Partial Restart of a Distributed Data Acquisition System

Youichi Igarashi

***Abstract*—Modern detector systems employ a large number of front-end electronics modules (FEEs). These FEEs can occasionally become unstable and may require a restart. In addition, to improve the signal-to-noise ratio, FEEs are sometimes mounted close to the detector. In such cases, the FEEs are placed near the beam line and may malfunction or stop operating because of radiation-induced single-event upsets (SEUs).**

In conventional data acquisition (DAQ), when an FEE malfunctions, the DAQ system is often stopped, the FEE is restarted, and the DAQ system is then restarted. If FEE malfunctions occur frequently, this procedure increases the time during which data acquisition is stopped, thereby affecting the statistical precision of the experiment.

In this study, we implemented a procedure in which the readout of only the malfunctioning FEE is stopped, the FEE is restarted, and the FEE readout is then resumed, without stopping the overall data acquisition. NestDAQ, the distributed data acquisition system currently under development, performs data acquisition through the coordinated operation of many single-function processes.

Among these processes, the TimeFrameBuilder (TFB) collects data from all FEEs and combines them into a single time frame. This process detects FEEs from which data are not received and reports the anomaly to an online database. A separate recovery process, called the Watchdog, receives a notification from the database and controls the Sampler process that reads out the faulty FEE. It stops the readout, restarts the FEE, and controls the reintegration of the FEE into the DAQ stream. This mechanism allows problems with FEEs to be resolved without stopping data acquisition, enabling data taking to continue.

The mechanism was implemented in the COMET CDC readout system with 104 RECBE front-end modules. In a restart test performed during data taking, the failed FEE was automatically detected, restarted through the remote FPGA firmware download system, and reintegrated into the DAQ stream in approximately one minute, while the other FEEs continued data taking. For the expected SEU-induced FEE failure rate in COMET Phase-I, the method eliminates an estimated 12.2 % global dead-time component that would otherwise be introduced by full DAQ restarts.

***Index Terms*—Data acquisition, System recovery**

I. Introduction

Modern detector systems employ a large number of front-end electronics modules (FEEs). These FEEs can occasionally become unstable and may require a restart. In addition, to improve the signal-to-noise ratio, FEEs are sometimes mounted close to the detector. In such cases, the FEEs are placed near the beam line and may malfunction or stop operating because of radiation-induced single-event upsets (SEUs).

Youichi Igarashi is with Instrumentation Technology Development Center, Institute of Particle and Nuclear Studies, High Energy Accelerator Research Organization (KEK), Tsukuba, Japan (e-mail: youichi.igarashi@kek.jp)

The COMET Phase-I experiment [1], which is currently under construction at J-PARC, searches for charged-lepton flavor violation by measuring muon-to-electron conversion events. In the CyDet detector system, which performs the physics measurement, the FEEs are mounted on the main detector, the Cylindrical Drift Chamber (CDC). Therefore, the FEEs can be affected by the beam and can experience SEUs. Fig. 1 shows the CyDet detector system of the COMET experiment and the FEE mounting area.

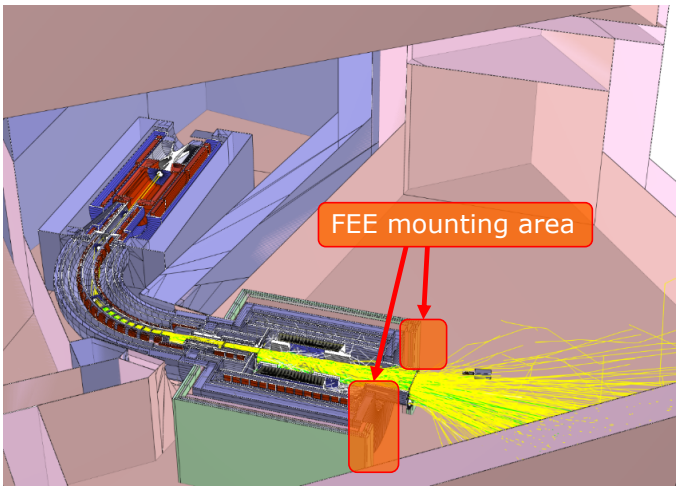

Fig. 1. The mounting area for the FEEs on the COMET detector system (CyDet).

Based on simulations and measurements of the radiation tolerance of the FEEs, FEE failures caused by SEUs are expected to occur once every 7.2 min at the beam intensity of COMET Phase-I. Under such conditions, if the entire data acquisition (DAQ) system is stopped and the FEE is restarted every time an FEE stops, the dead time introduced by the restart procedure cannot be neglected.

On the other hand, in a detector system with a large solid angle, the region traversed by particles in a single event is limited to a small part of the detector. The other parts of the detector are not involved in that event. Therefore, meaningful data can still be acquired even when a small part of the detector acceptance is temporarily missing.

For these reasons, restarting an FEE without stopping the DAQ system can make a significant contribution to improving the efficiency of data acquisition.

Motivated by this, we attempted to implement a partial restart mechanism in the distributed DAQ system currently under development.

The main contributions of this work are as follows:

1) extension of the TFB to handle incomplete time frames;
2) automatic identification of malfunctioning FEEs using database notifications;
3) coordinated restart of an FEE and its Sampler process without stopping the global DAQ run; and
4) demonstration of the mechanism in the COMET CDC readout system.

II. A Distributed Data Acquisition Software Framework: NestDAQ

The development of a framework for data acquisition in a distributed computing environment is currently underway. This framework is being named NestDAQ. [2] This DAQ software operates through the cooperation of many single-function processes. Each DAQ process communicates using ZeroMQ [3], allowing data readout, event building, online triggering, and data recording to be executed in a distributed manner on multiple computers connected via a network. Each DAQ process is managed and controlled through Redis [4], a key-value database. This DAQ software can be used for both streaming DAQ and triggered DAQ.

The framework is characterized by scalability with respect to the data acquisition rate and by a process-connection topology that can be described using simple configuration parameters. A conceptual diagram of the software during DAQ operation is shown in Fig. 2.

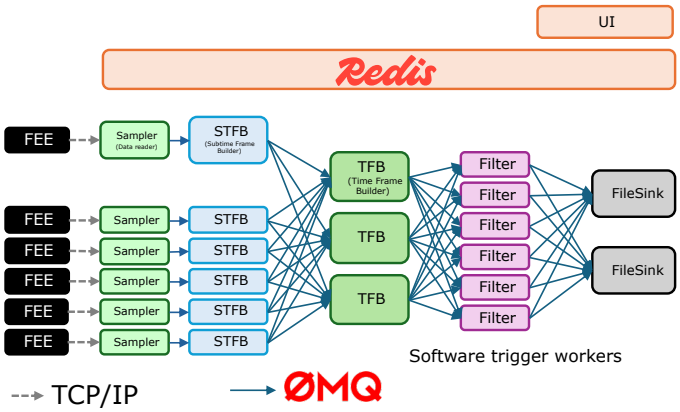

Fig. 2. DAQ processes and their connections. Data is passed from the left process to the right process. All DAQ processes reference the Redis database for control and management.

NestDAQ consists of several types of processes:

- Sampler: Reads out data directly from the FEE.
- SubTimeFrameBuilder (STFB): Extracts sub-time frames from the data received from the Sampler based on specific time intervals.
- TimeFrameBuilder (TFB): Aggregates sub-time frames from all FEEs to construct a complete time frame.
- Online Trigger Processes (Filter): A group of processes that process time frames to extract physically meaningful events in real-time.
- FileSink: Responsible for persistent data storage.

In a trigger-based DAQ configuration, data is transmitted from the FEE for each trigger signal. Consequently, the SubTimeFrameBuilder is not required, and the TimeFrameBuilder works as an event builder.

NestDAQ is currently deployed and utilized in experiments at several facilities, including J-PARC Hadron Hall and RCNP (Research Center for Nuclear Physics) of Osaka University. [5] We have implemented a partial restart mechanism within this framework to improve its robustness.

III. Method for Partial Restart

The TFB buffers data read from FEEs and sends time frames containing data from all FEEs to the downstream DAQ processes. In this operation, if a problem occurs in an FEE and its data are not sent correctly, a complete time frame cannot be built. When a timeout occurs or the buffer capacity limit is exceeded, the incomplete time frame is normally discarded.

The partial restart mechanism is realized by extending the functionality of the TFB and by using the Watchdog process, which monitors the TFB status and restarts the appropriate FEE when necessary.

A. Extension of the TFB Functions

The TFB was extended to handle incomplete time frames. If data from all FEEs have not been collected when the configured timeout or buffer-full condition is reached, the TFB flags the assembled data as an incomplete time frame and sends it to the downstream processes instead of discarding it. During this operation, the global DAQ run continues. The TFB also identifies the FEEs from which data have not arrived and records their IDs, together with the number of missing data segments per unit time, in the database.

B. Watchdog Process

The Watchdog is a process that detects time-frame building failures and restarts FEEs. Redis provides a notification mechanism. When the TFB records a time-frame building failure in the database, Redis notifies the Watchdog of the failure.

After receiving the notification, the Watchdog reads the database and identifies the anomalous FEE. If it confirms that time-frame building has been failing for a configured period of time, it executes the following restart sequence.

1) The state of the DAQ process (Sampler) that reads out the identified FEE is changed from RUN to STOP.
2) A program for restarting the FEE is executed with configuration parameters generated from the FEE ID.
   a) The executed program configures the firmware download path to the malfunctioning FEE and reloads and restarts the FPGA mounted on the FEE using the JTAG protocol.
3) After step 2) is completed, the DAQ process whose state was changed in step 1) is returned to RUN.

Through this sequence, the FEE is restarted, and the Sampler resumes reading data and sending them to the downstream processes. As a result, the overall DAQ system returns to normal operation.

This sequence of operations is shown in Fig. 3.

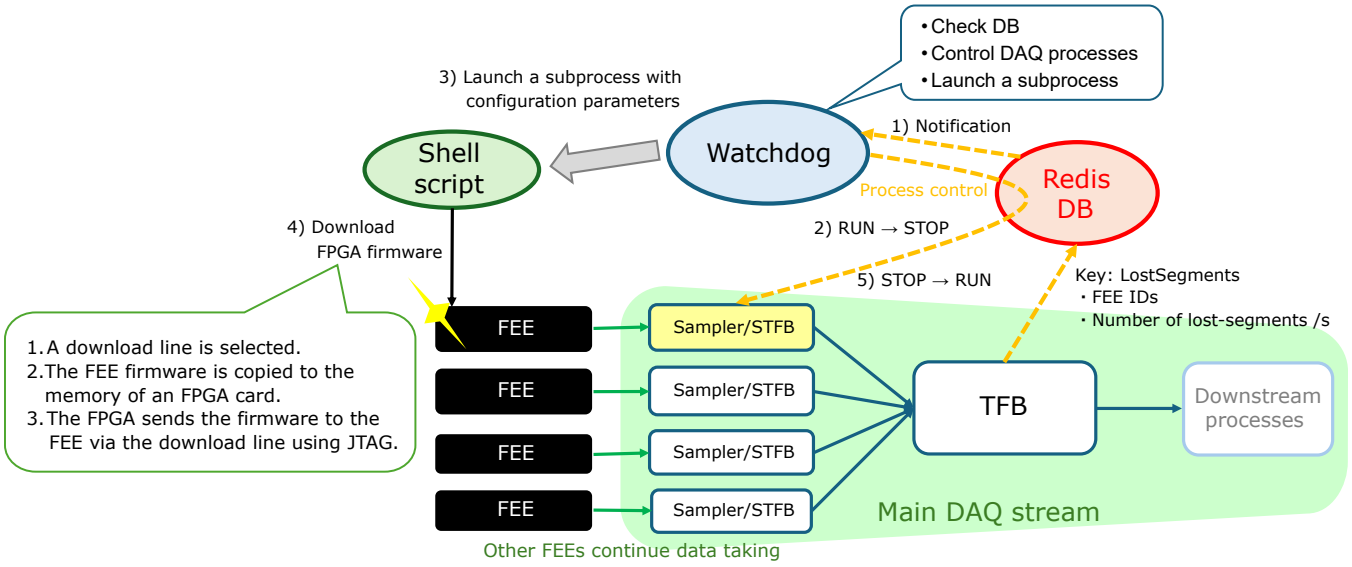

Fig. 3. Partial restart sequence. TFB reports the anomaly to the DB. The Watchdog, referring to the database, restarts the FEE identified by the TFB.

C. State Transition of the Sampler

NestDAQ uses ZeroMQ for inter-process communication. Therefore, explicit connection control is not required for communication between DAQ processes. However, the Sampler is the process that directly reads data from an FEE. The FEEs used in this system are read out through SiTCP [6], a TCP/IP engine implemented in an FPGA. For this reason, the Sampler directly uses a socket to read data from the FEE via TCP/IP.

When an FEE stops abnormally, the process reading out that FEE must close the socket and then open it again. The Sampler state is changed during the FEE restart to reconnect this socket. When the Sampler transitions from the RUN state to the READY state, the callback function 'PostRun()' is called. In this function, the Sampler closes the socket. Similarly, when the Sampler transitions from the READY state to the RUN state, the callback function 'PreRun()' is called. In this function, the Sampler opens the socket.

This operation maintains consistency between the FEE restart and the Sampler readout process, enabling the Sampler to read data normally again. Fig. 4 shows the state transition of the Sampler.

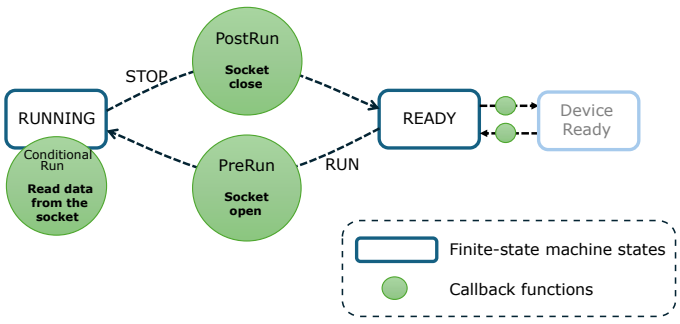

Fig. 4. State transition diagram for the sampler. The sampler opens and closes the socket as it transitions between READY and RUNNING states.

IV. Implementation in COMET CDC

The Central Drift Chamber (CDC) detector used in the COMET experiment is currently undergoing cosmic-ray tests. We introduced the FEE restart mechanism into the readout system of this detector.

The CDC is read out by chamber readout boards called RECBEs, which are equipped with an amplifier-shaper-discriminator (ASD) and an FPGA. These FEEs can be read out via a network. The configuration of the COMET CDC readout system is shown in Fig. 5.

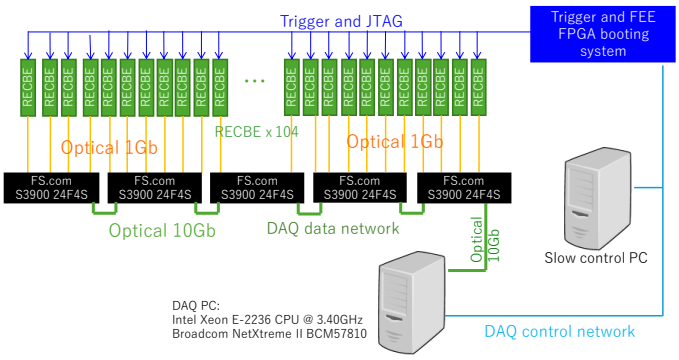

Fig. 5. Setup of the COMET CDC readout system. 104 RECBE(FEE) boards are connected to the DAQ PC via a 10 Gbps network. The Slow control PC and the DAQ PC are connected via another network. The Slow control PC can control the trigger system over the network. The trigger system not only distributes triggers but also features JTAG functionality for booting the FEE's FPGA.

The COMET detector also has a unified mechanism for downloading firmware to the FPGA mounted on each FEE and booting it remotely. This is realized using a JTAG tree. At each branch point of the tree, a demultiplexer is implemented for path selection, enabling firmware to be downloaded and FPGAs to be restarted remotely from the network for a specific FPGA or multiple FPGAs. Fig. 6 shows the remote firmware download mechanism for the CDC detector.

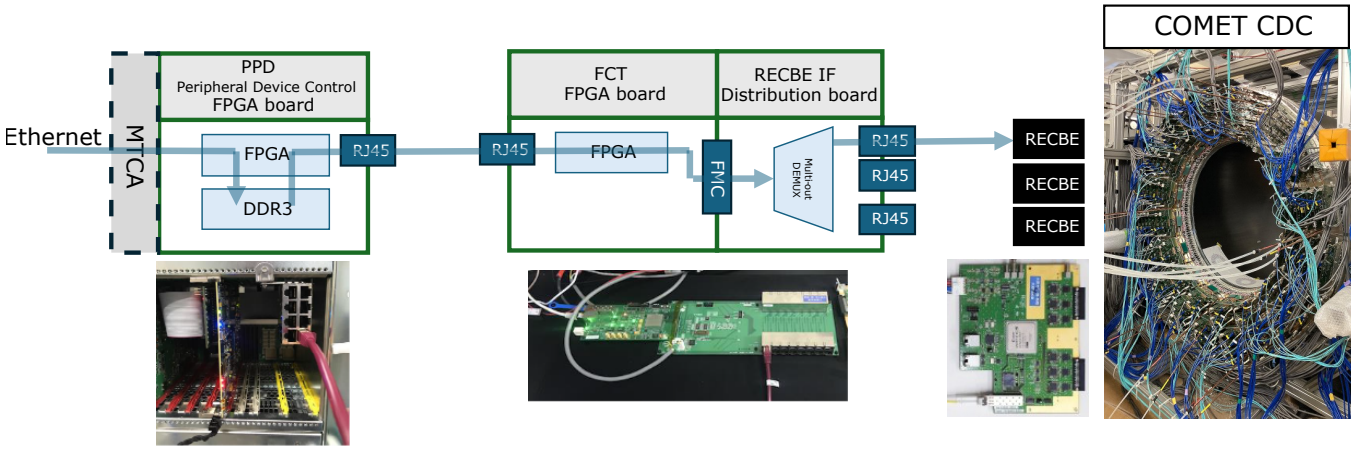

Fig. 6. Mechanism for remote firmware downloads for FEEs for the CDC. Firmware written to the PPD via the network is routed through the FCT and RECBE-IF and ultimately downloaded to a specified FEE.

The FEE restart test was performed while acquiring data from the CDC detector by controlling the registers of a RECBE from another device and forcibly stopping the RECBE.

V. Performance Evaluation

Fig. 7 shows the timeline when an FEE module is restarted. The Watchdog requires 20 to 40 s to confirm that the FEE module has indeed stopped, and then executes the script for re-downloading the firmware to the FEE and restarting it. Firmware re-download and FEE restart require approximately 35 s.

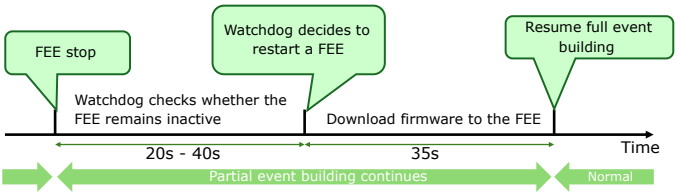

Fig. 7. Timeline of the recovery sequence. With the current settings, when the FEE stops, the Watchdog verifies that it has indeed stopped and then restarts the FPGA. Verifying that it has stopped takes 20-40 seconds, and restarting the FPGA takes 35 seconds.

Fig. 8 shows the time-frame building success rate reported by the TFB during this operation. The TFB calculates the success rate every second and sends the value to the database. In Fig. 8, the TFB success rate becomes 0 % when the FEE module stops. This is because data are not received from the stopped FEE module, and therefore the TFB cannot successfully build time frames. Here, the TFB success rate is defined as the fraction of complete time frames that contain data from all FEEs. Therefore, the success rate becomes 0 % when one FEE stops, even though incomplete time frames containing data from the remaining FEEs continue to be transferred downstream.

When the FEE module restarts, time-frame building succeeds again, and the TFB success rate returns to 100 %. The TFB success rate sometimes becomes 50 % during restart. This behavior is considered to be an artifact of the

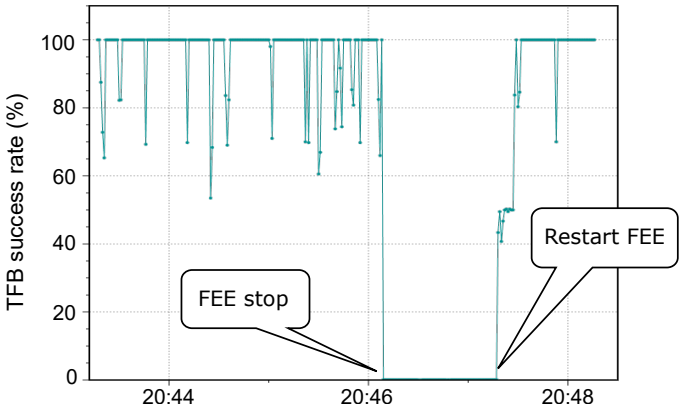

Fig. 8. Time-Frame Building Success Rate. When all FEEs are operating normally, the TFB success rate is 100 %, but it drops to 0 % when a FEE stops. It returns to 100 % when the FEE restarts. Even when the success rate is 0 %, incomplete time frames continue to be transferred downstream.

statistics calculation and buffering and did not affect the restart sequence. The TFB success rate is sometimes not 100 % even during normal operation because of instability in the trigger distribution system. This issue will be addressed in future work.

The restart procedure was tested over 50 times, and all trials succeeded. The tests were performed for multiple RECBE modules while data taking from the other FEEs was continued. These results confirmed that the FEE restart procedure functioned correctly.

VI. Discussion

The time required for firmware download is determined by the clock frequency at which JTAG can operate stably and by the firmware size. With the current cable configuration, it is difficult to shorten this time. The Watchdog accesses the database every 2 s and judges that an FEE module has stopped if data are not received from that FEE module five consecutive times. With this algorithm, it should be possible to determine that an FEE module has stopped within 10 s. However, at present, the determination takes 20 to 40 s. The

present delay was not caused by the restart sequence itself, but by ambiguity in the failure source during the test. Because of an instability in the trigger distribution system, missing-data information was sometimes recorded for FEEs that did not receive triggers, which made it difficult for the Watchdog to identify the actual stopped FEE immediately. A more robust classification of missing-data sources is expected to reduce the decision time toward the nominal value of 10 s.

By introducing this mechanism, the effective live time of data acquisition is expected to improve compared with the conventional method in which the run is stopped and the FEE module is restarted every time an FEE module stops.

Assuming that the mean time between failures (MTBF) due to SEUs for the total of 104 FEE modules is 7.2 min, and that the time from stopping the run to restarting it is 1 min, the global dead-time fraction introduced by a full DAQ restart can be estimated as

$$D_{\mathrm{full}} = \frac{T_{\mathrm{restart}}}{(T_{\mathrm{MTBF}} + T_{\mathrm{restart}})},$$

where T_{MTBF} is the mean time between FEE failures and T_{restart} is the time required for a full DAQ restart. Using $T_{\mathrm{MTBF}} = 7.2$ min and $T_{\mathrm{restart}} = 1$ min, D_{full} is $1/(7.2{+}1) = 12.2$ %.

This estimate evaluates only the global DAQ dead-time component and does not include the effect of the temporary local acceptance loss during the FEE restart. In the proposed method, this global dead time is replaced by a temporary local loss of detector acceptance corresponding to the restarted FEE.

The proposed method does not depend on the detailed structure of the COMET CDC readout electronics, except for the experiment-specific firmware download procedure. The essential requirements are that each FEE is read out by an independent Sampler process, that missing data fragments can be identified by the time-frame-building process, and that the corresponding FEE can be restarted remotely. Therefore, the same approach can be applied to other distributed readout systems in which local FEE failures should not stop the entire DAQ system. Table I compares the proposed method with a conventional full DAQ restart.

TABLE I
COMPARISON BETWEEN FULL DAQ RESTART AND PARTIAL FEE RESTART

Item	Full DAQ restart	Partial FEE restart
Data acquisition	Interrupted	Continued
Global DAQ state	STOP	RUN
Other FEEs	Stopped	Continue data taking
Data loss during recovery	All FEEs	Affected FEE only
Operator intervention	Manual or semi-automated	Automatic
Recovery impact	Global dead time	Local acceptance loss

VII. Summary

We implemented a mechanism for restarting an FEE without stopping the global DAQ run in the distributed DAQ software framework NestDAQ. The TFB was extended to handle incomplete time frames and to report missing FEE information to the database. In addition, a Watchdog process was developed to identify the malfunctioning FEE, stop the corresponding Sampler, restart the FEE, and reintegrate it into the DAQ stream.

The mechanism was implemented in the COMET CDC readout system with 104 RECBE front-end modules. In a restart test during data taking, the failed FEE was automatically restarted and the system returned to complete time-frame building in approximately one minute, while the other FEEs continued data taking. For the expected SEU-induced FEE failure rate in COMET Phase-I, the method can eliminate the global dead-time component that would otherwise be introduced by full DAQ restarts, replacing it with a temporary local acceptance loss.

Acknowledgment

This data acquisition system is being developed under the SPADI Alliance [7]. We would like to thank the members of the COMET experiment group, especially the members of the COMET CDC group, for their cooperation in implementing the system in an actual detector.

References

[1] COMET Collaboration, "Experimental Proposal for Phase-I of the COMET Experiment at J-PARC," 2012. [Online]. Available: http://j-parc.jp/researcher/Hadron/en/pac_1207/pdf/E21_2012-10.pdf Accessed: July 01, 2026.

[2] T. N. Takahashi, Y. Igarashi, R. Honda, H. Sendai, "Streaming DAQ Software Prototype at the J-PARC Hadron Experimental Facility," IEEE Trans. Nucl. Sci., vol. 70, no. 6, pp. 922-927, DOI: 10.1109/TNS.2023.3262061

[3] "ZeroMQ," [Online]. Available: https://zeromq.org/. Accessed: July 01, 2026.

[4] "Redis," [Online]. Available: https://redis.io/. Accessed: July 01, 2026.

[5] Y. Igarashi *et al.*, "Implementations of Streaming DAQ on Actual Detector Systems," IEEE Trans. Nucl. Sci., vol. 72, no. 3, pp. 421-428, 2025, DOI: 10.1109/TNS.2024.3506783

[6] T. Uchida, " Hardware-Based TCP Processor for Gigabit Ethernet", IEEE Trans. Nucl. Sci., vol. 55, no. 3, pp. 1631-1637, 2008. DOI: 10.1109/TNS.2008.920264

[7] "SPADI Alliance," [Online]. Available: https://spadi-alliance.rcnp.osaka-u.ac.jp/. Accessed: July 01, 2026.