EPICS ACCELERATOR CONTROL SYSTEM FOR THE IAC-RadiaBeam THz PROJECT

Anthony Andrews*, Y. Kim, C. Eckman, P. Buaphad,
T. Downer, C. O'Neill, B. Berls, K. Folkman, and J. Ralph,
Idaho Accelerator Center, Idaho State University, Pocatello, ID 83201, USA

Abstract

The Idaho Accelerator Center (IAC) of Idaho State University has been operating a 44 MeV L-band linac for various nuclear physics related applications [1]. However, for the past several years, this research has been done without the aid of a modern computer based control system. To obtain a better reproducibility and stability in operation, the EPICS accelerator control system has been applied to control various components of this linac. This has been done for the purpose of a joint THz research project between IAC and RadiaBeam that was performed in November 2012 [1, 2]. This paper describes the development of the EPICS accelerator control system used during this joint THz research experiment.

INTRODUCTION

To get a better reproducibility and stability for the IAC-RadiaBeam THz project, the analog control system for the 44 MeV was upgraded to a computer based accelerator control system. To do that, various control systems using the Experimental Physics and Industrial Control System (EPICS) have been developed. As shown in Fig. 1, the control system for the THz project had three different device types: seven TDK-Lambda ZUP magnet power supplies, one Lambda EMS magnet power supply, and two Prosilica GC1290 GigE CCD cameras [3–5]. The Lambda EMS magnet power supply used during this THz experiment has the embedded IEEE 488 controller. However, the development of a control system for some Lambda EMS magnet power supplies with the external RSTL controller will be described in the next conference paper. For the IAC-RadiaBeam THz project, the power supplies were connected to their own dedicated MOXA terminal server with an RS485 interface for the TDK-Lambda ZUPs, and an RS485-RS232 conversion interface for the Lambda EMS power supplies [6]. Then, terminal servers and two Prosilica GC1290 GigE CCD cameras were connected to the EPICS server via an isolated network using Ethernet cables and a 3Com network switching hub. After that, a means to communicate with the power supplies and CCD cameras was developed by using three EPICS modules (ASYN, StreamDevice, and areaDetector) supplied with the synApps package [7, 8]. Then, device support applications were manually programmed for the power supplies and the module areaDetector was used to control one of the CCD cameras. Finally, the current of the power supplies

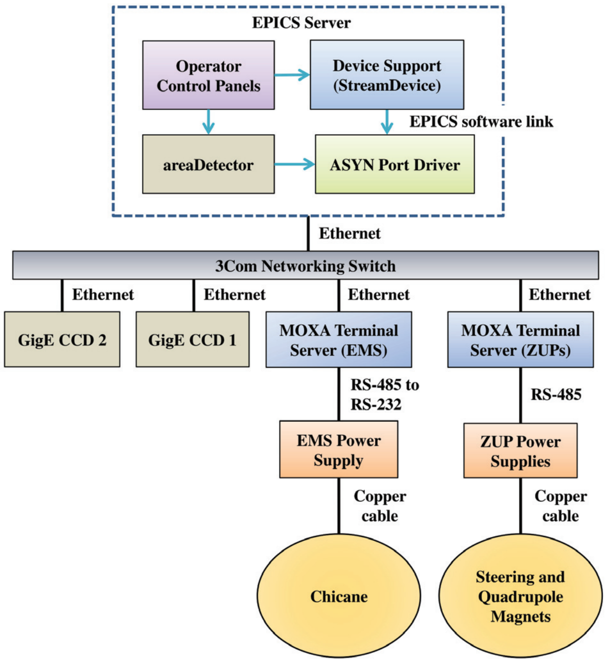

Figure 1: Experimental setup of EPICS server and hardware.

and various parameters of the cameras were controlled by manipulating process variables (PVs) using MEDM Operator Interface (OPI) panels.

DEVICE SUPPORT APPLICATIONS

In order to control the magnet power supplies and CCD cameras, the first step was to download and install the correct software to communicate to them. Then, in the case of the power supplies, make device support applications which contain the files required to create PVs. The device communication was accomplished by installing the synApps software package including ASYN, StreamDevice, and areaDetector [7, 8].

areaDetector

The program areaDetector is an EPICS module and a pre-made device support application that is specific to devices like the Prosilica GC1290 GigE CCD camera. This program contains PVs and OPI panels specific for the control of the CCD camera. So, once installed, the only things that are required are some configuration on the camera to get an image, and some file configuration in areaDetector [8]. [1]

* Mail: andranth@isu.edu

[1] For some details on ASYN and StreamDevice, see reference [9]

Copyright © 2013 by JACoW — cc Creative Commons Attribution 3.0 (CC-BY-3.0)

Camera EPICS Configuration

To get the camera to work with EPICS, the first step was to configure the Ethernet adapter with MTU 8228 and set a static IP address for the camera under Windows OS. This was accomplished by using the IPConfig program, which was packaged with the SampleViewer program [5, 10]. Then one camera was used on Linux to give live image acquisition of the beam using SampleViewer. The other was used with EPICS, also on Linux, for automatic emittance measurements [10]. To have EPICS recognize the CCD camera, the static IP address was added to the st.cmd file in the areaDetector software distribution directory. The entry looked similar to this:

- prosilicaConfig(PS1, 10.x.y.z, 50, -1)

The 10.x.y.z represents the static IP address of the camera. The variable PS1, represents a mapping for the static IP address. Whenever this PS1 is seen in files, it means the code is referring to the camera identified with 10.x.y.z. The 50 refers to the NDArray maximum buffer size for the camera. The -1 refers to the maximum bytes the driver for the camera can handle. For this project, the only thing that was changed from the st.cmd file default was the static IP address. After putting the static IP address in the st.cmd file, the control system for the camera needs to be able to locate the OPI displays used for control. To do this, all files with extension .adl, in the areaDetector software distribution directory, were copied to an "adls" directory under user home. Afterwards the envPaths file, in the same directory as *st.cmd*, was renamed envPaths.linux, and then the camera worked by running the start_epics file in that same directory [8].

Application Creation for Power Supplies

The creation of PVs for both types of power supplies were similar to each other. Therefore, in this paper, the control system for the Lambda-EMS will be treated. [2] First of all, before creating PVs for the Lambda EMS magnet power supply, the default echo setting on the Lambda EMS was disabled and a device support application was created for the PVs to live in. To create the application, the following commands were used [7]:

- makeSupport.pl -t streamSCPI dev
- makeBaseApp.pl -t ioc dev1
- makeBaseApp.pl -t ioc -i dev1

The first command makes a support directory where the files to create PVs are located. The second command creates a configure and an application directory. The third command creates an iocBoot directory, which contains the *st.cmd* file [7]. The "dev" and "dev1" are arbitrary names for the support directory, application directory, and the name of the directory, within iocBoot, where the *st.cmd* file is located. In order to create PVs, two types of files are required, protocol files and database files.

[2]For a guide on creating PVs for the TDK-Lambda ZUP magnet power supplies, see reference [9].

Protocol Files

The module StreamDevice uses protocol files to translate commands that follow the SCPI (Standard Commands for Programmable Instruments) standard into a format that EPICS records can use [9]. For example, the SCPI query command MEASURE:CURRENT? looks like this:

- getMC{out "MEASURE:CURRENT?"; in "%6f"; in "OK";}

In this case, getMC is a protocol, which sends the query command MEASURE:CURRENT? "out" to the device and gets a response "in" from the device. For the Lambda EMS there are two "in" responses, the first is a number with six floating digits, and the second is an "OK" response from the device that says that the device has received some kind of command. The other key component for making a PV is a record in a database file. [3]

Database File

The database files are where the records are located. The PVs are made by putting the protocol name (i.e. getMC) into the input or output fields in the record [7, 9]. Here is an example of a record to read the current:

```
record( ai, "$(P):BC_chicane:curr_r") {
field( DESC, "Measure Current")
field( SCAN, ".1 second")
field( DTYP, "stream")
field( INP, "@EMS.proto getMC $(PORT5) $(A)")
field( PREC, "3")
field( EGU, "Amps")
field( HOPR, "18")
field( LOPR, "0") }
```

In the record above, a field of particular interest is the INP field. This field sends the command defined as the protocol "getMC" located in the protocol file "EMS.proto" to the port and address of the device. The $(PORT5), $(A), and $(P) are all defined in the *st.cmd* file.

st.cmd File

This is an executable file that establishes a connection to the device by using an IP address, a portName, and an address. This file also loads all of the records that are in the database files. The way the IP address of either power supply was specified was by using a line similar to the following in the *st.cmd* file:

```
drvAsynIPPortConfigure("P5","134.50.A.B:4005",0,0,0)
```

Here P5 is a portName, which is an arbitrary identifier used to define $(PORT5) from records as an IP address (134.50.A.B) and a port (4005) of the MOXA terminal server [7]. In this case, P5 was used because this particular power supply was connected to the fifth port of the MOXA

[3]For more protocol file information and more on SCPI commands, see reference [9]

Copyright © 2013 by JACoW — cc Creative Commons Attribution 3.0 (CC-BY-3.0)

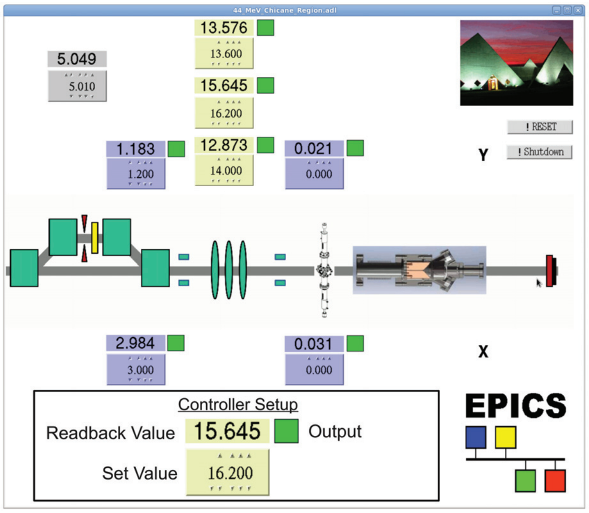

Figure 2: An MEDM OPI pannel to control and read the current of the power supplies.

terminal server. The rest (0,0,0) specifies *priority, noAutoConnect,* and *noProcessEos* [7]. To specify a database in the *st.cmd* file, a line similar to the following thing was used:

dbLoadRecords(“db/EMS.db”,“P=$(P),PORT5=P5,A=5”)

Here the EMS.db is the database file that contains all of the needed records. The macro P is used as a device identifier, in this case P is 44MeV. The PORT5 is the same as the portName. The address (A=5) for this set-up ran from 1 - 32, because there were 32 ports on each MOXA terminal server [7].

MEDM OPI

Motif Editor and Display Manager (MEDM) is an EPICS extension which provides a way to create simple OPI’s [9]. The OPI developed for use in the THz experiment to control the current of the power supplies is shown in Fig. 2. This display is cropped from the original to emphasize the magnets that were controlled by EPICS in this experiment. The boxes above each magnet control the power supply for that magnet (or magnets in the case of the chicane controller). A blown up representation of the controller is shown in Fig. 2 at the bottom. The operator changes the current set value digit by digit by clicking the arrows, and the actual readback value is represented in the box above the controller. For the TDK-Lambda ZUP power supplies, there is an additional indication box next to the readback that represents whether the output of the power supply is on (green color) or off (red color). The buttons on the right hand side of the display are to “RESET” the power supplies current values to 0, and “Shutdown” the power supplies at the end of operation. [4]

[4]For information on the OPI used for the camera control, see references [8, 10]

CONCLUSION

The goal for this project was to make an EPICS control system for a set of TDK-Lambda ZUP power supplies, Lambda EMS power supplies, and Prosilica GigE cameras. This was accomplished by using ASYN and StreamDevice to communicate with the power supplies and convert SCPI commands into PVs. Then an MEDM OPI was developed to control and monitor those PVs. To control the cameras, areaDetector was installed which came with the needed PVs and MEDM OPI displays. Using this EPICS accelerator control system, the operators and scientists involved in the joint IAC-RadiaBeam project were able to successfully detect THz radiation from the 44 MeV linac [11, 12].

ACKNOWLEGEMENTS

I would like to thank Jeff Bush, the technical group leader at TDK-Lambda Inc., for all of his help configuring and troubleshooting problems with the Lambda EMS power supplies through email correspondence. I would also like to thank Dr. Kukhee Kim and Dr. Debbie Rogind for teaching me a lot about developing displays through my internships at SLAC national accelerator laboratory in 2011 and 2012.

REFERENCES

[1] `http://www.iac.isu.edu`

[2] `http://www.radiabeam.com/index.html?gclid=CLDG1p2K1bYCFaF7QgodODoAmw`

[3] `http://www.tdk-lambda.com/products/sps/ps_adj/zup/indexe.html#`

[4] `http://www.us.tdk-lambda.com/hp/product_html/emspower1u.htm`

[5] `http://www.alliedvisiontec.com/emea/products/cameras/gigabit-ethernet/prosilica-gc/gc1290.html`

[6] `http://www.moxa.com/product/nport_6650.htm`

[7] E. Norum,“HowToDoSerial(StreamDevice),” Argonne National Laboratory; `http://www.aps.anl.gov/epics/modules/soft/asyn/HowToDoSerial_StreamDevice.html`

[8] M. Rivers, “areaDetector: EPICS Area Detector Support,” The University of Chicago; `http://cars9.uchicago.edu/software/epics/areaDetectorDoc.html`

[9] A. Andrews *et al.*, in *Proc. IPAC2012*, New Orleans LA, USA.

[10] C. Eckman *et al.*, in these proceedings.

[11] Y. Kim *et al.*, in these proceedings.

[12] A. V. Smirnov *et al.*, in these proceedings.

Copyright © 2013 by JACoW — cc Creative Commons Attribution 3.0 (CC-BY-3.0)