Sonix+ - the new instrument control at the IBR-2 reactor

A.S.Kirilov, N.V.Astakhova, S.M.Murashkevich, T.B.Petukhova, V.E.Yudin

FLNP of JINR
141980, Dubna, Moscow Reg., Russia
kirilov@nf.jinr.ru

VME-based systems with the Os-9 operating system and the software complex Sonix are still the main "workhorses" used to control the instruments at the IBR-2 reactor [1]. Permanent instrument modification as well as new software technologies achievements encouraged us to revise the conceptual decisions from the end of last century [2]. When designing the new software Sonix+ we had in mind long-term experience of the Sonix exploitation and recent trends [3, 4]. The VME/Os-9 platform was replaced with the PC/Windows. The Python is used as a script language and as a configuration description language too.

Replacement of hardware and software platform

Using PC with Windows operating system for instrument control reduces the overall costs of the system. Users are more experienced with the Windows environment. Many useful software products are available.
Existing VME hardware controllers can be connected to PC by means of VME-PCI adapters [2].

Main software conceptual improvements

The Sonix+ software inherited some basic solutions from the older Sonix complex. In particular, those are the modular organization using special database for device control and reflection of the current complex state, using script programming for batch instrument actions.
At the same time some important features were revised to make complex unified, flexible and comfortable for the user.
All the changes can be grouped as follows:

- structural changes
- complex configuration enhancements
- expansion of script language
- GUI unification
- introducing a technical data format etc.

Where are new possibilities also in spectra visualization and development.

Structural changes

The main object of the user attention in the Sonix+ complex became "device" not "module" or "server" as it was in the Sonix earlier. Device can represent a real hardware element of the instrument (stepper motor, detector etc.) or a virtual element. For instance, the "expo" device is designed for exposure control and spectra saving, including all detectors. Creating virtual devices and representing several devices as a single device became possible due to device interface formalization and due to introduction of unified protocol for communication between modules.

Configuration of the complex

Configuration of the complex now is done by a programmer to add new modules to the complex and by user to choose device set for the experiment and tune their parameters. The complex configuration is stored in a special file written in Python. This file contains information concerning both configuration itself and template device descriptors. The first is created and modified by a man with the help of

"Configuration Editor". The latter are created by modules. Each module implementing particular device is responsible to add appropriate device descriptor structures to the configuration file.

The database

As it was in Sonix all communications with the devices are performed via local database/3/. All the devices are represented in the database in a similar way. Each device has input and output post box and descriptor. The latter consists of "status" variable for dynamic parameters, "params" variable for static parameters and variables describing the structure of status and params. So the database at every moment contains complete information about the measurement, including configuration information and descriptions. This is particularly important for saving complex state in internal data format.

Script expansions

Replacement of the custom-made specialized but limited script language by Python [5], enables us to create experiment control scripts as complex as it necessary. For instance, the script enables rather easy implementation of the spectra transformation from technical (internal) format to "private" format of particular spectrometer.
More over, a library of typical instrument operations, created by the script, is very useful as an intermediate layer between GUI and hardware control modules. Editing the library is not related to the software development systems, like "Visual C++", so an instrument responsible can do it himself.
The Python language is used also as a configuration description language.
The Python language is documented very well and there are many free development tools for it. This simplifies Python programming greatly.

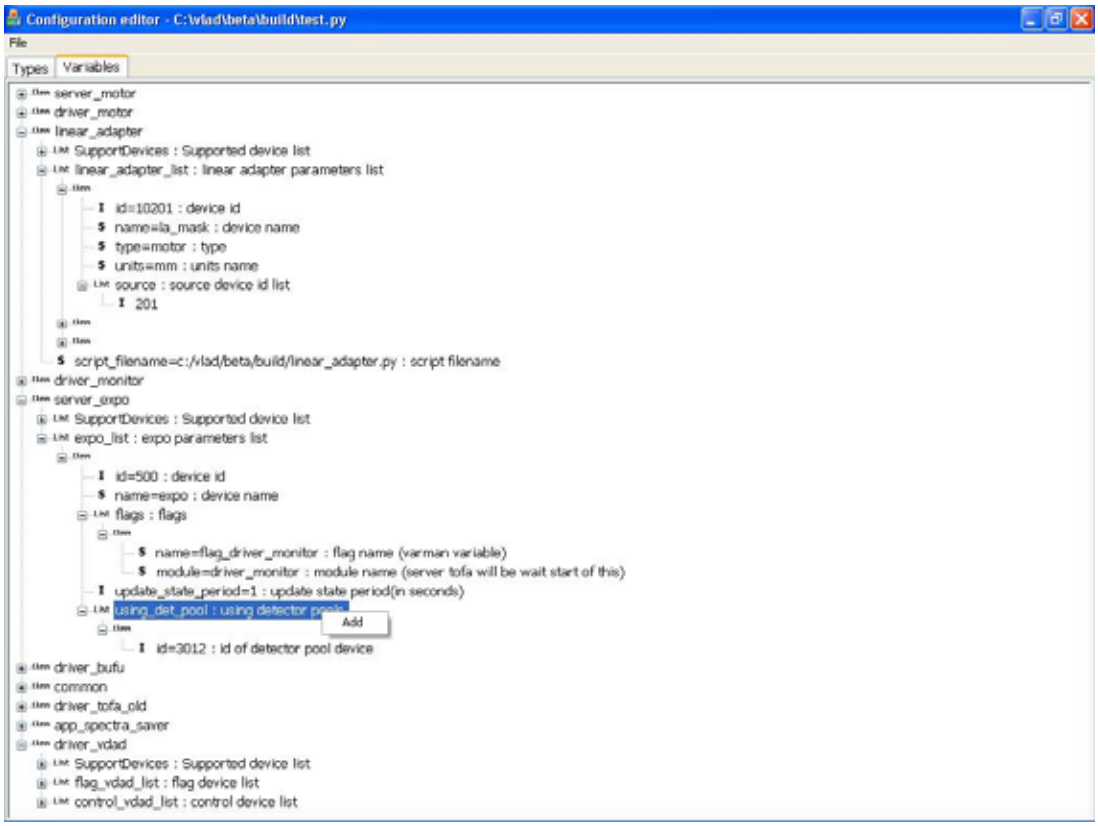

Fig.1. The Configuration editor

GUI

The Sonix GUI was changed radically. The structural changes of the complex enable us create universal components to show and control above measurements. Our approach in general is similar to the ideas of the NICOS project [6] by not exactly. The unified interface consists of

- configuration editor
- reflector program
- script interpreter
- log viewer
- spectra visualization program

There could be also private instrument GUI components for script generation, instrument tuning etc.

The Configuration Editor

This program is intended to edit the device set for the instrument and edit their parameters if necessary. There are also some operations to modify the tree itself (adding new devices, removing unnecessary devices, doubling etc).

The "Reflector" program

The Reflector (see Fig. 3) program enables representation of current information about all devices in the complex according information stored in the database. In a window a tree-like list of devices is drawn, grouped by device type. Clicking on particular device, one can see the complete information about it in a separate window. General device information (state, type, use_flag etc.) is displayed in the upper part of the window. Device specific information is displayed in the lower part as a tree again. Walking along that tree one can find urgent values of particular device parameters. Any device parameter can be included to the "Watch list" for permanent supervision. The contents of these windows are updated automatically.

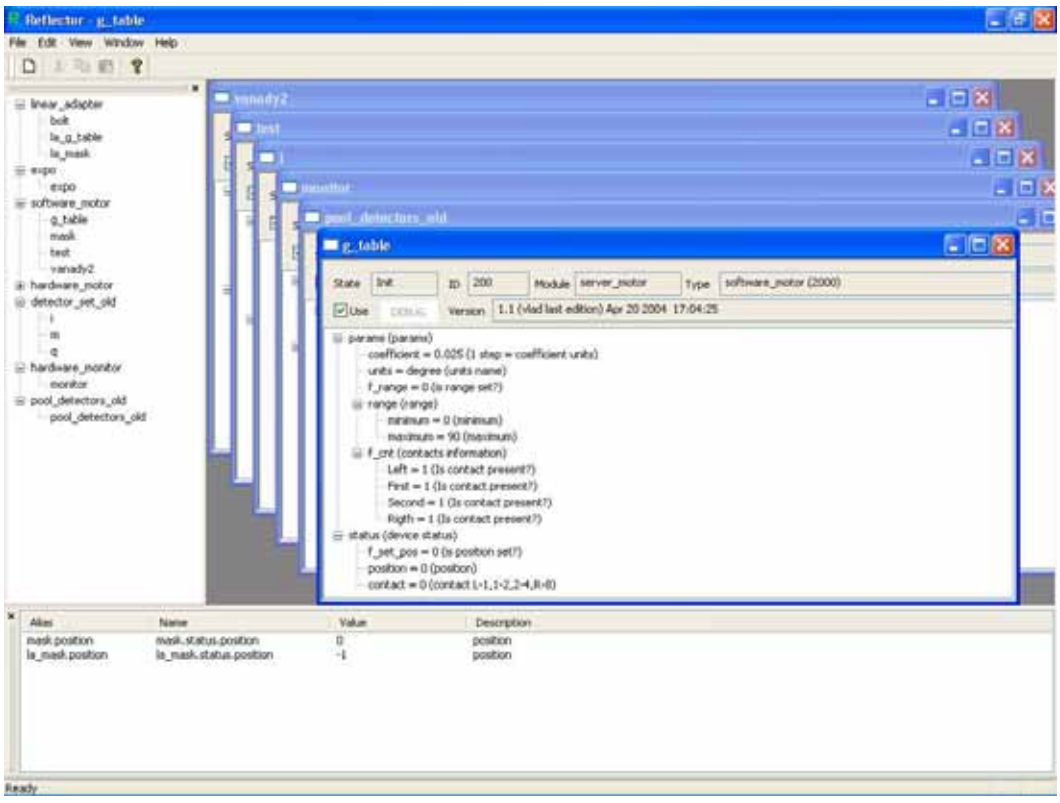

Fig.2. The Reflector program

The script interpreter

The script interpreter Pi (see Fig. 4) is intended:

- to start and stop the measurement procedure
- to suspend and resume the measurement procedure
- to display a script and show current position in the source
- to display selected script variables.

The Pi enables to launch script generators (see Fig. 5) to create script for typical instrument measurement by newcomers. For experienced users it enables to edit script with comfortable script editor.

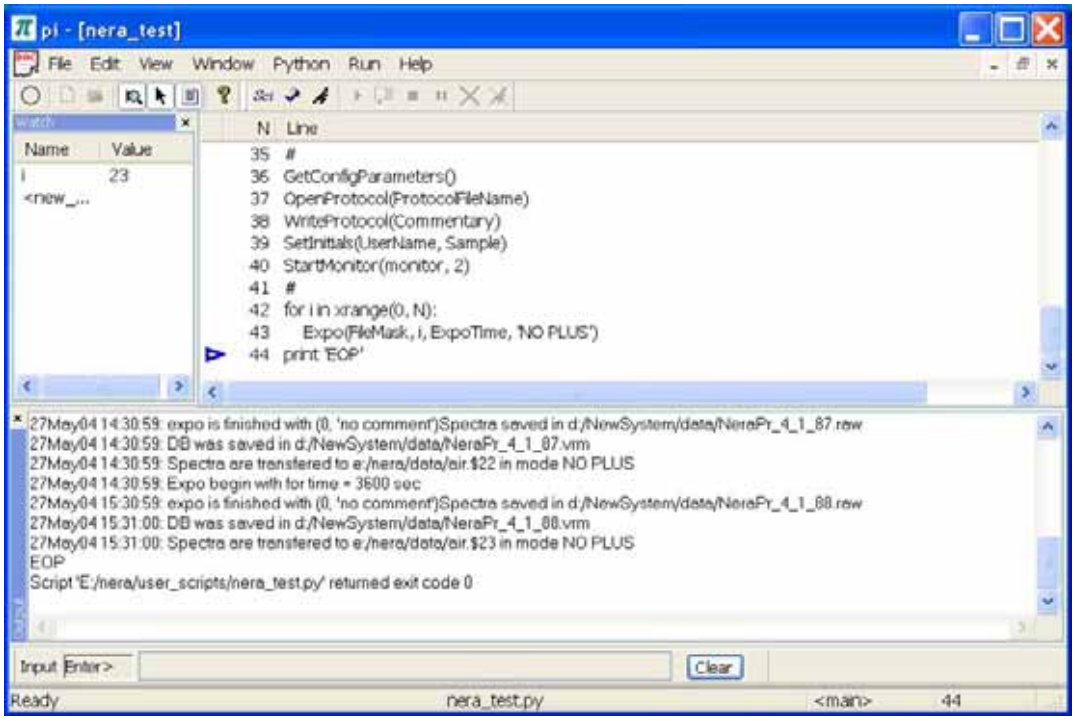

Fig.3. The Python interpreter typical view

The LogViewer program

This program enables to visualize log files. The log file is automatically initialized at the software complex start. It accumulates user, warning, error and info messages concerning both user actions and complex internal affairs. Each type of messages could have level from 1 to 5. The most important is level 1. The LogViewer helps the user to display sorted information concerning either current measurement or previous one. This program also helps the complex designer to understand what have happened in case of complex failures, simplifing a high level debugging.

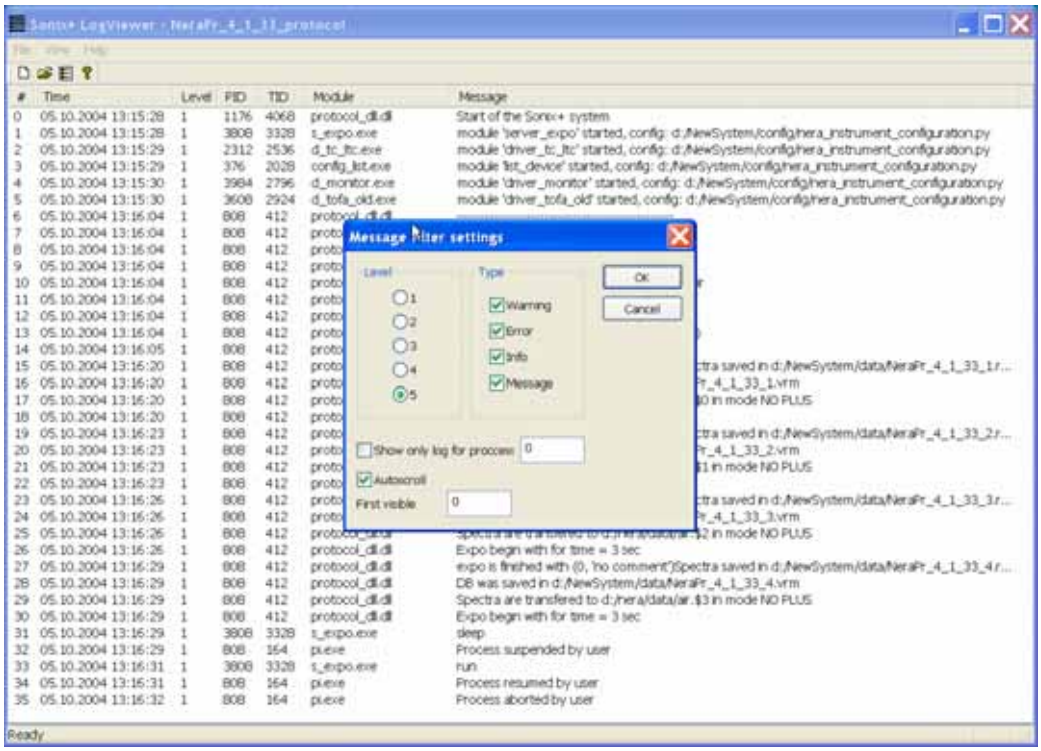

Fig.4. The LogViewer window

Private instrument programs

Although the Pi script interpreter can be used to create script from scratch we prepared template script

generators for each instrument. This is important for occasional users and newcomers especially if typical measurement procedure of the instrument is not simple. For instance, on the picture 5 there is a script generator window for the REMUR (spectrometer of polarized neutrons and reflectometers) instrument.

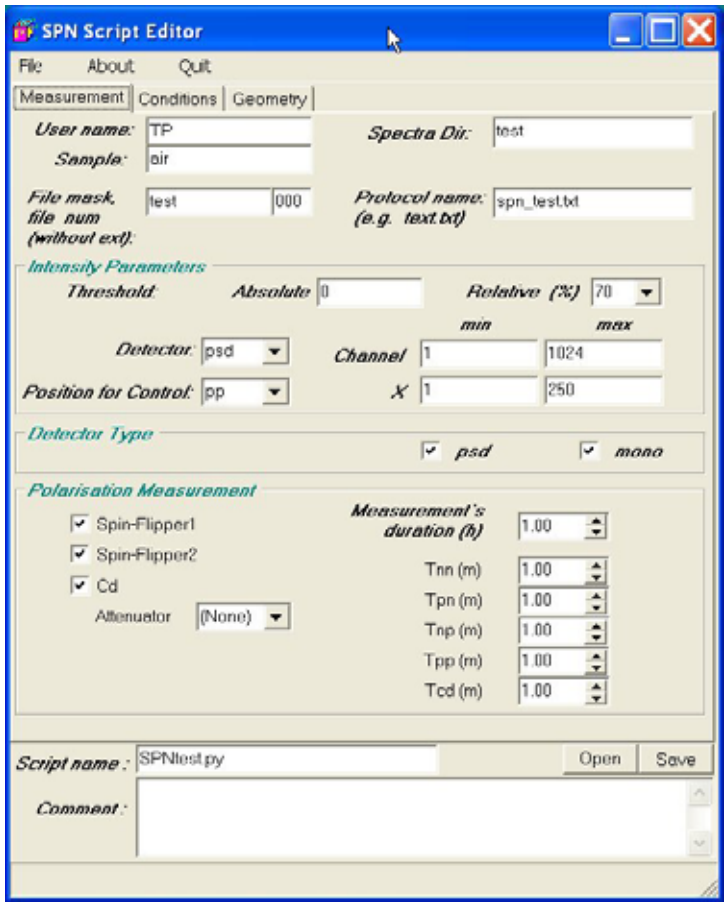

Fig.5. The REMUR instrument template script generator

The unified protocol for communication between modules enables to upgrade the complex with new components above the device level. As an example on the next picture there is program ICE intended for tuning the REMUR instrument. This program unites device control, spectra measurement and development as well as special visualization of results.

Spectra saving formats

Python script simplifies implementation of double spectra saving scheme. Initially, the spectra are saved in the internal “technical” format common for all instruments. Every file gets unique name to prevent rewriting of previously stored spectra. The database snapshot is stored together with the file. The snapshot contains all device parameters. This file may contain additional information required for spectra development. So this couple of files contains all information concerning particular measurement.

Subsequently, a special interpreter is called to transform the technical file to any desired “private” format. The file may be renamed according to the spectra name scheme accepted at particular instrument.

This transformation can be done later on to restore files or to transform files to another format, for instance, Nexus [7].

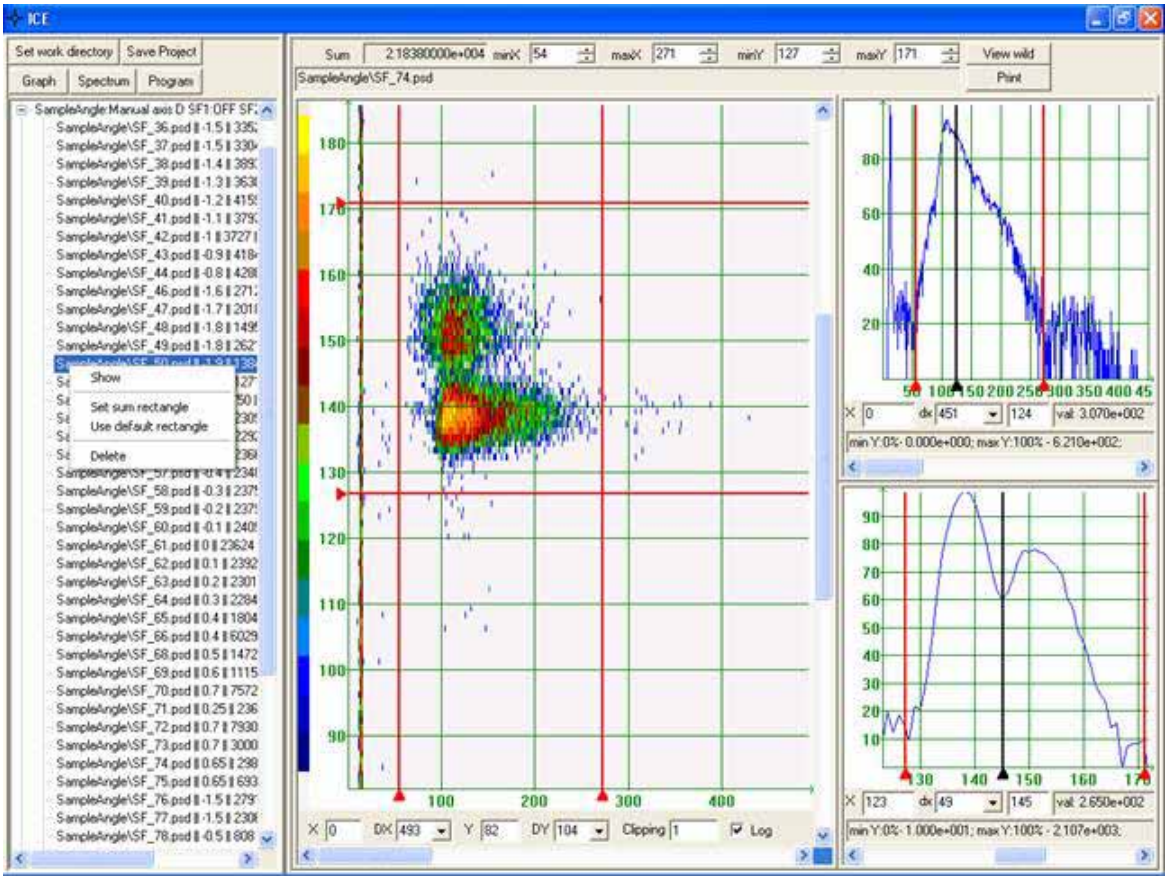

Fig.6. The ICE – the REMUR instrument tuning program

Spectra visualization and development

Python enables to easy connection of any visualization program to the measurement system in batch or manual mode. For on-line visualization an intermediate module to transfer spectra from detectors to visualization program is necessary. It is also possible to implement a communication protocol with visualization program by script.
All of above is true for spectra development programs. A script can be easily modified to include any new or existing program in measurement pipeline.

Conclusion

Due to the structural changes and using Python as a script language the Sonix+ complex become more powerful, flexible and simple in user control. At the same time it has become more unified and easily extendable.
At the moment the Sonix+ version for the Multicrystal inverted geometry tome-of-flight spectrometer NERA-PR has been tested without neutrons and is ready to be test at the facility. A development of version for the REMUR instrument, which is more complicated, is at the stage of final testing.

References

1. http://nfdfn.jinr.ru/~kirilov/Sonix/sonix_index.htm
2. http://nfdfn.jinr.ru/annual2002/AnRep-2002.pdf
3. http://webster.ncnr.nist.gov/events/nobugs2002/
4. http://www.esrf.fr/computing/bliss/
5. http://www.python.org
6. http://www.frm2.tu-muenchen.de/software/nicos/en/main.html
7. http://www.neutron.anl.gov/nexus/