J. Synchrotron Rad. (1998). **5**, 600–602

SCIPE – a simple control interface protocol for experimenters

J. P. G. Quintana

DND-CAT Synchrotron Research Center, APS/ANL Sector 5, Bldg 400, 9700 S. Cass Avenue, Argonne, IL 60439, USA. E-mail: jpq@guava.dnd.aps.anl.gov

(*Received 4 August 1997; accepted 17 September 1997*)

The simple control interface protocol for experimenters (SCIPE) used at Sector 5 of the Advanced Photon Source is presented. This interface provides a generic method for data-acquisition programs to control generic *actuators* and *detectors* on a synchrotron beamline. The implementation used by the DuPont–Northwestern–Dow Collaborative Access Team, based on readily available hardware, is also described.

Keywords: control interface; detectors.

1. Introduction

Control systems for synchrotron beamlines must integrate a wide variety of devices, including motor controllers, vacuum valves, vacuum controllers, counters *etc.*, as well as be able to accommodate equipment that synchrotron users bring to the beamline. In addition, the beamline hardware must easily be integrated into data-collection programs provided by experimenters or third-party vendors. Programmers and scientists that are not familiar with the beamline may need to work on these projects. Consequently, to ease integration, the beamline control system should provide a simple model and interface for application programs. In addition, the interface should also not present a high learning curve for developers. While Mooney *et al.* (1996) have proposed that EPICS (experimental physics and industrial control system) be used as a foundation for a beamline control system, the majority of beamline components (*e.g.* mechanical slits, tables, valves *etc.*, devices which communicate over RS232 communications) do not require the complexity or expense of the hard real-time processing capabilities that EPICS provides. At the DuPont–Northwestern–Dow Collaborative Access Team at Sector 5 of the Advanced Photon Source, we have been developing an ASCII-based simple control interface protocol for experimenters (SCIPE) to control the majority of beamline components, including monochromators, slits, counters, servo motors and stepper motors.

2. SCIPE protocol description

The SCIPE protocol describes two types of generic devices. These devices are *actuators* and *detectors*. All of the hardware on the beamline is placed into these two categories. Each actuator or detector can be multidimensional. An example of a multidimensional actuator would be an instrument table where each element of the actuator would be a motion of the table. An example of a multidimensional detector would be a multichannel analyser or a set of counters that are gated off using a monitor counter or a timer. To move an actuator, the actuator load register is preset to the desired location. Then a move command is issued to cause the motion. Each detector has a preset register. If the detector can be gated off using the signal from one of its channels, the preset register is used to set the value of this channel.

Table 1
Command list for 'scipe' objects.

⟨det1⟩, ⟨det2⟩ *etc.* denote the names of *detector* objects. ⟨act1⟩, ⟨act2⟩ *etc.* denote the names of *actuator* objects

Command	Command description
desc	Returns a description of this implementation
actcnt	Returns the number of actuators on interface
detcnt	Returns the number of detectors on interface
actname ⟨value⟩	Returns the name of actuator ⟨value⟩
detname ⟨value⟩	Returns the name of detector ⟨value⟩
version	Returns the current version of scipe
move ⟨act1⟩ ⟨act2⟩...	Issues a move command to act1, act2, ...
halt ⟨act1⟩ ⟨act2⟩...	Issues a halt command to act1, act2, ...
stop ⟨det1⟩ ⟨det2⟩...	Issues a stop command to det1, det2, ...
start ⟨det1⟩ ⟨det2⟩...	Issues a start command to det1, det2, ...
quit	Quits the current session

The protocol is an ASCII-based master/slave polling protocol. The application program sends ASCII commands to the SCIPE controller and receives a numerical response code followed by any subsequent information. The format of all commands is

⟨`object`⟩⟨`command`⟩⟨`argument 1`⟩⟨`argument 2`⟩⟨`argument 3`⟩....

The ⟨object⟩ is the name of the actuator or detector. There is a special object called 'scipe' which can be used to obtain information about the devices on the interface. The 'scipe' object can also be used to issue simultaneous commands to different objects on the interface. Responses from the controller take the form of:

⟨`response code`⟩⟨`optional text strings`⟩

Tables 1, 2 and 3 contain lists of the commands that apply to 'scipe', actuator and detector objects for SCIPE version 1.0. Response codes are given in Table 4.

The SCIPE protocol only includes those commands that would be required by a data-collection program to manipulate the devices. There are no generic set-up commands since these would not be the same for different types of actuators (*e.g.* a digital-to-analog converter does not have the same set-up parameters as a stepper motor). Set-up tools need to be provided by the SCIPE programmer for each device that is supported on the SCIPE interface. While this makes writing the driver slightly more complicated, it simplifies the interface as well as the data-collection program. By keeping the interface simple, data-collection programs do not require special precompiled libraries or drivers. Also, non-programmers can easily interact with the devices. Since the data-collection program can only perform 'safe' operations, there is no need to support extra layers of security in the data-collection application. There is also no generic 'set position' command as part of the interface. SCIPE actuators are expected to retain their own position information and retain the information through power on/power off cycles. However, in those cases where a beamline designer wishes to provide a 'set command' for a position or motor speed *etc.*, this is easily performed by creating a virtual actuator which sets those values in the controller. (For example, if there is a MonoEnergy actuator tied to a monochromator, a designer might have a MonoEnergySet actuator which sets the value of the MonoEnergy without moving the monochromator.)

Table 2
Command list for *actuator* objects.

Command	Command description
desc	Return an ASCII description of the actuator
dim	Return the dimensionality of the actuator
position	Return the current position
load ⟨value1⟩ ⟨value2⟩...	Load the values into load register(s)
move	Move actuator to position in load register(s)
halt	Halt the actuator
movenow ⟨value1⟩ ⟨value2⟩ *etc*...	Combined load and move
status	Returns status
upperlimit	Returns upper limit(s)
lowerlimit	Returns lower limit(s)

Offsets, combined motions, dial *versus* user coordinates can all be handled in this same way.

The SCIPE protocol does not specify a communications medium. It only specifies syntax. Consequently, it is suitable for use over serial communications or by using TCP/IP network sockets to create a distributed control system. Fig. 1 provides an example of a SCIPE session that was made by telneting into the scipe TCP/IP port on a remote controller.

3. Implementation

The hardware for the control system used by the DND-CAT at Sector 5 of the Advanced Photon Source utilizes commercial Intel class motherboards running the Linux operating system. Our choice for using Linux as the basis for this application came from previous work on integrating programmable logic controllers with Linux (Quintana & Jemian, 1996). Linux has also been used on PC/104 hardware to perform experimental data acquisition on board a US Space Shuttle flight (Kuzminsky, 1997). Since our beamline control workstations also operate under Linux, we only need to maintain one type of computer hardware and operating system for both workstations and controllers. By utilizing high-volume commercial computer components, we take advantage of low cost, high availability and upgrade paths provided by the computer marketplace. The low cost also makes it possible to construct a true distributed control system where each piece of major equipment (*e.g.* a monochromator or diffractometer) can be controlled with its own computer/controller. So far, we have built dedicated controllers using the SCIPE interface for white-beam and monochromatic beamline slits, monochromators, experimental shutters, table motions for a specially built table for a Rigaku *R*-axis IV single-crystal station, a two-circle diffractometer and an EXAFS experimental station. Beamline staff can modify configuration information such as motor speeds by editing configuration files from protected Linux computer accounts.

The SCIPE protocol is implemented by software processes written using Tcl (Ousterhout, 1994; Welch, 1997) with C extensions to support the hardware. While faster processing could be achieved using compiled C, we chose Tcl initially for the ease of programming. Since Tcl easily provides for embedding compiled C commands, we can recode any time-critical tasks in C as they are required. The controller machines contain only a floppy boot drive and hardware interface cards. The controllers boot the operating system kernel from the floppy and download their programs from a main server over the network. The SCIPE processes take their command input over the network using TCP/IP network sockets. Since the SCIPE protocol itself does not support access security, security is accomplished by only allowing

Table 3
Command list for *detector* objects.

Command	Command description
desc	Returns a description
dim	Returns the dimensionality of the detector
clear	Clears the detector
preset ⟨value⟩	Presets the monitor in the detector
start	Starts the counter
stop	Stops the counter
read	Reads the counter
status	Returns the counter status

Table 4
SCIPE response codes.

Response code	Response to command from	Meaning
100	All	OK
101	All	OK with result string following
200	*actuator*	Not moving
201	*actuator*	Moving
202	*actuator*	General fault
203	*actuator*	Upper-bound fault
204	*actuator*	Lower-bound fault
205	*actuator*	Requested load value exceeds upper limit
206	*actuator*	Requested load value is below lower limit
300	*detector*	Not collecting
301	*detector*	Collecting
302	*detector*	General fault
500	All	Command not found
501	All	Object not found
502	All	Error in command
503	All	Error in object processing

```
$ telnet akee1 8200
Trying 164.54.149.57...
Connected to akee1.dnd.aps.anl.gov.
Escape character is '^]'.
> scipe actcnt
< 101 8
> scipe actname 1
< 101 RaxisTable_x1
> scipe actname 2
< 101 RaxisTable_x2
> RaxisTable_x1 position
< 101 0.0
> RaxisTable_x1 status
< 200 NOT_MOVING
> RaxisTable_x1 movenow 2.0
< 100
> RaxisTable_x1 position
< 101 2.0015200376285764
> RaxisTable_z position
< 101 -0.0
> RaxisTable_z movenow 5.0
< 100
> RaxisTable_z status
< 201 BUSY
> RaxisTable_z position
< 101 5.0
> RaxisTable_z status
< 200 NOT_MOVING
> scipe quit
Connection closed by foreign host.
```

Figure 1
Sample TCP/IP 'scipe' session to control a custom table for an *R*-axis IV single-crystal crystallography station. < denote commands to the controller and > denote responses and are not part of the protocol. In this example, akee1 is the name of the scipe controller and 8200 is the TCP/IP port number that the controller is listening on.

specific machines to connect to the SCIPE ports. In practice, each SCIPE process becomes a slave to one or two beamline control computers. The typical round-trip response time to process an SCIPE request from a beamline computer (*e.g.* a simple command such as a detector start) is 5.2 ms over 10 Mbit s^{-1} Ethernet. The same command executing on a local SCIPE controller takes approximately 1.2 ms to execute. Consequently, the network latency is the limiting factor in processing SCIPE commands in the current implementation using Tcl. We have repeated these tests from a remote computer located 40 miles away from the beamline connected to the Internet through a commercial Internet service provider. The average round-trip processing time in this case is 250 ms.

The current hardware supported by the DND SCIPE implementation includes the McLennan SM9464 servo-motor controller used by the Daresbury-Laboratory-designed monochromator built by Vacuum Generators, stepper-motor drivers using the Technology 80 Model 24 Industry Pack module, servo-motor drivers using the Technology 80 Model 50A Industry Pack module, programmable logic controllers using the Koyo PLC DirectNet protocol, 32-bit presettable timer/counters using boards based on the AMD9513 chip, 16-bit 100 kHz analog-to-digital converters with a 4 MByte memory board from Computer Boards, a Computer Boards 12-bit D/A board and Stanford Research Systems SR570 current amplifiers. Since the SCIPE protocol operates at the device layer of a data-acquisition system, it could be used by more elaborate systems, such as EPICS (Mooney *et al.*, 1996).

4. Conclusions

We have presented the simple control interface protocol for experimenters (SCIPE) as a generic method for interfacing between data-acquisition applications and beamline components on a synchrotron beamline. We have described an implementation using readily available commercial hardware and public domain software. Currently, all of the major components on the DND-CAT beamlines at Sector 5 of the Advanced Photon Source are interfaced using this protocol.

This work was performed at the DND-CAT Synchrotron Research Center. The DND-CAT Synchrotron Research Center is supported by E. I. Du Pont de Nemours & Co., The Dow Chemical Co., the US National Science Foundation through grant DMR-9304725, and the State of Illinois through the Department of Commerce and the Board of Higher Education grant IBHE HECA NWU 96.

References

Kuzminsky, S. (1997). *Linux J.* **39**, 32–37.
Mooney, T. M., Cha, B. K., Goetze, K. A., Reid, D. R. & Winans, J. R. (1996). *Rev. Sci. Instrum.* **66**(9), 1–5.
Ousterhout, J. (1994). *Tcl and the TK Toolkit.* Reading, MA: Addison-Wesley.
Quintana, J. P. & Jemian, P. (1996). *Rev. Sci. Instrum.* **66**(9), 1–5.
Welch, B. (1997). *Practical Programming in Tcl and Tk.* Upper Saddle River, NJ: Prentice-Hall.