# CLAS12 RICH Slow Control System Documentation

**Justin Goodwill** [1]

Nathan Baltzell[2], Valery Kubarovsky[2], Fatiha Benmokhtar[1]

[1]Duquesne University, [2]Jefferson Lab

October 2016

## FORKING THE REPOSITORY/RUNNING THE SLOW CONTROL SYSTEM

1. Go to the URL: https://github.com/JeffersonLab/clas12-epics.

2. With your github account, click fork in the upper right-hand corner. Doing so will create a copy of the clas12-epics repository in your github account. Inside this repository are the graphical user interface (GUIs) for the RICH system and their necessary databases.

3. In order to make changes to the system, you first need to secure shell (ssh) into the clonsl2 or clonsl3 servers with your computer account from the terminal. Enter the following commands in the terminal to do so:

```
ssh yourusername@login.jlab.org  → Enter JLab credentials
ssh hallgw  →Enter 2-factor authorization credentials
ssh clonsl2  →Enter clonsl2 credentials
```

4. Once on this server, you need to create a copy of the repository on your account on the server. To create one called "clas12-epics", type the following command:

```
git clone https://github.com/***yourgithubusername***/clas12-epics clas12-epics
```

5. Next, you need to compile the code within all branches of the system. This will create the file system for the slow control system for all of CLAS12. In order to compile this code, type:

```
cd /home/yourusername/clas12-epics/
make –f Makefile
```

6. In order to view the GUI screens, you have to open Control System Studio (CS-Studio). CS-Studio interfaces with EPICS to display certain process variables such as voltage or current in a graphical format. To open CS-Studio, type in the terminal:

```
cd  /home/yourusername/clas12-epics/apps/
./css_launcher.sh
```

7. In CS-Studio, the file system will appear in the navigator pane. To get to the RICH detector's GUI screens in CS-Studio, navigate to CLAS12_Share/detectors/RICH. Various ".opi" files will be in this directory, named in accordance to the system they are displaying. "RICH_Master.opi" has access to all of the systems.

8. To start the simulated input/output controller, you first need to open a new terminal (as the other one is now dedicated to running CS-Studio) and ssh to the clonsl2 or clonsl3 server as in step 3. Once on the clonsl2 or clonsl3 server, type:

```
cd /home/yourusername/clas12-epics/apps/iocBoot/iocrichsim.
./st.cmd
```

   Note: A more detailed explanation of the input/output controller is given on pages 4-5.

9. In CS-Studio, go to a particular .opi file such as "RICH_Master.opi" and click the run-time mode option. With the IOC running and the .opi file in run-time mode, there is communication between the IOC and CS-Studio. If you have the .opi in run-time mode and the IOC is not running, there will be alarms in the .opi file indicating that there is no communication with the IOC.

10. To send a signal to the IOC, type in the terminal:

```
caput  processvariablename value
```

**DATABASES**

For each system, there is a database containing all of the process variables (PVs) that need to be monitored or changed. These databases are contained in the clas12-epics/apps/richsimApp/Db directory. In the databases are records that correspond to these PVs, an example of which is shown below in the figure.

```
record(ao, "B_HW_$(CrName)_Sl$(Sl)_Ch$(Ch):vmon")
{
   field(PREC,"2")
   alias("B_DET_$(Det)_$(Sys)_$(Element):vmon")
   alias("B_SYS_$(Sys)_$(Det)_$(Element):vmon")
   info(autosaveFields_pass0, "HIGH HIHI LOW LOLO HSV HHSV LSV LLSV")
}
```

This particular record is for the HV system, which will be used to monitor the voltage. Ultimately, CS-Studio will reference these particular PVs and display the information stored. In EPICS, there are more than 30 types of records that can be chosen from analog outputs to binary inputs—choosing which type of record to use is based on the particular application. Each record has various fields that can be changed as governed by the EPICS software [1].

For the high voltage, low voltage, and temperature systems, there are many channels that are repeated. Thus one standard database is created with macros. The specific values of these macros are defined in ".substitutions" file. For each channel, one set of macros is defined, which will then create a database for this particular channel. For example, for the HV system, there is a standard database called richhv_sim.db with a substitutions file called HVRICH.substitutions, which defines each channel. Likewise, this setup was configured for the low voltage and temperature systems.

In order to add these databases, one would have to insert a new record or change the fields of the other records. The documentation for the different types of records and their specific fields are given on the web at https://wiki-ext.aps.anl.gov/epics/index.php/RRM_3-14 [1].

*HV ALARM*

An intricate alarm system was developed for the HV system within EPICS using a subroutine record. This subroutine record is contained in the file:

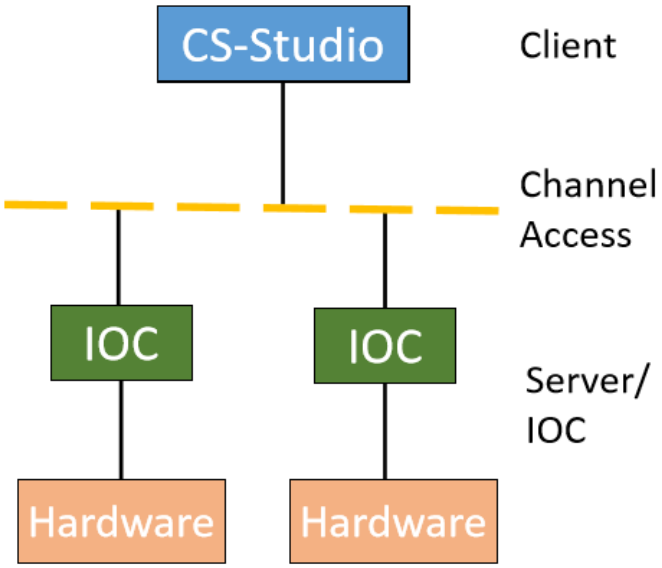
clas12-epics/apps/richsimApp/src/richalarm.c.

This subroutine uses the :comms and :stat records in the HV database. The :comms record is a binary input that signals a communications error: it is 0 when there is no communications error and 1 when there is. The :stat record is a 12-bit binary sequence that contains possible alarm states such as when the HV channel is tripped or if the maximum voltage is reached. The subroutine record called :alarm sets the precedence for these alarms. Via bitwise operations, it first checks for a communications error and subsequently whether the channel is on. It then checks the :stat record for the most important alarm, which corresponds to the most significant bit in the sequence. Finally, it checks whether the voltage is outside of some tolerance percentage.

Based on which alarm is most significant, the :alarm subroutine record outputs a value to a multi-bit binary input record called :alarmstate. Based on this value, the :alarmstate record sounds an alarm with either a MAJOR or MINOR severity and a string of text to identify the issue. To supplement this alarm, a script was attached to each rectangular widget that changed its background color based on the state of the :stat and :comms.

**INPUT/OUTPUT CONTROLLER**

In terms of data communication, the way in which hardware communicates with the software is through the input/output controller (IOC), which is basically a computer that determines what information is sent and received between hardware and software. Loaded on the IOC are databases containing all of the PV information for the detector. CS-Studio, which is comprised of the GUI interfaces, broadcasts the need for information on a certain process variable over a network.




The IOC recognizes this need and requests from the hardware the necessary information. The IOC then sends this information back to CS-Studio over the network, and it is displayed on the GUI in some fashion [2].

Currently, the IOC is being simulated since there is no hardware. To test it, signals are sent to the IOC as if they were coming from hardware and the software response to this information is examined. For example, one might check to see an alarm is raised if the voltage is too high.

To load different databases on the IOC, you must edit the st.cmd file in the clas12-epics/apps/iocBoot/iocrichsim/ folder.

To load a database, type the command, dbLoadRecords("yourdatabase.db"), in the st.cmd file.

**AUTO-SAVE**

The auto-save feature is important when operating the detector as it periodically creates multiple backups of all the alarm levels, their severities, and other pertinent information. For example, if the temperature is just a little too high such that it raises an alarm, the operator might slightly increase the alarm level. If the input/output controller (IOC) crashes at some point, the alarm level should not revert back to its initial value. Instead, it should be set to the value the operator gave it.

```
B_SYS_HV_RICH_SEC1_ROW10_PANEL1:type.HIGH 10
B_SYS_HV_RICH_SEC1_ROW10_PANEL1:type.HIHI 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL1:type.LOW 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL1:type.LOLO 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL1:type.HSV 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL1:type.HHSV 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL1:type.LSV 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL1:type.LLSV 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL2:type.HIGH 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL2:type.HIHI 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL2:type.LOW 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL2:type.LOLO 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL2:type.HSV 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL2:type.HHSV 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL2:type.LSV 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL2:type.LLSV 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL3:type.HIGH 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL3:type.HIHI 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL3:type.LOW 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL3:type.LOLO 0
B_SYS_HV_RICH_SEC1_ROW10_PANEL3:type.HSV 0
```

The auto-save feature is important when operating the detector as it periodically creates multiple backups of all the alarm levels, their severities, and other pertinent information. For example, if the temperature is just a little too high such that it raises an alarm, the operator might slightly increase the alarm level. If the input/output controller (IOC) crashes at some point, the alarm level should not revert back to its initial value. Instead, it should be set to the value the operator gave it.

Inside the st.cmd file for starting the IOC are functions to implement this auto-save feature. When it starts running, the IOC reads from the last .sav file stored in

/usr/clas12/DATA/autosave/iocrichsim/, which contains all of the alarm levels and severities. Shown to the left is what a .sav file consists of. If alarm values are changed at all, then the .sav files are updated.

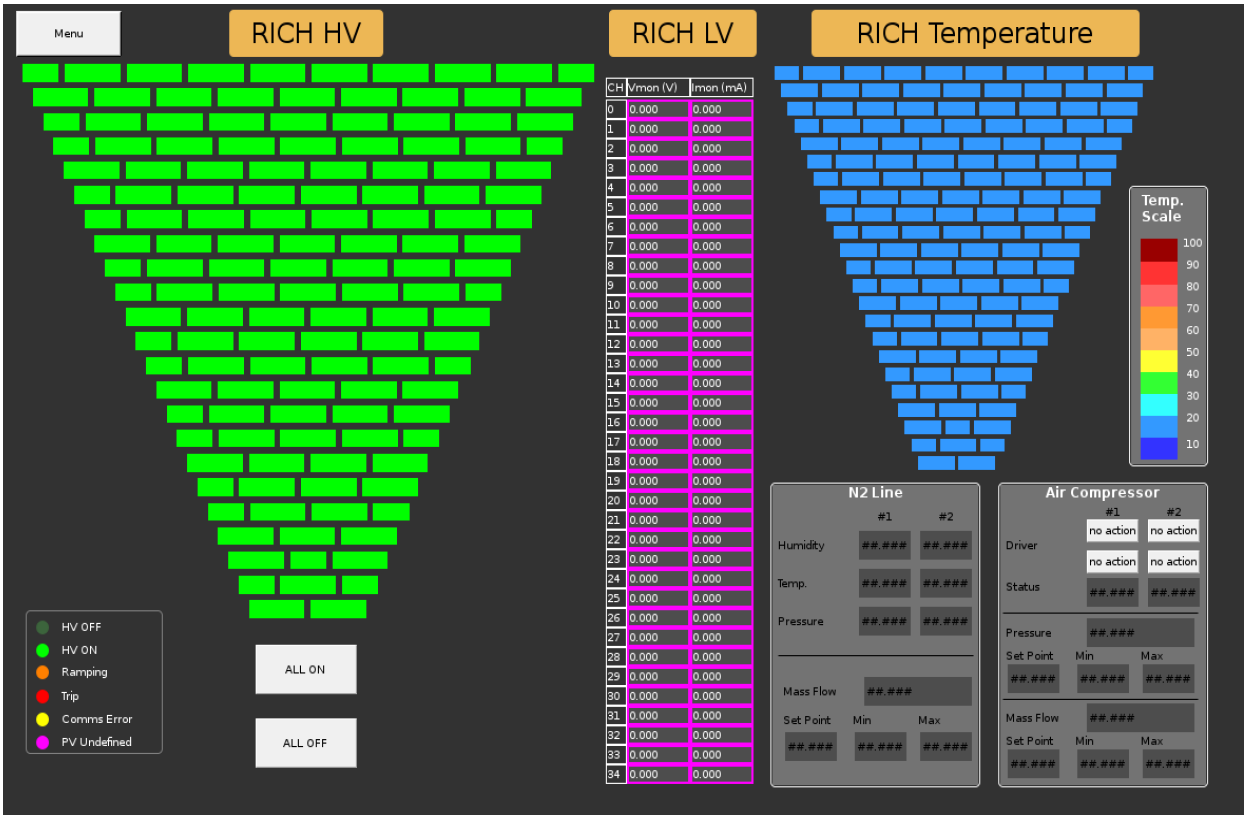
**GUI SCREENS**

GUIs were developed for the HV, LV, temperature, N2 gas, and air compressor systems. The table below lists the various .opi files that were created and a brief description of them.

| OPI file | Description |
|---|---|
| **RICH_Master.opi** | Master screen showing all systems on one page. Further information is given below. |
| **RICH_HV.opi** | The array of HV channels implemented using rectangular widgets. This .opi file is linked to the master screen. |
| **RICH_HV_channels_details.opi** | Displays detailed information of all HV channels on one screen, intended for operator use. |
| **RICH_HV_channels_novice.opi** | Displays a truncated view of the information for all HV channels on one screen. |
| **LV_forMaster.opi** | List of monitored voltages and currents for the LV system. This .opi file is linked to the master screen. |
| **RICH_LV_details.opi** | Displays detailed information of all LV channels on one screen, intended for operator use. |
| **RICH_LV_detail_individual.opi** | Displays detailed information of only one LV channeled. This is used when one clicks on a LV channel on the master screen. |
| **RICH_LV_novice.opi** | Displays a truncated view of the information for all LV channels on one screen. |
| **RICH_LV_Novice_Individual.opi** | Displays truncated view of information of only one LV channeled. This is used when one clicks on a LV channel on the master screen. |
| **RICH_temp.opi** | The array of temperature channels implemented using rectangular widgets. This .opi file is linked to the master screen. |

| | |
|---|---|
| **rich_N2Purge.opi** | Displays the circuit flow of the N2 purge system with valve pressures. |
| **CoolingCircuit.opi** | Displays the circuit flow of the cooling circuit system with valve pressures. |

*MASTER SCREEN*




The master screen displays the most important information from all screens including high voltage, low voltage, temperature, N2 lines and air compressor flow. All other screens listed in the table can be accessed from this screen. Clicking on the HV or temperature rectangles will bring up detailed information about the individual channel. The temperature of each of the channels is monitored in real time, and as such, the rectangles change color according to the color scale on the right side of the screen. If an alarm is raised for the high voltage, the border of the rectangle will change color according to the color code key in the bottom left corner. All-on and all-off buttons for the high voltage are placed below the high voltage. Clicking on the

channel number for the low voltage will bring up information about the individual channel. The menu button allows you to open the screen showing all channels on one screen for HV and LV.

**REFERENCES**

[1] Stanley, Philip, Anderson, Janet, and Marty Kraimer. "EPICS 3-14 Record Reference Manual." May 1998. https://wiki-ext.aps.anl.gov/epics/index.php/RRM_3-14

[2] Johnson, Andrew. "EPICS Database Principles." 1999. http://www.aps.anl.gov/epics/docs/USPAS2010/Lectures/Database.pdf