

# DEVICE CONTROL TOOL FOR CEBAF BEAM DIAGNOSTICS SOFTWARE\*

P. Chevtsov, Jefferson Lab, Newport News, VA 23606, USA

## Abstract

Continuously monitoring the beam quality in the CEBAF accelerator, a variety of beam diagnostics software created at Jefferson Lab makes a significant contribution to very high availability of the machine for nuclear physics experiments. The interface between this software and beam instrumentation hardware components is provided by a device control tool, which is optimized for beam diagnostics tasks. As a part of the device/driver development framework at Jefferson Lab, this tool is very easy to support and extend to integrate new beam instrumentation components. All device control functions are based on the configuration (ASCII text) files that completely define the used hardware interface standards (CAMAC, VME, RS-232, GPIB, etc.) and communication protocols. The paper presents the main elements of the device control tool for beam diagnostics software at Jefferson Lab.

## 1. INTRODUCTION

The CEBAF accelerator provides nuclear physics experiments at Jefferson Lab with high quality electron beams. High quality of beams means not only their excellent (more than 80%) polarization factor and precise position and shape on targets, but also an outstanding relative energy spread, which is as small as twenty parts in one million. The smaller the beam energy spread the better the resolution of nuclear physics experiments, and physicists can see more details inside nuclei.

CEBAF beam quality is continuously monitored by an advanced beam diagnostics system. The system includes such popular beam diagnostics devices as wire scanners, beam viewers, optical transition radiation (OTR) monitors, synchrotron light monitors (SLM) and interferometers (SLI) as well as a variety of software dealing with the main functions of these devices and their data analysis. The software is based on a device control tool that was designed and created at Jefferson Lab as a part of the device/driver development framework.

## 2. DEVICE CONTROL TOOL AND ITS BASIC COMPONENTS

The device control at Jefferson Lab is based on the EPICS toolkit [1], which allows easy system extensions at all control levels. To interface with control devices, the elements of a distributed real-time EPICS database or EPICS records need to know the device communication

protocols and must be equipped with device/driver support software. All this can easily be provided on the basis of the device/driver framework that has been created at Jefferson Lab. The main idea of the framework is to make the device control a relatively simple task even for non-specialists in accelerator control software. Based on this idea, the device/driver support software for beam diagnostics applications at Jefferson Lab was written to be responsible for all essential control functions for beam diagnostics devices as well as very efficient mechanisms for troubleshooting of any possible problems with these devices during accelerator runs. The software makes up a powerful device control tool consisting of the next key components: a Device Communication Hardware Handler, a Serial Port Handler, a Device Support Database Handler, and a Device Control and Data Processing Support Module (Fig.1).

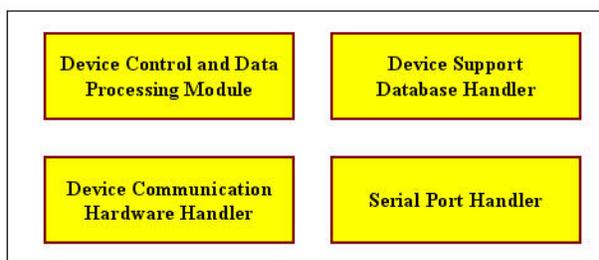


Figure 1. Main components of the Device Control Tool.

### 2.1 Device Communication Hardware Handler

The Device Communication Hardware Handler registers hardware components used by a control computer (IOC) to communicate with control devices. The Handler is based on CAMAC, IndustryPack (IPAC or IP), Com on Serial, and GPIB support libraries created at Jefferson Lab [2-4]. Hardware is registered with the use of a very limited number of library calls, the most important of which deals with the type of the hardware:

```
stat= initDeviceHw (HW TYP, PARAM_STR) (1)
```

Here stat is the registration status (traditionally, the status is equal to 0 if everything is OK and contains an error code in case of a failure), HW TYP is the device hardware type, and PARAM\_STR is a string consisting hardware specific initialization parameters.

If HW TYP is VME or CAMAC then the hardware does not require other registration calls but the PARAM\_STR must contain the information about a used VME board (such as HYTEC VSD 2992 CAMAC serial highway driver card or Maxvideo 200 image processor, for

\*Notice: Authored by Jefferson Science Associates, LLC under U.S. DOE Contract No. DE-AC05-06OR23177. The U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce this manuscript for U.S. Government purposes.

example), its I/O base addresses (A16, A24, A32), and an interrupt number (or interrupt numbers), if applicable.

When HWTYPE is IPAC, then the PARAM\_STRING must provide the information about a used carrier board (SYSTRAN, TVME200, VIPC616, etc.) type and its I/O base addresses as well as assign to this board its number (CBN) in the system. At that all IP modules used on slots available on this board require one additional registration call:

```
stat= initIPslot(CBN, SLN, PAR_STR), (2)
```

where stat is the module registration status, CBN is a carrier board number, SLN is a slot number on this carrier board, the PAR\_STR is a string with information about the module type (GPIB, IP-OCTAL232, IP-OCTAL485, etc.) and an interrupt number used by this module, if applicable.

When hardware is registered, the Device Communication Hardware Handler software makes several checks to make sure that specified cards are installed in a VME crate or on an IP carrier board as well as have valid manufacturer and model IDs.

## 2.2 Serial Port Handler

If an IP serial control module is installed in one of the slots on an IP carrier board, it can usually be used to talk with several serial devices connected to its data communication ports. For example, eight serial devices can be connected to each IP-OCTAL232 module. Serial ports available for device controls are registered by the Serial Port Handler software with the use of the next library call:

```
stat= initSerialPort(CBN, SLN, PN, P_STR) (3)
```

Here stat is the port registration status, CBN and SLN are carrier and slot numbers, PN is the port number, P\_STR is a string that defines basic serial communication parameters, such as the baud rate, number of stop bits, data word size, parity, and so on.

## 2.3 Device Support Database Handler

The Device Support Database Handler registers device communication protocols. For example, in case of GPIB or serial devices, each device protocol fragment consists of a sequence of input and output commands with or without data as well as some additional parameters affecting data transfer, such as separators, terminators, timeouts, etc. Protocol fragments have their names, which are usually associated with particular device actions (such as enable, start, stop, etc.). For each device type, all protocol fragments are combined in one device protocol file. All device protocol files reside in one "special" directory (~iocs/DATA/deviceProtocols) in the control computer file system. The protocol file name, the protocol fragment name, and the information about the data communication channel (carrier board number, slot number, and serial port number or GPIB address) are

referenced by INP and OUT fields of EPICS database records.

When an EPICS database is loaded on the IOC, the Device Support Database Handler parses protocol files referenced by records, defines the structure of all device control data streams and make them a part of the standard EPICS device/driver support.

## 2.4 Device Control and Data Processing Support Module

The Device Control and Data Processing Support Module consists of a set of device data processing, diagnostics, and service executable utilities and scripts. For each IOC and for each particular type of the used device control hardware, device hardware registration calls mentioned above are combined into one device configuration file. All these files are located in a second "special" directory (~iocs/DATA/devCommHw) in the control computer file system at Jefferson Lab. When the device support software is loaded on the IOC, service scripts read all device configuration files existing for this IOC and make registration calls (1)-(3) for required data communication channels.

Very important functions for accelerator operations are provided by Device Control and Data Processing Support Module software with respect to wire scanner controls and beam in age processing systems.

Images from dozens of beam viewers in the CEBAF accelerator are captured by TV cameras and displayed in the machine control room on the main control display wall as well as on numerous TV screens all over Jefferson Lab. The TV signals are also fed into pipelined high performance image processing systems Maxvideo 200. The main advantage of the pipeline technology is that the pixel manipulation can be done while the image is being digitized and directed to the image memory. As a result basic image processing operations can be implemented at the full 30 Hz frame rate of the standard NTSC video signal. To calculate the size of the beam from its viewer image, Maxvideo software needs to know such important parameters as pixel spacing in physical units (for example, mm per pixel), a region of interest to mask pixels outside it, an image acquire gain, etc. For each viewer, these parameters are determined during a standard viewer calibration procedure and are placed in a viewer calibration file. All these files reside in another "special" directory (~iocs/DATA/calibViewers) in the control computer system. When Maxvideo 200 begins processing images from a particular beam viewer, Device Control and Data Processing Support Module software reads the calibration file for this beam viewer and enters this information into a control system database, which makes it available for any application running on the accelerator control computer network.

The most popular beam profile measurement devices at accelerator facilities are wire scanners (WS). The idea of WS is transparent. A stepper motor moves thin wires through the beam. The amount of beam particles intercepting a wire changes depending on the beam

profile. The advanced wire scanner control software created at Jefferson Lab integrated WS implemented in different hardware standards (CAMAC and VME) into one homogeneous system that is very easy to use and support for accelerator operations [3]. The Device Control and Data Processing Support Module software has added more functionality to this system by implementing a semi-automatic WS mode. In this mode, the wires can be moved continuously back and forth between any two specified points in the beam pipe, with different velocities and acceleration. The software collects the information about beam profiles and settings of upstream magnets, which can immediately be used, for example, for beam emittance calculations.

The Device Control and Data Processing Support Module software also provides powerful mechanisms for troubleshooting of any problems with beam diagnostics devices in operations. For each registered device communication channel (serial port, GPIB address, etc.) it spawns a task monitoring data streams flowing through this channel. In case of any troubles with the data flow, the software analyzes the situation and generates EPICS alarms associated with the device failure sources (such as a serial CAMAC highway collapsed loop, a wire scanner stepper motor power supply failure, problems with video signals for Maxvideo systems, etc.). The EPICS Alarm Handler catches these alarms and provides accelerator operators and beam diagnostics specialists with standard Display Guidances [5] that explain in detail what should be done to solve device communication problems.

### 3. DEVICE CONTROL TOOL AT WORK

Above we have presented the basic elements of the Device Control Tool at Jefferson Lab. With the use of this Tool, the device control does not require any software coding for connecting a new device to the control system and is based on configuration (ASCII text) files, which define device communication hardware and protocols. We note that the main documentation web page of the controls software group at Jefferson Lab has detailed information about the Device Control Tool and numerous examples how to use it.

In particular, if we have, for instance, a serial (RS-232) device with the name `controlDEV` and want to connect it to one of the ports of an IP-octal232 module sitting on a TMS200 carrier board and control this device from the control computer with the name `iocABC`, then we must perform the following steps.

a) Log on any control software development computer, go to the directory `~iocs/DATA/devCommHw`, and add the information (if it doesn't exist already) about the used carrier board, slot, module and parameters of the serial port to the device configuration file with the name `iocABC.hw` in the form of the standard Tool library calls (1)-(3).

b) Go to the directory `~iocs/DATA/deviceProtocols` and create a file with the name `controlDEV` (if it doesn't

exist already there) consisting all fragments of the device communication protocol.

c) Create a new EPICS database handling this serial device.

d) Go to the directory `~iocs/iocABC` and add the information about a new control device to the startup all file.

e) Connect the device to the serial port with a proper cable.

d) Reboot the `iocABC` computer to activate the new device communication channel.

When the IDC is rebooted, the Device Control and Data Processing Support Module software reads the device configuration file located in the directory `~iocs/DATA/devCommHw` and activates the device communication serial port. The Device Support Database Handler examines the EPICS database associated with the device, reads the device protocol file in the directory `~iocs/DATA/deviceProtocols` and adjusts the Serial Port Handler software to deal with all device control data streams. At this point, serial port control (which is a part of the Serial Port Handler) and monitoring (that is provided by the Device Control and Data Processing Support Module) tasks are spawned and take control on the device.

### 4. CONCLUSIONS

The Device Control Tool has been in operations at Jefferson Lab for the last few years. Almost all control and image processing software for CEBAF beam diagnostics applications is based on this Tool. The software is extremely reliable and very easy to use, support, and extend to integrate new beam instrumentation components.

### ACKNOWLEDGMENTS

The author would like to thank M. Bickley, A. Hutton, and K. White for their support of this work.

### REFERENCES

- [1] B. Dalesio et al., "The Experimental Physics and Industrial Control System Architecture: past, present and future", *NIM*, A 352 (1994), p.179-184.
- [2] P. Chevtsov, S. Schaffner, "Information-Control Software for Handling Serial Devices in an EPICS Environment", *ICALEPCS-2001*, San-Jose, CA, USA, 2001.
- [3] P. Chevtsov, "New Control Software for CEBAF Wire Scanners", *ICALEPCS-2003*, Gyeongju, Korea, 2003.
- [4] M. Bickley, P. Chevtsov, T. Lamieau, "Device Configuration Handler for Accelerator Control Applications at Jefferson Lab", *ICALEPCS-2003*, Gyeongju, Korea, 2003.
- [5] J. Anderson, "Alarm Handler User's Guide", [www.aps.anl.gov/epics/extensions/alh/index.php](http://www.aps.anl.gov/epics/extensions/alh/index.php)