THE ANKA B-FIELD TEST FACILITY CONTROL SYSTEM, BASED ON A SPEC MACRO PACKAGE ENHANCED SETUP*

Karlheinz Cerff, Thomas Spangenberg, Wofgang Mexner, Institut for Synchrotron Radiation, (ISS)-ANKA, Karlsruhe Institut of Technology, (KIT)-Campus North, Germany.

Abstract

The ANKA B-field test facility provides users with a flexible tool to investigate magnetic field distributions of different setups of coils or permanent magnets, optimal sensor types, geometrical alignments of probes and the possibility to change the independent physical stimuli to generate and alter magnetic field distributions [1]. From the point of Software development it is taken as an example of a straight-forward device implementation with a recently introduced type of macro based 'building block system' for devices in SPEC, [2]. This macro package provides the C-like SPEC with an object orientated framework with a namespace and class concept to represent the power supplies of different brands, probe positioning devices and measurement amplifiers.

INTRODUCTION

The B-Field Test facility provides measurement data of magnetic field distributions of coils or permanent magnet structures, within the range of um spatial resolution, over positioning ranges up to meters, devices in use are,

- a stepper motor driven, encoder monitored linear positioning probe, equipped with a variable geometrical arrangement of Hall-sensors to measure B-field induced voltage gradients.
- Two power supplies, consisting of a main and a second, multiple power supply, driving individual shaped I-current ramping functions for corrector coils.
- A Digital Multi-Meter (DMM) of Keithley, type 'k2700' to read out, up to n Hall-probes.

The control software package should also generate a raw data fit for a polynomial of variable degree i (i<=9), for up to n Hall-probes. At last the control system monitors the safe operation of the Test facility, for example it shuts down the main power supply when a superconducting coil under test is quenching.

IMPLEMENTATION

In the context of the 'Macro package based Enhancemend of SPEC controlled Experimental Setup'[3], this means that the device properties are stored as elements of data structures (SPEC global associative arrays). The task of the software development is, to

- set up an abstract model of the B-Test Facility hardware devices.
- write the device drivers for B-Test Facility motor, power supplies and digital multi meters.
- linking the resulting SPEC macro functions to the Interface generated by enhanced macro package.

The introduction of a set of interfacing rules minimizes the risk of damage to existing SPEC-structures, furthermore it opens the possibility to port in this way generated SPEC-'classes' to other experimental facilities.

Table.1: B-Test facility, list of realized implementation of functions, devices, SPEC '-instances' and –'classes'.

physical function	device	SPEC-'instance'	SPEC–'class' (macro)
motor controller, one channel	OMS-Maxv	'm0'	Motor.mac
main power supply, 1 channel.	FUG NTV-1000	'fugbig'	Fug.mac
power-supply small, 8 channels	FUG NTV-100	'fug'	Fug.mac
Digital multi-meter Hall-probes	Keithley, K2700/7703	'k2770' 'Hall n'	Anka-Keithley.mac

Setting up the B-Test Facility, the two power supplies are defined as members of the 'class', represented by FUG.mac. They are both instantiated as objects 'fug' and 'fugbig' in the declared global associative array 'FUG', writing a set of device dependent standard-values to it. SPEC-associative Arrays offer as possible arguments arbitrary strings or numbers instead of integers [2]. In the 'class'-macro keithley_anka.mac, the Keithley DMM is instantiated as object "k2700" and the connected Hall-probes as objects "Hall-1"-"Hall-15. The minisetup class' macro contains the 'standardvalues' declarations and a data fit object to fit raw data to a polynomial up to the order of nine.

Benefit

- Two FUG devices, representing nine power supply 'objects' can be accessed by 11 (for the main power supply) and 73 (for the corrector power supply) standard-function calls obeying the naming rules introduced by the macro package.
- Up to fifteen Hall-probes have to be addressed by 255 standard function calls for the Hall-probes plus three functions for the K2770.

The advantages using the object oriented approach is clearly visible, there is no need to write, a set of 84 nearly identical conventional SPEC-functions for power supplies and additional 255 functions to handle the output, in addition existing ANKA-beamline driver modules for motors can be used.

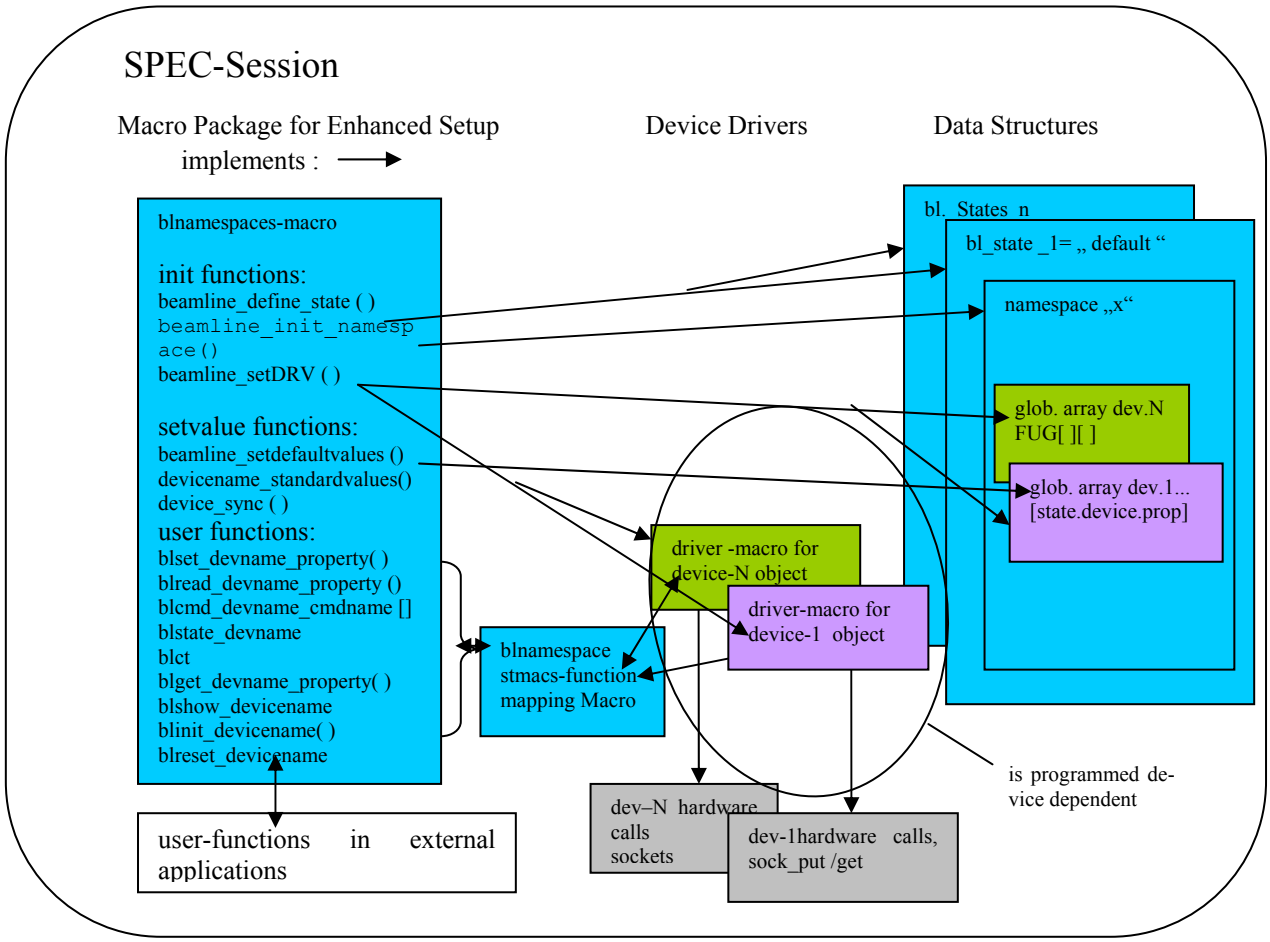

Figure 1: Macro Package generated structures (blue), driver software to be written (green and cyan), SPEC-built in functions (grey), interface function calls (white)

BUILDING THE B-TEST FACILITY DEVICE MODELS

Loading and executing, the blnamespaces-macro, which is the heart of enhanced setup, submits the functionality for setting up the namespace and global array structures:

beamline_define_state (“x”,”default”*)*
beamline_init_namespace (“x ..”*)*

Both function are processed only once, because all devices are instantiated in one state “default” and namespace “x”. In principle the concept allows multiple state definitions “others” which could be used for example to define different arrangements of Hall sensors The functions below instantinate the device-objects given in the first column:

beamline_setDRV(“fugbig . ”,”FUG”)
beamline_setDRV(“fug . ”,”FUG”)
beamline_setDRV(“k2700a . ”,”KEITHLEY”)
beamline_setDRV(“hall n . ”,”KEITHLEY”)

The power supplies are abstracted by:
status, ramping behaviour, address, type, set/get/ voltages, I-currents, I-current-rates. The device models are stored as sets of object variables in the associative arrays “FUG” and “KEITHLEY”, generated by the macro package init functions, s. Fig.1:

devn . property = “value” structure:

```
FUG["fug"]["$active"] = 0
FUG["fug"]["$adress"] = "192.168.4.4:23"
FUG["fug"]["$fugtype"] = "FUG-NTV 100"
FUG["fug"]["$maxcurrent"] = 10
FUG["fug"]["*current1"] = 0
FUG["fug"]["currentrate 1-n"] = 0.2
FUG["fug"]["dcpower 1-n"] = 0
FUG["fug"]["readout 1-n"] = 1
```

The prefix in the second array elements marks the state of properties: “private”, “read only”, “read/write” or “command “.

B-TEST FACILITY DEVICE DRIVERS

The program code which has to be written are the device driver macros for power supplies “fug” and “fugbig” and the digital multi meter with connected Hall- probes.

The functions can be grouped in :

- internal functions, like socket functions to set/get specific hardware register values, to reset or initialize devices, to address sub device and functions to process data strings received.
- Functions for data synchronisation with the pre-defined standard values in associative arrays or with the ongoing values of the hardware device of interest,
- functions to set /get device parameters by calling external measurement devices used at ANKA-beam lines
- functions, which are 'built in' SPEC, here used for the linear motor drive with encoders to position Hall-probes.

FUNCTION-MAPPING

A set of 'standard-' or user functions' for communication is generated automatically by the macro package. The bulky type of driver functions with long argument lists is mapped to a set of user friendlier functions. The functions have the general form:

```
def user_function (value, argument ) '{
<return> driver function ("device name . property")
}
```

The simplest user functions don't have arguments, for example a 'blct'-call, gives the outputs of all parameters of the assembly of power supplies, DMMs, and Hall-probes of the B-Test facility:
A generic example for function mapping, will be the 'setcurrentrate' user function for 8-fold power supply 'fug', device No 3, with a I-current rate of 0.2A/sec.
The user function call is ,

- *blset_fug_currentrate3(0.2)*

mapping to the device driver function :

- *FUG_setcurrentrate3("fug", 0.2, 3).*

This calls the SPEC socket functions of the driver to write an appropriate value to the hardware register sub address 3 of the power supply "fug", after command reference given in [5]:

```
def FUG_setcurrentrate3(device,quiet,value,) '{
__FUG_setcurrentrate(device,quiet,value,2)  }'

   # call of __internal driver function :
def __FUG_setcurrentrate(device,quiet,value,devnr) '{
   #which type of power supply ?:
if ( FUG[device]["$fugtype"]=="FUG-NTV 100") {
   # write external inputs for ps with devnr=2+1 to'
    value':
value =  NumberInput ("current rate", FUG [device]
[sprintf ("setcurrentrate%i",devnr+1)] ,0, 1, quiet,
value);
   # call subdevice 3, addressing, convention, s. command reference [5]
__FUG_sendcommand(device,sprintf ("%s>S%iR
%g\n",sprintf("#%i",int(devnr/2)),devnr-
2*int(devnr/2), value ));
   # The __internal function uses the basic 'built in'
    SPEC socket_put function:
def __FUG_sendcommand (device,command) '{
  sock_put(FUG["device"]["$address"],command);
        }
}'
   # value gets the formatted readback from subdev. 3:
value = __FUG_splitanswer(__FUG_readback
(device));

   # __internal function calls basic  sock_get function:
def __FUG_readback(device)  '{ local tmp;
     tmp=sock_get(FUG["device"]["$adress"]);
       }'
   # updates appropriate element of global array FUG
with current read back value:

FUG[device][sprintf("setcurrentrate%i",devnr+1)]=_
_FUG_readcurrentrate (device, quiet ? 0 :1,devnr);
 }
```

CONCLUSION

The object oriented implementation, by use of existing beam-line software modules make the procedure straightforward since only the missing drivers for power supplies, digital multi-meters and the raw data evaluation algorithm, have to be introduced. But synergy proceeds, the FUGs will be the power supplies of future insertion devices [4] at ANKA, so the Software modules written to control its devices can easily be ported to the control system of the next ANKA superconducting undulator.

REFERENCES

[1] *CASPER*- A magnetic measurement facility for superconducting undulators,
E Mashkina et al *2008 J. Phys*.: Conf. Ser. 97 012020
[2] www.certif.com, software SPEC
[3] Macro Package based Enhancement of SPEC controlled Experimental Setups, T. Spangenberg, K. Cerff, W. Mexner
Proceedings of PCaPAC2010, Canadian Light Source, Saskatoon, Canada , October 2010
[4] A modular control system based on ACS for present and future ANKA insertion devices
K. Cerff, W. Mexner, T. Spangenberg, M. Hagelstein,
Proceedings of PCaPAC2008, Ljubljana, Slovenia, October 2008
[5] FUG, Probus, ADDAT30, command reference, V2,13