Computing Developments at Hasylab

Th. Kracht
Hamburger Synchrotronstrahlungslabor HASYLAB at DESY
Notkestr. 85, 22607 Hamburg, Germany

October 4, 2004

Abstract

Spectra is the program for instrument control and data acquisition at Hasylab. This note describes the improvements since the previous Nobugs workshop.

Much work went into the upgrade of the graphical user interface, which is written in PerlTk. This was a natural choice, since Perl serves as the Spectra scripting language. All of the standard online applications, those which are of use at all beam lines, are supported by widgets now. In the meantime the development of GUIs that are customized for special experimental technique has started.

Another important issue is the upgrade of the Spectra server capabilities. Currently multiple clients can be served in an interactive session and Spectra is prepared to run as a detached process that listens continously to service requests from various sources.

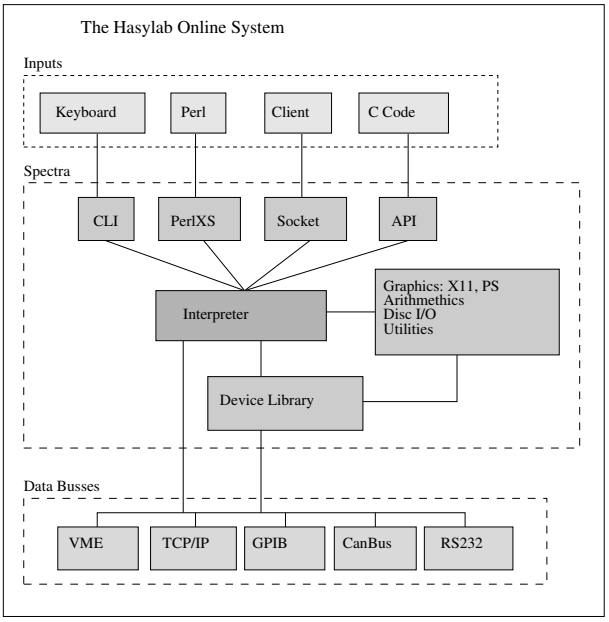

Figure 1: The Hasylab Online System

1 Introduction

At Hasylab a standard online system has been developed. It is implemented at 21 end stations and some test setups. In the meantime the Hasylab system has been ported to other locations (Uni Dortmund, Uni Aachen). It supports measurements in various fields, including material science, hard condensed matter, interfaces and surfaces, environmental science and soft condensed matter.

The next section of this note gives a short overview of the Hasylab online system. It is followed by a discussion of the Spectra - Perl interface. The last section describes the improvements of the GUIs and the upgrade of the server mode.

2 The Hasylab Online System

Linux PCs serve as our experiment computers. They are coupled to VME, GPIB and CanBus via PCI adaptors, avoiding any network overhead during data transfers to these busses. Ethernet connections

via TCP/IP sockets become increasingly important. They are mostly used for the communication with detector hardware or between computers. Serial lines are still needed. They are provided by terminal servers.

Most of our end stations are equipped with an additional Windows PC, offering our guests the opportunity to run pre-analysis applications that have been developed for this platform.

The experiment PCs are backuped each day by the TSM system which is provided by the Desy-central IT group. It allows a disk restore after hardware failures and also single file restores by users. The Linux PCs have a link to the dCache which is the front end to the Desy mass storage. The dCache has been developed for the high energy physics experiments and offers our users virtually unlimited storage space. In general data are transfered across the network to the home institutes of our guests. But there is also a public DVD/CD recorder that is accessible from all experiment computers.

The program Spectra is used for experiment control, data acquisition and online monitoring [1], [2]. It consists of an interpreter, an interface to the device library, graphics routines (X, Postscript), disk I/O functions, a comprehensive list of command verbs and an interface to Perl/PerlTk [3]. In addition there is an application programmers interface. The Spectra components are displayed in figure 1.

The device library provides the interface to the electronic equipment. Motors, timers, counters, ADCs, MCAs, etc. can be accessed in a generic way, hiding the module specific features from the user. Stepping motors can be operated individually or as composite devices (monochromators, diffractometer angles, slit systems). The device library is constantly updated whenever new hardware has to be installed. The aim is to standardize the electronic system. It is achieved by generating a list of supported devices, which are recommended for hardware upgrades or for new experiments.

A web interface to the online resources allows beam line scientists to control the running experiments.

Spectra is very flexible. The identical code runs at all the mentioned end stations. Those features that are beam line specific are configured in a startup file and inside scan macros.

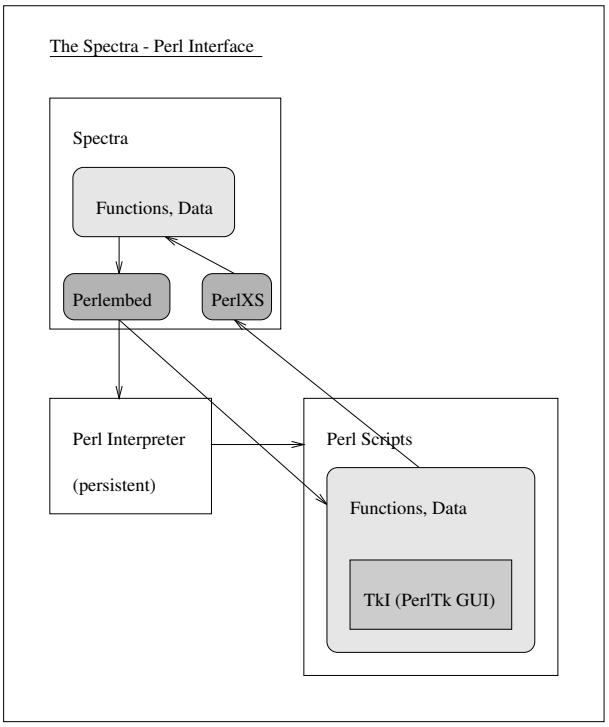

Figure 2: The Spectra - Perl Interface

3 The Spectra - Perl Interface

Using Perl as the scripting language means that two interfaces had to be implemented: Perl → Spectra and Spectra → Perl. The first direction (Perl → Spectra) has been programmed in the PerlXS framework [4]. All online functions had to be mapped into Perl and also the Spectra internal data structures had to be made accessible from Perl. Details can be found in [5]. The purpose of the Perl → Spectra interface is obvious. This might not be the case for the other direction: Spectra → Perl. It is needed by the graphical user interface which has been written in PerlTk. Consider that a stepping motor is moved by a widget. The widget passes an appropriate move command via PerlXS to Spectra, which executes the move

synchronously. That means it returns control to the widget after the move completed. This procedure is, as it has been explained so far, not acceptable, since the user wants to observe how the move proceeds. In other words: the widgets containing the motor positions have to be updated regularly and also the widgets that display status information like "Moving", "Stopped", "Doing backlash", etc. The update procedure has been implemented in the following way: Spectra (C code) starts the motor movements and while the motors are in motion it calls the PerlTk update procedure (Perl code). This is one place where the Spectra → Perl interface comes into action. Figure 3 displays the relations.

Perl scripts that make use of Spectra features are always executed within a Spectra session. They are invoked from the command line or from other Perl scripts. Otherwise the execution of a single script would correspond to a complete Spectra session, including the overhead that is involved by the startup and exit procedures: loading the device list, checking motor positions, maintaining log files, etc. The other important reason for running scripts as part of the Spectra session is performance: the Perl interpreter is loaded only once. It happens before the first Perl script is executed. The following scripts are executed by the persistent interpreter.

4 The Graphical User Interfaces

Perl was chosen to be the scripting language for Spectra. Consequently, the graphical user interface is written in PerlTk. It is named TkI - the Tk interface.

The standard applications, like stepping motor alignment, scanning procedures, crystal orientation, MCA measurements, etc. are well supported by widgets.

The main TkI widget is displayed in figure 3. It shows the list of motors and a log window for informational messages. At the top of the main widget a menu bar can be found which allows the user to start scans, work with the diffractometer, set different options or invoke miscellaneous applications.

A double-click on one of the motor names starts the motor alignment widget, see figure 6. It displays a signal while the motor is moved. The motor can be moved to an absolute position, it can be moved step by step or it can be moved with a slider bar. The slew rate can be reduced and the window limits can be narrowed in order to do efficient peak searches. Both parameter changes have only a temporary effect. The original values are restored when the widget exits.

Another important widget is the general scan menu, see figure 6. It can be used to scan the energy, motors, slits or even a dummy motor. This doesn't actually move any hardware, instead it can be thought of as a time scan.

There is a special option for energy scans: EXAFS. If selected, the scan points are defined by a list of regions, all having different step width and sample time.

The measurement that is carried out during a scan is specified with an extra widget. It is displayed in figure 6. On the left hand side the user may specify the timer and counters and whether the counter contents are displayed during the scan or just written to the output file.

In the mid part of the widget the user may chose MCAs and define regions of interest (SCA) within the MCAs. The contents of the regions of interest are displayed like counters. If MCAs are selected, the output files are written to an extra directory. This feature has been implemented, because MCA scans produce a lot of output files.

On the right hand side of the select scan device widget virtual counters are introduced. A virtual counter is a piece of Perl code that replies to two methods: reset and read. It is left to the user what is done, when these methods are invoked. Upon return from the read method Spectra expects that some number is returned. It is treated like a counter reading. The virtual counters have been implemented for those devices that are used rarely or for tests of new hardware.

In addition to these standard widgets other applications have been developed that are customized for special techniques: standing waves, diffraction, 3D scans, etc. It is expected that more experimental methods will be supported by dedicated widgets.

5 The Server Mode

Since long Spectra was able to serve external clients. This operation mode was intended to be used by our guest groups who operate own equipment. They are mostly interested in using the experiment computer as a front end to the monochromator or slit systems. So far Spectra served these clients synchronously, allowing for one input source only. If a client was connected to an online session, control was given to it exclusively. Command line control resumed after the client disconnected.

Spectra is now able to serve multiple clients while the session remains active for the user. The asynchronous server mode is entered by creating a thread that waits for client connections. A client can connect at any time to an active Spectra session. After a client connected, the thread re-enters the accept state.

Although there may be several input sources, i.e. clients or the active user, there is no parallelism. Command execution is serialized. A new command can only be executed, after the preceding command finished. For fast interactions, like counter readings, this is not a problem. But motor movements may last for a considerable amount of time, during which the other input streams are blocked. This problem can be overcome by splitting a movement into several pieces which are quickly executed: setup-move, start-move and repeated check-moving. In between these pieces other clients can be served. It is entirely up to the user to configure procedures that are well behaved. The user is also responsible for avoiding race conditions: if some hardware receives contradicting commands from various clients, the last client wins.

Spectra serves clients at well defined points:

- During read-keystroke wait states.
- During timer wait states.
- During scans, before the measurements are performed.
- During MCA wait states.
- The TkI has a timer and a time-out service function installed. On timer expiration this function checks for client commands and executes them.

The projects to come (VUV-FEL, Petra-3) require Spectra to run detached from an interactive user, the daemon mode. The daemon will be installed on an extra beam line PC which gives the experiments at these new facilities common access to certain beam line components like: monochromators, mirrors or absorbers. One experiment will be in charge, i.e. it has the right to change the setup. The others will be given read access only. The daemon mode is close to the asynchronous server mode with the addition that the daemon starts during the boot procedure of the PC. An important issue for the daemon is the verification of the stepping motor positions (absolute position encoders are installed only for a small fraction of the stepping motors). In the past users were prompted what to do when Spectra detected differences between the stepper controller registers and the device list (The device list is a disc file, which is created when Spectra exists). These differences can be due to trivial reasons like re-powering the VME crate, or they may indicate hardware failures or corrupted log files. However, a Spectra daemon has to find the correct motor positions on its own. The decision is made upon the exit state of the preceding Spectra session, the actual values in the controller registers, the device list and other log files. This procedure has just been implemented also for interactive sessions.

6 Conclusion

In summary the concept of choosing Linux PCs as the central part of the Hasylab beam line control system has proven to be successful. The Perl and PerlTk interface to the online program Spectra has been considerably extended, making graphical user interfaces now available not only for the standard procedures, but also for customized, beam line specific applications. The server capabilities of Spectra have been improved. If required, the program can now run as a detached process, answering requests from multiple network clients.

References

[1] www-hasylab.desy.de/services/computing/spectra/spectra.html

[2] www-hasylab.desy.de/services/computing/online/online.html

[3] www.perl.com

[4] www.perldoc.com

[5] www-hasylab.desy.de/services/computing/perl_spectra/perl_spectra.html

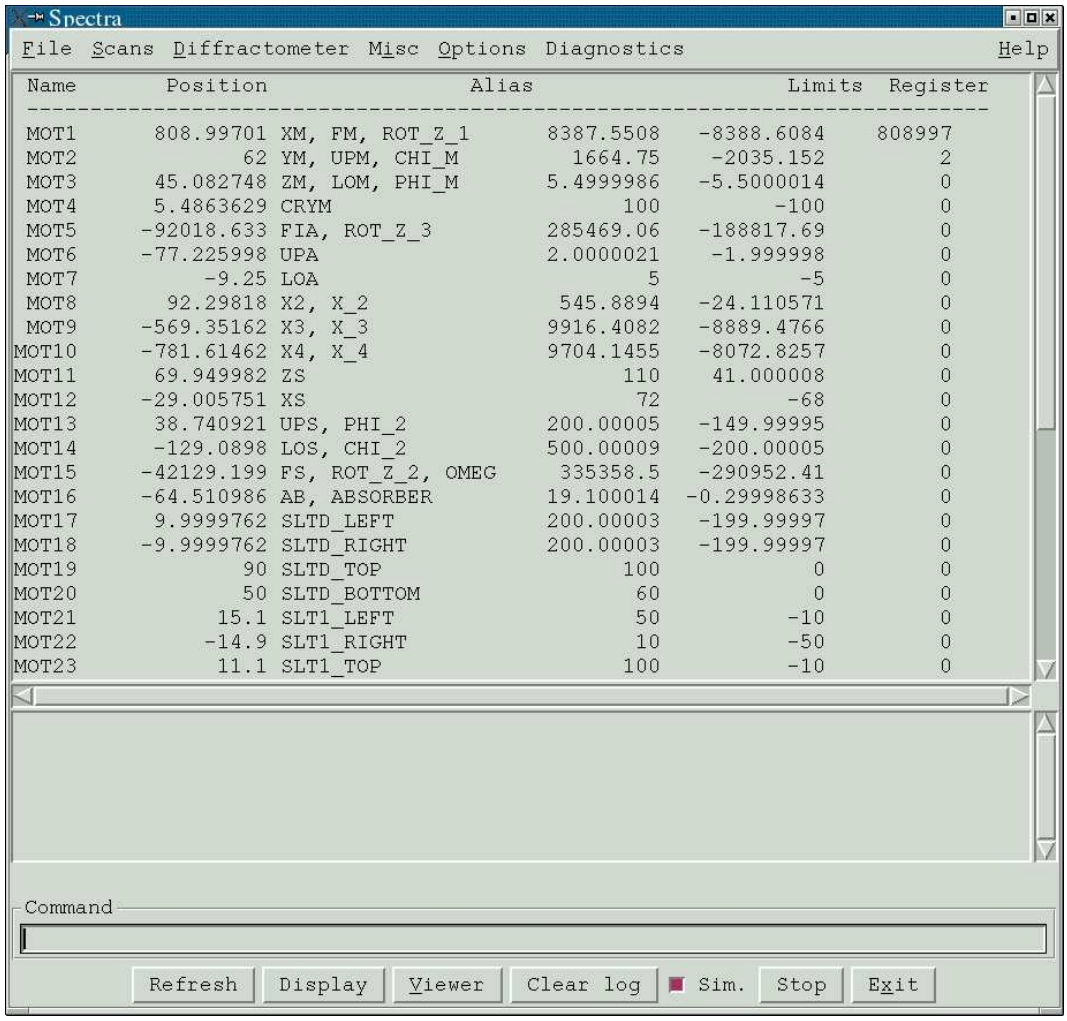

Figure 3: The Main TkI Widget

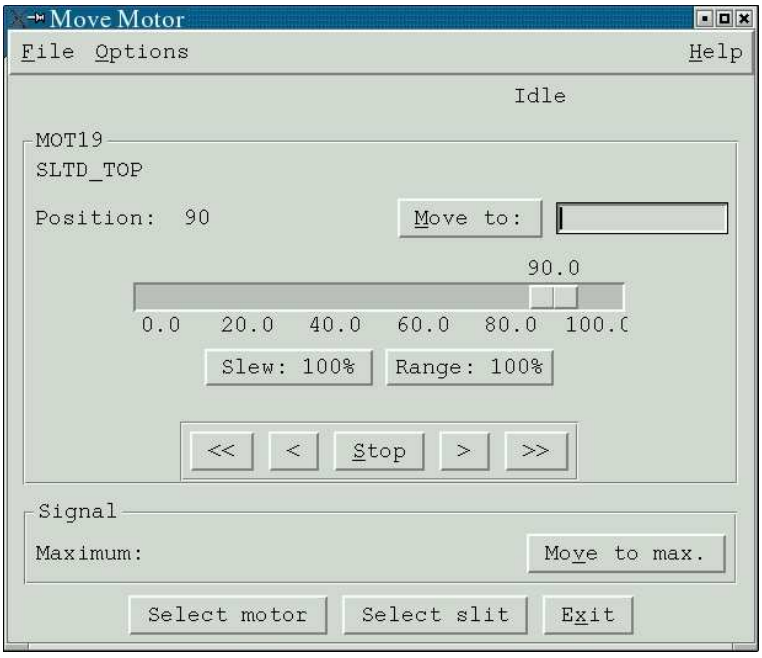

Figure 4: The Motor Alignment Widget

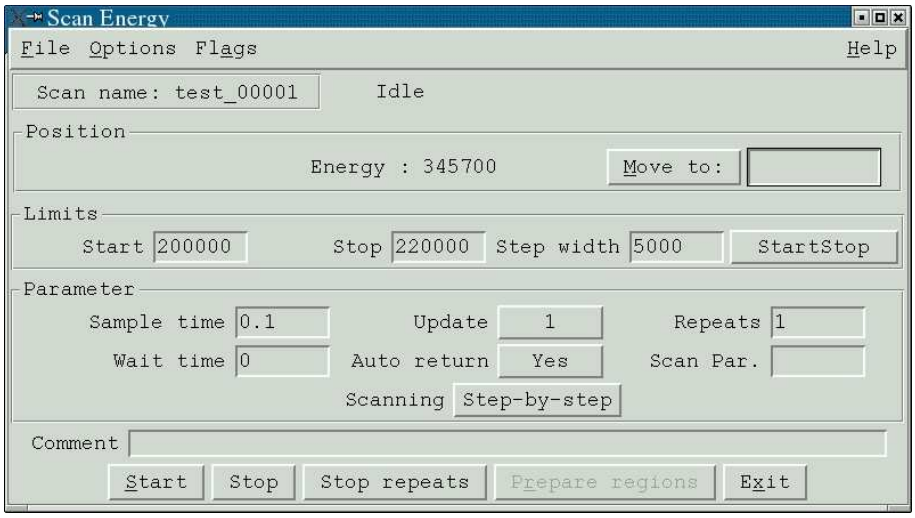

Figure 5: The Scan Widget

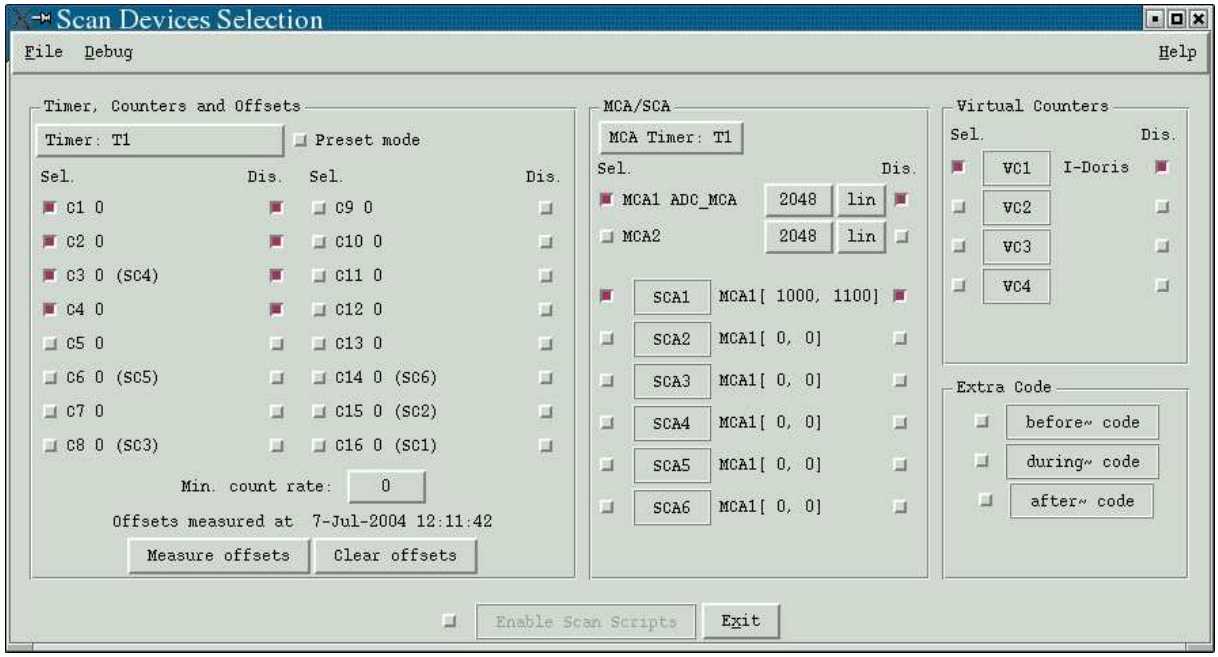

Figure 6: The Select Scan Devices Widget