AMOR–Reference Manual

April 4, 2005

Dr. Mark Könnecke
Labor für Neutronenstreuung
Paul Scherrer Institut
CH–5232 Villigen–PSI
Switzerland

Contents

1 Introduction **4**
- 1.1 Interaction with SICS . . . 4
  - 1.1.1 SICS Invocation . . . 4
  - 1.1.2 The AMOR Client . . . 6

2 General User Commands **9**
- 2.1 Drive commands . . . 9
- 2.2 AMOR Motors . . . 9
  - 2.2.1 Physical Motors . . . 9
  - 2.2.2 AMOR 2 Theta . . . 11
- 2.3 Logging your activity . . . 12
  - 2.3.1 LogBook command . . . 12
  - 2.3.2 The Commandlog . . . 13
- 2.4 Batch Processing in SICS . . . 13
  - 2.4.1 The Batch Buffer Manager . . . 13
  - 2.4.2 The Token Command . . . 17
- 2.5 Miscellaneous AMOR Commands . . . 17
  - 2.5.1 Shutter Control . . . 17

3 AMOR in Single Counter Mode **18**
- 3.1 AMOR in Single Counter Mode . . . 18
- 3.2 The Scan Command . . . 18
  - 3.2.1 Center Scan . . . 19
  - 3.2.2 Simple Scan . . . 19
  - 3.2.3 Peak And Center . . . 20

4 AMOR in Time–Of–Flight Mode **21**
- 4.1 AMOR in Time Of Flight Mode . . . 21
  - 4.1.1 Configuring the Histogram Memory . . . 21
  - 4.1.2 Configuring the Chopper . . . 22
- 4.2 Counting Commands . . . 22
- 4.3 AMOR Data Storage . . . 22

5 Advanced Topics **24**
- 5.1 Sample Environment Devices . . . 24
  - 5.1.1 SICS Concepts for Sample Environment Devices . . . 24
  - 5.1.2 Sample Environment Error Handling . . . 24

5.1.3 General Sample Environment Commands . . . 25
5.1.4 Special Environment Control Devices . . . 28
5.2 Histogram memory . . . 35
5.2.1 Configuration . . . 36
5.2.2 Histogram Memory Commands . . . 37
5.3 SICS motor handling . . . 38
5.3.1 The motor parameters . . . 39
5.3.2 Motor Error Handling Concepts . . . 40
5.4 SICS counter handling . . . 41
5.5 Serial Port Direct Access . . . 42
5.5.1 Invocation . . . 42
5.5.2 Usage . . . 42
5.5.3 System Commands . . . 43
5.5.4 Configuration Commands . . . 44
5.6 SICS Trouble Shooting . . . 45
5.6.1 Looking at Log Files . . . 45
5.6.2 Starting SICS . . . 46
5.6.3 Stopping SICS . . . 46
5.6.4 Restart Everything . . . 47
5.6.5 Checking SICS Startup . . . 47
5.6.6 Getting New SICS Software . . . 48
5.6.7 Hot Fixes . . . 48
5.6.8 HELP debugging!!!! . . . 48

Chapter 1

Introduction

1.1 Interaction with SICS

1.1.1 SICS Invocation

SICS means SINQ Instrument Control System. SICS is a client server system. This means there are at least two programs necessary to run the experiment. The first is the SICServer which does the actual instrument control work. A user rarely needs to bother about this server program as it is meant to run all the time. See instructions below if things go wrong.

Then there are client programs which interact with the instrument control server. These client programs implement the status displays and a command line application which forwards commands to the SICS server and displays its response. Graphical User Interfaces may be added at a later time. The user has only to deal with these SICS client programs. SICS Clients and the SICServer communicate with each other through the TCP/IP network.

Currently the following SICS clients are available:

- A command line control client for sending commands to the SICS server and displaying its repsonses.
- A status display for the powder diffractometers DMC and HRPT.
- A status display for TOPSI.
- A status display for SANS and SANS2.
- A status display for FOCUS.
- A AMOR control and status program.
- A triple axis control and status program.
- A SICS variable watcher. This application graphically logs the change of a SICS variable over time. Useful for monitoring for instance temperature controllers.
- A graphical client for TRICS.

Steps necessary for logging in to SICS

The following actions have to be taken in order to interact with the SICS server through a client:

- Start the client application.
- Connect the client to a SICS server.
- In case of command line clients: authorize yourself as privileged SICS user.

Starting SICS client applications

These programs can be started on a DigitalUnix system by issuing the following commands at the command prompt:

sics & for the control client.

powderstatus & for the DMC status display client.

topsistatus & for the TOPSI status display.

sansstatus & for the SANS status display.

focustatus for the FOCUS status display.

amor & for the AMOR status display and control application.

tas & for the triple axis status display and control application.

varwatch & for the variable watcher.

trics-& for the starting the TRICS graphical client.

On a PC you may find icons for starting the different programs on the desktop. Each of these clients has usage instructions online which can be displayed through the help/about menu entry.

Another option to start SICS clients is the Java Webstart mechanism which is available for most platforms. Java webstart requires both Java and Java webstart to be installed on the computer running the client. Then clients can be started directly from a WWW-page. The advantage is that clients are automatically updated in this system as soon as new version have been copied to the WWW-site. Installation instructions for Java webstart and links to start all SICS clients though this mechanism can be found at: the SICS webstart[1] page. This service is only accessible within the PSI network.

Connecting

After startup any SICS client is not connected to a SICS server and thus not active. A connection is established through the connect menu of the client.

[1] See URL http://lns00.psi.ch/sics/wstart

Authorization

SICS is a multi user instrument control system. In order to prevent malicious manipulations of the instrument SICS supports a hierarchy of user rights. In order to run an experiment you need at least user level privilege. In order to achieve this privilege you have to invoke the User Parameter/Set Rights dialog. There you have to enter the apropriate username and password kindly provided by your instrument scientist.

Restarting the Server

The SICS server should be running all the time. It is only down if something went wrong. You can check for the presence of the SICS server by loging in to the instrument computer and typing **CheckSICS** at the command prompt. The output will tell you what is happening. If you need to restart the SICS server log in as the instrument user at the instrument computer and invoke the appropriate command to start the server. These are:

DMC Computer = lnsa05, User = DMC

TOPSI Computer = topsi, User = TOPSI

SANS Computer = sans, User = SANS

TRICS Computer = lnsa18, User = TRICS

HRPT Computer = lnsa11, User = HRPT

FOCUS Computer = lnsa16, User = FOCUS

AMOR Computer = lnsa14, User = AMOR

TASP Computer = tasp, User = TASP

POLDI Computer = poldi, User = POLDI

For starting the SICS server type **startsics**. This is a shell script which will starts all necessary server programs. This script works only on the instrument computer and in the appropriate instrument account.

If all this does not help look under trouble shooting SICS (cf. Section 5.6).

1.1.2 The AMOR Client

The AMOR Client is the dedicated SICS client program for the reflectometer AMOR. It is a Java application which runs on all computers for which a Java runtime better then jdk 1.1.6 is available.

Starting the AMOR Client

There are various ways to start the AMOR client. On the unix systems, simply type *amor &* at the command prompt. For PC's and proper Macintosh systems (proper means MacOS > 10) it is recommended to install Java WebStart and start the application from: http://lns00.psi.ch/sics/wstart. The AMOR client can be exited through the File/Exit menu option.

Connecting to a SICS Server

Before anything useful can be done with the AMOR client, it has to be connected to a SICS server. This can be done through the Connect menu in the application. Normally choose AMOR in this menu and everything will be fine. The option Custom Connect is used if the SICS server had to be relocated to another computer (because of a hardware problem) and the connection parameters: computer and port number have to be given explicitly.

Using the AMOR Client

After starting the AMOR client you see a menu, a row of buttons beneath the menu and a central display area, showing AMOR's fantastic logo. Now, the AMOR client has six different views of the instrument. These views can be selected through the button row below the menubar. The views are:

AMOR logo The fantastic AMOR logo. IMHO, AMOR got the nicest logo.

AMOR Schema A drawing of AMORS components annotated with motor names.

Command The main interaction window for typing commands to the instrument. I/O with the server is displayed in the large central text area. Commands can be typed into the text field at the bottom. Then there is a yellow line displaying the progress of the current counting operation. To the left of the command entry field is a little red button labelled *Interrupt*. This button aborts any current operation. There is a menu point corresponding to this window. This allows to select the following functions:

User Rights A dialog for gaining privileges in SICS. You need to specify a username and a password.

Open Logfile Open a log file on the local machine running the client.

Close Logfile Close a local logfile.

Histogram This window displays a histogram of the current data available. What can be seen here depends on AMOR's mode of operation: In single detector mode, the current scan data is displayed. In TOF mode, the counts summed over a rectangular region of the detector (defined in the Area Detector view) is displayed against time-of-flight. Yoy may *zoom in* on selected regions of the histogram by dragging a rectangle from top to bottom. You can *zoom out* through the opposite movement. Further histogram commands hide under the Histogram menu entry. Here you can:

Reset Reset the plot boundaries to show everything.

Print print the current plot into a postscript file.

Logarithmic Toggle the logarithmic flag which, when set, causes the counts to be displayed on a logarithmic basis.

Area Detector This display only makes sence in TOF-mode, with a PSD. It shows the data in the histogram memory projected along the time axis. In the text area below the picture, the current x, y position and the current value at the cursor position

are displayed. Double clicking on the display opens a dialog which allows to set things like: *scale, colour mapping, logarithmic mapping and the mapping range*. Dragging a rectangle in this display will pop up a dialog asking you for a name. The rectangle selected then becomes a *named region* which will then be summed and plotted in the Histogram display. Named regions can be removed through the Update Control/Clear TOF Regions menu entry.

Parameter Shows selected instrument parameters in textual form.

Updating the Display In single detector mode scan data will be automatically updated. In TOF mode, updates to either the histogram display or the area detector display have to obtained either:

- Manually by hitting the green Update button at the bottom of the screen.
- or automatically at certain time intervalls configured through the Update Control/Update Intervall and enabled by toggling the Update Control/Automatic Update flag.

Chapter 2

General User Commands

2.1 Drive commands

Many objects in SICS are **drivable** . This means they can run to a new value. Obvious examples are motors. Less obvious examples include composite adjustments such as setting a wavelength or an energy. This class of objects can be operated by the **drive, run, Success** family of commands. These commands cater for blocking and non-blocking modes of operation.

run var newval var newval ... can be called with one to n pairs of object new value pairs. This command will set the variables in motion and return to the command prompt without waiting for the requested operations to finish. This feature allows to operate other devices of the instrument while perhaps a slow device is still running into position.

Success waits and blocks the command connection until all pending operations have finished (or an interrupt occured).

drive var newval var newval ... can be called with one to n pairs of object new value pairs. This command will set the variables in motion and wait until the driving has finished. A drive can be seen as a sequence of a run command as stated above immediatly followed by a Success command.

2.2 AMOR Motors

!!!!!!! WARNING !!!!!!

The left and right slit motors of diaphragms 1 - 5 are NOT independent motors, but are handled through a single driver. Which motor is driven is selected MANUALLY through a little turn switch close to the motor controller in the orange 19" rack besides the instrument.

!!!!!!! WARNING END !!!!!!

2.2.1 Physical Motors

D1L Diaphragm 1 left slit.

D1R Diaphragm 1 right slit.

D1T Diaphragm 1 top slit.

D1B Diaphragm 1 bottom slit.

MOZ Polarizer omega table up and down

MOM Polarizer omega.

MTY Polarizer y movement.

MTZ Whole polarizer up and down

D2L Diaphragm 2 left slit.

D2R Diaphragm 2 right slit.

D2T Diaphragm 2 top slit.

D2B Diaphragm 2 bottom slit.

D3L Diaphragm 3 left slit.

D3R Diaphragm 3 right slit.

D3T Diaphragm 3 top slit.

D3B Diaphragm 3 bottom slit.

STZ sample height

SOM sample omega

SCH sample chi

SOZ sample table height

STB magnet height

D4L Diaphragm 4 left slit.

D4R Diaphragm 4 right slit.

D4T Diaphragm 4 top slit.

D4B Diaphragm 4 bottom slit.

AOZ Analyzer table height

AOM Analyzer omega

ATZ Analyzer omega height

D5L Diaphragm 5 left slit.

D5R Diaphragm 5 right slit.

D5T Diaphragm 5 top slit.

D5B Diaphragm 5 bottom slit.

COZ Counter table height

C3Z second single counetr height.

COM Counter omega

COX Counter x.

2.2.2 AMOR 2 Theta

The movement off the two theta is quite complex at the reflectometer AMOR. The angle of the detector itself is expressed by translations along the optical bench and in height. Then the angle of the detector needs to be adjusted as well. Moreover some diaphragms along the way from the sample to the detector need adjustment as well. In polarizing mode the analyzer position must be corrected as well. In order to do this a virtual motor has been created which takes care of this movement. As such this object has no real user interface except the usual printing of the position when the name of the virtual motor is typed. However, in order to do his work this virtual motor needs a lot of parameters. These are described below. It is assumed that the virtual motors name is s2t. There is another virtual motor called aom2t which is the analyzer two theta angle.

Parameters needed by the AMOR virtual two theta motor

detectord distance of the detector from the sample.

d4d distance of diaphragm 4 from the sample.

d5d distance of diaphragm 5 from the sample.

sampleh An additive factor describing the height of the sample, i.e. the height of the table itself.

detectorh The base height of the detector.

d4h The base height of diaphragm 4.

d5h The base height of diaphragm 5.

interrupt The interrupt to issue if this motor fails to operate.

anah Height of the analyzer.

anad distance analyser - sample.

anaflag Flag if analyzer movement should be calculated or not. -1 if not, positive if yes.

c2h height constant for single detector 2.

aomconst constant part of analyzer omega angle.

The values of parameters can be inquired by typing:

```
s2t parname
```

and set by:

```
s2t parname newval
```

For example:

```
s2t detectord
155.
s2t detectord 300.
```

2.3 Logging your activity

SICS offers not less then three different ways of logging your commands and the SICS server's responses:

- The SICS command line client allows to open a log file on your local computer and on your account. This can be achieved through the File/Open Logfile menu entry. Select a file name and hit save. From then on, any output in the SICS clients terminal area will be written into the selected file. There is a gotcha: ouptut may not be immediately visible in the file. This is due to buffering of I/O by the operating system. If you want to flush, either open a new file or exit the client. Flushing I/O at each line written is possible, but would have a massive and unacceptable performance impact.
- You may create a similar per client log file on the computer running the SICS server through the logbook (cf. Section 2.3.1) command.
- Then there is a way to log all activity registered from users with either user or manager privilege into a file. This means: all commands which affect the experiment regardless from which client they have been issued. This is accomplished with the commandlog (cf. Section 2.3.2) command.

2.3.1 LogBook command

Some users like to have all the input typed to SICS and responses collected in a file for further review. This is implemented via the LogBook command. LogBook is actually a wrapper around the config file command. LogBook understands the following syntax:

LogBook alone prints the name of the current logfile and the status of event logging.

LogBook file *filename* sets the filename to which output will be printed. Please note that this new filename will only be in effect after restarting logging.

LogBook on This command turns logging on. All commands and all answers will be written to the file defined with the command described above. Please note, that this command will overwrite an existing file with the same name.

LogBook off This command closes the logfile and ends logging.

2.3.2 The Commandlog

The commandlog is a file where all communication with clients having user or manager privilege is logged. This log allows to retrace each step of an experiment. This log is normally configured in the startup file or can be configured by the instrument manager. There exists a special command, commandlog, which allows to control this log file.

commandlog new *filename* starts a new commandlog writing to filename. Any prior files will be closed. The log file can be found in the directory specified by the ServerOption LogFileDir. Usually this is the log directory.

commandlog displays the status of the commandlog.

commandlog close closes the commandlog file.

commandlog auto Switches automatic log file creation on. This is normally switched on. Log files are written to the log directory of the instrument account. There are time stamps any hour in that file and there is a new file any 24 hours.

commandlog tail [***n***] prints the last n entries made into the command log. n is optional and defaults to 20. Up to 1000 lines are held in an internal buffer for this command.

commandlog intervall Queries and configures the intervall in minutes at which time stamps are written to the commandlog.

It is now possible to have a script executed whenever a new log file is started. In order to make this work a ServerOption with the name logstartfile must exist in the instrument configuration file. The value of this option must be the full path name of the file to execute.

2.4 Batch Processing in SICS

Users rarely wish to stay close to the instrument all the time but appreciate if the control computer runs the experiment for them while they sleep. SICS supports two different ways of doing this:

- SICS has a built in macro programming (cf. Section 2.4.1) facility based on the popular scripting language Tcl. The most primitive usage of this facility is processing batch files.
- Second there is the LNS Rünbuffer (cf. Section 2.4.1) system.
- Third there is the new Batch File (cf. Section 2.4.1) execution system.

2.4.1 The Batch Buffer Manager

The batch buffer manager handles the execution of batch files. It can execute batch files directly. Additionally, batch files can be added into a queue for later processing. The batch buffer manager supports the following command described below. Please note, that the examples assume that the batch manager has been configured into SICS under the name of exe.

exe buffername directly load the buffer stored in the file buffername and execute it. The file is searched in the batch buffer search path.

exe batchpath [**newpath**] Without an argument, this command lists the directories which are searched for batch files. With an argument, a new search path is set. It is possible to specify multiple directories by separating them with colons.

exe syspath [**newpath**] Without an argument, this command lists the system directories which are searched for batch files. With an argument, a new system search path is set. It is possible to specify multiple directories by separating them with colons.

exe info prints the name of the currently executing batch buffer

exe info stack prints the stack of nested batch files (i.e. batch files calling each other).

exe info range [**name**] Without an argument prints the range of code currently being executed. With a parameter, prints the range of code executing in named buffer within the stack of nested buffers. The reply looks like: number of start character = number of end character = line number.

exe info text [**name**] Without an argument prints the code text currently being executed. With a parameter, prints the range of code text executing in the named buffer within the stack of nested buffers.

exe enqueue buffername Appends buffername to the queue of batch buffers to execute.

exe clear

Clears the queue of batch buffers

exe queue Prints the content of the batch buffer queue.

exe run Starts executing the batch buffers in the queue.

exe print buffername Prints the content of the batch buffer buffername to the screen.

exe interest Switches on automatic notification about starting batch files, executing a new bit of code or for finishing a batch file. This is most useful for SICS clients watching the progress of the experiment.

Macro Commands

SICS has a built in macro facility. This macro facility is aimed at instrument managers and users alike. Instrument managers may provide customised measurement procedures in this language, users may write batch files in this language. The macro language is John Ousterhout's Tool Command Language (TCL). Tcl has control constructs, variables of its own, loop constructs, associative arrays and procedures. Tcl is well documented by several books and online tutorials, therefore no details on Tcl will be given here. All SICS commands are available in the macro language. Some potentially harmful Tcl commands have been deleted from the standard Tcl interpreter. These are: exec, source, puts, vwait, exit,gets and socket. A macro or batch file can be executed with the command:

fileeval *name* tries to open the file name and executes the script in this file.

batchrun *name* prepends to name a directory name configured in the variable batchroot and then executes that batchfile. The usage scenerio is that you have a directory where you keep batch files. Then the variable batcroot is set to contain the path to that directory. Batchrun then allows to start scripts in that directory without specifying the full path. Then there are some special commands which can be used within macro-sripts:

ClientPut sometext1 ... Usally SICS suppresses any messages from SICS during the processing of batch files. This is in order not to confuse users with the output of intermediate results during the processing of batch files. Error messages and warnings, however, come through always. Clientput now allows to send messages to the user on purpose from within scripts.

SICSType object allows to query the type of the object specified by object. Possible return values are

- **DRIV** if the object is a SICS drivable object such as a motor
- **COUNT** if the object is some form of a counter.
- **COM** if the object is a SICS command.
- **NUM** if the object is a number.
- **TEXT** if object is something meaningless to SICS.

SICSbounds *var newval* checks if the new value newval lies within the limits for variable var(example: SICSbounds D1HL 10). Returns an error or OK depending on the result of the test.

SICSStatus *var* SICS devices such as counters or motor may be started and left running while the program is free to do something else. This command inquires the status of such a running device. Return values are internal SICS integer codes. This command is only of use for SICS programmers.

SetStatus *newval* sets the SICS status to one of: Eager, UserWait, Count, NoBeam, Driving, Running, Scanning, Batch Hatl or Dead. This command is only available in macros.

SetInt *newval, GetInt* sets SICS interrupts from macro scripts. Not recommended! Possible return values or new values are: continue, abortop, abortscan, abortbatch, halt, free, end. This command is only permitted in macros. Should only be used by SICS programmers.

Rünbuffer Commands

LNS scientists have got used to using Rünbuffers for instrument control. A Rünbuffer is an array of SICS commands which typically represent a measurement. This Rünbuffer can be edited at run time. This is very similar to a macro. In contrast to a macro only SICS commands are allowed in Rünbuffers. When done with editing the Rünbuffer it can be entered in a Rünlist. This is a stack of Rünbuffers which get executed one by one. While this is happening it is possible (from another client) to modify the Rünlist and edit and add additional Rünbuffers on top of the stack. This allows for almost infinite measurement and gives more control than a static batch file. In order to cater for this scheme three commands have been defined:

The **Buf** object is responsible for creating and deleting Rünbuffers. The syntax is:

- **Buf new *name*** creates a new empty Rünbuffer with the name name. name will be installed as a SICS object afterwards.
- **Buf copy *name1 name2*** copies Rünbuffer name1 to buffer name2.
- **Buf del *name*** deletes the Rünbuffer name.

After creation, the Rünbuffer is accessible by his name. It then understands the commands:

- ***NAME* append *what shall we do with a drunken sailor*** will add all text after append as a new line at the end of the Rünbuffer.
- ***NAME* print** will list the contents of the Rünbuffer.
- ***NAME* del *iLine*** will delete line number iLine from the Rünbuffer.
- ***NAME* ins *iLine BimBamBim*** inserts a new line **after** line iLine into the Rünbuffer. The line will consist of everything given after the iLine.
- ***NAME* subst pattern *newval*** replaces all occurences of pattern in the Rünbuffer by the text specified as newval. Currently this feature allows only exact match but may be expanded to Unix style regexp or shell like globbing.
- ***NAME* save *filename*** saves the contents of the Rünbuffer into file filename.
- ***NAME* load *filename*** loads the Rünbuffer with the data in file filename.
- ***NAME* run** executes the Rünbuffer.

The Rünlist is accessible as object **stack** . Only one Rünlist per server is permitted. The syntax:

- **stack add *NAME*** adds Rünbuffer name to the top of the stack.
- **stack list** lists the current Rünlist.
- **stack del *iLine*** deletes the Rünbuffer iLine from the Rünlist.
- **stack ins *iLine NAME*** inserts Rünbuffer name after Rünbuffer number iLine into the Rünlist.
- **stack run** executes the Rünlist and returns when all Rünbuffers are done.
- **stack batch** executes the Rünlist but does not return when done but waits for further Rünbuffers to be added to the list. This feature allows a sort of background process in the server.

2.4.2 The Token Command

In SICS any client can issue commands to the SICS server. This is a potential source of trouble with users possibly issuing conflicting commands without knowing. In order to deal with this problem a "token" mechanism has been developed. In this context the token is a symbol for the control of an instrument. A connection can grab the token and then has full control over the SICS server. Any other connection will not be privileged to do anything useful, except looking at things. A token can be released manully with a special command or is automatically released when the connection dies. Another command exists which allows a SICS manager to force his way into the SICS server. The commands in more detail:

token grab Reserves control over the instrument to the client isssuing this command. Any other client cannot control the instrument now. However, other clients are still able to inspect variables.

token release Releases the control token. Now any other client can control the instrument again. Or grab the control token.

token force password This command forces an existing grab on a token to be released. This command requires manager privilege. Furthermore a special password must be specified as third parameter in order to do this. This command does not grab control though.

2.5 Miscellaneous AMOR Commands

2.5.1 Shutter Control

Even the shutter can be controlled from within SICS. This is safe because the shutter will not open if the door to the instrument is open. In Local Beam Control (LBC) speak this status is named "Enclosure is broken". Be careful anyway because some idiots may climb the fence..... The following SICS commands control the shutter:

shutter The command shutter without arguments returns the status of the shutter. This can be one of open, closed, Enclosure is broken.

shutter open opens the shutter when possible.

shutter close closes the shutter.

Chapter 3

AMOR in Single Counter Mode

3.1 AMOR in Single Counter Mode

In this mode the chopper is off. The first task is to adjust diaphragms and monochromators in such a way that the neutron beam hits the sample and afterwards finds its way into the detector. Then scans can be peformed with the normal scan commands. Most often scans will be performed varying two theta. Now, two theta is varied through a complicated, coordinated movement of angles and distances at AMOR. For this to work properly a lot of parameters have to be entered manualy. See, the documentation for a2t (cf. Section 2.2.2) for details. Scan results are stored in NeXus files. Do not forget to set those SICS variables which need to be written to the data file as described in the data storage (cf. Section 4.3) section.

3.2 The Scan Command

An important concept in neutron scattering instrument control is a "scan". For a simple scan a range of instrument positions is divided into equidistant steps. The instrument then proceeds to drive to each of these points and collects data at each of them.

The general idea of the scan object is, that you configure the scan by typing commands at the command line. Once, the configuration is finished the requested scan is started. A data file will be written automatically to the default location. The scan command can not only scan over motors but also about some variables which relate to motors. For instance lamda for the wavelength. Scan can scan over more then one variable. The syntax of the scan command in some detail:

scan clear Clears current scan parameters.

scan list lists current scan parameters.

scan var *name start step* Defines a parameter (motor) to be scanned. The name of the parameter, a start value and a step width need to be given. More then one scan variable can be specified.

scan modvar *name start step* Modifies the scan parameters for scan variable name to the new values given.

scan getvars Returns a list of currently active scan variables terminated with the string -END-.

scan np *num* Sets the number of scan points.

scan preset [***val***] Sets the Preset value for the scan. Without a parameter, inquires the current value.

scan mode [***val***] Sets the count mode for the scan. Without a parameter, inquires the current value. Possible values are *timer* or *monitor*.

scan run Executes the scan.

scan cinterest This call enables automatic printing of scan counts to your connection when new values arise. This command is primariliy of interest for status display clients.

scan pinterest This function makes the scan command send a notification (the string ScanVarChange) to you whenever the scan variables get modified. This command is primarily of interest for status display clients.

3.2.1 Center Scan

Center scan is a convenience command which starts a scan around a specified center value. This mostly used for centering purposes. The syntax is like this:

cscan *var center delta np preset*

All parameters must be specified. The parameters and their meanings:

- **var** is the variable which is to be center scanned. Only one can be specified.
- **center** is the value to use as center of the scan.
- **delta** is the step width to use for the scan.
- **np** is the number of points to scan in each direction.
- **preset** is the preset to use for the counter. As the counter mode, the mode currently configured active in the scan object is used.

example: **cscan** *d2hr 0.5 0.01 10 10000*

3.2.2 Simple Scan

Simple scan is a convenience command which starts a scan for one to several variables with a simplified syntax. The syntax is like this:

sscan *var1 start end var2 start end ... np preset*

All parameters must be specified. The parameters and their meanings:

- **var1 start end** This is how the variables to scan are specified. For each variable scanned the name of the variable, the start value and the end value of the scan must be given. More then one triplet can be given in order to allow for several scan variables.
- **np** is the number of points to scan.
- **preset** is the preset to use for the counter. As the counter mode, the mode currently configured active in the scan object is used.

3.2.3 Peak And Center

These two commands are related to the scan command insofar as they act upon the results of the last scan still in memory. The command **peak** prints the position, FWHM and maximum value of the peak in the last scan. The command **center** drives the first scan variable to the peak center of the last scan. Both peak and center use a rather simple but effective method for locating peaks. The prerequisite is that the peak is approximatly gaussian shaped. The algorithm first locates the peak maximum. Then it goes to the left and right of the maximum and tries to find the points of half maximum peak height. The two points are interpolated from the data and the peak position calculated as the middle point between the two halfheight points.

Chapter 4

AMOR in Time–Of–Flight Mode

4.1 AMOR in Time Of Flight Mode

AMOR can be operated in time of flight mode with a large position sensitive detector. Measuring in this mode involves:

- Aligning the instrument.
- Configuring the histogram memory.
- Configuring the chopper.
- count for some time.

4.1.1 Configuring the Histogram Memory

In order to use the histogram memory, it has to be configured. Two things have to be taken care of:

- Configuring the resolution.
- Configuring the time binning.

See also the general histogram memory (cf. Section 5.2) section.

The resolution of the PSD in pixels can be tailored to the experiment at hand. To this purpose the command psdconfigure is available:

psdconfigure hm *xsize ysize* xsize and ysize are the resolution of the detector in x direction (beam width) and y direction (two theta).

Usually this should already have been set up for you.

Configuring the time binning is a two step process:

- Generate the time binning in SICS with the command:

  hm genbin *start step nBins* This generates an equidistant time binning starting at time start, with stepwidth step and nBins time bins.

- Configure the histogram memory to use the new time binning with:

– hm init

Please note, that non equidistant time binnings are possible as well. See the main histogram memory (cf. Section 5.2) documentation for details.

4.1.2 Configuring the Chopper

The most important thing about the chopper, its rotation speed, can not be controlled from the instrument control system. It has to be adjusted **MANUALLY** at the chopper control PC at the hall floor. There are two other parameters, however, which are needed by the detector electronics in order to process the chopper synchronisation signal properly. The commands are:

aw Read the current value of the acceptance window.

aw *val* Set the acceptance window to val.

td Read the current value of the time to delay to start.

td *val* Set the time delay to start to val.

4.2 Counting Commands

count [***mode preset***] Does a count operation in mode with a given preset. *Mode* can be *timer* or *monitor*. The parameters mode and preset are optional. If they are not given the count will be started with the current setting in the histogram memory object.

repeat num mode preset. Calls count num times. num is a required parameter. The other two are optional and are handled as described above for count.

Both commands make sure, that measured data is written to files.

4.3 AMOR Data Storage

Data files at AMOR are stored in NeXus format, based on HDF-5. This is a portable binary format. For more information see the NeXus WWW-pages[1]. Data storage happens normally without user intervention during scans or after a count command has finished. A couple of things are noteworthy, however:

- Data Storage in TOF-mode may take several minutes due to the amount of data to transfer. In this time the SICS server will be quite unresponsive.
- Data file writing can be interrupted. This is a unique feature at AMOR. As a consequence, a data file is NOT automaticaly created when a measurement has been interrupted.
- Data file writing can be forced by typing: *storeamor*.

[1]See URL http://lns00.psi.ch/NeXus/

- Data files can be found in directories: /home/AMOR/data/YYYY on the instrument computer or (after a litlle delay) at /data/lnslib/data/AMOR/YYYY . The YYYY is a placeholder for the year of data collection.
- The last data file written can be overwritten with command: *killfile*. Managers privilege is required for this command.

In order to have complete information in the data files a couple of SICS variables have to be set manually. A SICS variable's value can be interrogated by typing the name of the variable, and set by typing the name of the variable followed by the new value. The following variables are relevant:

chopperrotation Chopper Rotation speed.

user User name

email User e-mail address.

fax User fax number

phone User phone number

adress User address.

sample Sample name.

title Measurement title

Chapter 5

Advanced Topics

5.1 Sample Environment Devices

5.1.1 SICS Concepts for Sample Environment Devices

SICS can support any type of sample environment control device if there is a driver for it. This includes temperature controllers, magnetic field controllers etc. The SICS server is meant to be left running continously. Therefore there exists a facility for dynamically configuring and deconfiguring environment devices into the system. This is done via the **EVFactory** command. It is expected that instrument scientists will provide command procedures or specialised Rünbuffers for configuring environment devices and setting reasonable default parameters.

In the SICS model a sample environment device has in principle two modes of operation. The first is the drive mode. The device is monitored in this mode when a new value for it has been requested. The second mode is the monitor mode. This mode is entered when the device has reached its target value. After that, the device must be continously monitored throughout any measurement. This is done through the environment monitor or **emon**. The emon understands a few commands of its own.

Within SICS all sample environement devices share some common behaviour concerning parameters and abilities. Thus any given environment device accepts all of a set of general commands plus some additional commands special to the device.

In the next section the EVFactory, emon and the general commands understood by any sample environment device will be discussed. This reading is mandatory for understanding SICS environment device handling. Then there will be another section discussing the special devices known to the system.

5.1.2 Sample Environment Error Handling

A sample environment device may fail to stay at its preset value during a measurement. This condition will usually be detected by the emon. The question is how to deal with this problem. The requirements for this kind of error handling are quite different. The SICS model therefore implements several strategies for handling sample environment device failure handling. The strategy to use is selected via a variable which can be set by the user for any sample environment device separately. Additional error handling strategies can be added with a modest amount of programming. The error handling strategies

currently implemented are:

Lazy Just print a warning and continue.

Pause Pauses the measurement until the problem has been resolved.

Interrupt Issues a SICS interrupt to the system.

Safe Tries to run the environment device to a value considered safe by the user.

5.1.3 General Sample Environment Commands

EVFactory

EVFactory is responsible for configuring and deconfiguring sample environment devices into SICS. The syntax is simple:

EVFactory new name type par par ... Creates a new sample environment device. It will be known to SICS by the name specified as second parameter. The type parameter decides which driver to use for this device. The type will be followed by additional parameters which will be evaluated by the driver requested.

EVFactory del name Deletes the environment device name from the system.

emon

The environment monitor emon takes for the monitoring of an environment device during measurements. It also initiates error handling when appropriate. The emon understands a couple of commands.

emon list This command lists all environment devices currently registered in the system.

emon register name This is a specialist command which registers the environment device name with the environment monitor. Usually this will automatically be taken care of by EVFactory.

emon unregister name This is a specialist command which unregisters the environment device name with the environment monitor. Usually this will automatically be taken care of by EVFactory Following this call the device will no longer be monitored and out of tolerance errors on that device no longer be handled.

General Commands UnderStood by All Sample Environment Devices

Once the evfactory has been run successfully the controller is installed as an object in SICS. It is accessible as an object then under the name specified in the evfactory command. All environemnt object understand the common commands given below. Please note that each command discussed below MUST be prepended with the name of the environment device as configured in EVFactory!

The general commands understood by any environment controller can be subdivided further into parameter commands and real commands. The parameter commands just print the name of the parameter if given without an extra parameter or set if a parameter is specified. For example:

Temperature Tolerance

prints the value of the variable Tolerance for the environment controller Temperature. This is in the same units as the controller operates, i. e. for a temperature controller Kelvin.

Temperature Tolerance 2.0

sets the parameter Tolerance for Temperature to 2.0. Parameters known to ANY envrironment controller are:

Tolerance Is the deviation from the preset value which can be tolerated before an error is issued.

Access Determines who may change parameters for this controller. Possible values are:

- 0 only internal
- 1 only Managers
- 2 Managers and Users.
- 3 Everybody, including Spy.

LowerLimit The lower limit for the controller.

UpperLimit The upper limit for the controller.

ErrHandler. The error handler to use for this controller. Possible values:

- 0 is Lazy.
- 1 for Pause.
- 2 for Interrupt
- 3 for Safe.

For an explanantion of these values see the section about error (cf. Section 5.1.2) handling above.

Interrupt The interrupt to issue when an error is detected and Interrupt error handling is set. Valid values are:

- 0 for Continue.
- 1 for abort operation.
- 2 for abort scan.
- 3 for abort batch processing.
- 4 halt system.
- 5 exit server.

SafeValue The value to drive the controller to when an error has been detected and Safe error handling is set.

MaxWait Maximal time in minutes to wait in a drive temperature command. If maxwait is set to 0: If the temperature is not reached within tolerance, it waits indefinitely.

Settle Wait time [minutes] after reaching temperature. Indicates how long to wait after reaching temperature. If the temperatures goes again out of tolerance during the settling time, the time outside tolerance is not taken into account.

Additionally the following commands are understood:

send par par ... Sends everything after send directly to the controller and return its response. This is a general purpose command for manipulating controllers and controller parameters directly. The protocoll for these commands is documented in the documentation for each controller. Ordinary users should not tamper with this. This facility is meant for setting up the device with calibration tables etc.

list lists all the parameters for this controller.

no command, only name. When only the name of the device is typed it will return its current value.

name val will drive the device to the new value val. Please note that the same can be achieved by using the drive command. and **log frequency** (both below)

Logging

The values of any sample environement device can be logged. There are three features:

- Logging to a file wih a configurable time intervall between log file entries.
- Sums are kept internally which allow the calculation of the mean value and the standard deviation at all times.
- A circular buffer holding 1000 timestamps plus values is automatically updated.

The last two systems are automatically switched on after the first drive or run command on the environment device completed. This system is run through the following commands.

name log clear Resets all sums for the calculation of the mean value and the standard deviation.

name log getmean Calculates the mean value and the standard deviation for all logged values and prints them.

name log frequency val With a parameter sets, without a parameter requests the logging intervall for the log file and the circular buffer. This parameter specifies the time intervall in seconds between log records. The default is 300 seconds.

name log file filename Starts logging of value data to the file filename. Logging will happen any 5 minutes initially. The logging frequency can be changed with the name log frequency command. Each entry in the file is of the form date time value. The name of the file must be specified relative to the SICS server.

name log flush Unix buffers output heavily. With this command an update of the file can be enforced.

name log status Queries if logging to file is currently happening or not.

name log close Stops logging data to the file.

name log tosicsdata dataname copies the content of the circular buffer to a sicsdata buffer. This is used by graphical clients to display the content of the circular buffer.

name log dump Prints the content of the circular log buffer to screen.

name log dumptofile filename Prints the content of the circular log buffer into the file specified as filename. Note, this file is on the computer where the SICS server resides.

5.1.4 Special Environment Control Devices

This section lists the parameters needed for configuring a special environment device into the system and special parameters and commands only understood by that special device. All of the general commands listed above work as well!

LakeShore Model 340 Temperature Controller

This is *the* temperature controller for cryogenic applications and should replace at least the Oxford & Neocera controllers at SINQ.

The control is handled by a seperate server process TECS (TEmperature Control Server) and is initialized by default on most instruments. If there is already an other device selected, it must be deleted with:

```
EVFactory del temperature
```

and TECS must be reinstalled with:

```
tecs on
```

(This is just an abbreavation for EVFactory new temperature tecs)

More details can be found on the Sample Environment Home Page[1]

ITC-4 and ITC-503 Temperature Controllers

These temperature controller were fairly popular at SINQ. They are manufactured by Oxford Instruments. At the back of this controller is a RS-232 socket which must be connected to a terminal server via a serial cable.

[1]See URL http://sinq.web.psi.ch/sinq/sample.env/tecs.html

ITC-4 Initialisation An ITC-4 can be configured into the system by:

EVFactory new Temp ITC4 computer port channel

This creates an ITC-4 controller object named Temp within the system. The ITC-4 is expected to be connected to the serial port channel of the serial port server porgramm at localhost listening at the specified port. For example:

EVFactory new Temp ITC4 localhost 4000 7

connects Temp to the serial port 7, listening at port 4000.

ITC-4 Additional Parameters The ITC-4 has a few more parameter commands:

timeout Is the timeout for the SerPortServer waiting for responses from the ITC-4. Increase this parameter if error messages contaning ?TMO appear.

sensor Sets the sensor number to be used for reading temperature.

control Sets the control sensor for the ITC-4. This sensor will be used internally for regulating the ITC-4.

divisor The ITC4 does not understand floating point numbers, the ITC-503 does. In order to make ITC4's read and write temperatures correctly floating point values must be multiplied or divided with a magnitude of 10. This parameter determines the appropriate value for the sensor. It is usually 10 for a sensor with one value behind the comma or 100 for a sensor with two values after the comma.

multiplicator The same meaning as the divisor above, but for the control sensor.

Installing an ITC4 step by step

1. Connect the ITC temperature controller to port 7 on the terminal server box. Port 7 is specially configured for dealing with the ideosyncracies of that device. No null modem is needed.

2. Install the ITC4 into SICS with the command:
   evfactory new temperature localhost 4000 7
   You may choose an other name than "temperature", but then it is in general not stored in the data file. Please note, that SICS won't let you use that name if it already exists. For instance if you already had a controller in there. Then the command:
   evfactory del name
   will help.

3. Configure the upper and lowerlimits for your controller appropriatetly.

4. Figure out which sensor you are going to use for reading temperatures. Configure the sensor and the divisor parameter accordingly.

5. Figure out, which sensor will be used for controlling the ITC4. Set the parameters control and multiplicator accordingly. Can be the same as the sensor.

6. Think up an agreeable temperature tolerance for your measurement. This tolerance value will be used 1) to figure out when the ITC4 has reached its target position. 2) when the ITC4 will throw an error if the ITC4 fails to keep within that tolerance. Set the tolerance parameter according to the results of your thinking.

7. Select one of the numerous error handling strategies the control software is able to perform. Configure the device accordingly.

8. Test your setting by trying to read the current temperature.

9. If this goes well try to drive to a temperature not to far from the current one.

ITC-4 Trouble Shooting If the ITC-4 **does not respond at all**, make sure the serial connection to is working. Use standard RS-232 debugging procedures for doing this. The not responding message may also come up as a failure to connect to the ITC-4 during startup.

If error messages containing the string **?TMO** keep appearing up followed by signs that the command has not been understood, then increase the timeout. The standard timeout of 10 microseconds can be to short sometimes.

You keep on reading **wrong values** from the ITC4. Mostly off by a factor 10. Then set the divisor correctly. Or you may need to choose a decent sensor for that readout.

Error messages when **trying to drive the ITC4**. These are usually the result of a badly set multiplicator parameter for the control sensor.

The ITC4 **never stops driving**. There are at least four possible causes for this problem:

1. The multiplicator for the control sensor was wrong and the ITC4 has now a set value which is different from your wishes. You should have got error messages then as you tried to start the ITC4.

2. The software is reading back incorrect temperature values because the sensor and divisor parameters are badly configured. Try to read the temperature and if it does have nothing to do with reality, set the parameters accordingly.

3. The tolerance parameter is configured so low, that the ITC4 never manages to stay in that range. Can also be caused by inappropriate PID parameters in the ITC4.

4. You are reading on one sensor (may be 3) and controlling on another one (may be 2). Then it may happen that the ITC 4 happily thinks that he has reached the temperature because its control sensor shows the value you entered as set value. But read sensor 3 still thinks he is far off. The solution is to drive to a set value which is low enough to make the read sensor think it is within the tolerance. That is the temperature value you wanted after all.

Haake Waterbath Thermostat

This is sort of a bucket full of water equipped with a temperature control system. The RS-232 interface of this device can only be operated at 4800 baud max. This is why it has to be connected to a specially configured port. The driver for this device has been

realised in the Tcl extension language of the SICS server. A prerequisite for the usage of this device is that the file hakle.tcl is sourced in the SICS initialisation file and the command inihaakearray has been published. Installing the Haake into SICS requires two steps: first create an array with initialisation parameters, second install the device with evfactory. A command procedure is supplied for the first step. Thus the initialisation sequence becomes:

inihaakearray name-of-array localhost name port channel
evfactory new temperature tcl name-of-array

An example for the SANS:

inihaakearray eimer localhost 4000 1
evfactory new temperature tcl eimer

Following this, the thermostat can be controlled with the other environment control commands.

The Haake Thermostat understands a single special subcommand: **sensor**. The thermostat may be equipped with an external sensor for controlling and reading. The subcommand sensor allows to switch between the two. The exact syntax is:

temperature sensor val

val can be either intern or extern.

Bruker Magnet Controller B-EC-1

This is the Controller for the large magnet at SANS. The controller is a box the size of a chest of drawers. This controller can be operated in one out of two modes: in **field** mode the current for the magnet is controlled via an external hall sensor at the magnet. In **current** mode, the output current of the device is controlled. This magnet can be configured into SICS with a command syntax like this:

evfactory new name bruker localhost port channel

name is a placeholder for the name of the device within SICS. A good suggestion (which will be used throughout the rest of the text) is magnet. bruker is the keyword for selecting the bruker driver. port is the port number at which the serial port server listens. channel is the RS-232 channel to which the controller has been connected. For example (at SANS):

```
evfactory new magnet bruker localhost 4000 9
```

creates a new command magnet for a Bruker magnet Controller connected to serial port 9. In addition to the standard environment controller commands this magnet controller understands the following special commands:

magnet polarity Prints the current polarity setting of the controller. Possible answers are plus, minus and busy. The latter indicates that the controller is in the process of switching polarity after a command had been given to switch it.

magnet polarity val sets a new polarity for the controller. Possible values for val are **minus** or **plus**. The meaning is self explaining.

magnet mode Prints the current control mode of the controller. Possible answers are **field** for control via hall sensor or **current** for current control.

magnet mode val sets a new control mode for the controller. Possible values for val are **field** or **current**. The meaning is explained above.

magnet field reads the magnets hall sensor independent of the control mode.

magnet current reads the magnets output current independent of the control mode.

Warning: There is a gotcha with this. If you type only magnet a value will be returned. The meaning of this value is dependent on the selected control mode. In current mode it is a current, in field mode it is a magnetic field. This is so in order to support SICS control logic. You can read values at all times explicitly using magnet current or magnet field.

The Eurotherm Temperature Controller

At SANS there is a Eurotherm temperature controller for the sample heater. This and probably other Eurotherm controllers can be configured into SICS with the following command. The eurotherm needs to be connected with a nullmodem adapter.

evfactory new name euro computer port channel

name is a placeholder for the name of the device within SICS. A good suggestion is temperature. euro is the keyword for selecting the Eurotherm driver. port is the port number at which the serial port server listens. channel is the RS-232 channel to which the controller has been connected. **WARNING:** The eurotherm needs a RS-232 port with an unusual configuration: 7bits, even parity, 1 stop bit. Currently only the SANS port 13 is configured like this! Thus, an example for SANS and the name temperature looks like:

```
evfactory new temperature euro localhost 4000 13
```

There are two further gotchas with this thing:

- The eurotherm needs to operate in the EI-bisynch protocoll mode. This has to be configured manually. For details see the manual coming with the machine.
- The weird protocoll spoken by the Eurotherm requires very special control characters. Therefore the send functionality usually supported by a SICS environment controller could not be implemented.

The Risoe A1931 Temperature Controller

This is a temperature controller of unknown origin (probably built at Risoe) which is coming with the Risoe instruments. This temperature controller is connected to the computer systems through a GPIB bus and controller. A A1931 temperature controller is configured into SICS through the command:

evfactory new temperature-name a1931 gpib-controller-name gpibaddress

This creates a new command temperature-name. gpib-controller-name is the name of a GPIB controller within SICS. A GPIB controller is configured into SICS with the command MakeGPIB as described in the SICS managers documentation. gpibaddress is the address of the A1931 on the GPIB bus.

A A1931 temperature device understands a couple of additional commands on top of the standard set:

temperature sensor val The A1931 can switch control to various sensors. This command allows to query the control sensor (command without parameter) or set the control sensoe (command with parameter).

temperature file filename The A1931 can be configured through files containing calibration commands. Sich file can be loaded into the A1931 through the file subcommand. The full path of filename must be given.

The PSI-EL755 Magnet Controller

This is magnet controller developed by the electronics group at PSI. It consists of a controller which interfaces to a couple of power supplies. The magnets are then connected to the power supplies. The magnetic field is not controlled directly but just the power output of the power supply. Also the actual output of the power supply is NOT read back but just the set value after ramping. This is a serious limitation because the computer cannot recognize a faulty power supply or magnet. The EL755 is connected to SICS with the command:

evfactory new name el755 localhost port channel index

with port and channel being the usual data items for describing the location of the EL755-controller at the serial port server. index is special and is the number of the power supply to which the magnet is connected. An example:

```
evfactory new maggi el755 localhost 4000 5 3
```

connects to power supply 3 at the EL755-controller connected to lnsa09 at channel 5. The magnet is then available in the system as maggi. No special commands are supported for the EL755.

PSI-DSP Magnet Controller

The PSI-DSP magnet controller has been developed by the PSI electronics group, most notably by Lukas Tanner, for the SLS. However, these controllers are now being used at SINQ as well. This controller has a binary command protocoll and thus the send command does not work for it. In order to handle this protocoll SICS has to bypass the usual SerPortServer mechanism for communicating with serial devices and to connect to the terminal server directly. This also implies one gotcha: **The PSI-DSP works only at specially configured terminal server ports**.The terminal server port to which the PSI-DSP is connected **MUST** be configured to: 115200 baud, 8 data bits, 1 stop bit, odd parity. In general a system manager is required to do this. The PSI-DSP also requires a null-modem connector between the box and the terminal server. Once these hurdles have been mastered, the PSI-DSP can be configured into SICS with the command:

evfactory new name psi-dsp terminalservername port

with name being the name of the magnet in SICS, terminalservername the name of the terminal server, for example psts224 and port being the address of the binary port on the terminal server. This is usually the serial port number at the terminal server plus 3000. An example:

evfactory new maggi psi-dsp psts224 3016

configures a magnet named maggi which is connectd to port 16 at the terminal server psts224. maggi can now be read and driven like any other environment device.

Old Dilution Cryostat (Obsolete)

This is a large ancient device for reaching very low temperatures. This cryostat can be configured into SICS with the command:

```
EVFactory new Temp dillu computer port channel table.file
```

Temp is the name of the dilution controller command in SICS, dillu is the keyword which selects the dilution driver, computer, port and channel are the parameters of the Macintosh-PC running the serial port server program. table.file is the fully qualified name of a file containing a translation table for this cryostat. The readout from the dilution controller is a resistance. This table allows to interpolate the temperature from the resistance measurements and back. Example:

```
evfactory new temperature dillu lnsp19.psi.ch 4000 1 dilu.tem
```

installs a new dilution controller into SICS. This controller is connected to port 1 at the Macintos-PC with the newtwork adress lnsp19.psi.ch. On this macintosh-PC runs a serial port server program listening at TCP/IP port 4000. The name of the translation table file is dilu.tem.

The dilution controller has no special commands, but two caveats: As of current (October 1998) setting temperatures does not work due to problems with the electronics. Second the dilution controller MUST be connected to port 1 as only this port supports the 4800 maximum baud rate this device digests.

Old CryoFurnace Controller (Obsolete)

The CryoFurnace at PSI is equipped with a Neocera LTC-11 temperature controller. This controller can control either an heater or an analag output channel. Futhermore a choice of sensors can be selected for controlling the device. The LTC-11 behaves like a normal SICS environment control device plus a few additional commands. An LTC-11 can be configured into SICS with the following command:

evfactory new name ltc11 computer port channel

name is a placeholder for the name of the device within SICS. A good suggestion is temperature. ltc11 is the keyword for selecting the LTC-11 driver. Computer is the name of the computer running David Maden's SerPortServer program, port is the port number at which the SerPortServer program listens. Channel is the RS-232 channel to which the controller has been connected. For example (at DMC):

```
evfactory new temperature ltc11 localhost 4000 6
```

creates a new command magnet for a LTC-11 temperature Controller connected to serial port 6 at lnsp18.

The additional commands understood by the LTC-11 controller are:

temperature sensor queries the current sensor used for temperature readout.

temperature sensor val selects the sensor val for temperature readout.

temperature controlanalog queries the sensor used for controlling the analog channel.

temperature controlanalog val selects the sensor val for controlling the analog channel.

temperature controlheat queries the sensor used for controlling the heater channel.

temperature controlheat val selects the sensor val for controlling the heater channel.

temperature mode queries if the LTC-11 is in analog or heater control mode.

Further notes: As the CryoFurnace is very slow and the display at the controller becomes unusable when the temperature is read out to often, the LTC-11 driver buffers the last temperature read for 5 seconds. Setting the mode of the LTC-11 is possible by computer, but not yet fully understood and therefore unusable.

5.2 Histogram memory

Histogram memories are used in order to control large area sensitive detectors or single detectors with time binning information. Basically each detector maps to a defined memory location. The histogram memory wizard takes care of putting counts detected in the detector into the proper bin in memory. Some instruments resolve energy (neutron flight time) as well, than there is for each detector a row of memory locations mapping to the time bins. As usual in SICS the syntax is the name of the histogram memory followed by qualifiers and parameters. As a placeholder for the histogram memories name in your system, HM will be used in the following text.

A word or two has to be lost about the SICS handling of preset values for histogram memories. Two modes of operation have to be distinguished: counting until a timer has passed, for example: count for 20 seconds. This mode is called timer mode. In the other mode, counting is continued until a control monitor has reached a certain preset value. This mode is called Monitor mode. The preset values in Monitor mode are usually very large. Therefore the counter has an exponent data variable. Values given as preset are effectively 10 to the power of this exponent. For instance if the preset is 25 and the exponent is 6, then counting will be continued until the monitor has reached 25 million. Note, that this scheme with the exponent is only in operation in Monitor mode.

5.2.1 Configuration

A HM has a plethora of configuration options coming with it which define memory layout, modes of operation, handling of bin overflow and the like. Additionally there are HM model specific parameters which are needed internally in order to communicate with the HM. In most cases the HM will already have been configured at SICS server startup time. However, there are occasion where these configuartion option need to enquired or modified at run time. The command to enquire the current value of a configuration option is: **HM configure option**, the command to set it is: **HM configure option newvalue**. A list of common configuration options and their meaning is given below:

HistMode HistMode describes the modes of operation of the histogram memory. Possible values are:

- Transparent, Counter data will be written as is to memory. For debugging purposes only.
- Normal, neutrons detected at a given detector will be added to the apropriate memory bin.
- TOF, time of flight mode, neutrons found in a given detector will be put added to a memory location determined by the detector and the time stamp.
- Stroboscopic mode. This mode serves to analyse changes in a sample due to an varying external force, such as a magnetic field, mechanical stress or the like. Neutrons will be stored in memory according to detector position and phase of the external force.

OverFlowMode This parameter determines how bin overflow is handled. This happend when more neutrons get detected for a particular memory location then are allowed for the number type of the histogram memory bin. Possible values are:

- Ignore. Overflow will be ignored, the memory location will wrap around and start at 0 again.
- Ceil. The memory location will be kept at the highest posssible value for its number type.
- Count. As Ceil, but a list of overflowed bins will be maintained.

Rank Rank defines the number of dimensions the detector has, minus the time channle when applicable. 1 is a linear detector, 2 a area detector etc.

BinWidth determines the size of a single bin in histogram memory in bytes.

dim0, dim1, dim2, ... dimn define the logical dimensions of the histogram.

extrachan Extra time channels as used at AMOR and SANS for time-of-flight monitors. They get appended to the main hm data but are treated separately.

In addition to these common options there exist additional options for the EMBL position sensitive detectors (PSD) installed at TRICS and AMOR. These PSDs can be operated at different pixel resolutions. The position of a neutron event on these detectors is encoded in a delay time value which is digitized into a range between 0 to 4096. This

resolution exceeds the resolution available from instrument physics by far. Useful resolutions are obtained by dividing this raw range by a factor. In addition, the coordinates of the center of the detector have to given as well (usually size/2).This is done through the configuration options:

xFac x direction division factor

yFac y direction division factor

xOff Offset of the detector center in x.

yOff Offset of the detector center in y.

Do not forget to change the standard options dim0, dim1 and length as well when changing the PSD resolution.

For time of flight mode the time binnings can be retrieved and modified with the following commands. Note that these commands do not follow the configure syntax given above. Please note, that the usage of the commands for modifying time bins is restricted to instrument managers.

HM timebin Prints the currently active time binning array.

HM genbin *start step n* Generates a new equally spaced time binning array. Number n time bins will be generated starting from start with a stepwidth of step (example: HM genbin 10 1 5).

HM setbin *inum value* Sometimes unequally spaced time binnings are needed. These can be configured with this command. The time bin iNum is set to the value value.

HM clearbin Deletes the currently active time binning information.

HM notimebin returns the number of currently configured timebins.

5.2.2 Histogram Memory Commands

Besides the configuration commands the HM understands the following commands:

HM preset [***newval***] with a new value sets the preset time or monitor for counting. Without a value prints the current value.

HM exponent [***newval***] with a new value sets the exponent to use for the preset time in Monitor mode. Without a value prints the current value.

CountMode [***mode***] with a new values for *mode* sets the count mode. Possible values are Timer for a fixed counting time and Monitor for a fixed monitor count which has to be reached before counting finishes. Without a value print the currently active value.

HM init after giving configuration commands this needs to be called in order to transfer the configuration from the host computer to the actual HM.

HM count starts counting using the currently active values for CountMode and preset. This command does not block, i.e. in order to inhibit further commands from the console, you have to give Success afterwards.

HM countblock starts counting using the currently active values for CountMode and preset. This command does block, i.e. you can give new commands only when the counting operation finishes.

HM initval *val* initialises the whole histogram memory to the value val. Ususally 0 in order to clear the HM.

HM get *i iStart iEnd* retrieves the histogram number i. A value of -1 for i denotes retrieval of the whole HM. iStart and iEnd are optional and allow to retrieve a subset of a histogram between iStart and iEnd.

HM sum *d1min d1max d2min d2max .... dnmin dnmax* calculates the sum of an area on the detector. For each dimension a minimum and maximum boundary for summing must be given.

5.3 SICS motor handling

In SICS each motor is an object with a name. Motors may take commands which basically come in the form

motorname command ; example: D2HR list.

Most of these commands deal with the plethora of parameters which are associated with each motor. The syntax for manipulating variables is, again, simple.

Motorname parametername

will print the current value of the variable.

Motorname parametername newval

will set the parameter to the new value specified. A list of all parameters and their meanings is given below. The general principle behind this is that the actual (hardware) motor is kept as stupid as possible and all the intracacies of motor control are dealt with in software. Besides the parameter commands any motor understands these basic commands:

- ***Motorname list*** gives a listing of all motor parameters.
- ***Motorname* reset** resets the motor parameters to default values. This is software zero to 0.0 and software limits are reset to hardware limits.
- ***Motorname* position** prints the current position of the motor. All zero point and sign corrections are applied.
- ***Motorname* hardposition** prints the current position of the motor. No corrections are applied. Should read the same as the controller box.
- ***Motorname* interest** initiates automatic printing of any position change of the motor. This command is mainly interesting for implementors of status display clients.

Please note that the actual driving of the motor is done via the drive (cf. Section 2.1) command.

5.3.1 The motor parameters

- **HardLowerLim** is the hardware lower limit. This is read from the motor controller and is identical to the limit switch welded to the instrument. Can usually not be changed.
- **HardUpperLim** is the hardware upper limit. This is read from the motor controller and is identical to the limit switch welded to the instrument. Can usually not be changed.
- **SoftLowerLim** is the software lower limit. This can be defined by the user in order to restrict instrument movement in special cases.
- **SoftUpperLim** is the software upper limit. This can be defined by the user in order to restrict instrument movement in special cases.
- **SoftZero** defines a software zero point for the motor. All further movements will be in respect to this zeropoint.
- **Fixed** can be greater then 0 for the motor being fixed and less then or equal to zero for the motor being movable.
- **InterruptMode** defines the interrupt to issue when the motor fails. Some motors are so critical for the operation of the instrument that all operations are to be stopped when there is a problem. Other are less critical. This criticallity is expressed in terms of interrupts, denoted by integers in the range 0 - 4 translating into the interrupts: continue, AbortOperation, AbortScan, AbortBatch and Halt. This parameter can usually only be set by managers.
- **Precision** denotes the precision to expect from the motor in positioning. Can usually only be set by managers.
- **AccessCode** specifies the level of user privilege necessary to operate the motor. Some motors are for adjustment only and can be harmful to move once the adjustment has been done. Others must be moved for the experiment. Values are 0 - 3 for internal, manager, user and spy. This parameter can only be changed by managers.
- **Sign** reverses the operating sense of the motor. For cases where electricians and not physicists have defined the operating sense of the motor. Usually a parameter not to be changed by ordinary users.
- **failafter** This is the number of consecutive failures of positioning operations this motor allows before it thinks that something is really broken and aborts the experiment.
- **maxretry** When a motor finishes driving, SICS checks if the desired position was reached. If the position read back from the motor is not within precision to the desired value, the motor is restarted. This is done at max maxretry times. After maxretry retries, the motor throws an error.

- ignorefault If this is bigger then 0, positioning faults from the motor will be ignored.

5.3.2 Motor Error Handling Concepts

As mechanical components motors are prone to errors. SICS knows about two different classes of motor errors:

HWFault This is when there is a problem communicating with the motor, a limit is violated etc. SICS assumes that such errors are so grave that no fix is possible. If such a HWFault is detected a configurable interrupt (see parameter InterruptMode) is set which can be used by upper level code to act upon the problem.

HWPosFault This is a positioning failure, i.e. The motor did not reach the desired position. Such a positioning problem can come from two sources:

- The positioning problem is reported by the motor driver. SICS then assumes that the driver has done something to solve the problem and promotes this problem to a HWFault.
- The motor driver reported no error and SICS figures out by itself, that the desired position has not been reached. SICS thinks that this is the case if the difference between the desired position and the position read from the motor controller is greater then the parameter precision. If SICS detects such a problem it tries to reposition the motor. This is done for the number of times specified through the parameter maxretries. If the position has not been reached after maxretries repositionings, a HWFault is assumed.

In any case lots of warnings and infos are printed.

If SICS tries to drive an axis which is for some reason broken to often hardware damage may occur (and HAS occurred!). Now, SICS has no means to detect if the mispositioning of a motor is due to a concrete block in the path of the instrument or any other reason. What SICS can do though is to count how often a motor mispositions in sequence. This means SICS counts mispositionings if it cannot drive a motor, if the motor is driven succesfully, the count is cleared. If the count of mispositionings becomes higher then the parameter failafter, SICS thinks that there is something really, really wrong and aborts the measurement and prints an error message containing the string: MOTOR ALARM.

There are some common pitfalls with this scheme:

You want upper level code to be signalled when your critical motorfails. Solution: set the parameter interruptmode to something useful and check for the interrupt in upper level code.

SICS falsly reports mispositionings. Solution: increase the precision parameter.

You know that a motor is broken, you cannot fix it, but you wantto measure anyway. Solution: increase the precision parameter, if SICS finds the positioning problem, increase maxretries, increase the failafter parameter. In the worst case set the ignorefault parameter to greater 0, this will prevent all motor alarms.

5.4 SICS counter handling

A counter in SICS is a controller which operates single neutron counting tubes and monitors. A counter can operate in one out of two modes: counting until a timer has passed, for example: count for 20 seconds. Counting in this context means that the noutrons coming in during these 20 seconds are summed together. This mode is called timer mode. In the other mode, counting is continued until a specified neutron monitor has reached a certain preset value. This mode is called Monitor mode. The preset values in Monitor mode are usually very large. Therefore the counter has an exponent data variable. Values given as preset are effectively 10 to the power of this exponent. For instance if the preset is 25 and the exponent is 6, then counting will be continued until the monitor has reached 25 million. Note, that this scheme with the exponent is only in operation in Monitor mode. Again, in SICS the counter is an object which understands a set of commands:

- **countername setpreset *val*** sets the counting preset to val.
- **countername getpreset** prints the current preset value.
- **countername preset *val*** With a parameter sets the preset, without inquires the preset value. This is a duplicate of getpreset and setpreset which has been provided for consistency with other commands.
- **countername setexponent *val*** sets the exponent for the counting preset in monitor mode to val.
- **countername getexponent** prints the current exponent used in monitor mode.
- **countername setmode *val*** sets the counting mode to val. Possible values are Timer for timer mode operation and Monitor for waiting for a monitor to reach a certain value.
- **countername getmode** prints the current mode.
- **countername *mode val*** With a parameter sets the mode, without inquires the mode value. This is a duplicate of getmode and setmode which has been provided for consistency with other commands. Possible values for val are either monitor or timer.
- **countername setexponent *val*** sets the exponent for the counting preset in monitor mode to val.
- **countername getcounts** prints the counts gathered in the last run.
- **countername getmonitor *n*** prints the counts gathered in the monitor number n in the last run.
- **countername count *preset*** starts counting in the current mode and the given preset.
- **countername status** prints a message containing the preset and the current monitor or time value. Can be used to monitor the progress of the counting operation.

- **countername gettime** Retrieves the actual time the counter counted for. This excludes time where there was no beam or counting was paused.
- **countername getthreshold *m*** retrieves the value of the threshold set for the monitor number m.
- **countername setthreshold *m val*** sets the threshold for monitor m to val. WARNING: this also makes monitor m the active monitor for evaluating the threshold. Though the EL7373 counterbox does not allow to select the monitor to use as control monitor in monitor mode, it allows to choose the monitor used for pausing the count when the count rate is below the threshold (Who on earth designed this?)
- **countername send *arg1 arg2 arg3 ...*** sends everything behind send to the counter controller and returns the reply of the counter box. The command set to use after send is the command set documented for the counter box elsewhere. Through this feature it is possible to diretclly configure certain variables of the counter controller from within SICS.

5.5 Serial Port Direct Access

At SINQ serial devices are connected to a UNIX/LINUX computer. On this machine runs a serial port server which allows to read and write data through TCP/IP sockets to a serial port connected to the machine. This document describes a simple interface for communicating with such serial devices.

5.5.1 Invocation

The interface to a serial device connected to a UNIX/LINUX computer is initialised with the following command given at the Tcl prompt:
Controller name computer port channel
This command opens a connection to the serial port on the UNIX/LINUX machine and installs a new command in order to interact with it. The parameters:

- **name** is the name of the new command to generate for the connection in Tcl.
- **computer** is the computer name of the UNIX/LINUX machine.
- port: is the TCP/IP port number at which the UNIX/LINUX machine serial port server is is listening. Usually this is 4000.
- channel: is the number of the RS-232 port to connect to.

5.5.2 Usage

Once the connection has been initialised name is available as a new command in Tcl. Let us assume, MC as the name for the purpose of this description. MC then can be used as follows:
MC -tmo value
Configures the timeout for the connection to value. Value is in microseconds.

MC arg1 arg2 ..... argn
Everything after MC is written to the serial port. The reply received from the port is returned.

All these commands can return errors. Mostly these refer to the wrong device being specified on initialisation. The others are network problems.

5.5.3 System Commands

sics_exitus . A single word commands which shuts the server down. Only managers may use this command.

wait *time* waits time seconds before the next command is executed. This does not stop other clients from issuing commands.

resetserver resets the server after an interrupt.

dir a command which lists objects available in the SICS system. Dir without any options prints a list of all objects. The list can be restricted with:

dir var prints all SICS primitive variables

dir mot prints a list of all motors

dir inter driv prints a list of all drivable objects. This is more then motors and includes virtual motors such as environment devices and wavelength as well.

dir inter count Shows everything which can be counted upon.

dir inter env Shows all currently configured environment devices.

dir match wildcard lists all objects which match the wildcard string given in wildcard.

status A single word command which makes SICS print its current status. Possible return values can be: Eager to execute commands, Scanning, Counting, Running, Halted. Note that if a command is executing which takes some time to complete the server will return an ERROR: Busy message when further commands are issued.

status interest initiates automatic printing of any status change in the server. This command is primarily of interest for status display client implementors.

backup [***file***] saves the current values of SICS variables and selected motor and device parameters to the disk file specified as parameter. If no file parameter is given the data is written to the system default status backup file. The format of the file is a list of SICS commands to set all these parameters again. The file is written on the instrument computer relative to the path of the SICS server. This is usually /home/INSTRUMENT/bin.

backup motsave toggles a flag which controls saving of motor positions. If this flag is set, commands for driving motors to the current positions are included in the backup file. This is useful for instruments with slipping motors.

restore [***file***] reads a file produced by the backup command described above and restores SICS to the state it was in when the status was saved with backup. If no file argument is given the system default file gets read.

killfile decrements the data number used for SICS file writing and thus consequently overwrites the last datafile. This is useful when useless data files have been created during tests. As this is critical command in normal user operations, this command requires managers privilege.

5.5.4 Configuration Commands

SICS has a command for changing the user rights of the current client server connection, control the amount of output a client receives and to specify additional logfiles where output will be placed. All this is accessed through the following commands:

The SICS server logs all its activities to a logfile, regardless of what the user requested. This logfile is mainly intended to help in server debugging. However, clients may register an interest in certain server events and can have them displayed. This facility is accessed via the **GetLog** command. It needs to be stressed that this log receives messages from **all** active clients. GetLog understands the following messages:

- **GetLog All** achieves that all output to the server logfile is also written to the client which issued this command.
- **GetLog Kill** stops all logging output to the client.
- **GetLog OutCode** request that only certain events will be logged to the client issuing this command. Enables only the level specified. Multiple calls are possible.

Possible values for OutCode in the last option are:

- **Internal** internal errors such as memory errors etc.
- **Command** all commands issued from any client to the server.
- **HWError** all errors generated by instrument hardware. The SICS server tries hard to fix HW errors in order to achieve stable operations and may not generate an error message if it was able to fix the problem. This option may be very helpful when tracking dodgy devices.
- **InError** All input errors found on any clients input.
- **Error** All error messages generated by all clients.
- **Status** some commands send status messages to the client invoking the command in order to monitor the state of a scan.
- **Value** Some commands return requested values to a user. These messages have an output code of Value.

The **config** command configures various aspects of the current client server connection. Basically three things can be manipulated: The connections output class, the user rights associated with it, and output files.

- The command **config OutCode val** sets the output code for the connection. By default all output is sent to the client. But a graphical user interface client might want to restrict message to only those delivering requested values and error messages and suppressing anything else. In order to achieve this, this command is provided. Possible values Values for val are Internal,Command, HWError,InError,Status, Error, Value. This list is hierarchical. For example specifying InError for val lets the client receive all messages tagged InError, Status, Error and Value, but not HWError, Command and Internal messages.

- Each connection between a client and the SICS server has user rights assocociated with it. These user rights can be configured at runtime with the command **config Rights Username Password** . If a matching entry can be found in the servers password database new rights will be set.

- Scientists are not content with having output on the screen. In order to check results a log of all output may be required. The command **config File name** makes all output to the client to be written to the file specified by name as well. The file must be a file accessible to the server, i.e. reside on the same machine as the server. Up to 10 logfiles can be specified. Note, that a directly connected line printer is only a special filename in unix.

- **config close num** closes the log file denoted by num again.

- **config list** lists the currently active values for outcode and user rights.

5.6 SICS Trouble Shooting

There is no such thing as bug free software. There are always bugs, nasty behaviour etc. This document shall help to solve these problems. The usual symptom will be that a client cannot connect to the server or the server is not responding. Or error messages show up. This section helps to solve such problems.

5.6.1 Looking at Log Files

The first thing to do, especially when confronted with confusing statements from either users or instrument scientists, is to look at the SICS servers log files. The last 1000 lines of the instrument log are accessible from any SICS client or through the WWW interface. The SICS commands:

commandlog tail shows the last 20 lines of the log.

commandlog tail n shows the last n lines of the log.

will show you the information available. In order to see more, log in to the instrument account. There the following unix commands might help:

- **sicstail** shows the last 20 lines of the current log file and its name

- **sicstail n** shows the last n lines of the current log file.

In order to see some more, cd into the log directory of the instrument account. In there are files with names like:

```
auto2001-08-08@00-01-01.log
```

This means the log file has been started at August, 8, 2001 at 00:01:01. There is a new log file daily. Load appropriate files into the editor and look what really happened.

Another good ideas is to use the unix command grep on assorted log files. A grep for the strings ERROR or WARNING will more ofteh then not give an indication for the nature of the problem.

The log files show you all commands given and all the responses of the system. Additionally there are hourly time stamps in the file which allow to narrow in when the problem started. Things to watch out for are:

MOTOR ALARM This message means that the motor failed to reach his position for a couple of times. This is caused by either a concrete shielding element blocking the movement of the instrument, badly adjusted motor parameters, mechanical failures or the air cushions not operating properly.

EL734__BAD_EMERG_STOP Somebody has pushed the emergency stop button. This must be released before the instrument can move again. Moreover the motor controller will not respond to further commands in this mode. Thus restarting SICS on this error message will make SICS fail to initialize the motors affected!

EL*__BAD_PIPE, BAD_RECV, BAD_ILLG, BAD_TMO, BAD_SEND** Network communication problems. Can generaly be solved by restarting SICS.

EL737__BAD_BSY A counting operation was aborted while the beam was off. Unfortunately, the counter box does not respond to commands in this state and ignores the stop command sent to it during the abort operation. This can be safely ignored, SICS fixes this condition.

5.6.2 Starting SICS

An essential prerequisite of SICS is that the server is up and running. The system is configured to restart the SICServer whenever it fails. Only after a reboot or when the keepalive processes were killed (see below) the SICServer must be restarted. This is done for all instruments by typing:

```
startsics
```

at the command prompt. startsics actually starts two programs: one is the replicator application which is responsible for the automatic copying of data files to the laboratory server. The other is the SICS server. Both programs are started by means of a shell script called **keepalive**. keepalive is basically an endless loop which calls the program again and again and thus ensures that the program will never stop running.

When the SICS server hangs, or you want to enforce an reinitialization of everything the server process must be killed. This can be accomplished either manually or through a shell script.

5.6.3 Stopping SICS

All SICS processes can be stopped through the command:

```
killsics
```

given at the unix command line. You must be the instrument user (for example DMC) on the instrument computer for this to work properly.

5.6.4 Restart Everything

If nothing seems to work any more, no connections can be obtained etc, then the next guess is to restart everything. This is especially necessary if mechanics or electronics people were closer to the instrument then 400 meters.

1. Reboot the histogram memory. It has a tiny button labelled RST. That' s the one. Can be operated with a hairpin, a ball point pen or the like.
2. Wait 5 minutes.
3. Restart the SICServer. Watch for any messages about things not being connected or configured.
4. Restart and reconnect the client programs.

If this fails (even after a second) time there may be a network problem which can not be resolved by simple means.

5.6.5 Checking SICS Startup

Sometimes it happens that the SICServer hangs while starting up or hardware components are not properly initialized. In such cases it is useful to look at the SICS servers startup messages. In order to do so, both the SICServer and its keepalive process must be killed first. On the instrument acount issue the command:

```
ps -A | grep SICS
```

A message like this will be printed:

```
  23644 ??       I        0:00.00 ksh keepalive SICServer focus.tcl
 23672 ??       R       59:24.05 SICServer focus.tcl
  7119 ttyp6    S  +     0:00.00 grep SICS
```

Remember the numbers in the first columns (the PID's) and kill both programs by issuing the command:

```
kill -9 pid pid
```

Example:

```
kill -9 23644 23672
```

Note, the numbers are those displayed with the ps -A command. Then cd into the bin directory of the instrument account and issue the unix command:

```
SICServer inst.tcl | more
```

Replace inst.tcl with the name of the appropriate instrument initialisation file. This allows to page through SICS startup messages and will help to identify the troublesome component. The proceed to check the component and the connections to it.

5.6.6 Getting New SICS Software

Sometimes you might want to be sure that you have the latest SICS software. This is how to get it:

1. Login to the instrument account.
2. If you are no there type cd to get into the home directory.
3. Type **killsics** at the unix prompt in order to stop the SICS server.
4. Type **sicsinstall exe** at the unix prompt for copying new SICS software from the general distribution area.
5. Type **startsics** to restart the SICS software.

5.6.7 Hot Fixes

When there is trouble with SICS you may be asked by one of the SICS programmers to copy the most recent development reason of the SICS server to your machine. This is done as follows:

1. Login to the instrument account.
2. cd into the bin directory, for example: /home/DMC/bin.
3. Type **killsics** at the unix prompt in order to stop the SICS server.
4. Type **cp /data/koenneck/src/sics/SICServer .** at the unix prompt.
5. Type **startsics** to restart the SICS software.

!!!!!! WARNING !!!!!!!. Do this only when advised to do so by a competent SICS programmer. Otherwise you might be copying a SICS server in an instable experimental state!

5.6.8 HELP debugging!!!!

The SICS server hanging or crashing should not happen. In order to sort such problems out it is very helpful if any available debugging information is saved and presented to the programmers. Information available are the log files as written continously by the SICS server and posssible core files lying around. They have just this name: core. In order to save them create a new directory (for example dump2077) and copy the stuff in there. This looks like:

```
/home/DMC> mkdir dump2077
/home/DMC> cp log/*.log dump2077
/home/DMC> cp core dump2077
```

The `/home/DMC>` is just the command prompt. Please note, that core files are only available after crashes of the server. These few commands will help to analyse the cause of the problem and to eventually resolve it.