MADOCA II INTERFACE FOR LabVIEW

Y. Furukawa, T. Fujita, M. Ishii, T. Matsumoto
SPring-8/JASRI, Kouto, Sayo-cho Hyogo, 679-5198, Japan.

Abstract

LabVIEW is widely used for experimental station control in SPring-8. LabVIEW is also partially used for accelerator control, while most software for the SPring-8 accelerator and beamline control are built on Message And Database Oriented Control Architecture (MADOCA) control framework. As synchrotron radiation experiments advance, there is a requirement for complex data exchange between the MADOCA and LabVIEW control systems which has not been realized. We have developed a next-generation MADOCA called MADOCA II, as reported in this ICALEPCS (T.Matsumoto et. al.). We ported the MADOCA II framework to Windows and we developed a MADOCA II interface for LabVIEW. Using the interface, variable length data can be exchanged between MADOCA and LabVIEW-based applications. As a first application, we developed a readout system for an electron beam position monitor with NI's PCI-5922 digitizers. A client application sends a message to a remote LabVIEW based digitizer readout application via the MADOCA II middle-ware and the readout system sends back waveform data to the client. We plan to apply the interface to various accelerator and synchrotron radiation experiment controls.

INTRODUCTION

The Message And Database Oriented Control Architecture (MADOCA) [1] was developed for the SPring-8 storage ring and beamline control system and it has extended its coverage area to the injector and booster synchrotron [2] of the SPring-8, SACLA[3] accelerator, etc. MADOCA has also been applied to the experimental station controls in SACLA[4].

MADOCA was designed as a client/server model to be applied to a distributed control system such as a large-scale accelerator. MADOCA client programs issue simple text-based messages to remote I/O controlling computers and receive result messages.

With advances in accelerator technology and experiments, there are many demands which cannot be handled by the MADOCA control system such as variable-length data transfer and Windows-based control systems.

MADOCA II, as reported at this conference [5], has been developed to solve the above problems. MADOCA II has many advantages such as 1) MADOCA II can handle variable-length data such as images and waveforms, and 2) MADOCA II can run in the Windows environment.

LabVIEW is a powerful tool for developing control and data acquisition programs in Windows. To integrate a LabVIEW-based program into a MADOCA–II-based control system, a MADOCA II–LabVIEW interface was developed.

As the first application of the MADOCA II–LabVIEW interface, we built a waveform readout system based on the NI's PCI-5922 digitizer for the SPring-8 beam position monitors(BPMs).

MADOCA II – LABVIEW INTERFACE

The MADOCA II protocol uses ZeroMQ [6] and MessagePack [7]. The MADOCA II client program packs a text-based message formatted as “S/V/O/C” and related information into a packed binary using the MessagePack library. Then the program issues it to the “message server” (MS) which is a messaging middle-ware of the MADOCA II framework, using the ZeroMQ library.

The parts of the message format are as follows: “S” denotes the program, and the framework automatically defines it. “V” denotes the action of command; mainly “put” or “get” is used, and supplementary “ask” and “show” are also used. “O” is an “object name” which identifies the target of the message, and “C” is an action parameter.

Server programs which control devices register an object list to the MS running on the same computer. The MSs exchange their contained object lists. The MS decides the destination of a message using the “O” (object) part of the message. and issues the message to one of the server program or another MS running on a remote computer.

The MADOCA II–LabVIEW interface is written using LabVIEW-zmq binding [8] and MessagePack for LabVIEW [9] as shown in Fig. 1. The MADOCA II–LabVIEW interface communicates with MS via LabVIEW-zmq binding. MADOCA II–LabVIEW receives a message from the MS and unpacks it using MessagePack for LabVIEW.

An application block in Fig. 1 receives an unpacked message from the MADOCA II–LabVIEW interface, interprets it, executes it and sends back the execution result to the interface. Then the interface packs results using MessagePack for LabVIEW and sends the result message to the MS using LabVIEW-zmq binding.

In the initialization phase, the MADOCA II–LabVIEW interface sends an object list, which is described in a configuration file, to the MS, so the MS can transfer messages from clients to the interface.

Note that the entire MADOCA II–LabVIEW interface was written in LabVIEW, no other language or helper library was required except for LabVIEW-zmq binding and MessagePack for LabVIEW.

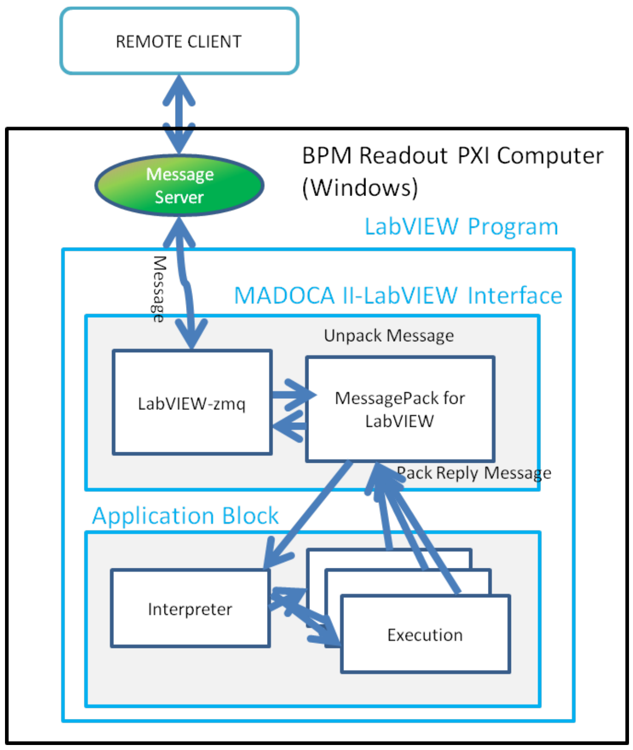

Figure 1: The MADOCA II–LabVIEW Interface. A message from a client via the message server (MS) is received by LabVIEW-zmq and unpacked by MessagePack for LabVIEW. After execution in the application block, the result message and data are packed by the MessagePack for LabVIEW and sent back to the client by LabVIEW-zmq.

BEAM POSITION MONITOR AND WAVEFORM READOUT

An outline of the system is shown in Fig. 2. There are 2 BPMs installed at cell 15 and cell 27 in SPring-8. Analogue blocks amplify signals from electrodes and convert them to horizontal and vertical position signals. The position signals are converted to digital data by a PXI-5922 ADC with a 50k/s sampling rate and 24bit resolution.

As shown in Fig. 2, the BPM readout software consists of two blocks. The first block is to readout BPM data continuously and the other block is to send back BPM data (waveform) as a request from a remote client program using the MADOCA II–LabVIEW interface.

The BPM data is read out continuously by the readout block just after execution of the BPM read out program. After receiving a “start” message from a client program via the MADOCA II middle-ware, the readout block sends decimated data at 5ks/s, 500s/s and 50 s/s to MADOCA II–LabVIEW interface block using LabVIEW’s data queue.

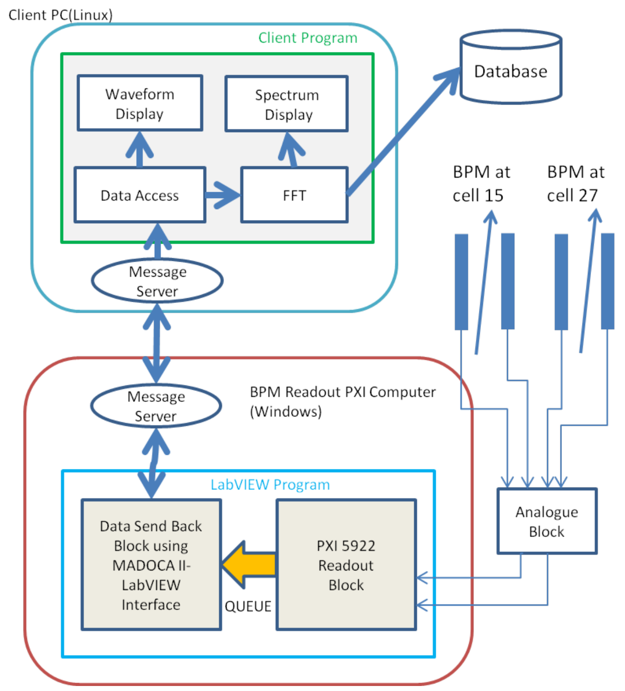

Figure 2 : Structure of the BPM readout system. The BPM signals are fed to the analogue block and converted to position signals. The position signals are read out by PXI 5922 ADC and transferred to the client program using MADOCA II–LabVIEW interface.

The MADOCA II–LabVIEW interface sends a reply message with a packed floating expression data array with a size of 5003, 503 or 53 for every one-second waveform. The additional 3 data contain start time, end time and time resolution.

The client program running on the LINUX (SUSE Linux Enterprise 11.0) accesses waveform data once every second and displays waveforms on the graphical user interface panel as shown in Fig 3. The client program also displays the frequency–domain spectrum by a fast Fourier transform and stores the waveforms in the relational data base (RDB) system every 15min. Storage rate is limited of the performance of the RDB system and will be switched to the NoSQL data base system reported at this conference [10].

The transfer time for 5000 data points from the BPM read out–program to the client for two BPMs was about 550ms. This is fast enough to obtain 2 BPM data every second. After opretating for several weeks, it is confirmed that the system works well for long–term operation.

The system will be used for beam orbit monitoring beam position displacements up to 100Hz and to investigate the source of drift of the beam position.

ISBN 978-3-95450-139-7

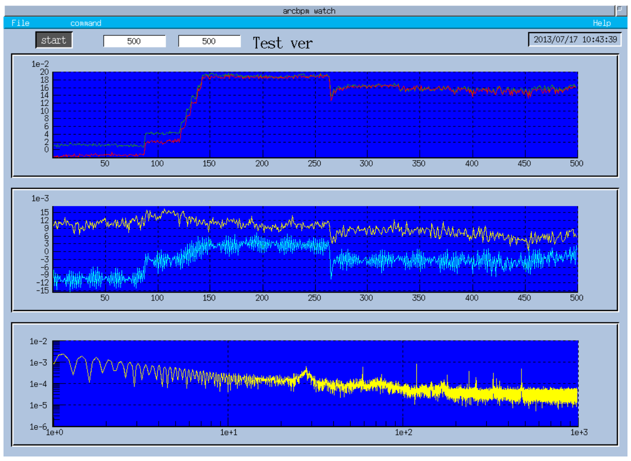

Figure 3: Example of a client program. Upper and middle panes display waveforms for horizontal and vertical position. The bottom pane shows an FFT–analysed spectrum. This Figure shows beam position during storage ring tuning, so the horizontal position was changed.

SUMMARY

We developed a MADOCA II–LabVIEW interface to integrate LabVIEW–based software into the MADOCA II control system. As the first application we also developed a PXI based BPM readout system. We introduced it into the SPring-8 control system and it had good enough performance to obtain BPM data. We plan to use the interface for other many applications such as a LabVIEW–based detector control system, a piezo-transducer control system for the fine position tuning of X-ray monochromators, etc.

ACKNOWLEDGMENT

We would like to thank Mr. Yamagata of National Instruments Japan for developing the MADOCA II–LabVIEW interface. We also would like to thank Dr R. Tanaka, A. Yamashita and M. Kago at SPring-8 for useful discussions and suggestions.

REFERENCES

[1] R. Tanaka, et.,al., “The first operation of control system at the SPring-8 storage ring”, Proc. of ICALEPCS'97, Beijing, China, (1997) p.1
[2] N. Hosoda, et., al., “Integration of the Booster Synchrotron Control System to the SPring-8 Control System”, Proc. of ICALEPCS'99, Trieste, Italy, (1999) p.93
[3] R. Tanaka, et., al., “Inauguration of the XFEL Facility, SACLA, in SPring-8”, Proc. of ICALEPCS 2011, Grenoble, France, (2011) p.585
[4] M. Yamaga, et., al., “Event-Synchronized Data-Acquisition System for SPring-8 XFEL”, Proc of ICALEPCS 2009, Kobe, Japan, (2009) p.69
[5] T. Matsumoto et al., “Next-generation MADOCA for SPring-8 control framework”, TUCOCB01, these proceedings.
[6] http://www.zeromq.org
[7] http://msgpack.org
[8] http://www.zeromq.org/bindings:labview
[9] http://sourceforge.net/p/msgpack-labview/wiki/Home
[10] M. Kago et al., “Development of a Scalable and Flexible Data Logging System Using NoSQL Databases”, TUPPC012, these proceedings.