EPICS IOC PROTOTYPE OF FRIB MACHINE PROTECTION SYSTEM*

L. Wang†, M. Davis, Z.Y. Li, S. Zhao, G.B. Shen, M. Ikegami, FRIB, East Lansing 48823, USA

Abstract

The FRIB Machine Protection System (MPS) is designed to protect accelerator components from damage by the beam in case of operating failure. MPS includes master and slave nodes, which are controlled by MPS IOC. In this paper, we present design of MPS IOC and status of its prototyping.

INTRODUCTION

The FRIB MPS includes one master and fifty-six slave nodes, and it is to be aware of the machine mode/beam mode/ion type and constantly monitors critical signals. Once the condition to stop beam under the selected machine mode/beam mode/ion type is detected, it issues command to evoke the mitigation devices to switch off beam. MPS interfaces with all devices that require to stop beam quickly. It interfaces with Run Permit System (RPS), Global Timing System (GTS) and MPS IOC as well [1]. MPS IOC communicates with MPS nodes via a customized UDP-based protocol, which is shared by a couple of FRIB UDP-based subsystems and includes the definition of register access, waveform upload, flash memory access and DDR raw data upload. The function of MPS IOC includes register control, remote update of FPGA firmware and DDR raw data transfer. MPS IOC is developed based on asynPortDriver module and the GUI is developed with CS-Studio BOY.

SYSTEM OVERVIEW

There are three virtual machines used for MPS IOC: master IOC, slave IOC and CS-Studio OPI. Master IOC communicates with master node and slave IOC communicates with all the fifty-six slave nodes. MPS network architecture is shown in Fig. 1. Each node has a static IP address, and each IOC can be configured which node to communicate with.

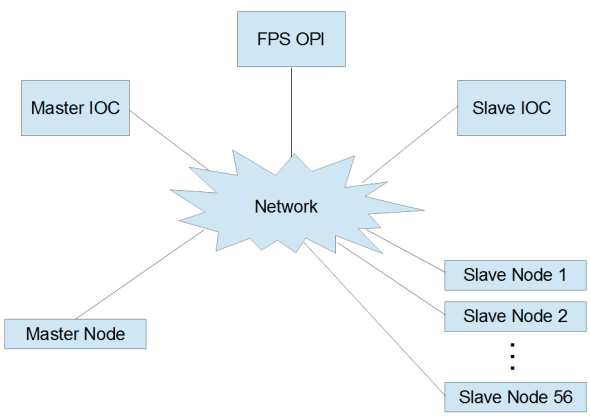

Figure 1: MPS Network Architecture

MPS master and slave nodes are separate hardware controllers and provide UDP control interface, so MPS IOC is required to control MPS controllers via UDP packets. The required functions for MPS IOC are as follows:

- Set/read status and parameters in MPS controllers.
- Remotely update the FPGA firmware in MPS controllers.
- Transfer up to 256MB of DDR raw data from each MPS controller to MPS IOC in an acceptable time period.

LCP PROTOCOL

LCP (LLRF Control Protocol) is a UDP-based application layer protocol originally defined by FRIB LLRF (Low-Level Radio Frequency) and is so far also used for FRIB MPS and Prebuncher systems.

Command Definition

It defines register read/write, waveform upload, persistent memory erase/read/write and DDR raw data upload UDP commands for communication between driver and controllers. Each command has request and response packets, which have defined format.

Register Definition

LCP defines three groups of registers: read-only, write-anytime and write-once. All of them are read from controllers periodically, write-anytime register is written to controllers periodically and write-once register is written to controllers only if its value is changed and the writing command is sent only once.

Two threads are responsible for reading or writing data from or to controllers. Sync thread is used in request/response mode and async thread is used in streaming mode. In request/response mode, the sent and received packet are in pair, whereas in streaming mode, one packet that specifies response interval is sent, after which the response packet is received periodically. LCP infrastructure is shown in Fig. 2, the communication between LCP driver and controller and the one between LCP driver and EPICS record are asynchronous.

Implementation

The source code of LCP is divided into two parts:

- LCP protocol layer: a C++ base class, which is derived from asynPortDriver class and is responsible for UDP communication with controllers. Different LCP-based drivers share this part.
- Application layer: a C++ derived class, which handles application-specific issues like register definitions and data process. Different LCP-based drivers build their own copies of this part.

* Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661
† wanglin@frib.msu.edu

ISBN 978-3-95450-169-4

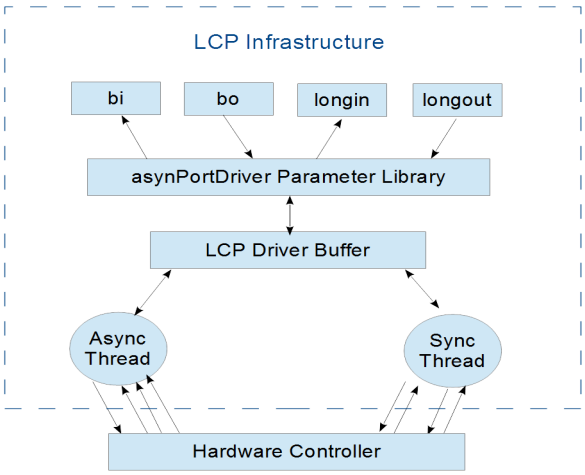

Figure 2: LCP Infrastructure.

MPS IOC BASIC FUNCTIONALITY

MPS master IOC is responsible for controlling master node and MPS slave IOC is responsible for controlling all slave nodes. However, most functionalities in slave IOC are also covered in master IOC, so this paper will mainly describe functionalities for master IOC.

Figure 3 shows the main page of MPS master OPI, it displays beam status, MPS state and mitigation devices status, and it also provides links to the page for each functionality.

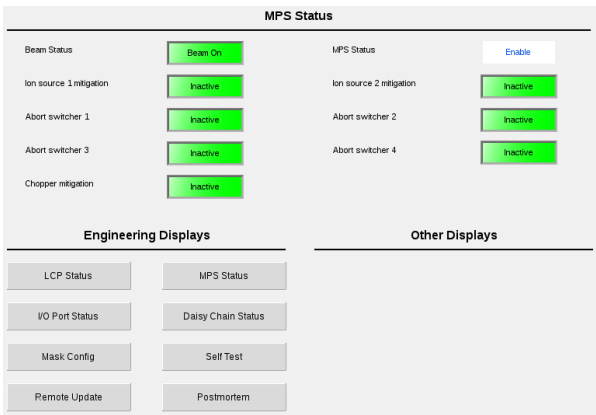

Figure 3: MPS Master OPI.

LCP Status

MPS IOC reads hardware information from controller, monitors UDP connection status and communication thread status, saves and restores IOC/controller restart time, calculates UDP sent/received packets and provides a reboot command to reset controller. It is a useful diagnostic feature to track UDP communication between IOC and controller. LCP status page is shown in Fig. 4.

MPS State Control

MPS system has four states: fault, disable, monitor and enable. MPS IOC provides the interface to set and read MPS state as well as the interface to read beam status and mitigation device status.

I/O Port Status

MPS master or slave controller has a number of rear panel and front panel ports that can interact with other systems, and each port has an INP/OUT configuration and OK/NOK status. Both configuration and status are determined in controller, and the IOC is responsible for reading the status from the controller.

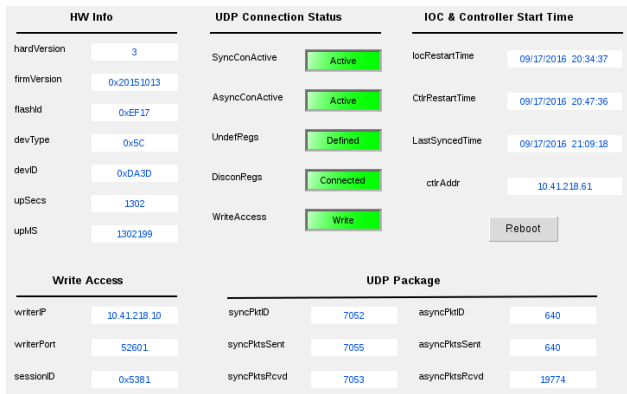

Figure 4: LCP Status Page.

Daisy Chain Status

MPS master node communicates with slave daisy chains, each of which includes eight slave nodes. IOC will get daisy chain diagnostic information including daisy chain number, identified slaves in each chain, I/O error, chain status, timeout error, checksum error and data valid error provided by master node.

If MPS trips, MPS master node will determine which slave node detects NOK signal from the sensor in other system, then daisy chain ID, slave ID, slave node timestamp, master node timestamp and the 96 port status will be latched in master node. IOC is responsible for getting this information.

Mask Config

Each MPS slave node has 96 ports, each of which connects to the sensor in other system and detect OK/NOK signal, if NOK signal is detected, MPS trips. But at times some ports do not connect to any sensor, in this case, we need to mask those ports. Therefore, MPS IOC provides a mask bit for each slave port, if a mask bit is set to “mask”, MPS will not check the OK/NOK status of the corresponding sensor thus no further actions will be taken. All the mask bits are configured in master IOC.

Self-Test

In order to provide the hardware engineer a convenient way to test the components on MPS master and slave controllers, IOC provides self-test interface for DDR test, LED test, front panel test, rear panel test, and so on. After sending self-test command to the controller, IOC can get the response PASS/FAIL result from the controller after it finishes the test.

Response Time Calculation

When MPS slave node detects NOK signal from the sensor in other system, it will latch the timestamp and inform master node via fibre, then MPS master node will latch its own timestamp, trigger mitigation device and switch MPS state to "fault". The response time is from when slave node detects NOK signal to when the corresponding mitigation device is triggered.

IOC monitors MPS state, if it switches to "fault", then calculates the response time according to master timestamp, slave timestamp and GTS event clock (80.5MHz) as follows:

respTime (ns) = (ts_master - ts_slave) * 1000 / 80.5

REMOTE UPDATE

MPS remote update feature is to update FPGA firmware to the flash memory in master or slave node via UDP packet. The image that needs to be written to flash is divided into three parts:

- Header: Contains basic command and address information for loading image from flash into FPGA.
- Golden image: The backup image, which will be loaded when working image does not work.
- Working image: The primary image, which will be loaded into FPGA when powered up.

The user interface is implemented as PV, and update command and update progress for each MPS node is provided. Figure 5 shows remote update OPI.

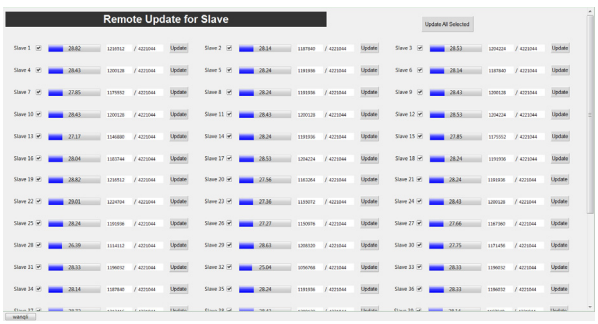

Figure 5: Remote Update User Interface.

Performance Test

There are fifty-six MPS slave nodes and they may need to be updated together, so the performance for remote update concurrently is important. Table 1 shows the test result of updating one node as well as updating five, ten and fifty-six concurrently. The test is done between IOC and controller simulator, which is a customized program that simulates the behavior of the controller.

Table 1: Remote Update Performance Test

Nodes	1st	2nd	3rd
1	2m 02s	2m 04s	2m 01s
5	2m 02s	2m 02s	2m 00s
10	2m 03s	2m 02s	2m 03s
56	2m 22s	2m 21s	2m 22s

DDR RAW DATA TRANSFER

When MPS trips, sometimes the operator needs to acquire ADC raw data from controllers for post-mortem, and the raw data for MPS is up to 256MB which is stored in DDR memory on the controller. Therefore, MPS IOC needs to transfer this data from controllers.

In order to improve the performance, DDR raw data is transferred in streaming mode, which means IOC sends only one request and MPS controller will respond a number of packets.

One transfer thread, one receive buffer and two UDP sockets are created to receive and process the UDP packets, the workflow is as follows:

- Transfer starts and IOC requests data blocks from the controller one by one, each block includes a number of packets.
- For each block, a flag for each packet is created to record whether the packet has been received.
- For each block, IOC sends a request packet and receives the streaming response packets for the block via socket sFd1. After all the response packets arrive, IOC checks and resend request for all the lost packets (if any) one by one via socket sFd2. All the received packets will be temporarily stored in the receive buffer.
- Once a whole block data has been received, IOC will save it into a file.
- If all the blocks have been received and saved in the file, transfer ends.

Performance Test

Currently, the DDR raw data transfer in streaming mode is tested with the controller simulator instead of the real hardware. Table 2 shows the result of performance test.

Table 2: DDR Raw Data Performance Test

Test	Time(s)
1st	21
2nd	22
3rd	18
4th	20
5th	15
6th	22
7th	17
8th	20
9th	18
Avg	19.2

CONCLUSION

By now, all the MPS features for FRIB Front-End commissioning have been finished and the IOC has been deployed in FRIB test network. The next step is: 1) test streaming-mode register read and DDR raw data transfer with real hardware; 2) improve the remote update feature; 3) implement the post-mortem feature.

ISBN 978-3-95450-169-4

REFERENCES

[1] ZhiYong Li, *et al.,* "FRIB Fast Machine Protection System: Engineering for Distributed Fault Monitoring System and Light Speed Response", presented at *LINAC'16*, East Lansing, MI, USA, Sep. 2016, paper THPLR046, this conference.