ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA189

LCLS MACHINE PROTECTION SYSTEM HIGH LEVEL INTERFACE IMPROVEMENTS

C. Bianchini, S. Hoobler, SLAC National Laboratory, Menlo Park, CA 94025, U.S.A.

Abstract

The Linac Coherent Light Source (LCLS) is a free electron laser (FEL) facility operating at the SLAC National Accelerator Laboratory (SLAC). The LCLS Machine Protection System (MPS) contains thousands of inputs and hundreds of protection interlocks. Control room operators use a high-level Graphical User Interface (MPSGUI) to view and manage faults [1].

MPSGUI contains a wealth of useful information, from hardware input details to high-level logic flow, but in its first version it was difficult for accelerator operators to take full advantage of this. A recent project has greatly improved the workflow and usability of MPSGUI.

INTRODUCTION

The purpose of the MPS is to prevent damage to beamline components due to beam. The MPS monitors the states of devices throughout the accelerator. If it detects a condition that may lead to damage, it turns off the beam.

MPSGUI, a Java application, is the primary operator interface to the MPS. Operators use it to identify, diagnose, and manage faults. This paper will describe the enhancement provided by this project on MPSGUI.

MPSGUI

The MPS defines its static input and logic configuration in SQLite database files. Real-time state information is hosted by EPICS signals. The MPSGUI uses this combination of static and dynamic data to provide detailed fault and diagnostic information to operators.

The information is distributed in the MPSGUI tabs, accessible at the interface's bottom (Figure 1) [2].

- Summary: displays current rates, current faults, and bypasses
- Faults: details of MPS inputs;
- Logic: details of MPS logic, how inputs translate to rate limits
- Ignore logic: condition under which logic can be ignored
- History: full history of MPS input state changes;
- Recent Faults: last 1000 MPS faults that affected beam (faults that clear quickly, except sub second may not appear here)

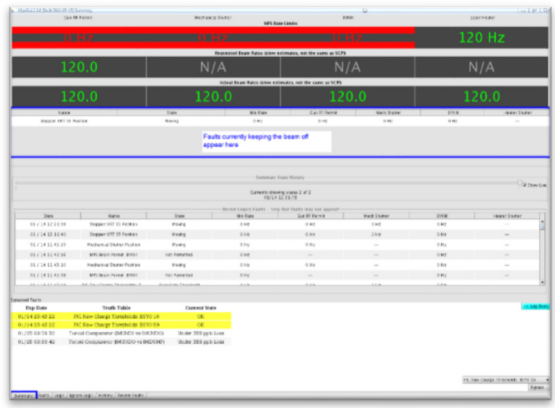

Figure 1: MPSGUI summary tab.

ENHANCEMENTS

MPSGUI contains a lot of information available to describe the MPS details but not fully utilized by Operations. In fact, the navigation from High Level GUI down to the logic fault description, hardware level bits related to this fault requires cross-reference and the use of several screens.

The complete requirements list was defined during a series of meetings with control room operators. A task list was made based on MPSGUI's maintenance tickets, user feedback and feasibility balanced with the limited resources of time and budget.

The intent of this project was to solve the following main MPSGUI's issues:

- Hard to find inputs associated with a given piece of logic.
- Missing information in displays.
- Challenging to identify faults that clear quickly.
- Difficult to associate a fault to the related logic details.
- Resolve issues that were discouraging operators from using the GUI.

Faults History Server

The most important of Operation's requests was the desire to identify fast non-latching recurring faults. This category of faults would appear repeatedly and clear at a very fast rate. Originally, MPSGUI was providing information about "current faults", operated by a separate JAVA thread. The thread was running on the user launched interface process, increasing the CPU load on the user side. The "current faults" information was not available when launching a new interface, instead was

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA189

populated real-time. An additional server process was developed to solve this issue. The server process continuously runs in the background identifying and storing logic faults. The Faults History Server is hosted on a production server where the process increments a PV monitored by the alarm system. In rare cases where the server process stops working, the alarm system reports the status to Operations to re-start the process.

The Faults History information is stored into a JSON file. The JSON file contains up to 1000 faults and updates using a FIFO methodology. The "Recent Faults" tab added to MPSGUI displays the fault history as shown in Figure 2.

It is now possible to access days of fault history on freshly launched GUI instances.

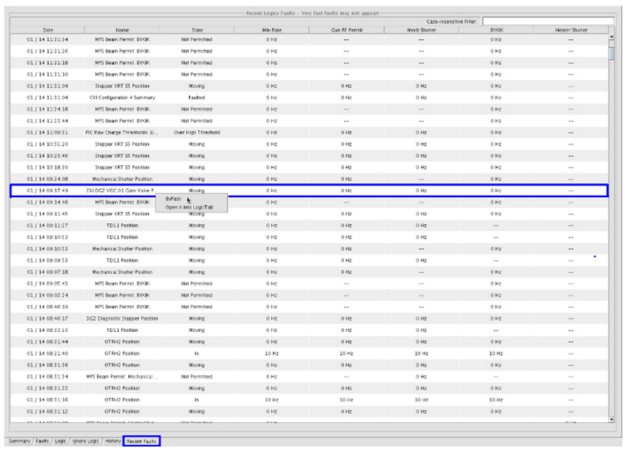

Figure 2: MPSGUI recent faults tab.

Another new tab's feature is the context menu to bypass or to show details for the selected piece of logic.

Logic Input Information Accessibility

The MPS's logic is displayed into MPSGUI's logic tab, it contains details how inputs translate to rate limits.

The information provided was incomplete and required to cross-reference multiple displays in order to map hardware input to high-level logic. A table appear while clicking on one of the listed logic, containing sensitive information and the link to the link node information.

The additional search feature by fault name is very useful for troubleshooting and identifying a fault occurrence. Prior to this search feature becoming available, it was not possible to easily find a fault in a Logic tab unless knowing the exact name.

In Figure 3, the blue rectangle is showing the related information bottom and the related display to link node's display.

The same feature is available for the "ignore logic" tab.

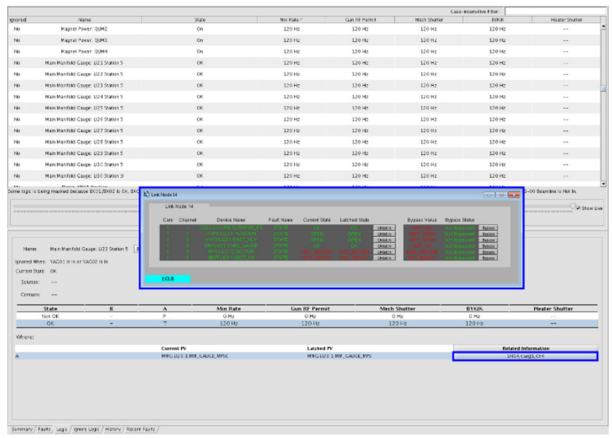

Figure 3: MPSGUI logic tab.

Freeze History Tab

The MPSGUI History tab was updating at a very fast rate. Many MPS changes occur very quickly making difficult to use the History tab, continuously updating.

Added a "freeze" button which allows users to control the GUI and stop receiving updates to the list of input state changes, Figure 4.

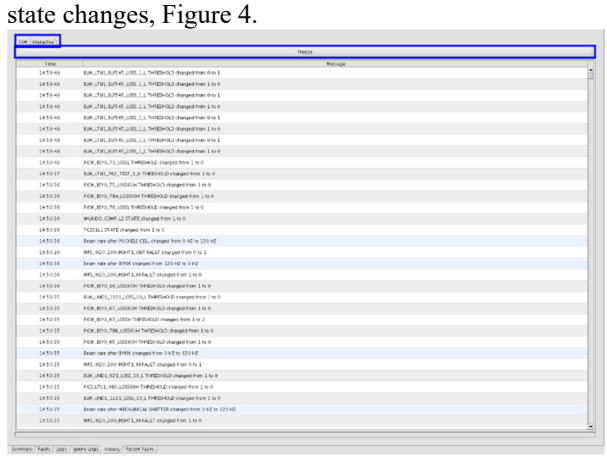

Figure 4: MPSGUI history tab.

CONCLUSION

The new features now allow operators to quickly identify MPS faults and diagnose problems. The troubleshooting time has been reduced, increasing the uptime for FEL delivery to user experiments. The MPS Logic Faults History server, in particular, has had a strong positive impact, allowing users to access days of recent fault history with freshly launched GUI instances.

ACKNOWLEDGEMENT

Acknowledgement to Sonya Hoobler for her guidance and support; to the SLAC operators for the teamwork to write the project requirements.

REFERENCES

[1] SLAC: MPS documentation on confluence, https://confluence.slac.stanford.edu/display/ACCSOFT/MPS

[2] SLAC: Ops Wiki, MPS GUI User Guide, http://ad-ops.slac.stanford.edu/wiki/index.php