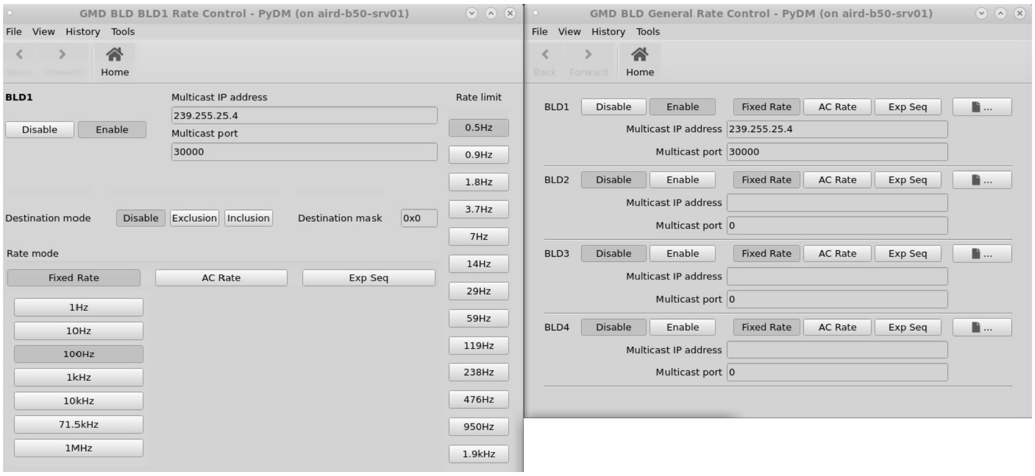
Up to 31 32-bit processed data variables can be sent upstream per event. Up to four BLD rates can be configured simultaneously, each accessible at its configured IP destination and port, with a maximum rate of 1 MHz. The software receives the packet from the firmware, processes the data in the software, identifies which BLD destination it needs to send a broadcast packet to, and transmits it. Users interested in acquiring any of the four BLD rates need to register in the corresponding IP destination at the multicast target switch to receive a copy of the multicast packet on their respective receiver software. Data channels and BLD rates can be disabled and enabled per application. Figure 2 shows a high-level diagram of the data flowing from the ATCA to the registered clients.

Figure 3 shows the BLD packet format from the FPGA to the software and from the software to the multicast clients.

Generic interfaces developed in PyDM[9] are also provided to the user to control the BLD service. The BLD standard interface using PyDM is depicted in Fig. 4.

Content from this work may be used under the terms of the CC BY 4.0 licence (© 2023). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

Content from this work may be used under the terms of the CC BY 4.0 licence (© 2023). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI



BLD1 Rate Control:

Multicast IP address: 239.255.25.4
Multicast port: 30000
Destination mask: 0x0

General Rate Control:

BLD1 Multicast IP address: 239.255.25.4
BLD1 Multicast port: 30000
BLD2 Multicast IP address:
BLD2 Multicast port: 0
BLD3 Multicast IP address:
BLD3 Multicast port: 0
BLD4 Multicast IP address:
BLD4 Multicast port: 0

Figure 4: BLD PyDM interfaces.

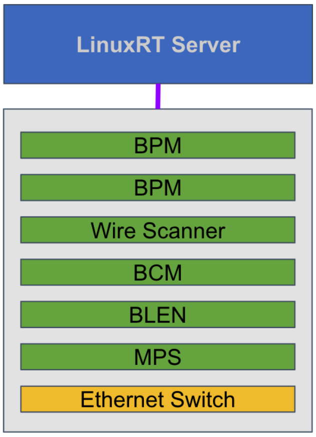


Figure 5: One LinuxRT server connected to one ATCA crate.

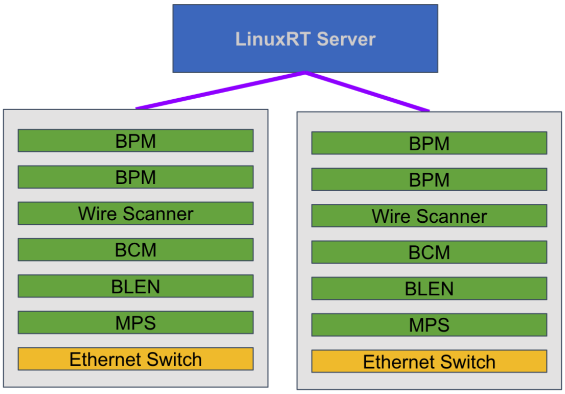


Figure 6: One LinuxRT server connected to two ATCA crates.

PERFORMANCE

It is worth noting that since we have four BLD destinations, for each packet coming from firmware, four packets to four different destinations will be transmitted (given that the BLD rates of all four are identical). In this scenario, the bandwidth of multicasted packets will need to be almost four times that of the data coming from firmware. This may saturate bandwidth easily and has serious consequences if BLD is not tuned carefully. As an example, if 5 32-bit variables were transmitted from firmware to software, and all 4 BLD rates were enabled and configured at 1 MHz, that generates an upstream bandwidth of almost 1 Gbps.

At SLAC we are standardizing that each BLD rate will be filtered with a different destination for the beam. As the beam can't reach two different destinations, this means that no BLD rate will have the same packet being transmitted by the software to the multicast network. In this case, the maximum event rate when summing up all four BLD rates will never exceed 1 MHz if the standard is configured correctly.

Regarding CPU usage, we estimate approximately 20% of the server time being dedicated to packet reception and multicast packet generation, and another 16% spent in the kernel when transmitting the data over UDP. For one single IOC running BLD in the LinuxRT server, this poses no harm. But when three or more IOCs are running BLD at the same time that other IOCs are running, we start to see CPU hogging during stress tests.

SLAC uses two different connection styles between LinuxRT servers and ATCA crates. One server with one ATCA crate (Fig. 5) or one server with two ATCA crates (Fig. 6). At SLAC, ATCA crates fully loaded have 6 carrier boards and we've been running one EPICS IOC per carrier board. So, a server with two ATCA crates can have up to 12 IOCs running.

Figure 7 shows a real case scenario at SLAC with 11 IOCs running: 8 beam position monitors (BPM), 2 machine protection systems (MPS), and one wire scanner, none of them running BLD. The figure shows how the CPU usage increases as each IOC boots one after the other. The rightmost bar on the chart shows the scenario with all 11 IOCs running. With more than 90% of the CPU in use, it is easy to see how turning BLD on can be a problem.

Indeed, in this scenario, once three IOCs had BLD turned on, we observed slowness in threads with lower priorities translating into EPICS process variables (PVs) presenting

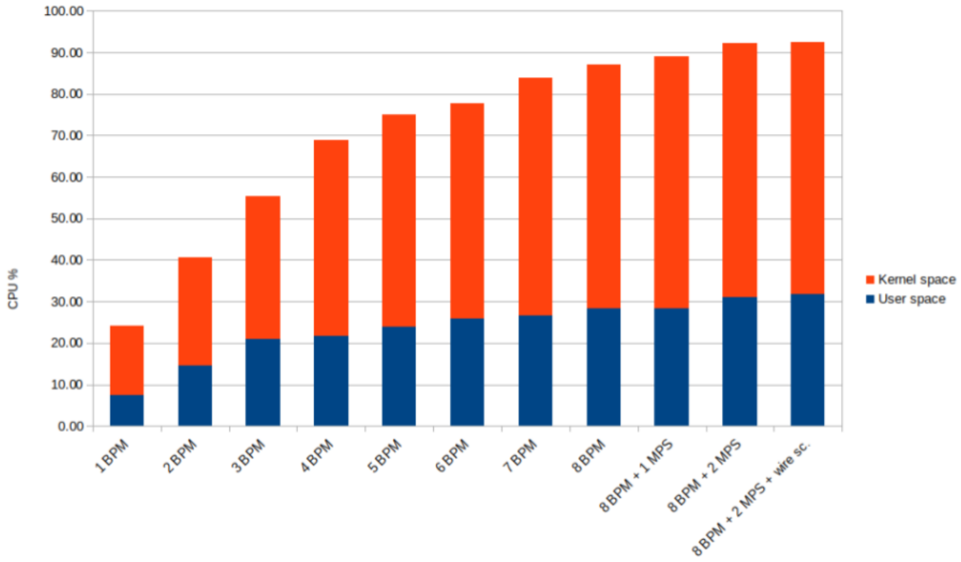


Figure 7: CPU usage increases as IOCs boot one after the other.

wrong numbers. With four IOCs running BLD, hogging started and we observed the following effects at the server:

- Linux *top* command took about one minute to start when it should answer immediately.
- Linux *ifconfig* command was slow to answer.
- In the Linux terminal, auto-complete with Tab took seconds to respond.
- Accessing the EPICS IOC console took around one minute when it should answer immediately.
- *iocManager*, one type of EPICS IOC at SLAC that monitors other systems including other IOCs, was showing the heartbeat status changing: PRESENT - INTERMITTENT - ABSENT, meaning that the IOC was unable to post monitors for seconds.

After a few minutes, one or more IOC processes gave up and ended up closing as the CPU couldn't deliver the network packets from the ATCA in a timely manner, leading the IOC to consider that a network connection was severed.

CONCLUSION

The ATCA-based BLD software implementation for SLAC's LCLS-II timing system has been presented. Up to four BLD rates can be configured simultaneously, each accessible at its configured IP destination and port, with a maximum rate of 1 MHz. Up to 31 32-bit processed data variables can be sent upstream per event. If proper care is not taken when tuning BLD, the upstream network bandwidth may get saturated easily, as well as the CPU usage.

The BLD upgrade has been successfully running since the end of August 2023 in the Gas Monitor Detector (GMD) system.

ACKNOWLEDGEMENTS

The authors would like to thank the Advanced Control Systems department team, namely, Eric Gumtow, Michael Skoufis, and Ernest Williams for their feedback and discussions.

REFERENCES

[1] M. P. Donadio, A. S. Fisher, and L. Sapozhnikov, "Upgrade of the Bunch Length and Bunch Charge Control Systems for the New SLAC Free Electron Laser", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, p. 1186. doi:10.18429/JACoW-ICALEPCS2019-WEPHA044

[2] A. Young *et al.*, "Design of LCLS-II ATCA BPM System", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, pp. 477–479. doi:10.18429/JACoW-IPAC2017-MOPAB149

[3] J. M. D, J. C. Frisch, B. Hong, K. H. Kim, J. J. Olsen, and D. V. Winkle, "Performance of ATCA LLRF System at LCLS", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, pp. 4817–4820. doi:10.18429/JACoW-IPAC2017-THPVA152

[4] J. C. Frisch *et al.*, "A FPGA Based Common Platform for LCLS2 Beam Diagnostics and Controls", in *Proc. IBIC'16*, Barcelona, Spain, Sep. 2016, pp. 650–653. doi:10.18429/JACoW-IBIC2016-WEPG15

[5] EPICS, https://epics-controls.org/

[6] K. H. Kim, S. Allison, T. Straumann, and E. Williams, "Real-Time Performance Improvements and Consideration of Parallel Processing for Beam Synchronous Acquisition (BSA)", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 992–994. doi:10.18429/JACoW-ICALEPCS2015-WEPGF122

[7] T. Straumann *et al.*, "The SLAC Common-Platform Firmware for High-Performance Systems", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 1286–1290. doi:10.18429/JACoW-ICALEPCS2017-THMPL08

[8] asynDriver, https://epics.anl.gov/modules/soft/asyn/R4-38/asynDriver.html

[9] PyDM, https://slaclab.github.io/pydm

Content from this work may be used under the terms of the CC BY 4.0 licence (© 2023). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI