Karlsruhe Institute of Technology

Institut für Synchrotronstrahlung (ISS), IT
ANKA

Macro package based Enhancement of SPEC controlled Experimental Setups

Thomas Spangenberg*), Karlheinz Cerff, Wolfgang Mexner

*) thomas.spangenberg@kit.edu

Certified Scientific Software's program package spec for X-Ray diffraction and data acquisition provides reliable instrument control to scientists at synchrotrons and other facilities worldwide. A new object oriented like software development concept for spec is proposed. A few naming rules plus a macro package in combination with a single client-server-application expand the manageability and options to control experiments considerably. As main goal spec gets an object-like handling and a standardized user interface of newly introduced devices. A generic server-client based interface allows a smooth integration of spec in more complex control environments via TANGO.

Device model

a)

b)

private:	address
read	current
read/write	voltage
command	HV on/off

Fig. 1: a) real device, a vacuum pump controller; b) logical abstraction as a set of independent properties; c) device declaration in SPEC‘s macro language.

The driver function implementation is following a strict naming scheme.

SPEC macro language

Certified Scientific Software spec™
X-Ray Diffraction Software

c)

```
#device declaration
def VPC_standardvalues(device,address) '{
VPC[device]["$adress"] = address
VPC[device]["*current"] = 0
VPC[device]["voltage"] = 0
VPC[device]["!on"] = "VPC_poweron"
VPC[device]["!off"] = "VPC_poweroff" }'

#driver function implementation
def VPC_readcurrent(device,verbose) '{... }'
def VPC_readvoltage(device,verbose) '{... },
def VPC_setvoltage(device,quiet,value) '{... }'
def VPC_cmdon(device,opt) '{... }'
def VPC_cmdoff(device,opt) '{... }'
```

Object like handling

```
VPC["vc1"]["$address"] ="x.x.x.x:y"
VPC["vc1"]["*current"] = 0.763
VPC["vc1"]["voltage"] = 2000
VPC["vc1"]["!on"] = "VPC_poweron"
VPC["vc1"]["!off"] = "VPC_poweroff"

blread_vc1_current(verbose);
blread_vc1_voltage(verbose);
blset_vc1_voltage(quiet,value);
blcmd_vc1on [opt];
blcmd_vc1_off [opt];

blread vc1.current
blread vc1.voltage
blset vc1.voltage [value]
blcmd vc1.on [opt]
blcmd vc1.off [opt]
```

Fig. 3: N devices can be instantiated without significant effort. The automatic generation of the interface functions guarantees the provision of all specific ‚object functions‘. The clear structure facilitates the rapid comprehension of (even unknown) device implementations.

Interface generation by macro package

```
#package initialization
beamline_define_state("x", "default")
beamline_init_namespace("x..")

#device instantiation
beamline_setDRV("vc1", "VPC")
VPC_standardvalues("vc1", "x.x.x.x:y")
. . .
beamline_setDRV("vc1", "VPC")
VPC_standardvalues("vcN", "x.x.x.x:yN")

#interface generation
beamline_defaultmacros()
```

Fig. 2: Very small package overhead and just *two* lines per device are needed to implement it. The whole interface will be generated and loaded by the last function call.

The whole object like functionality may be exported by SPEC. Without additional organization effort any macro package supported device is exported by its values and methods. A TANGO binding was realized. SPEC driven devices may be used or visualized from other API without disturbing or modifying SPEC driven experiments.

Conclusions

The macro package in combination with a few naming rules offers a straight forward approach to an object like access for device implementations with SPEC’s macro language. Additionally a systematic generated user interface and a device export via socket with a small time impact to SPEC can be provided at the same time.
A TANGO server for the SPEC macro package managed devices is provided and permits a remote control of them by other programs.

References

[1] http://www.certif.com
[2] http://www.tango-controls.org