TRICS–Reference Manual

Version July, 2001

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

Contents

Chapter 1

The Four Circle Diffractometer TRICS

1.1 Introduction

The four circle diffractometer TRICS is used for the study of dynamic and magentic structures by neutron diffraction at single crystals. TRICS can be operated in two modes: 1.) with a single counter very much like a traditional four circle diffractometer and 2.) with three position sensitive detectors operating like a oscillation camera as used by protein crystallographers. This manual describes the operation of TRICS using the SICS software.

TRICS is situated at one of the beamlines for thermal neutrons at the spallation source SINQ at PSI, Switzerland. The incident white neutron beam first hits a monochromator crystal. The selected wavelength is not variable. A monochromator lift with two monochromators and the possiblity to select different reflections peaks however make it possible to select between a few different neutron wavelength. The sample is held in a standard eulerian cradle with the usual angles omega, chi and phi. Up to three position sensitive detectors are held on detector holders at different angles in gamma and nu. Or a single detector is fixed to this setup. The three position sensitive detectors may be moved up and down on a sphere around the sample. This allows to access different lattice planes when the eulerian cradle has been replaced by some cryostat or other sample environment devices to heavy to be used in conjunction with an eulerian cradle.

The control of a four circle diffractometer requires a tight integration between measurement procedures and specialized data anlaysis software. For the actual control of the instrument, its movements and the actual measurement of reflections the SICS software is provided. Preliminary data analysis and special four circle calculations are done with a set of F77 programs. These are mostly programs which originate from the ILL, France.

Given this setup the rest of this manual logically has to have three parts:

- A general part (cf. Section 2) describing the principal operation of the SICS software.
- A chapter describing the operation of TRICS with a single counter (cf. Section 3).
- A chapter describing the operation of TRICS with position sensitive (cf. Section 4) detectors.

Chapter 2

General Operation of the SICS instrument control software

This chapter contains information about:

- Basic SICS concepts (cf. Section 2.1)
- Driving motors (cf. Section 2.2)
- Configuring motor parameters (cf. Section 2.3).
- TRICS motor list (cf. Section 2.4).
- Counting and monitors (cf. Section 2.5).
- Sample Environment Device Control (cf. Section 2.6).
- Log I/O to a client logfile (cf. Section 2.7) at the server, all command output to a commandlog (cf. Section 2.8) or to your local machine using the open log file option in the File menu of the SICS command line client.
- Doing batch processing using batch files (cf. Section 2.9) or the LNS specific Rünbuffer (cf. Section 2.10) system.
- **Interrupt** SICS by hitting the interrupt button near the bottom of the SICS command line client.
- How to perform crystallographic computations (cf. Section 2.11) online.
- Some specialist commands for configuring certain aspects of your client connection (cf. Section 2.12) and for in depth interaction with the SICS server (cf. Section 2.13).

2.1 Basic SICS concepts

2.1.1 General structure

SICS is a client server system. The application the user sees is usually some form of client. A client has two tasks: the first is to collect user input and send it to the SICS server which then executes the command. The clients second task is to listen to the the server messages and display them in a readable format. This aproach has two advantages: clients can reside on machines across the whole network thus enabling remote control from everywhere in the world. The second advantage is that new clients (such as graphical user interface clients) can be written in any feasible language without changes to the server.

2.1.2 SICS Command Syntax

SICS is an object oriented system. This is reflected in the command syntax. SICS objects can be devices such as motors, single counters, histogram memories or other hardware variables such as wavelength or Title and measurement procedures. Communication with these objects happens by sending messages to the target object. This is very simply done by typing something like: object message par1 par2 .. parn. For example, if we have a motor called A1:

```
A1 list
```

will print a parameter listing for the motor A1. In this example no parameters were needed. There exist a number of one-word commands as well. For compatability reasons some commands have a form which resembles a function call such as:

```
drive a1 26.54
```

This will drive motor a1 to 26.54. All commands are ASCII-strings and usually in english. SICS is in general CASE INSENSITIVE. However, this does not hold for parameters you have to specify. On a unix system for instance file names are case sensitive and that had to be preserved. Commands defined in the scripting language are lower case by convention.

2.1.3 Authorisation

A client server system is potentially open to unauthorised hackers who might mess up the instrument and your valuable measurements. A known problem in instrument control is that less knowledgeable user accidentally change instrument parameters which ought to be left fixed. In order to solve these two problems SICS supports authorisation on a very fine level. As a user you have to specify a username and password in order to able to access SICS. Some clients already do this for you automatically. SICS support four levels of access to an instrument:

- **Spy** may look at everything, request any value, but may not actually change anything. No damage potential here.
- **User** is privileged to perform a certain amount of operations necessary to run the instrument.
- **Manager** has the permission to mess with almost everything. A very dangerous person.
- **Internal** is not accessible to the outside world and is used to circumvent protection for internal uses. However some parameters are considered to be so critical that they cannot be changed during the runtime of the SICS-server, not even by Managers.

All this is stated here in order to explain the common error message: You are not authorised to do that and that or something along these lines.

2.1.4 SICS variables

Most of the parameters SICS uses are hidden in the objects to which they belong. But some are separate objects of their own right and are accessible at top level. For instance things like Title or wavelength. They share a common syntax for changing and requesting their values. This is very simple: The command *objectname* will return the value, the command *objectname newvalue* will change the variable. But only if the authorisation codes match.

2.1.5 The SICS Command Line Client

The most common client for controlling SICS is the **SICS command line client**. This application can be started by typing the command:

```
sics &
```

at the Unix prompt. Before this program is ready to collaborate with you you have to connect it to an instrument using the options in the connect pulldown menu. The screen is roughly divided in three areas: The top area shows all input to and output from the server. The middle area shows the command history. At the lower end is a text entry field which allows you to type commands to be sent to the SICS server. For more information about this client consult the online help of this application.

2.2 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.

2.3 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* . 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 2.2) command.

2.3.1 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.

2.4 TRICS Motors

This chapter has still to be defined as there is a war waging in LNS about names. Final names will be entered after the end of hostilities. More then one name may be stated as SICS supports aliases. For now, you have to do with:

- OM = A3 = omega motion of the eulerian cradle.
- CH = chi circle of the eulerian cradle.
- PH = phi circle of the eulerian cradle
- TH = A4 = two theta angle of the detector.

2.5 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 the preset 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.

2.6 Sample Environment Devices

2.6.1 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.

2.6.2 SampleEnvironment 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.

2.6.3 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 2.6.2) 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.

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 two 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.

The last system is 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. 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 DigitalUnix 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.

2.6.4 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. If there is already an other device selected, it must be deleted and TECS must be reinstalled:

EVFactory del temperature
EVFactory new temperature tecs

The sample environment device is selected automatically by a coding in the plug of the sensor/heater cable(s). If this does not work (plugs without coding or temporarely use of a wrong cable) you may select the device with

temperature device *device*

You may want to verify the selected device with

temperature device

The actually known devices (April 2000) are:

- orange cryostats: **ill1** (50mm), **ill2** (70mm), **ill3** (cryofurnace), **ill4** (FOCUS), **ill5** (100 mm)
- closed cycles: **cti1**, **cti2**, **cti3**, **cti4**, **cti5** (maxi), **cti6** (FOCUS), **apd** (TriCS), **ccr4k** (4K)
- other: **hef4c** (He-flow cryostat 4circle), **sup4t** (4 T supraconducting magnet)

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 Macintosh computer running the SINQ terminal server program via a serial cable. Please make sure with a different Macintosh or a PC that the serial line is OK and the ITC-4 responding before plugging it in.

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 at the Macintosh computer computer running the SINQ terminal server program listening at port port. For example:

EVFactory new Temp ITC4 lnsp22.psi.ch 4000 7

connects Temp to the Macintosh named lnsp22, serial port 6 (7 above is no typo!), listening at port 4000.

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

timeout Is the timeout for the Macintosh terminal server program waiting for responses from the ITC-4. Increase this parameter if error messages containg ?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 6 on the Macintosh serial port extension box. Port 6 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 name Macintoshname 4000 7
   Thereby replace name with the name you want to address the ITC4 in SICS. A good choice for a name is temperature, as such a value will be written to data files. 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. Macintoshname is the name of the instrument Macintosh PC.

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 the Macintosh 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 the serial printer port of the Macintosh serial port server computer. This makes the channel number to use for initialisation a 1 always. 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 macintosh-computer name port channel
evfactory new temperature tcl name-of-array

An example for the SANS:

inihaakearray eimer lnsp25.psi.ch 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.

Dilution Cryostat

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.

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 Mac-PC Mac-port Mac-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. Mac-PC is the name of the Macintosh PC to which the controller has been connected, Mac-Port is the port number at which the Macintosh-PC's serial port server listens. Mac-channel is the RS-232 channel to which the controller has been connected. For example (at SANS):

```
evfactory new magnet bruker lnsp25.psi.ch 4000 9
```

creates a new command magnet for a Bruker magnet Controller connected to serial port 9 at lnsp25. 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 CryoFurnace.

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 Mac-PC Mac-port Mac-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. Mac-PC is the name of the Macintosh PC to which the controller has been connected, Mac-Port is the port number at which the Macintosh-PC's serial port server listens. Mac-channel is the RS-232 channel to which the controller has been connected. For example (at DMC):

```
evfactory new temperature ltc11 lnsp18.psi.ch 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.

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 Mac-PC Mac-port Mac-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. Mac-PC is the name of the Macintosh PC to which the controller has been connected, Mac-Port is the port number at which the Macintosh-PC's serial port server listens. Mac-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 Macintosh port 13 (the last in the upper serial port connection box) is configured like this! Thus, an example for SANS and the name temperature looks like:

```
evfactory new temperature euro lnsp25.psi.ch 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 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 Mac-PC Mac-port Mac-channel index

with Mac-PC, Mac-port and Mac-channel being the usual data items for describing the location of the EL755-controller at the Macintosh 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 lnsa09.psi.ch 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.

2.7 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 This command 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.

2.8 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.

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.

2.9 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. Then there are some special commands which can be used within macro-sripts:

ClientPut sometext1 ... writes everything after ClientPut to the client which started the script. This is needed as SICS supresses the output from intermediate commands in scripts. Except error messages and warnings. With clientput this scheme can be circumvented and data be printed 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 varaible var. 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.

2.10 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.

2.11 4 Circle Diffractometer Setting Calculation

An essential part of operating a 4 circle diffractometer is the calculation of the setting angles for the diffractometer for a given reciprocal lattice point from the UB matrix and the wavelength. SICS does this through the hkl object. The hkl object can calculate the required settings both for normal 4 circle configuration and normal beam configuration. It is possible to specify if low chi or high chi values are preferred. The wavelength can be dealt with in two ways: it can be set manually. Or a variable controlling the wavelength can be specified. The hkl object will then be updated automatically with the newest value for the wavelength whenever the wavelength changes.

The hkl object understands the following commands:

hkl list Prints a listing of all relevant settings calculation parameters.

hkl current Prints the value of the last calculated reflection.

hkl lambda val Manually sets the wavelength to val. No automatic updates of the wavelength will be performed.

hkl lambdavar val Sets the name of the variable controlling the wavelength. The wavelength in the hkl object will be automatically updated whenever the wavelength is modified by driving variable val. This is a user command because the TRICS spectrometer has more then one monochromator.

hkl setub a11 a12 a13 a21 a22 a23 a31 a 32 a33 SetUB sets the UB matrix to the nine values given.

hkl nb val Switches the mode for normal beam calculation. If val is 1 a normal beam calculation is performed, else normal four circle calculations are done.

hkl quadrant val Defines the chi quadrant to prefer. The parameter val can be 0 for low chi and 1 for high chi.

hkl calc h1 h2 h3 psi hamil Calculates and prints the setting angles for the reflection (h1,h2,h3). Optionally a psi value and a hamilton position can be specified.

hkl run h1 h2 h3 psi hamil Calculates the setting angles for the reflection (h1,h2,h3) and starts the motors to run to that position. This command will return immediately and will not wait for the diffractometer to arrive at the setting angles requested. Optionally a psi value and a hamilton position can be specified.

hkl drive h1 h2 h3 psi hamil Calculates the setting angles for the reflection (h1,h2,h3) and starts the motors to drive to that position. This command will wait for the diffractometer to arrive at the setting angles requested. Optionally a psi value and a hamilton position can be specified.

2.12 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.

2.13 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 single word command which lists all objects available in the SICS system in its current configuration.

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.

Chapter 3

Running TRICS with a Single Counter

In this mode TRICS simulates a conventionell four circle diffractometer much like a x-ray diffractometer as commercially available. The tasks which have to be solved are:

- Locate Reflections
- Index reflections and refine a UB-matrix.
- Measure a couple of reflections.
- Furthermore there are some specialities.

There are two ways to achieve all this: The older way uses some built in SICS functionality plus some external programs inherited from the ILL. This is called the ILL operation. Then a complete four circle package called DIFRAC from P. White and Eric Gabe was integrated into SICS. This is The Difrac way of operation.

3.1 DIFRAC

The DIFRAC commands are accessed by prepending the difrac commands with **dif**. For example: `"dif td"` calls the difrac td command. For more information on DIFRAC commands see the separate DIFRAC manual.

3.2 ILL operation

3.2.1 Locate Reflections

If you know x-ray single crystal diffractometers you'll expect sophisticated reflection search procedures here. Nothing is available in this field in SICS. It was deemed inapropriate for neutron research. The first reflections must be found by hand. Something which may help in this is a quick scan facility which allows to run a motor and print counts while the motor is moving. This can be invoked by a command like this:

```
susca var start end time
```

The parameters are:

- var: the motor or variable to scan.
- start: the start position from which to scan.
- end: the end position for this scan.
- time: The maximum counting time.

Be aware that this is inprecise and liable to changes in the source current. But it may help to locate the aproximate position of a peak.

Once a peak has been found, its position can be optimised and centered with the peak optimiser (cf. Section 3.4).

The next thing to do is to store the reflection and find other ones. Once a few reflections have been found, the need to be written to disk. This can be accomplished with the object rliste which has the following subcommands:

rliste clear clears all entries from the list

rliste store saves the current diffractometer position into the list

rliste write file Writes the contents of the reflection list to the file specified.

3.2.2 Indexing Reflections and Refining UB-Matrix

For these purposes the external programs INDEX and RAFIN are provided. These programs are courtesy of the ILL, France.

3.2.3 Measuring Reflections

Before measuring reflections a list of reflections to measure must be created. This is done with the external program HKLGEN. Then reflections this reflection list can be fed into SICS using the mess (cf. Section 3.6) command. mess creates two output files: a .col file containing the reflection profiles of all the relfections and a .dat files which contains the HKL,F,sig(F),TH,OM,CH,PH for each reflection. Intensity has then be integrated within SICS. The .col files can be processed by the program REFRED which allows to perform more advanced data reduction chores and has a choice of integration methods for reflection data. Please note, that SICS does not automatically measure standard reflections. It is your task to add suitable standard reflections into the reflection list.

3.3 Special Commands

As of current this section only holds the hklscan commmand (cf. Section 3.7) which allows to express a scan in Miller indizes. This is in fact a scan in reciprocal space.

3.4 The Peak Optimiser

In instrument control the need may arise to optimise a peak with respect to several variables. Optimising means finding the maximum of the peak with respect to several variables. This is useful during instrument calibration, for example. Four circle diffractometers use this facility on a day to day basis for finding and verifying the exact position of reflections. In order to support both usages a more general module has been implemented. The algorithm is like this:

```
while errors gt precision and cycles lt maxcycles
  for all variables
      do a scan
      Try find the maximum, two halfwidth points and the peak center.
      if failure extend the scan.
      if success shift the variable, remember last shift.
      If shift lt precicison mark this variable as done
   end for
end while
```

Possible outcomes of this procedure are: success, the peak was lost or the maximum number of cycles was reached. This routine requires that the instrument is currently placed somewhere on the peak and not miles away.

The Peak Optimiser is implemented as an object with the name opti. It understand the following commands:

opti clear clears the optimiser.

opti addvar name step nStep precision This command adds a variable to optimise to the optimiser. The user has to specify the name of the variable, the step width to use for scanning, the number of steps needed to cover the full peak when scanning and the precision which should be achieved when optimising the peak. The step width and number of steps parameters should cover the whole peak. However, the Optimiser will extend the scan is the specified range is not sufficient.

opti run Starts the optimiser. It will then optimise the peak. This may take some time.

The behaviour of the optimiser can be configured by modifying some parameters. The synatx is easy: **opti parameter** prints the value of the parameter, **opti parameter newval** sets a new value for the parameter. The following parameters are supported:

maxcycles The maximum number of cycles the optimiser will run when trying to optimise a peak. The default is 7.

threshold When a peak cannot be identified after a scan on a variable, the optimiser will check if there is a peak at all. In order to do that it searches for a count rate higher then the threshold parameter. If such a rate cannot be found the optimiser will abort and complain that he lost the peak.

channel The counter channel to use for scanning. The default is to use the counter. By modifying this parameter, the optimiser can optimise on a monitor instead.

countmode The counting mode to use when scanning. Possible values are **timer** or **monitor**.

preset The preset value to use for counting in the scan. Depending on the status of the countmode parameter this is either a preset time or a preset monitor.

3.5 External FORTRAN 77 Programs

3.5.1 INDEX

The program indexes reflections on the basis of observed 2Theta, Omega, Chi, Phi angles when the cell constants and wavelength are known. It does not take into account systematic extinctions.

The process, when successful, has three steps.

First, it calculates, for each set of observations, all possible HKL's for which theta(calc) lies within theta(obs) +/- delta theta. delta theta is given - see below -. For delta theta = 0, the value defaults to 0.05.

Second, it finds for all combinations of two sets of observations, the angle between the indexed HKL's for which angle(calc) lies within angle (obs) +/- delta. delta given - see below -. Delta = 0 causes default to 0.2

Finally, it finds all sets of indexed HKL's that explain all angles between the observed sets of Omega, Chi and Phi.

The user will normally be presented with several possible sets of HKL's which fit within the limits given. - they are already tested for right-handedness - and he must now choose which set he likes the most.

If he wishes he may now specify which set of reflections he likes and the program will then set up the input file for the program rafin, - see next section -. The program ensures that the first two reflections are acceptable to rafin. The user must say whether he wants the ub matrix written directly into lsd (for instant use) and which file he wants his rafin input to come from (usually rafin.dat). The program rafin is then automatically started.

Input to index can be done either from the terminal or from a file index.dat The format is the same, an example is given here.

```
THIS IS A DUMMY EXAMPLE       ( text )
5.82 16.15 4.09 90 103.2 90 .84 .15 (cell constants, lambda delta)
16.18 9.01 34.71 14.74 0  (2Th, Om, Ch, Ph, delta theta)
13.21 7.8 .71 -56.13 0.1  (2Th, Om, Ch, Ph, delta theta)
etc. etc.
0  ( end list with 2Th = 0 )
```

The program will only suggest sets of indexed HKL's if all reflections are explained. If not, the user must himself look through the list of observed and calculated angles to find a partial list.

3.5.2 RAFIN

This program determines orientation matrix (ub) from two or more sets of orientation angles for reflections, and refines (optionally) wavelength; zeroes of 2Theta, Omega or Chi; a, b, c, alpha, beta or gamma.

To call the program, type :– rafin after having set up the input file :

The input data are on rafin. dat (teletype input can be used, but is cumbersome)

Use teco to make or correct the file.

An example of the input file (comments in parentheses) :–

```
0 1     (second no. 0/1 for  ub not transferred/transferred to lsd)
0       (always)
0 -4 -2 28.01 13.75  81.59  42.05  (H K L 2Theta Omega Chi Phi)
4 -6  7 50.84 25.37  34.04  18.41
-2 -6  0 41.55 20.53  66.93  59.99
4  0  4 19.74  9.94 -16.92  -5.40
1 -5 -3 35.59 17.70  82.32   1.40
6  0  0 18.47  9.26  -2.32 -46.95
0 .8405 (0/1 do not/do refine lambda; and lambda)
0 0.0 1 0.0 1 0.0 (0/1 for do not/do refine, 2Theta zero,
-0/1 for do not/do refine, Om zero,
-0/1 for do not/do refine, Chi zero.
0 15.9158 0 7.1939 0 14.277 0 90 0 98.72 0 90 (ditto for  a,  b,  c,  alpha,  beta and  gamma)
2 0 0 (H K L list for angles to be calculated)
0 2 0
0 0 2
0 0 0 (end of list)
-1 (end of input file)
```

Ensure that lsd is not running if you wish to transfer the matrix and wavelength directly into its parameter section, otherwise it may not be successful. rafin will never modify the zeroes for you. This is for you to do by adding them to the values in zer of par. Remember that for a well aligned diffractometer, they will never change by very much.

Note: the first two reflections should be far away enough in reciprocal space to define a plane. They must be at least 45 deg apart in Phi and only one may have Chi greater than 45 deg.

Note also that higher angle (Theta) reflections usually give a better fit.

You cannot, obviously, refine lambda and your cell at the same time.

Acknowledgements

The index program was written by M.S.Lehmann, and J.M.Savariault.

The rafin program was implemented at the ill by A.Filhol and M.Thomas. It was implemented on the pdp 11 system by A.Barthelemy.

3.5.3 HKLGEN

THIS PROGRAM IS USED TO GENERATE A LIST OF HKL's which can be used for input to the measurement routines of lsd. Indices can be generated internally in lsd, but it is generally considered easier to create a list, and measure from this.

hklgen will generate HKL's according to min and max specified indices, and will write them into output file(s) if they are inside the Theta limits.

If chi and phi limits are specified, the program will also look to see if the hkl is measurable inside these machine limits.

If not, it will see if the Friedel Pair is inside limits

If this is also outside limits, it will see if the reflection can be measured for hkl psi=180 - note this option means chi = 90 -¿ 180 i.e. up-side-down.

If measurement is not possible for any of these conditions, the hkl is declared blind.

Comments like fr.pr hichi blind indicate these on tty output.

To run the program do :–

hklgen Input to the program is either from the terminal or from a file hklgen.dat, already created by the user.

Input from Terminal

The first question asked under this option is whether a file hklgen.dat should be created. If subsequent runs are envisaged, this might be a good idea. In this case teco can be used to make small changes to the input and the program can be quickly re-run. hklgen then asks for the following parameters :–

1. Title. Up to 80 characters to be displayed at the top of the output.

2. Wavelength and the 9 'rules limiting possible reflections' - see appendix C – If you give wavelength = 0, the wavelength, extinction rules, and orientation matrix will be taken from lsd's parameter files. chi and phi software limits are also taken but an opportunity is given to over-write them. – If the wavelength is given explictly, followed by up to 9 numbers for the extinction rules, the orientation matrix must then be given line by line.

3. Theta limits. (Note not 2-Theta limits). chi and phi limits (if required) must also be given in this line. – If zeroes are given, no limits will be included in the calculations. – If nothing is given, LSD's limits will be used if data was taken from the parameter files, or it will default to zeroes if the data above was given by the user.

4. Three numbers indicating the relative speed of variation of h, k and l. First number is the slowest changing index, third number is the fastest changing index. 1/2/3 is used to represent h/k/l. e.g. 3 2 1 means L changes slowest, then K, with H changing fastest.

5. Minimum and maximum indices in hkl are now requested. You must give hmin hmax kmin kmax lmin lmax. Note however that before starting the calculations, hklgen calculates itself what is the maximum value for each index for the specified Theta range and if this is inside these values, they will be modified. Therfore, if, for example 0 999 0 999 -999 999 is given, hklgen will calculate the maximum values and give HKL's for positive H, positive K, positive and negative L.

6. Four numbers concerning various outputs from the program. a) The first npunch concerns the hkl output file. 0 = no file for output 1 = file for output containing hkl only in 3I4 format. 2 = file for d15-ren (not for d8-d9) 3 = file for output containing hkl and setting angles. b) The second ipara concerns machine geometry.

0 = Bissecting geometry - (normal for d8-d9). 1 = Normal beam geometry - (rarely used on d8-d9) 3 = d15 lifting counter mode (used with npunch = 2) c) The third number nbites concerns hkl output. 1 = write hkl for each case in four separate files. 0 = write all HKL's in one single file FOR00x. dat (x specified below). d) The fourth number nlist concerns terminal output. 0 = write each hkl with angles and comment on terminal. (can take time and consume paper). 1 = suppress most of the output on tty.

7. If in previous line FOR00x. dat was specified for hkl output, X must be given. This is the last line in the input but is not always necessary.

The program then generates as specified, creating file(s) if required. It gives a resume at the end and exits.

Input from file hklgen.dat

Input is given in exactly the same order as above, so for a more detailed description of each parameter see previous section.

Two possible examples are given below.

```
KNUDDERKRYSTAL LAVET AF AARKVARD, 120K      (Text, 80 characters)
0.8405 0 0 0 0 0 0 0 0 0   (Wavelength and the 9 Extinction rules. )
0.043361 -.04190  .5399      (  ub given in three separate lines
-.046409 -.032053 .03721      - as wavelength is given explicitly
-.00256  -.12861 -.02687
0 36 -20 95       ( Theta limits and Chi limits - note - no limits on Phi. )
2 1 3       ( relative speeds of  hkl. - K slowest - L fastest. )
-99 -1 0 5 -99 99       ( Hmin,Hmax,Kmin, etc. - note all L's with all neg
1 0 0 1       ( a) Output file of  hkl.
        b) Bisecting geom - usual.
c) All HKL's in  for00x.dat.
d) Suppress most  tty output. )
3 (  hkl file on  for003.dat )
```

hklgen is a program which has evolved in the hands of : A.Filhol, S.Mason. A.Barthelemy and J.Allibon.

3.5.4 Encoding of Extinction Rules

```
HKL
              0 : NO CONDITIONS
      1 : H + K + L = 2n
      2 : H, K, L all even or all odd
      3 : -H + K + L = 3n
      4 : H = K + L = 3n
      5 : H + K = 2n
      6 : K + L = 2n
      7 : H + L = 2n
      8 : H + K + L = 6n
```

```
      9 : H, K, L all even
      10 : H, K, L all odd
           11 : If H - K = 3n, then L = 6n
H K 0
              0 : No conditions
      1 : H = 2n
      2 : K = 2n
      3 : H + K = 2n
      4 : H + K = 4n
0 K L
              0 : No conditions
      1 : K = 2n
      2 : K + L = 2n
      3 : K + L = 3n
      4 : K + L = 4n
      5 : L = 2n
H 0 L
      0 : No conditions
      1 : L = 2n
      2 : H = 2n
      3 : L + H = 2n
      4 : L + H = 4n
H H L
      0 : No conditions
      1 : L = 2n
      2 : H = 2n
      3 : 2H + L = 4n
```

3.6 Reflection List Processor

This section describes the means for doing a standard single counter four circle diffractometer measurement with SICS. A prerequisite for that is a file with a list of reflections to measure. This is a simple file with three floating point values per line giving the HKL of the reflection to measure. Do not forget to put standard reflections into that file any now and then. Another prerequisite is, that the UB-matrix had been determined beforehand and that SICS has the updated values. Also check the value of lambda in the hkl-object.

The measurement procedure is rather simple: If a reflection is accessible the diffractometer is positioned on that reflection. Then a scan is done for the reflection and data written to file. The scans all run with a fixed scan widths, counter preset and countmode. There is a choice of omega scan or omega two theta scan. It is known that there are more sophisticated measurement schemes for four circle diffraction, but as TRICS is only temporarily operated with a single counter not much optimisation seemed necessary.

Three files will be written starting from a root such as tricsnumberyear. For instance trics05601998 means file number 560 in 1998. The file ending in .log will contain the console log. This is extremely verbose. Another file ending with .col will contain the reflection, diffractometer settings and the measured profile. The third file, ending with .rfl will contain for each refelction, the HKL, the diffractometer settings and the intensity and sigma intensity as calculated by the SICS internal integration routine. It does a Grant Gabe integration (see J.Appl. Cryst (1978), 11, 114-120).

For the purpose of the command description it is assumed, that this facility is accessible as object mess within SICS. Interaction with this object happens through the following commands:

mess start Creates a new set of files and writes some header info.

mess measure filename iSkip Starts a measurement. Reads reflections from the file filename. iSkip is an optional parameter which allows to skip iSkip lines in the file. This is for recovery in cases of accidental or purposeful interruption of the measurement.

mess genlist filename iSkip Mesures reflection from filename. The file is expected to have been created by hklgen and to include all the angle settings. The optional parameter iSkip determines the number of lines to skip in the file. This feature allows to continue measurement on not fully processed files.

mess reopen filename Reopens an already existing file set for appending. Only the file root without directory info or endings needs to be given.

mess close Closes the current data file set.

mess file Prints the current data file name.

Then there are a few parameter commands. They follow the general scheme:

mess parameter Prints the current value of the parameter

mess parameter value Sets the parameter to the new value.

This object knows about the following parameters:

countmode The counting mode to use. Possible values are timer or monitor.

preset The preset to use for counting

mode The measurement mode. Posssible values are omega for omega scans and omega2theta for omega two theta scans.

np number of points to collect for each profile.

step The step width in omega to use for scanning.

compact Determines if the scan data output to the SICS is in normal (compact = 0) or condensed (compact = 1) form. The default is 1.

mess supports two geometries: the first is the usual bisecting geometry. The second is the normal beam geometry where the detector is moved out of plane. This si accounted for by two switches:

mess bi switches into bissectiong mode. This is the default.

mess nb switches into normal beam mode.

This object supports some file management functionality. It caters for the problem that experiments may need to be continued. Thus reopening files and continuation of reflection processing at a point way down the reflection file is supported. Consequently the start of a new experiment requires the following steps:

- Create a new set of files with **mess start**.
- Configure the scans with the parameter commands.
- Start processing a reflection file with either the **mess genlist** or **mess measure** commands.

If you need to continue reflection file processing after an abort or after solving a problem the following steps are required:

- Determine the file number you were working at and the line number in the reflection file where you wish to continue processing.
- Set the file root with the **mess reopen** command.
- Configure the scan parameters again.
- Restart the measurement with either **mess genlist** or **mess measure** but specify the iSkip parameter according to the position in the reflection file where processing should continue.

3.7 Hklscan

Hklscan is a command which allows to scan in reciprocal space expressed as Miller indizes on a four circle diffractometer. Prerequisite for this is the existence of a scan object and the hkl-object for doing crystallographic calculations. Make sure the properties of the hkl object (UB, wavelength, NB) have some reasonable relation to reality, otherwise the diffractometer may travel to nowhere. Also it is a good idea to drive the diffractometer to the end points of the intended scan in reciprocal space. hklscan will abort if the requested scan violates diffractometer limits. The commands implemented are quite simple:

hklscan start fH fK fL sets the start point for the HKL scan. Three values required, one for each reciprocal axis.

hklscan step sH sK Sl sets the step width in reciprocal space. Three values required, one for each reciprocal axis.

hklscan run NP mode preset executes the HKL scan. NP is the number of points to do, mode is the counting mode and can be either timer or monitor and preset is the preset value for the counter at each step.

Data is written automatically into a slightly modified TOPSI data format file. The status display with topsistatus or scanstatus might be slightly erratic as it uses two theta as x-axis.

Chapter 4

TRICS with Position Sensitive Detectors

TRICS with a PSD requires the following special features.

- Instructions for dealing wih histogram memory (cf. Section 4.1).
- NeXus (cf. Section 4.2) data handling for TRICS.
- A peak search (cf. Section 4.3) command.
- A TRICS specific count and scan (cf. Section 4.5) command.

4.1 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.

4.1.1 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 histograms in memory.

Length gives the length of an individual histogram.

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

dim0, dim1, dim2, ... dimn define the logical dimensions of the histogram. Must be set if the the sum command (see below) is to be used. This is a clutch necessary to cope with the different notions of dimensions in the SINQ histogram memory and physics.

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.

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.

4.1.2 Histogram Memory Commands

Besides the configuration commands the HM understands the following commands:

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

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

CountMode with a new values 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 command sthis 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 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 amd 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.

4.2 TRICS NeXus File Writing Object

TRICS writes its data files in the upcoming NeXus data format standard for neutron scattering and X-ray diffraction. The user may interact with this object through the following commands:

nexus start Starts a new NeXus file.

nexus reopen filename Reopens a NeXus file which has already been written to.

nexus dumpframe Writes a frame of data at the current settings to the NeXus file.

nexus file Prints the filename of the data file currently in use.

4.3 TRICS PSD Peak Search

For almost any measurement at TRICS a UB matrix has to be determined beforehand. In order to do this a couple of peak must be located by some means. This section describes how the computer can help in finding an initial set of peaks.

The algorithm is quite simple: It consists of a big loop over ranges of the four circle angles two theta, omega, chi and phi. At each position a counting operation is performed. Then peaks are located on all three detectors through a local maximum search. If a candidate peak is found, it is refined in omega and written to a file. The tricky bit is the adjustement of the local maximum search parameters in order to minimize false maxima caused by a spicky background or powder lines.

The peak search facility need a lot of parameters in order to operate. This includes angle ranges, count parameters and the maximum search parameters. Commands are provided for adjusting these parameters. The general operation of these commands follow a pattern: typing the command alone prints the current values of the parameters. In order to set new values the command name must be typed plus new values for all the parameters listed by this command. An Example:

```
ps.sttrange
```

prints the range in two theta for the peaksearch.

```
ps.sttrange startval endval step
```

sets new values for the two theta range and prints them afterwards. The following commands are provided:

ps.sttrange adjustment of the two theta range for the peak search.

ps.omrange adjustment of the omega range for the peak search.

ps.chirange adjustment of the chi range for the peak search.

ps.phirange adjustment of the phi range for the peak search.

ps.countpar adjustment of the counting parameters for the peak search.

ps.scanpar adjustment of the parameters used by ps.scanlist for scanning located peaks. See below.

ps.maxpar Adjusts the maximum finding parameters for the peak search. These parameters need some explanation:

- **window** window is the size of the quadratic area which will be searched around each point in order to determine if it is a local maximum.
- **threshold** This is a minimum intensity a candidate local maximum must have before it is accepted as a peak. The value given is multiplied with the average counst on the data frame before use. This threshold is the strongest selection parameter.

steepness A candidate peak should drop of towards the sides. This is tested for by checking if the pixels on the borders of the local maximum detection window are below maximum value - steepness.

cogwindow In order to refine the peaks position a center of gravity calculation is perfomed. For this calculation pixels within the cogwindow around the candidate peak position are considered.

cogcontour In order not to base the COG calculation on background pixels, only pixels above cogcontour * maxvalue are used for the calculation. With the spicky background at TRICS .5 seems a good value.

ps.list lists all parameters for the peak search.

ps.listpeaks lists all the peaks already found.

ps.run filename starts the peak search and stores peaks identified in file filename.

ps.continue continues a peak search which was interrupted for one reason or another.

ps.scanlist performs an omega scan for each reflection found in the current peak list.

4.4 The Search Module

For debugging purposes and in order to find reasonable values for the parameters for the peakseacrh it is helpful to know the commands understood by the lower level SICS module which performs the actual search. This is the lowmax module (for local maximum). All commands act upon the current content of the histogram memory. The following commands are understood:

lowmax stat hm calculates the average and the maximum count in the frame currently held in histogram memory hm.

lowmax search hm searches the frame held in histogram memory hm for local maxima. Local maxima are returned as sets of three numbers which are the x and y coordinates and the intensity. Each set of numbers is separated from the next one by the @ symbol.

lowmax cog hm x y calculates the center ogf gravity for the pixel at coordinates x and y in histogram memory hm. Four numbers are returned: the new x and y cooredinates, the intensity of the peak and the number of points contributing to the peak.

lowmax steepness val accesses the steepness parameter for the peak search. With a parameter val sets a new value, without print the current value.

lowmax window val accesses the window parameter for the peak search. With a parameter val sets a new value, without print the current value.

lowmax threshold val accesses the thresholds parameter for the peak search. With a parameter val sets a new value, without print the current value.

lowmax cogwindow val accesses the cogwindow parameter for the peak search. With a parameter val sets a new value, without print the current value.

lowmax cogcontour val accesses the cogcontour parameter for the peak search. With a parameter val sets a new value, without print the current value.

All parameters have the same meaning as described above, in the description of the peaksearch program.

4.5 PSD-TRICS Count and Tricsscan Command

Two special commands have been defined for TRICS with a PSD:

count `mode preset` counts with all three detectors. The parameter mode defines which counting mode is used, supported are **preset** for counting up to a preset monitor or **timer** for counting for a specified time intervall. The second prameter preset is either the preset monitor or the preset counting time, depending on the mode choosen. Both parameters are optional, if they are notc specified values from the last run will be used. count does not store any data.

tricsscan `start step np mode preset` This command creates a new data file and then performs a scan in omega, storing meausured data after each step. `start step np` define the scan range in omega. Start is the start position, step the step width to use and np is the number of steps to do. The optional parameters mode and preset have the same meaning as in the count command described above. Mode and preset how data is collected at each step in omega.

psdrefscan filename step np mode preset reads reflection HKL values from file filename and performs tricsscans for each reflection. These will be done eith step width step, the number of steps np with counting mode mode and a preset of preset. These parameters have the same meaning as described above.

Chapter 5

TRICS PSD Data Analysis

As of now two packages are provided:

- A data analysis package based on XDS (cf. Section 5.1).
- An experimental package based on a novel volume matching (cf. Section 5.2) approach.

5.1 TRICS Data Analysis with XDS

A set of programs exist for TRICS data analysis which have been derived from the XDS package designed and written by Wolfgang Kabsch. Due to the different diffraction geometry at TRICS the program had to be subdivided. Data Analysis with this system requires four steps:

1. Location of strong diffraction spots with the program **spots**.
2. Indexing of diffraction spots and refining of a UB matrix with programs of your choice.
3. Integration of the diffraction spots with the program **reflex**.
4. Optionally, reflections collected in various runs can be merged with the program **xscale**.

The main limitation of this software is that only reflections at normal lattice positions can be analysed. Magnetic or superstructure reflections will not be integrated due to the fact that XDS uses predicted reflection positions for integration and has no facilities to predict either magnetic or superstructure reflections.

5.1.1 LEGAL STUFF

The programs **spots**, **reflex**and **xscale** are no official versions of XDS. The responsability for these programs lies with PSI and not with Wolfang Kabsch. Binaries of the above mentioned programs may be distributed, but according to an agreement with Wolgang Kabsch the source code may not be redistributed. If you are interested in an official version of XDS, please contact Wolgang Kabsch directly.

5.1.2 Spots and reflex Control File

The programs **spots** and **reflex** both require a control file to be specified as a command line parameter. The format of this control file resembles a Windows .ini file and is common for both programs. The syntax is: keyword = value.

5.1.3 Running spots

The purpose of **spots** is to search for strong diffraction spots in the data and write them out in a format suitable for indexing. spots can be started by typing:

```
spots controlfile
```

at the unix command prompt. All necessary parameters live in the control file. spots recognizes the following keywords in the control file:

numfiles The number of files to process.

fileXX Replace XX by the number of the file. For instance file00 is the first file to process. The value for this keyword is the filename to process.

numdetectors The number of detectors to process. TRICS can have up to three detector banks, if the electronics group finally makes them available by an act of grace.

det1dist, det2dist, det3dist The respective distances of the detectors from the sample positions.

det1x, det2x, det3x The number of pixels each detector supports in the x-direction.

det1y, det2y, det3y The number of pixels each detector supports in the y-direction.

det1pixx, det2pixx,det3pixx The size of a detector pixel in x-direction in mm for each detector.

det1pixy, det2pixy,det3pixy The size of a detector pixel in y-direction in mm for each detector.

wavelength The neutron wavelength.

bifile Switches on the writing of reflection positions converted to bissecting positions as from a normal four circle diffractometer. The value is the name of the file to which to write the list.

nbfile Switches on the writing of reflection positions converted to normal beam positions as from a normal beamdiffractometer. The value is the name of the file to which to write the list.

xyzfile Switches on the writing of reflection positions in XYZ format. The value is the name of the file to which to write the list.

bifile, nbfile or xyzfile are choices. Chhose the one which fits best with your preferred indexing program.

5.1.4 Indexing and UB Matrix Refinement

For indexing a variety of programs are available:

- The ancient combination of index and rafin from ILL. For a description see the four circle single detector section.
- **orient** A modern indexing program extracted from Difrac. It has originally been written by R. A. Jacobsen, Ames Research laboratory. orient will not only index the reflections found and determine a UB matrix. It will also refine the UB matrix based on the reflections given to it and tries to determine the space group as well.

Running orient

In order to start orient, type **orient** at the unix prompt. A selection dialog for the file type will show up. Select 2, then give the path to the file created with the spots option bifile. You will also be asked for the neutron wavelength. The following dialogs are self explaining. When orient finishes, the new UB matrix can be found in either the LPT1 or printer.out file.

5.1.5 Running reflex

reflex is controlled through the same style control.ini file as used by spots. The options specified for **spots** have to be present in the control file for reflex as well. Additionally the following options are required:

ub1, ub2, ub3 The three rows of the UB-matrix as determined by one of the indexing programs.

axis=0 0 -1 These are the coordinates of the rotation axis in XDS's own coordinate system. Leave this at the values stated, everything else is shit if you are using TRICS.

beam=0 1 0 These are the coordinates of the incoming neutron beam in XDS's own coordinate system. Leave this at the values stated, everything else is shit if you are using TRICS.

polarisation=.5 1 0 0 Some values for handling X-ray polarisation. Leave at the values given.

spacegroup Set this to the space group selected. Expected is the number of the space group as given in the international tables.

divergence The beam divergence. See below for a comment.

mosaic The crystal mosaic. mosaic and divergence together determine the size of the box in reciprocal space which will be integrated for each reflection. reflex writes a representation of the integration box and of the reflection to its output file (PROFIT.LP). Inspect this carefully. If reflections are cut of in the reflection box or the reflection box is to large, modify these values in order to achieve a good fit. As more experience is gathered, the instrument scientist will be able to provide you with reasonable defaults for these values.

reflex is run by typing **reflex control.ini** at the unix prompt. control.ini is the name of the control file. PROFIT.LP is the main log file which shows what has been done. PROFIT.HKL is a binary file holding the reflections integrated.

5.1.6 Running xscale

xscale has not been modified since it has been received from W. Kabsch. Therefore the original documentation, reproduced below is still valid.

```
C*******************************************************************
C********************* DESCRIPTION OF FILES *************************
C*******************************************************************
C                                                                  *
C                     XSCALE.INP     (formatted sequential)         *
C                     ==========                                   *
C                                                                  *
C This file contains the input parameters you have to provide to run   *
C the XSCALE program.(free format)                                 *
C                                                                  *
C line #            DESCRIPTION OF INPUT PARAMETERS                *
C                                                                  *
C   1      Resolution shell limits (Angstrom). Only the high resolution*
C          limit of each shell is given. Up to NRES (20) resolution   *
C          shells will be accepted. The shell limits must be specified *
C          in decreasing order. The resolution shells are used to     *
C          report statistical properties of the data sets as a function*
C          of resolution.                                          *
C   2      Space group number and unit cell parameters             *
C          (Angstrom and degrees)                                  *
C   3...   Each line describes a reflection file used for scaling    *
C          and contains the following items:                       *
C          >Optional control character - or * of the following meaning *
C            -: ignore this data set (this line will be skipped)      *
C            *: put all data sets to the same scale as this one;      *
C               default is the first data set.                     *
C          >File name of data set used for scaling.                *
C            The name must not be longer than 50 characters and       *
C            intervening blanks are not allowed.                   *
C          >File type must be one of the three following keywords     *
C            DIRECT: the file is of type XDS.HKL as generated by XDS.  *
C            UNIQUE: the file is of type UNIQUE.HKL as produced by XDS.*
C            OLDHKL: the ASCII file consists of free format records    *
C                    H,K,L,INTENSITY,SIGMA                          *
C                    The standard deviation SIGMA may be omitted and   *
C                    is estimated then as SIGMA=0.1*INTENSITY         *
C                    Reflection data files of type UNIQUE or OLDHKL    *
C                    may be unsorted and the reflection indices need   *
```

```
C                    not be the asymmetric indices. This simplifies    *
C                    the scaling of data sets generated by other       *
C                    programs than XDS.                                *
C          >Resolution window for accepting reflections from this file *
C            low  resolution limit (Angstrom)                          *
C            high resolution limit (Angstrom)                          *
C          >Frame separation (mandatory for data sets of type DIRECT)  *
C            specifying the maximum number of frames between FRIEDEL-  *
C            pairs to be included in the estimated anomalous intensity *
C            difference.                                               *
C          >Number of batches (optional for data sets of type DIRECT)  *
C            This number gives the number of subdivisions of the       *
C            rotation range covering the data set. Typically, it is    *
C            the total rotation range divided by 2.5...5 degrees, but  *
C            should not exceed a value of 36. This leads to at most    *
C            9*36=324 scaling factors for a single data set. The total *
C            number of scaling factors from all data sets together     *
C            must not exceed the value given by "MAXFAC" (1000).       *
C          >SAVE=file-name (optional); default file-name is XSCALE.HKL *
C            The type of the SAVE-file produced is UNIQUE. Symmetry    *
C            related reflections from input data sets sharing the same *
C            SAVE-file are used after scaling to estimate a mean       *
C            intensity, an anomalous intensity difference, and their   *
C            standard deviations. Scaling factors for each data set    *
C            are determined from all symmetry related reflections      *
C            regardless whether they go to different SAVE-files.       *
C                                                                      *
C***********************************************************************
C                                                                      *
C                     XSCALE.LP      (formatted sequential)            *
C                     =========                                        *
C                                                                      *
C This file contains the printed messages and results from running the *
C XSCALE-program.                                                      *
C                                                                      *
C***********************************************************************
C                                                                      *
C Description of XSCALE input file format of type DIRECT as produced   *
C by XDS.                                                              *
C                                                                      *
C                       XDS.HKL  (unformatted direct access)           *
C                       =======                                        *
C                                                                      *
C The corrected reflection intensities are saved on this unformatted   *
C direct access file of record length 68 bytes for each reflection.    *
C The file is sorted with respect to the unique reflection indices.    *
C This means:                                                          *
```

```
C  For each reflection with the original indices H,K,L all symmetry    *
C  equivalent indices are generated including Friedel related ones.    *
C  Among all these indices we choose the unique reflection indices     *
C  HA,KA,LA in the following order: HA is the largest H-index, among   *
C  those with the same HA-value select those with the largest K-index  *
C  which is KA, and finally the largest L-index which is called LA.    *
C  The unique indices HA,KA,LA thus found are packed into a 32-bit     *
C  word  KEY=(LA+511)+(KA+511)*1024+(HA+511)*1048576  .                *
C  The reflections are then sorted in growing values of KEY.           *
C                                                                      *
C                    Record structure                                  *
C                                                                      *
C  16bit-WORD #                      CONTENTS                          *
C      1         HA    (The last record is indicated by HA=10000)      *
C      2         KA     HA,KA,LA are the unique reflection indices.    *
C      3         LA     Any two reflections have the same unique       *
C                       indices if and only if they are related by     *
C                       symmetry. (HA,KA,LA are integer*2)             *
C      4          H     Original reflection indices H,K,L.             *
C      5          K     H,K,L are integer*2.                           *
C      6          L                                                    *
C      7          S     Identifying number of symmetry operator used   *
C                       to go from original to unique indices.         *
C                       (integer*2). A negative sign indicates that    *
C                       a mirror operation has been applied. This      *
C                       information may be useful if a special         *
C                       treatment for anomalous differences is         *
C                       required which goes beyond the method of       *
C                       the XDS-program.                               *
C      8        IPEAK   Percentage of observed reflection intensity.   *
C                       A value less than 100 indicates either a       *
C                       reflection overlap or bad spots in the profile*
C      9        ICORR   Percentage of correlation (integer*2) between  *
C                       observed and expected reflection profile.      *
C    10,11      FFADD   LP-corrected intensity of this reflection      *
C                       obtained by straight summation of counts       *
C                       within spot region ( background subtracted).   *
C                       The intensity is also corrected for radiation  *
C                       damage and absorption.      (real*4)           *
C    12,13      SDADD   Standard deviation of FFADD.(real*4)           *
C    14,15      RLP     Reciprocal LP-correction factor (real*4)       *
C     16       ABSCAY   Combined absorption and decay correction       *
C                       factor*1000 (integer*2).                       *
C                       In case you want to remove this calculated     *
C                       correction, divide intensities and standard    *
C                       deviations by ABSCAY/1000.0  .                 *
C     17        IALFA   IALFA and IBETA (both integer*2) are polar-    *
```

```
C       18       IBETA   coordinates of the spindle axis in units of a *
C                        hundreth of a degree. The lab coordinates of  *
C                        the spindle axis are:                         *
C                        Ux=sin(BETA)*cos(ALPHA)                       *
C                        Uy=sin(BETA)*sin(ALPHA)                       *
C                        Uz=cos(BETA)                                  *
C                        where ALPHA=IALFA/5729.578,                   *
C                              BETA =IBETA/5729.578.                   *
C       19       IFRM    Frame number at diffraction of this reflection*
C                        (integer*2)                                   *
C       20       PHI     Calculated spindle position for this          *
C                        reflection at diffraction in units of a       *
C                        hundreth of a degree. (integer*2)             *
C       21        IX,    Calculated detector x- and y-coordinates for  *
C       22        IY     this reflection at diffraction in units of a  *
C                        tenth of a pixel times 512.0/NX and 512.0/NY, *
C                        respectively. NX, NY are the numbers of pixels*
C                        along the detector X- and Y-axis.             *
C                        IX,IY are integer*2.                          *
C     23-28       S0     Laboratory coordinates of direct beam wave-   *
C                        vector ( rec. Angstroem). S0 points from the  *
C                        x-ray source towards the crystal.             *
C     29-34       S1     Laboratory coordinates of scattered beam wave-*
C                        vector. Length is 1.0/lambda (rec. Angstroem) *
C                        S0 and S1 are real*4 arrays of length 3. S1   *
C                        points from the crystal towards the detector. *
C                        At diffraction, laboratory coordinates of the *
C                        reflection H,K,L are: S1(.)-S0(.)             *
C                                                                      *
C***********************************************************************
C                                                                      *
C Description of XSCALE input file format of type UNIQUE as produced   *
C by XDS.                                                              *
C                                                                      *
C                       UNIQUE.HKL (formatted sequential)              *
C                       ==========                                     *
C                                                                      *
C              DESCRIPTION OF SHORT OUTPUT FILE                        *
C                                                                      *
C Symmetry related reflections are averaged and written with           *
C FORMAT(3I5,4E12.4). Each record consists of                          *
C                                                                      *
C   HA,KA,LA,INTENSITY,STANDARD DEVIATION OF INTENSITY,                *
C   ANOMALOUS INTENSITY DIFFERENCE,STANDARD DEVIATION OF DIFFERENCE    *
C                                                                      *
C where HA,KA,LA are the unique reflection indices. The file is sorted *
C with respect to these unique reflection indices. The last record     *
```

```
C is indicated by HA=10000.                                           *
C Unobserved ANOMALOUS INTENSITY DIFFERENCE and its STANDARD DEVIATION *
C are both set to zero.                                                *
C                                                                     *
C**********************************************************************
```

xscale can be started by typing **xscale** at the unix prompt. Please note that xscale expects an input file named XSCALE.INP in the current directory.

5.2 TRICS Data Analysis with Autocloud

Autcloud is an experimental software which integrates reflections from a 3D data volume without any prior information. Reflections are located through a template matching approach in 3D. Template Matching is a common digital signal processing method for enhancing a signal against a background. This program may still contain very major algorithmical and programming errors as it could not yet be tested against real data. Data Analysis with autocloud requires two steps:

- Reflection integration (cf. Section 5.2.1) with autocloud.
- Indexing of the reflections found. This can be done with the program **orient**. Just select data type 3 and answer the questions asked.

5.2.1 Autocloud

With the advent of position sensitive detectors in X-ray and neutron diffraction the problem arises how integrated reflection intensities may be extratcted from the collected volumes of data. Typically a series of frames is measured while rotating the crystal under investigation in omega. Autocloud implements a novel approach for the extraction of reflection intensities from such data. Other currently used integration packages use a UB-matrix to predict the position of a reflection on the detector and then integrate the intensity in a box around the predicted position. In contrast autocloud tries to determine reflection positions and intensities directly from the data. In order to do so a template matching algorithm is used. One advantage of this approach is that crystals with magnetic or incommensurate structures can be easily analysed. Typically packages for intensity integration do not have facilities for predicting such reflections. The other advantage is ease of use. Data analysis with autocloud requires only two steps: Integration followed by indexing.

Running Autocloud

The syntax is:

```
autocloud options datafile
```

The following options are known:

-a val Selects the algorithm to use. The following algorithms are currently supported:

max perform only a local maximum search

template Perform template matching. This is the default.

cross Perform template matching using the cross correlation function.

-b AAxBBxCC For the evaluation of the initial template a preliminary box size is needed. This can be specified through this option. Three values separated by the character 'x' are required, one for each dimension in the order x, y, z.

-d val After the correlation of the data volume with the template another maximum search is started in order to locate the reflections. In order to suppress spurious peaks, a minimum steepness of the candidate peak can be set with the -d option.

-e val Some systems store frames a single files. With the -e option the end file number of the frame files can be set.

-m val When the maximum search only option is set a, a threshold is required for suppressing spurious peaks. This threshold can be set with the -m option.

-o file Redirects output to the file name specified. By default all output is written to stdout.

-s val Some systems store frames a single files. With the -s option the start file number of the frame files can be set.

-t type This option sets the type of the data file. Currently understood are:

sxd For NeXus data from SXD at ISIS.

trics For NeXus data files from TRICS, SINQ

debug An internal format used during software testing.

-v val Increases the verbosity of the output.

The Autocloud Algorithm

The autocloud algorithm has the following steps:

1. Location of strong peaks for template evaluation.
2. Background Subtraction.
3. Evaluation of a template for volume matching.
4. Correlation of the template with the data volume.
5. Location of maxima in the correlated data.
6. Integration of the reflections found.

Location of Strong Peaks for Template Evaluation This is basically a local maximum detection scheme. A local maxima must be the strongest intensity within a 7 by 7 by 7 volume. All maxima smaller then 10% of the largest maximum found are discarded.

Background Subtraction Background subtraction is done with essentially the same algorithm XDS uses. For each x, y coordinate in the frame values are summed along the third dimension. Points belonging to a local maimum are excluded. The background for this x,y coordinate is then the average of the values summed. The data volume is then corrected for the background with these values. This works well as long as the assumption holds that the background varies mostly across the detector and not much with the third dimension.

Template Evaluation The template to be used for template matching later on is calculated by summing all local maxima first. Then the limits of the reflection are calculated for each scanline using the Lehmann-Larsen algorithm. The reflection thus found is scaled to a value of 1 and used as the template.

Template Matching For the actual correlation of the template with the data two variantes can be used: Normal simple correlation or cross correlation.

Peak Detection This is again a local maximum detection within a 7 by 7 by 7 box. Another criterium for the supression of wrong identifications is a minimum steepness. This means that the candidate local maximum must at least be higher by a certain amount (the steepness) then the points at the border of its 7 by 7 by 7 box.

Peak Integration A scale factor is calculated for each candidate reflection between the data and the template. The intensity is derived from this scale factor and the standard deviation is calculated as the squared difference between the scaled template and the data. This scheme is the same as learnt profile fitting as described by Ford for the 1- and 2d cases.