Progress in Data Transmission and Storage for Long Pulse Data Acquisition System

Yu Gu¹, Mengqi Fan¹, Chuanqian Tang¹, Guojing Zhang¹, Xiaodan Zhang¹*

¹Department of Computer Technology and Applications, Qinghai University, Xining, Qinghai, 810016, China
*Corresponding author’s e-mail: xdzhang@qhu.edu.cn

ABSTRACT. Since the study in the field of fusion has gradually developed toward the long-pulse experiment mode, long-pulse data has gradually become one of the main data types for pulsed experiments in the field of fusion. For long-pulse data, which is a kind of pulse-type data, it will be more difficult to transmit and store than short-pulse data because of its significant characteristics. In addition, in the design of data acquisition and control system (DACS) in fusion field, Experimental Physics and Industrial Control System (EPICS) has now gradually become the main framework of experimental control system to meet the diversity of devices and complexity of subsystems in large experimental systems. However, due to the limitation of EPICS, its effectiveness in handling data transmission and storage under high speed data acquisition is not satisfactory. To solve the data transmission and storage under high-speed sampling, this paper proposes a data transmission and storage solution based on TCP/IP protocol and MDSplus database, which is designed with the concept of segmentation, i.e., data generated from experiments longer than 100 seconds are uploaded and stored in a segmented form. Currently, this system has been tested and applied, and the test result shows that the solution is feasible and the overall test system operates stably and reliably.

1. Introduction
At present, long pulse experiments have become an important study direction in the field of fusion. Through long pulse experiments, experimenters can more accurately analyse the operating state of each subsystem and the behaviour of large experimental devices during the experiments, which will help to study the effects of different experimental parameters on the experimental results and further upgrade the experimental devices. Compared with short-pulse experiments, the duration of long-pulse experiments usually exceeds 100s, so the experimenter usually needs to analyse the data only after the experiments are finished. Therefore, it is necessary to explore a method to synchronize the transmission, storage and visualization of long pulse data. With the study development of the experimental system, the control system of large fusion hosts is now usually developed using EPICS[1], which is first applied to the KSTAR system in Korea [2], and then the ITER organization and Huazhong University of Science and Technology have adopted EPICS to improve their own systems[3-4]. Compared to some data acquisition frameworks, EPICS provides input output controller (IOC) and channel access (CA) protocol, making it possible to support most current bus architectures on the one hand, and on the other hand, those control systems developed based on EPICS can precisely control subsystems through reliable CA protocol. However, nothing is perfect, the characteristics of EPICS make it ineffective in terms of data storage and data transmission for high speed data acquisition. In fact, EPICS does not provide reliable data retrieval and storage, so for data
storage processing, an additional database is usually required. For data transmission, EPICS can achieve data transmission at low acquisition rate through channel archiver [5], which creates a predefined buffer and archives the collected data in real time according to the scanning rate of EPICS. However, when the data acquisition rate is higher than 1kHz, EPICS will have a certain delay in scanning the data. Therefore, based on the above data transmission mechanism, it is easy to conclude that it is difficult to achieve better data transmission through channel archiver, which means that the system designed based on EPICS needs to consider data transmission at high speed data acquisition to avoid data loss.

Inspired by the ITER CODAC system, the current DACS has been upgraded based on the EPICS framework [6], and the architecture of it is shown in Figure 1. DACS is composed of three layers, namely the central control system, the server control system, and the sub-control system. The central control system is the human-machine interface (HMI), which includes both remote and local HMIs; the server control layer is mainly composed of a data server and a network server, the data server is used for long-term storage for experimental data, while the network server is used to monitor the data server and provide remote services for users; the sub-control system runs on an industrial personal computer (IPC), which can be composed of multiple IPCs to form a workgroup. The overall workflow of DACS can be briefly described as follows: the experimenter issues a trigger command from the HMI, the relevant configuration data will be obtained by the sub-control system, and at the same time, the sub-control system will start the data acquisition task, and the collected data will be asynchronously uploaded to the data server for long-term storage and visualization on the HMI. This paper focuses on the design of the DACS for data transmission and data storage in the case of high-speed sampling.

![Figure 1. The architecture of DACS](image)

2. **The overall design of data transmission and storage**

As mentioned above, EPICS is deficient in data transmission and storage. Therefore, MDSplus [7] is used for data storage. MDSplus database is a software that integrates data storage, data processing, and data visualization which is specially designed for long pulse experiments by MIT. It has been widely used for long pulse data processing and storage due to its easy operation and user-friendly interface. As for data transmission, the data transmission process here refers to the real-time transmission of the experimental data from the IOC to the data server for long-term storage. As mentioned earlier, MDSplus provides channel archiver to archive data from the EPICS IOC to MDSplus. However, limited by the working mechanism of the channel archiver, this data transmission method is not suitable for high speed data acquisition. In view of the fact that data transmission under low speed data
acquisition can be completed by channel archiver, this paper only discusses data transmission under high speed data acquisition. The collected data is finally transferred to MDSplus in the data server for long-term storage for later analysis and utilization by the researchers. The process of data transmission and storage under high speed data acquisition is shown in Figure 2.

In the design and implementation about the scheme, Data acquisition tasks at high sampling speeds are performed by IPC and then the collected data will be scanned by EPICS IOC and read in the process variable (PV). In order to reduce the network transmission load and save storage space, the raw experimental data will be compressed by a data compression algorithm. After that, the compressed data will be integrated into a defined and standardized data structure, which is then transmitted to the data server via TCP/IP based data transmission structure [8]. The packet will then be parsed and the data will be stored in MDSplus using data segmentation techniques. When the experimenter needs to access the experimental data, it can be obtained based on the specific sampling time and channel number of a particular experiment. Finally, the waveform of the raw data will be visualized on the HMI.

It is worth noting that the storage format needs to be specified in advance. Data obtained by different data acquisition channels in the same experiment number will be stored in different data files to prevent data confusion; while experimental data from different experiment numbers but under the same channel should be stored by experiment number to prevent data overwriting. In order to easily display data waveforms to the user, the data segmentation requirements must be met in a specific data format. The requirements of this system under high-speed sampling can be summarized as follows.

- Implementation of communication between the IPC and the data server for data transmission and reception at high speed data acquisition.
- Correct parsing of the data structure to facilitate access to the original data and prevent data loss.
- Long pulse data segmentation processing.
- Experimental data segmentation storage.
- Accessing segmented data from the MDSplus database according to pre-set parameters.
- A clear, simple and user-friendly data visualization interface.

3. The detailed design and implementation

3.1. High speed data acquisition
Given that the overall framework of EPICS is implemented through C and C++, the data acquisition program is also implemented using the C programming language in order to maximize compatibility
between the data acquisition program and EPICS. In addition, since EPICS provides a variety of PVs for implementing different functions, we establish a generic device model based on 'ai (analog input)', 'bo (binary output)', and 'calc (calculation)' in order to enhance the compatibility of the system with different DAQ cards. The 'ai' is used to read the collected data, the 'bo' is used to control the experimental devices, and the 'calc' is used to monitor the status of those devices. This device model unifies the abstract programming standard of experimental devices, especially for DAQ cards, so it provides great convenience for the subsequent maintenance, update and expansion of DACS. Before the data acquisition task starts, the experimenter needs to configure the experimental devices in detail through HMI, such as the number of acquisition channels, the selection of voltage range, the setting of acquisition frequency, and the acquisition time. At the same time, the program will retrieve the specified directory to determine the current experiment number to prevent the new experiment data from overwriting the original experiment data. When the experimental system is on standby, the IPC will be in a waiting state, ready to receive the 'START' command from the central control system, after which the data acquisition task will be started with the obtained configuration. Considering the need for real-time access in long pulse experiments, the steps of data acquisition, data transmission and data segment storage will be performed synchronously. Through the high speed data acquisition program, the data collected by the DAQ devices will be continuously scanned into the EPICS IOC, and the next step is to transfer and store the collected data for further processing.

In this study, PCI-9112 DAQ card is used on the IPC for data acquisition [9]. The card has 12-bit sampling accuracy and 16 channels, the highest sampling rate can reach 110 kHz for each independent channel. In addition, the minimum hardware and software environment required for the overall system operation should meet the following requirements: CPU performance of 2.0 GHz or above; memory performance of 4 GB or above; hard disk storage capacity of at least 500 GB. The software platform is a C/C++ programming environment based on the Linux operating system. Moreover, MDSplus database and Linux shell programming language are required for the successful implementation of the system.

3.2. Data transmission at high speed sampling

Once data acquisition is finished, one thing that needs to be considered is data transmission. Data acquisition at high speed sampling temporarily stores the collected data in a file named after the channel number that is used for data acquisition in this experiment. In order to reduce network transmission load and save data storage space, this data file will be compressed at this time using a data compression algorithm. The LZO (Lempel-Ziv-Oberhumer) data compression algorithm [10] is chosen for data compression after considering the support for data segmentation and the rate of compression and decompression and space size. After testing, the size of original data file processed by LZO can be reduced by 30%.

Lossless transmission and reception of raw data is crucial, which is why we adopt the TCP/IP protocol to achieve reliable data transmission under high-speed sampling. The data transmission structure based on TCP/IP protocol is shown in Figure 3. As can be seen from the figure, the data structure contains three parts, which are TCP header, data header and compressed data file. The TCP header includes the IP address of the IPC host, the IP address of the data server, the size of this segment of the data structure and CRC checksum. In this design, the IP of IPC is used to mark the source address of data, and the IP of the data server is used to indicate the address to which this data will be sent, and finally, the CRC is used to verify the correctness of the packet after transmission. The data header mainly includes some configuration information about the acquisition of this data, including the acquisition channel number, signal name, sampling frequency, sampling duration, sampling accuracy and voltage range. This information is used to accurately parse the data structure in the data parsing phase and store the data in the correct node in the MDSplus tree. And the last part is the compressed data file. After comparison, the collected data can be transmitted correctly in this way.
3.3. MDSplus-based data segmentation storage

Now, the collected data has been transmitted to the data server through the above data structure. As mentioned before, the collected data needs to be compressed by LZO, and the original data file will be changed to a lzo file after being compressed. Since MDSplus does not support LZO files for storage, in order to ensure that the data files can be stored successfully, it is necessary to perform decompression operation after the data has been transmitted to the data server. As for decompression, LZO's library provides decompression function that can be called directly at the time of use. After decompression, the compressed file will be converted to a txt file and the txt file is the one that will be stored in MDSplus.

The next step is data segmentation and storing the segmented data into the MDSplus. MDSplus stores all experimental data with a tree structure [11]. On the one hand, it unifies the data storage format; on the other hand, it can clearly express the logical relationship between tree nodes. There are two types of trees in the MDSplus: the model tree and the pulse tree. The model tree mainly stores the structure information of the tree, such as the node name and other specific tree structures. The pulse tree will be built after the experimental data was stored in the model tree. After this, if the structure of the model tree is changed again afterwards, the logical structure of the created pulse tree will not change again [12]. In other words, the change of the logical structure of the model tree does not affect the structure of the pulse tree. The relationship between them is shown in Figure 4.

![Figure 3. The data transmission structure based on TCP/IP Protocol](image)

![Figure 4. The relationship between the model tree and pulse tree](image)

3.3.1 MDSplus Model Tree and pulse tree. Although the model tree specifies the logical structure of the MDSplus tree, no data is actually stored in it. The overall framework of model tree can be built using the TCL language, and then the jTraverser provided by MDSplus can be used to help optimize
the structure of the model tree, which can save a lot of development time. In this paper, in order to better present the data storage structure of the MDSplus after data transmission, a four-level structure of the MDSplus tree was created in the test server, consisting of the experiment number, device number, channel number and data. The final MDSplus tree is shown in Figure 5.

In this MDSplus tree, ‘EXPNUM X’ represents the experiment number, ‘DEVICE X’ represents the device number, ‘CH X’ represents the channel number and the ‘DATA X’ represents the segmented data file. As mentioned before, the DAQ card has 16 data acquisition channels, so, there are 16 one-to-one corresponding subtree nodes named after the channel number of the DAQ card in this tree structure. In this design, the model tree of this MDSplus consists of four branches, including parameter, timing, raw and process. The parameter branch stores some constant data, such as amplification, data type, etc. The timing branch is used to store the trigger time, data acquisition time and the frequency of this experiment. The raw branch stores the raw experimental data. The processing branch stores the corresponding signal information of the transmitted packets after processing. As mentioned before, the pulse tree will be built after store the data into the model tree. Therefore, if the experiment starts, the data server needs to stay connected to each IPC all the time when the data server receives a data packet from the IPC, the server program will first parse the packet and write the data to the corresponding MDSplus tree node according to the parsed result. In fact, the parsed data is saved in three files according to the file type in the pulse tree, including tree file, data file and characteristic file. The tree file stores information about the structure of the MDSplus tree; the data file stores the real data. In theory, all data from a long pulse experiment will be written to the node of the pulse tree, while the actual data will only be stored in the data file of the tree node, and some information about this experiment, such as the offset address, will be stored in the access log of the node. MDSplus only maps the "*.tree" file to memory [13], and as for the actual data, it is stored in the hard disk, in other words, the data and the tree information are stored separately, which has the advantage of greatly increasing the data transfer speed.

3.3.2 Data segmentation storage and accessing. In the specific MDSplus data storage format, segmented storage is used to store data. The core design concept of segmented storage is the "data segment", by which MDSplus can achieve continuous data reads and writes. The key technology of
data segmentation is the segmented reading and segmented writing of the collected data. Segmented write means that each segment data is appended to the data node, instead of overwriting the original data, as storage space allows; segmented read means that each segment can be read from MDSplus in a separate form, without being read only with the overall data. By reading and writing in segments, the collected data can be stored in MDSplus in the form of segments without waiting for the end of the experiment. In addition, the experimenter can read the stored experimental data in time during the experiment. The principle of this technique is that when retrieving data from segmented records, MDSplus will select the corresponding segment based on the timestamp, then read all segments that match the retrieval time and join these segments to finally construct the final result that satisfies the retrieval criteria and return it to the user. Through data segmentation storage, the experimenter can read data from MDSplus that is in the experimental period but has been collected and stored. The associated program can then visualize the target data on HMI. The process of accessing data from MDSplus is shown in Figure 6.

![Figure 6. The process of accessing data from MDSplus](image_url)

4. Testing
In order to verify the integrity of the experimental data and the reliability of the segmented storage after processing by this system at high speed sampling, after the segmented experimental data is transmitted and stored in MDSplus, NBWave [14] is used for accessing the same experimental data in segment and in whole respectively, and the segmented data is compared with the original data to check whether the data is correctly stored in MDSplus. The test experiment is a long pulse experiment of up to 100 seconds with a square wave and 100 kHz data sampling frequency. In this test, 10 seconds is chosen as the segmentation basis. Due to the high speed data acquisition, for better display of the comparison results, a segment of data is taken here, whose time length is 10s, and then it is divided into 10 segments on the basis of 1 second. The raw data waveform for 10 seconds under this configuration is shown in Figure 7. Then the 2nd to 3rd seconds data stored in MDSplus is randomly selected and then visualized in the other data visualization window and compared with the original data waveform. The result of comparison between them is shown in Figure 8. In this figure, above the middle red line is the original data waveform, and below is the data of the 2nd to the 3th seconds. It can be seen that the upper part and the lower part are exactly the same in terms of the y-axis value that
corresponds to any value of the x-axis. According to the comparative analysis of the waveform chart, the results are consistent with the expectations.

Figure 7. The raw data waveform for 10 seconds

Figure 8. Comparison results between the data stored using segmentation and the original data.

5. Conclusion
This paper presents the progress of data transmission and storage for long pulse data acquisition system. It proposes a scheme to cope with high speed data acquisition based on EPICS and designs a reasonable data transmission structure based on TCP/IP protocol, and also proposes a segmented storage scheme based on MDSplus for the data storage in long-pulse experimental mode. It integrates many tools, including EPICS system, MDSplus database, LZO data compression algorithm, TCP/IP network transmission protocol and data segmentation processing technology. The application of the program makes it possible to access real-time data for high speed data acquisition in long-pulse experimental mode. After testing, the results show that the method is effective and fully meets the requirements of practical applications.
Acknowledgments
The authors gratefully acknowledge the supports of the open project of state key laboratory of plateau ecology and agriculture, Qinghai university (2021-ZZ-05), the experimental education teaching research project of Qinghai university under No.SYJG202012, the scientific research cooperation project (IPP-ZC-19071806).

References
[1] Experimental Physics and Industrial Control System Website. Available from: http://www.aps.anl.gov/epics/
[2] Kwon M, Choi IS, Choi JW, et al. The control system of KSTAR. Fusion Engineering and Design. 2004;71(1).
[3] Lister JB, Farthing JW, Greenwald M, et al. The ITER CODAC conceptual design. Fusion Engineering and Design. 2007;82(5).
[4] W. Zheng et al, “The J-TEXT CODAC system design and implementation,” Fusion Engineering and Design, 89, 5, (2014); 10.1016/j.fusengdes.2014.03.048.
[5] MDSplus Channel Archiver for EPICS. Home Page. Available from: https://www.mdsplus.org/index.php/Documentation:Users:EPICS.
[6] Y Gu, X Zhang, Y Zhao, et al., The redesign and implementation of data acquisition and control system for EAST NBI[J], Fusion Eng. Des. 164 (10) (2021), 112219.
[7] MDSplus Home Page. Available from: http://www.mdsplus.org/.
[8] X. Xie. Computer Networks. Publishing House of Electronics Industry, Beijing. (2017).
[9] ADLINK Official website [Online]. Available: https://www.adlinktech.com/Products/Data_Acquisition/Multi-FunctionDAQ/PCI-9112?lang=zh-cn, Accessed on: may. 21, 2020.
[10] LZO [Online] http://www.oberhumer.com/opensource/lzo/lzofaq.php.
[11] D. T. Lin, B. J. Xiao, and Q. P. Yuan, “MDSplus Data Storage Application,” Microcomputer Information, 2010, 26(34): 111-113.
[12] T. Fredian, J. Stillerman, G. Manduchi. “MDSplus extensions for long pulse experiments”. Fusion Engineering and Design,2007,83(2).
[13] W. J. Huang, H. Z. Wang, F. Yang, S. Li, and B. J. Xiao, “MDSplus Real-Time Data Acquisition System,” Measurement and Control Technology, 2014, 33(02): 16-19.
[14] L.J. Xiao, J.R. Luo, P. Sheng, Z.M. Liu, The design and implementation of data publishing system for NBI [J], Microcomputer Inf. 27 (05) (2011), 123-125+179. (In Chinese).