Basic SPEC commands for a diffraction experiment

Donald A. Walko
Beamline 7ID, Advanced Photon Source, Argonne National Laboratory, Argonne, IL 60439
d-walko@anl.gov

January 18, 2016

Abstract

This document lists some common commands (plus a few other hints) for casual users of the diffractometer control program SPEC. This is meant to briefly list common ways to use some of the most useful commands, not to be an exhaustive list nor a complete description of a command's syntax. Refer to the SPEC manual or help files for additional information [1], such as the structure of data files, setting motor positions or software limits, plotting and printing scans, and geometry-specific macros. The beamline staff may also be able to help with these issues, and probably should be consulted before users make significant changes such as resetting a motor's position or soft limits.

Starting SPEC

From your local beamline contact, determine how to log in to the appropriate computer to run SPEC, open any xterminal that may be necessary, and start the SPEC program. Typically, the program name is the geometry name, such as fourc, kappa, or psic.

The basic SPEC commands

SPEC is a command-line based program. Therefore it is important to know the correct commands; fortunately, some commands will list the type of parameters needed if you enter the wrong type (or number) of parameters. Actually, most "commands" (and many variables) are actually macros and could be redefined, which is not something one should normally do. But the writing and implementation of new macros is quite straightforward; users often write shortcut macros which are combinations of a number of commands. It is also important to note that SPEC is case-sensitive; most commands and variable names are lower-case, while certain special variables and macros are upper-case.

"Information" commands

`pa`
: I still don't know if this stands for "parameters" or "print all," but it is a very useful way to list parameters such as the orientation matrix, lattice parameters, operating mode, wavelength, etc.

`wh`
: where; lists positions of the diffractometer motors (in user units; see below), reciprocal lattice coordinates, and some relevant angles.

`wa`
: where all; lists positions of all the SPEC motors (in user and dial units; see below).

`wm` *motor_name(s)*
: where motor; lists where the motor is (user and dial units) and its software limits.

`lm` [*motor_name(s)*]
: limits; same as `wm`, except `lm` will list all motors if none are specified.

`p` *some expression*
print; can be used to print a variable's value:
`p F_CHI`
or as a calculator:
`p 5*sin(PI/4)`

`help`
calls the SPEC help utility, and lists a sizable number of topics which are described to varying degrees of detail.

The following commands may be more helpful for advanced users:

`prdef` *macro_name*
print definition; prints the definition of a macro.

`lsdef`
list defined; lists the names and sizes of all currently defined macros.

`lscmd`
list command; lists built-in commands and functions.

`whats` *something*
identifies an object, as keyword, function, macro, or variable

`syms` [*name*]
list of known symbols: all or only those which match *name* (you may use the wildcard characters * or ?)

Miscellaneous commands

`ct` [*time*]
counts and lists results for all scalers. The counting is for *time* sec or for the default time if *value* is omitted (such as 1 sec). But if *value* is negative, then counting continues until the monitor reaches *value* counts. Examples of use:
`ct`
`ct 10`
`ct -200000`
The monitor is typically a scaler keeping track of the incident beam.

`sleep(`*time*`)`
pauses for *time* sec. Useful, e.g., if you want walk over to a motor to watch it move. Example:
`sleep(10); umvr th 2`

`startup`
initializes a variety of parameters. Includes the following macros, which could also be called directly:
`newsample`: title for scan headers
`newfile`: begins a new data file
`setscans`: sets a few scan options
`setplot` [*value*]: options for plotting. Each option has a number, so you can add them up and include *value* as a shortcut, once you know which options you want.
`startgeo`: calls geometry-dependent set-up macros

`do` *command_file*
reads commands from a text file. It's usually good form for the file name to have a .mac extension

`qdo` *command_file*
quiet do; same as do, but doesn't print the commands to the screen.

`save` [*file_name*]
saves numerous parameters to a text file. Allows you to recover things like the orientation matrix if you want to make temporary changes. Use the `do` or `qdo` command to read the file back into SPEC.

`comment "`*whatever you want to say*`"`
writes a time-stamped comment to the data file.

`u` [*unix_command*]
unix shell: executes *unix_command* (if included) or goes to the unix command line. To escape from the unix command line, type exit. Some unix commands are directly accessible without typing the `u`:
`cd`: change directory
`ls`: list directory contents
`l`: list directory contents (the unix `ls -l` command)
`pwd`: present working directory

`quit`
quit out of SPEC. "exit" does not do this.

Be careful to understand the effects of the following commands before executing them. They may, for example, be safe on a simple rotary stage, but not on a more complex diffractometer:

`set` *motor_name position*
sets a motor's position (in user units).

`set_lm` *motor_name low high*
sets a motor's limits (in user units).

`set_dial` *motor_name position*
sets a motor's position (in *dial* units).

`config`
calls the hardware configuration editor. Allows configuration of motors, scalers, and other devices. These are mostly advanced options which should not be altered casually.

Simple motor motions

`umv` *motor_name position*
move *motor_name* to absolute *position* (in user units). Examples:
`umv th 20`
`umv th CEN`
See below under `dscan` for information on the variable `CEN`.

`umvr` *motor_name rel_position*
move *motor_name* by the relative amount *rel_position* from its current position.

`ubr` *H K L*
move motors to the reciprocal lattice point (Bragg point) defined by the Miller indices *H K L*

`ca` *H K L*
calculate the motor position for the reciprocal lattice point *H K L*. It's a good habit to do `ca` before `ubr`, to avoid unexpected motor motions.

`tw` *motor_name delta*
tweak; interactive subroutine to move *motor_name* by *delta*. Once in the subroutine, each time you hit `Enter` the motor moves by *delta*. You can change direction with p/n or +/-, and also change *delta* by entering a new value. Escape by hitting `CTRL-C`, or some other letter or symbol, followed by `Enter`.

The 'u' in `umv`, `umvr`, and `ubr` stands for 'update;' the motor positions are regularly updated on the screen while they move. This is not required, but is preferable to `mv`, `mvr`, and `br` since it is not obvious from the latter commands when the motion is completed: SPEC may appear to have hung, since the prompt appears but won't respond to new commands until the motors are done moving.

Basic scans

A main use of SPEC is to scan motors and collect data. If you are ever unsure of the order of parameters for a certain scan, just type the scan name and SPEC will list the parameters in order.

For all these scans, one enters the number of intervals, which is one more than the number of points. Thus, the step size is (ending point) - (starting point)/intervals. The unit of time is seconds per point if positive, or monitor counts per point if negative.

`loopscan` *npts [count_time [sleep_time]]*
time-lapse scan: sit at current conditions and count for *npts* points without moving motors

`timescan` *[count_time [sleep_time]]*
indefinite time-lapse scan, i.e., a loopscan with *npts*=0

Motor scans

`ascan` *motor_name start end intervals time*
absolute scan: *motor_name* starts at *start* and ends at *end* (in user units). At the end of the scan, *motor_name* stays at *end*. Example:

```
ascan th 5 7 30 1
```

`dscan` *motor_name rel_start rel_end intervals time*
relative (differential) scan: *motor_name* starts at *start + current_position* and ends at *end + current_position*. At the end of the scan, *motor_name* returns to its previous position. This is the same as a `lup` (line up) scan. Example:

```
dscan th -1 1 30 1; umv th CEN
```

The variable `CEN` (all caps) is calculated after each scan, and is the absolute position of the peak's center (as given by the FWHM, not the highest position or the center-of-mass). As long as there is a peak in the scan, this is a good way to line up to it. Note that if you typed `umvr th CEN` or `umv phi CEN` you could get into big trouble! It may also give weird results if the FWHM couldn't be calculated from the scan (e.g., because of a background value higher than 50% of the peak value).

`a2scan` *motor_name1 start1 end1 motor_name2 start2 end2 intervals time*
absolute scan of two motors: *motor_name1* starts at *start1* and ends at *end1*, while *motor_name2* starts at *start2* and ends at *end2*. `a3scan` and `a4scan` operate similarly, `d2scan`, `d3scan`, and `d4scan` are multimotor relative scans.

`mesh` *motor_name1 start1 end1 intervals1 motor_name2 start2 end2 intervals2 time*
motor mesh scan. A scan of *motor_name1* is done for each point of *motor_name2*, all of which is stored as one SPEC scan. Example:

```
mesh th 5 7 50 tth 10 14 30 1
```

In this example, the full scan contains 51*31=1581 points.

`resume` [*n*]
resumes an aborted scan. If a positive integer n is included, then n points are skipped. If a negative integer is included, then the last n points are repeated.

Reciprocal space scans

`hscan` *h_start h_end intervals time*
linear scan in reciprocal space along the H axis. The values of K and L during this scan are based on the previous position in reciprocal space, so you may need to use the `ubr` command to first move to the appropriate point. Example:

```
ubr 1 1 1; hscan .9 1.1 20 -20000
```

`kscan` *k_start k_end intervals time*
same as `hscan` but along the K axis.

`lscan` *l_start l_end intervals time*
same as `hscan` but along the L axis.

`hklscan` *h_start h_end k_start k_end l_start l_end intervals time*
: linear scan in reciprocal space along a general direction. For example, if you wanted to scan in some direction along H and K thru the (111) Bragg peak:

```
hklscan .9 1.1 1.2 0.8 1 1 20 1
```

`hklmesh` *Q1 start1 end1 intervals1 Q2 start2 end2 intervals2 time*
: reciprocal space mesh scan. *Q1* and *Q2* are literally `H`, `K`, or `L`. Thus this type of mesh scan is limited to be along the principal axes of reciprocal space. The value of the third reciprocal space coordinate during this scan is based on the previous position in reciprocal space, so you may need to move there first. For example, if you wanted to scan in the *H-L* plane thru the (111) Bragg peak:

```
ubr .8 1 .9; hklmesh H .8 1.2 20 L .9 1.1 20 1
```

More complicated scans in reciprocal space (e.g., radially, or along circles) are possible. See the SPEC manual for details [1].

The orientation matrix and other issues

An important function of SPEC is as a calculator, for the transformation between diffractometer angles and reciprocal lattice coordinates. The most important aspect of this is the orientation matrix, i.e., the angular position of the crystal lattice. The basic commands for setting up an orientation matrix are given here, but see the SPEC manual [1] or your beamline contact for additional information.

The orientation matrix is set by finding two nonparallel Bragg reflections. Since the orientation matrix is never perfect, the primary reflection will exactly agree with the orientation matrix (to a scale factor), but the secondary reflection will not exactly agree. Use the commands `or0` and `or1` to set the primary and secondary reflections, respectively, if the diffractometer is at the reflection, or use `setor0` and `setor1` if you know the appropriate angles but the diffractometer is at some other location. The lattice parameters are set with the `setlat` command, and the x-ray wavelength is given by the value of the variable `LAMBDA`.

There are three other aspects of determining a unique set of angles for a given *(HKL)* Bragg point. These vary depending on the particular SPEC geometry, so are only briefly mentioned here:

modes there are generally more diffractometer angles than there are dimensions in reciprocal space (i.e., 3); additional constraint(s) is/are set via the selection of a mode (e.g., constraining the angle of incidence, or constraining the position of a particular motor).

sectors this is how SPEC selects between sets of angles that are geometrically equivalent. For example, the angle pair (*twotheta, omega*) is equivalent to (*-twotheta, 180-omega*), but usually one prefers positive values of twotheta.

cut points this determines how to break the 360° degeneracy of the circles to avoid wraparound situations. For example, suppose the phi cut point is -180°, phi is currently at -175°, and you want to move to -185°. Then the phi motor will *not* make a -10° move but will in fact make a +350° move to +175°.

User units vs. dial units

SPEC maintains an important distinction between the "user units" and "dial units" of a motor. The dial units are the actual values which are read on a motor's physical dial (when such a dial exists). SPEC keeps track of a motor's position using the dial units (for example, in calculating whether a given motion would violate a software limit). Dial units are algebraically converted to user units, which are the units SPEC uses in calculations such as reciprocal lattice coordinates. As an example, if the detector is positioned to intercept the direct beam but the tick mark on the dial points to 90, then tth = 0 in user units but 90 in dial units. If you become concerned that a motor has somehow lost its position, you can go into the hutch and read the number off the dial (if it exists) to compare with the dial units listed in SPEC.

Counters

Data are recorded by SPEC via counters (also called scalers). Little will be said here, since so much depends on the particular hardware. Two special scalers are set in the config file, the timebase (units of seconds) and the monitor. A "`ct` n" command will count to the timebase if n is positive or to the monitor

if n is negative. Thus, monitor is typically a counter for the incident beam to normalize the signal to the incident flux.

Unfortunately, there is another convention wherein one scaler is called DET and another is called MON; these can be set by giving values to these variables (e.g., `DET=2`) or by the command `counters`. MON is *not* necessarily the same as the monitor defined above, but I think these counters occupy special columns in the SPEC data file. Finally, the relatively new command `plotselect` lets you choose which counter(s) are plotted on the screen during a scan.

References

[1] The Certified Scientific Software website has an online manual and help pages for SPEC at `http://www.certif.com`