Controls User Guide

Controls Camera User Guide

We have a convenience script for Gige cameras that can be called from the command line called "gige". In case "which gige" does not return anything, this script is located at /reg/g/pcds/engineering_tools/<hutch>/scripts. This should be on the path for <hutch>opr accounts if the currently existing .bashrc was sane enough.

"gige -h" will list the available options. If no option is passed, you will get a list of cameras to choose from. By default, the camera viewer described here will be opened for the requested camera. The camera name can be passed using the "-c <cam #/cam name>" options. If you pass "-m" in addition to the "-c XXX" argument, you will get the expert edm screen. Should the viewer not work (e.g. not update), open the edm screen and start simple trouble shooting described here (to be made). You can reboot the IOC either using the iocmanager or adding "-r" to your gige command. "gige -c camID -n -" will ask the IOC to start acquiring images again: when you put a camera back online, it will often not acquire and needs to be told to. A reboot does the same thing, but with a lot more overhead. If you have a gige object defined in <hutch> python, you can also start the expert screen by using the <gige>.expert_screen() command.

If you would like to record your controls camera in the xtc stream along with the DAQ data, contact your PCDS-POC. The IOC will need to be build with support for timestamps so that images can be assigned to events. After that, they can be added to the DAQ under the "Camera IOCs" section.

Motor expert screens

Motor screens can be found in various formats at various places in the different hutches. If you are interested in the "expert" screen for most of our motor types, you can use

!/reg/g/xpp/scripts/motor-expert-screen <PVNAME> where PVNAME is e.g. XPP:SB2:MMS:19

This script will bring up the right kind of motor screen (old & new IMS as well as Newport screens). In hutch python, you can also start the motor expert screens my using <motor>.expert_screen().

Newport screens are described here.

EPICS trouble shooting

If a PV gives you trouble, a first good step is to see if the IOC is up stably and possibly reboot it. For that you will use the IOC manager which has a user guide here and is started via iocmanager. This guides describes how to start it and how to it works. More detailed information for commonly used IOCs can be found below.

For a given PV you can also determine which server the IOC is running on and if rebooting the IOC does not help you can consider rebooting the server. Be mindful that this will restart all the IOCs on that server. One way to do that is using "serverStat <machine name> cycle" for any machine with ipmi.

IPIMB/Wave8

The configuration GUI for your IPIMB or Wave8 boxes can be started using: "ipmConfigEpics <b boxname>". If you don't pass the "-b" option, you get a list of ipimb boxes relevant for your hutch.

A troubleshooting guide can be found here: IPIMB Troubleshooting for Controls IPIMBs.

Motor setup

Newport

Turn XPS off before connecting motor!

Once everything is connected, turn XPS back on. Go to the IOC edm screen and reconnect, then click the "autoconfig" button. On success, a screen should come up telling you the motors the XPS found. Now your next steps are to initialize all motors and home or reference them. Now they should be accessible though the IOC.

To get them into your hutch python, you either have to:
(old python): add them to the correct epicsArch file. All motors found in the user/experiment specific file will be added to the "x" module (XPP/XCS).

(new python): motors added to the questionnaire (where they should be anyways) will be instantiated automatically.

**IMS - smart motors**

If you have a (old) hutch python (XPP/XCS), add the motor to the epicsArch file. A motor python object will be instantiated upon restart. This motor object has a function called "<motor>.auto_setup" which will initialize the motor, clear the power cycle and check for a configuration based on matching the motor controllers serial ID in the parameter manager. Alternatively, you can open the expert screen from the motor python object <motor>.expert_screen(), re-initialize from there & clear the power cycle. You can then use "<motor>.pmgr.diff()" to see the difference of current and saved configurations. "<motor>.pmgr.apply_config()" will apply the configuration. Check after applying the config that all parameters "made" it by calling pmgr.diff() on the same motor again.

**IMS - dumb motors (HXR)**

This works very similar to the smart motor procedure, except that the serial number cannot be used to find the configuration. You will need to pass the configuration name to the "diff" or "apply_config" functions. If you don’t, you will be asked. There is a search option where smart string matching is used when you don’t know the config name. It is planned to alliviate this issue by labelling the stages with their config name (HXR). Dumb motor recognition does rely on the parameter manager knowing the serial numbers of the dumb motor controllers. Should you have a new chassis or a newly repaired one, this might not work and PCDS personnel will have to fix this.

Note: When running Navitar motors on the new MForce chassis there are two additional settings in the controller MCode that need to be set. D1 and D2 need to be set to 50. This can be done through the :CMD PV.

**New Hutch-python**

The new hutch python has extensive documentation at https://pcdshub.github.io/hutch-python/.

**Expert guides (restricted viewing)**

Link to the old "Controls User Guide"