NEW GPIB CONTROL SOFTWARE AT JEFFERSON LAB*

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

ABSTRACT

The control of GPIB devices at Jefferson Lab is based on the GPIB device/driver library. The library is a part of the device/driver development framework. It is activated with the use of the device configuration files that define all hardware components used in the control system to communicate with GPIB devices. As soon as the software is activated, it is ready to handle any device connected to these components and only needs to know the set of commands that the device can understand. The old GPIB control software at Jefferson Lab requires the definition of these commands in the form of a device control software module written in C for each device. Though such modules are relatively simple, they have to be created, successfully compiled, and supported for all control computer platforms. In the new version of GPIB control software all device communication commands are defined in device protocol (ASCII text) files. This makes the support of GPIB devices in the control system much easier.

INTRODUCTION

The General Purpose Interface Bus or GPIB was invented in the late of 1960's by the Hewlett Packard company. The idea was to provide a reliable bus system for connecting instruments from different manufactures and computers. In the beginning of 1970's GPIB became IEEE 488 standard. The standard defines the interface that can easily be added to almost any instrument. Instruments have a very limited control over the interface. The control functions for all instruments on the bus are performed by the active bus controller.

Hundreds of various control devices at Jefferson Lab are handled over GPIB. The control is based on the GPIB device/driver library that has been created at the Lab [1]. The library consists of a common GPIB control block and a GPIB hardware interface block (see Fig. 1).

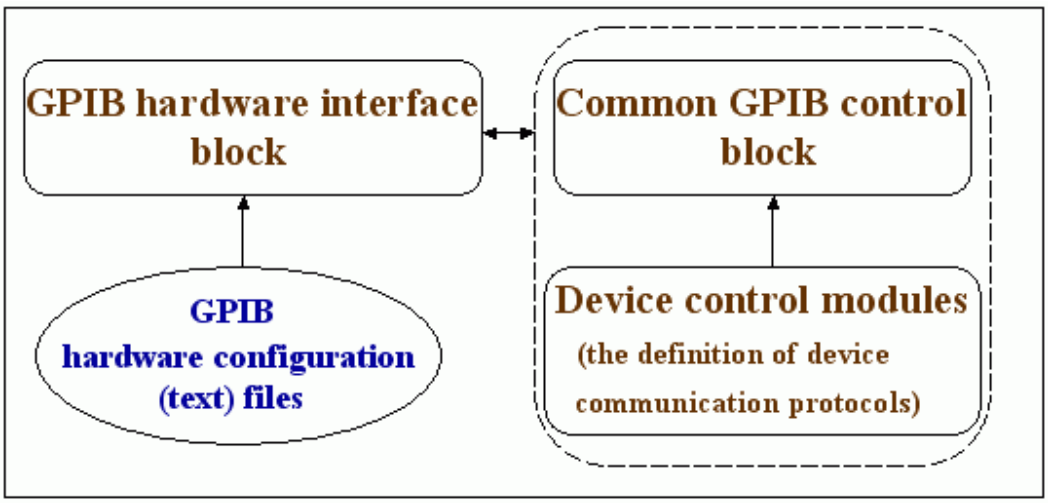

Figure 1: Basic components of the GPIB device/driver library at Jefferson Lab.

GPIB HARDWARE INTERFACE BLOCK

The hardware interface block deals with GPIB communication hardware, the most popular of which is based on IP-488 cards in the Industry Pack (IPAC) standard [2]. The Industry Pack bus is a point-to-point bus from a carrier board to an IPAC card. IPAC cards are passive on the bus and can only be accessed by a carrier board. The Industry Pack provides a convenient and inexpensive way of implementing a wide range of control, I/O, analog and digital functions and is used at Jefferson Lab as

*Supported by DOE Contract #DE-AC05-84ER40150.

the basic connection method for integrating control devices into the accelerator control system.

Each GPIB communication card is activated with the use of only one GPIB device/driver library call:

status = initGpibLib(CBN, SLN, intNum).

Here **status** is the card activation (or registration) status (OK or failure) that is returned by the software library, **CBN** is a carrier board number, **SLN** is the name (starting from "A") of the slot housing the card and **intNum** is the interrupt vector number that will be used by the card for I/O operations.

During the activation process, the GPIB device/driver library performs several checks to make sure that the specified carrier board and slot are available in the system as well as the interface card is installed and has valid manufacturer and model ID labels in its PROM. It also calls all necessary GPIB initialization routines and connects the interrupt service routine to a particular interrupt vector number. Each activated card provides the communication bus for controlling GPIB devices connected to it. For each control computer (IOC) talking to GPIB devices, all GPIB card activation calls are combined into one GPIB configuration file. All such files reside in one directory in the control computer file system. At start up time, a specially designed real-time OS (vxWorks) shell script downloads the proper GPIB configuration file into the IOC and activates the required communication lines.

COMMON GPIB CONTROL BLOCK

The common GPIB control block is responsible for the device communication protocols. Each GPIB interface card is served by a separate control task. The task handles the data streams between the IOC and GPIB devices and the operation timeouts. Internal data parsing procedures implemented in the common GPIB control block make these data streams available for any application in the system. It is very important for fast and efficient troubleshooting of any possible device communication problems.

With the use of the common GPIB control block in an EPICS environment (the basic control environment at Jefferson Lab), to add a new GPIB device to the control system, all that has to be done by application developers is to define the communication protocol for this device and create a standard EPICS database implementing this protocol.

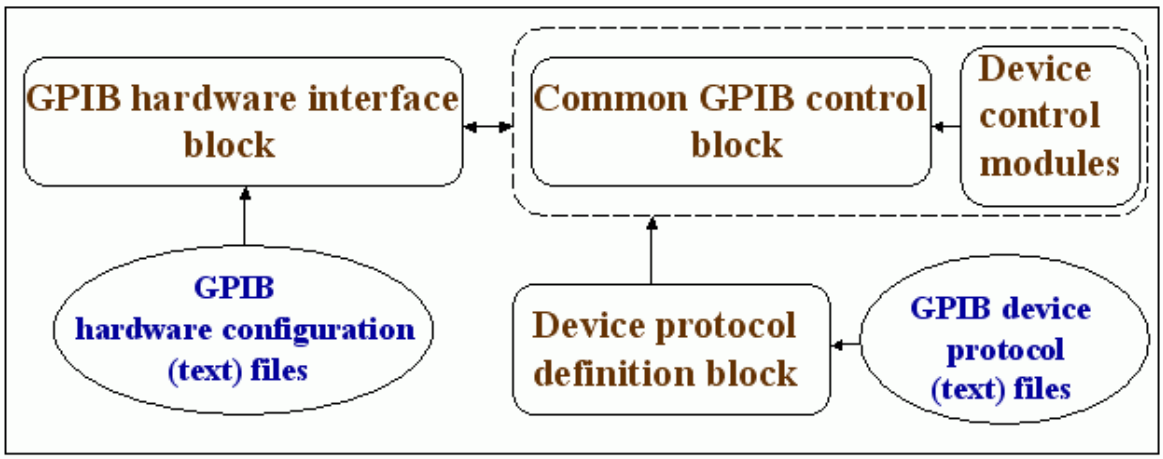

Figure 2: GPIB device/driver library after its recent modifications.

RECENT MODIFICATIONS OF GPIB SUPPORT SOFTWARE

The GPIB device/driver library has successfully been working at Jefferson Lab for more than two years. Until recently, the definition of the device communication protocol required writing a device control module in the C language for each type of device. Though such modules are relatively simple, they have to be created, successfully compiled, and supported for all control computer platforms. It was the only place in the GPIB device/driver library that required software coding before connecting a new device to the control system. To eliminate this requirement, the common GPIB control block was modified and a new component, a GPIB device protocol definition block, was added to the GPIB support software (see Fig. 2).

After these modifications, the device communication protocols can be defined not only by writing and compiling C files (the old way) but also with the use of very simple GPIB device protocol (ASCII text) files. These files look similar to the stream device protocol files developed by D. Zimoch and described in detail in his presentation [4].

GPIB device protocol files contain GPIB protocol fragments. Each protocol fragment has a name usually associated with some control action of the device as well as a valid command or a sequence of commands for this action, supported data formats, and device operation timeouts. All protocol fragments for each device type are combined in one GPIB protocol file. All GPIB protocol files reside in one directory of the computer control system. The protocol file name, the protocol fragment name, and the information about the device address on the bus are referenced by INP and OUT links of EPICS database records responsible for the device control. At the IOC startup time, the GPIB device protocol block software analyzes the protocol file and defines the device control functions for all such records. This makes the device available for the control system.

CONCLUSION

The new GPIB device/driver library at Jefferson Lab eliminates the requirement of any software coding for connecting GPIB devices to the control system. The definition of the data communication hardware and control protocol can now be based on the configuration (ASCII text) files that are very easy to understand, create, and support. The library makes the GPIB device control at Jefferson Lab a relatively simple task even for non-specialists in accelerator control software. After numerous successful tests of the new GPIB device/driver library, we plan to gradually switch the control of all GPIB devices to this software.

REFERENCES

[1] M. Bickley, P. Chevtsov, T. Larrieu, "Device Configuration Handler for Accelerator Control Applications at Jefferson Lab", ICALEPCS 2003, Gyeongju, Korea, 2003.

[2] M. Timmerman, "Comparison of Different Mezzanine Buses", Real-Time Magazine, 97-1.

[3] P. Chevtsov, S. Schaffner, "Information-Control Software for Handling Serial Devices in an EPICS Environment", ICALEPCS 2001, San Jose, CA, USA, 2001.

[4] D. Zimoch, "Stream Device Driver for EPICS 3.14", EPICS Collaboration Meeting, Diamond, UK, 2003. www.diamond.ac.uk/Technology/Controls/EPICS_programme.htm