The Bnldsp Fesa class quartet BnldspSIGNAL, BnldspTRIG, BnldspMUX, BnldspSCOPE forming LEIR's RF beamcontrol signal interface to the OASIS data presentation layer.

E. Bracke, A. Butterworth
(AB/RF-cs)

Tables of Contents and Figures.

Introduction.

LEIR's RF beamcontrol signals, that are produced / acquired in the same BNLDSP hardware that forms the actual beamcontrol system, are used by the accelerator physicists and operations specialists for diagnosis and subsequent optimization of the whole accelerator process.

The AB/CO group provides the 'OASIS' general purpose interface for the graphical data presentation layer, originally intended for the presentation in control rooms of remotely acquired samples from commercially available analogue digitizer boards.

By presenting our internal BNLDSP signals via an interface, consisting of 4 specially designed and closely collaborating Fesa classes, it was possible to conform to the OASIS signal and control interface requirement that is also used for the analogue hardware digitizers. We thus could profit of the oscilloscope like capabilities of an already existing and, for main control room personnel, familiar tool.

This note starts with a description of the order in which a measurement should be setup. It is intended as an example of how to use the system while defining a minimum set of parameters such as to actually deliver an array of 'samples' acquired from the hardware. This useful exercise is done because, for testing purposes, sometimes we can not / do not want to use OASIS for control and presentation, we rather must be using a more 'directly to the hardware' tool like the Fesa Navigator.

Next is described how the Bnldsp Fesa class quartet, that provides the OASIS interface, is actually collaborating. Special attention is given to the organization of the Fesa design on the issue of 'equipment links' where is implemented the aforementioned close collaboration between the 4 classes.

Finally a 'make recipe' section of this note goes into some detail of the order in which these Fesa classes need to be compiled for forming a correctly working ensemble.

This note is stored in the `BNLDSP/DOCUMENTATION` directory which can be found in the CERN CVS file repository, currently at:
http://isscvs.cern.ch/cgi-bin/viewcvs-all.cgi/?root=abrfcs
(and also available on Erik's home page at:
http://bracke.home.cern.ch/bracke/HTML/LEIR_Development/LEIR_Development.htm).

Note:
Also have a look at the `readme.txt` file in the Bnldsp C code modules' `BNLDSP` root directory. It might have some interesting (last minute) info.

The Bnldsp Fesa class quartet.

Four Fesa classes form the interface between the BNLDSP VME hardware (currently 3 boards) and the OASIS data presentation layer system:

- BnldspSIGNAL.
  Instances of this class define a few constant parameters used by OASIS for naming interesting signals, scaling and labeling of measurement axes, buttons etc. on screen. For our application we can consider the instances as an enumeration of all available Bnldsp signals; each instance is a 'container' of constants. Examples of these: Full Scale Range value, input impedance of an oscilloscope when this signal would be connected to a real oscilloscope (no meaning in Bnldsp, but required for Oasis), a linear translation formula (offset and a scaling factor) to be applied before data is displayed, the unit in which the data is expressed (V, A, etc).
  There is no RT action in this class that accesses the hardware, only operator read only access from the Fesa server level (class properties).

- BnldspTRIG.
  Another 'container' class, similar usage as for BnldspSIGNAL. Here we enumerate all possible trigger signals, instances within a cycle, where 'interesting' things happen. They are to be seen as 'external triggers' for connection to an external trigger input of a normal oscilloscope. Examples of constants in a BnldspTRIG instance: delay, to be added to the absolute time '0' before the trigger is valid, input impedance of an oscilloscope when this trigger would be connected to a real oscilloscope's external trigger input (no meaning in Bnldsp, but required for Oasis).
  It is to be noted that the absolute start (time '0') of all instances of the class is the start of the current cycle.
  There is no RT action in this class that accesses the hardware, only operator read only access from the Fesa server level (class properties).

- BnldspMUX.
  Two types of input to output multiplexers exist: one for signals and one for triggers (mutually exclusive). Multiplexers virtually connect signals and triggers to oscilloscopes; i.e. instances of the BnldspSCOPE class (see below).
  One signal multiplexer is instantiated fore each Bnldsp VME board and one trigger multiplexer services all Bnldsp VME boards.
  The multiplexers are the non-RT 'workhorses' of our interface. When an operator sets-up the connections between a scope and a signal and / or trigger, the multiplexer properties take data from all other 3 classes, transform it as required before posting the result into a connected BnldspSCOPE instance's data store (Fesa fields) and issues a notify command to the scope, such as to validate its (now modified) settings. If trigger delay settings are involved, this notification then also triggers an action in the scope concerned that changes the delay setting in the to this measurement channel dedicated BnldspTIM instance for the 'start of measurement' timing setting.
  There is no RT action in this class that accesses the hardware, only operator read / write access from the Fesa server level (class properties).

- BnldspSCOPE.
  Scopes finally do access the hardware, effectively setting up a measurement in BNLDSP VME boards and getting the acquired data out of them. After the trigger (instance of BnldspTIM) has started data taking, the data from the hardware is copied by an RT action into a Fesa field serving as a buffer, at a moment (LTIM driven) that data taking is guaranteed to be finished.
  When OASIS now has subscribed to this data, then every time BnldspSCOPE refreshes it data store, it gets a new copy of this buffer via the network (middleware).

Setting-up a measurement.

The entry point for setting-up a measurement with the OASIS quartet is the configuration of 2 multiplexers. It is the multiplexer (instance of BnldspMUX class), one for input signal selection and one for trigger selection, that forms the main operator configuration interface towards a scope (instance of the BnldspSCOPE class).

The multiplexer gets the name information of the various available input signals and triggers from the data space of the container class instances BnldspSIGNAL and BnldspTRIG. These signal and trigger names are for operator selection of an internally used 'index' into tables with values that are subsequently 'plugged' into the data space of a selected scope (instance of BnldspSCOPE class) and used by the latter for setting-up in real time the internal hardware connections in the Bnldsp VME boards such that a required measurement can indeed be performed.

Of course it should be remembered that measurements in the BNLDSP boards take place in a cycle-to-cycle fashion (PPM Users in LEIR terms). This means that correctly setup of the timing (enabling of the required sub-cycle in LTIM and BnldspTIM instances) is also needed for a measurement to be successful. Notably, an important detail is that data acquisition in the BNLDSP hardware is **only** done during the 'Beam Control' phase of the hardware state machine, which means that the 'End Beam Control' instance of BnldspTIM (EAX.EDSP, enumeration value: #2) should have a delay value that is sufficiently large to allow the required number of acquisition samples to be gathered in the time left after the selected scope trigger has started the acquisition. The important timing signals generation is done with CTR-V (VME) or CTR-P (PCMCIA) modules controlled by Fesa LTIM instances and internally to the LEIR BNLDSP system, by software timing instances of Fesa class BnldspTIM. Notably of concern, but not further discussed here, are:

- LTIM instances:
  - EAX.DSPCLKA, B, C: Clock signal generators for the VME synchronous logic electronics.
  - EAX.SDSPA, B, C: Start of cycle generation for the VME electronics.
  - EEX.PRE-EBC, EEX.EBC-RD, EEX.EBC-WR, EEX.RST-DSP, EEX.POST-EBC: Interrupt drive for the Fesa BNLDSP RT software.
- BnldspTIM instances:
  - EAX.EDSP: 'End Beam Control' toggles the BNLDSP state machine to 'Hardware access allowed' state. This is the first instance of BnldspTIM and has the ordinal value of enumTim Fesa field always defined as '2'.
  - EAX.SAQA1..4, B1..4, C1..4: 'Start of Acquisition' Measurement start trigger. These instances are in the examples of this note, displayed as 'timingNames' in the figures for 'Finding indexes for signals / triggers and scopes'.

Setting-up a measurement is here illustrated with the Fesa Navigator; it is a 'close to the hardware' view of the system that allows acquiring a good comprehension of the configuration steps to be done for a successful measurement.

Input signal selection.

All the available BNLDSP signals, to be connected to the OASIS system as input signals, are in the BNLDSP system a function of the particular BNLDSP VME board where the signals are produced. Input signal selection is therefore done on a 'per VME board' instance of BnldspMUX

We use the property 'Internals' of the BnldspMUX instance of the VME board involved to find out which index value for the desired input signal and oscilloscope instance we need for setting up the required connection in the hardware for doing the measurement. With this property, we can inspect all signals and scopes that are available for the particular VME board.

In this example we want to connect the signal 'ppcrf10_EA.FGFREVCORR-DS', generated in BNLDSP board A, to the oscilloscope 'ppcrf10_EA.SCOPE-A2'. We find for this MUX input that the signal index is '4' and that for the oscilloscope, the index is '2'. Note that internally, in the Fesa class code, arrays are numbered from '0' onwards, whilst the 'indexes', as used here, start with '1'.

On the picture below (Figure 1 Finding indexes for signals and scopes.) we can also see, for information, that the start of the measurement in the BNLDSP hardware will be governed by BnldspTIM instance 'ppcrf10_TIM_test29' (the scope's index '2' indicates in the triggerNames field the BnldspTIM instance that services the selected scope). Manipulation of the delay value in this timing is done 'on line' by the BnldspSCOPE instance (see later), 'ppcrf10_EA.SCOPE-A2' in our case.

Furthermore note that in the field 'triggerNames' nothing has been defined here. This is correct; the currently selected instance of BnldspMUX is a 'signal MUX' and therefore the field of triggerNames is irrelevant.

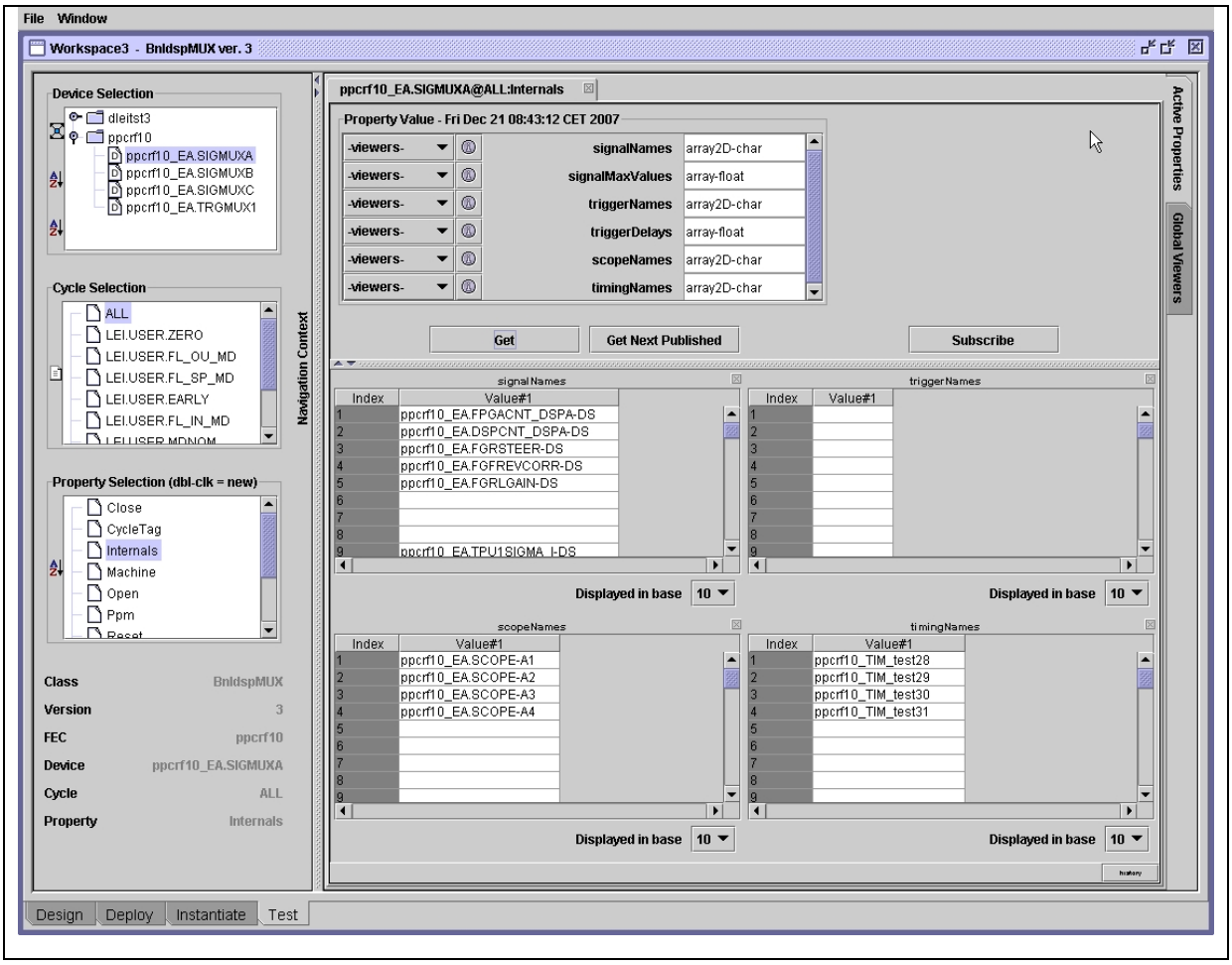

Figure 1 Finding indexes for signals and scopes.

To finalize the connection, we use property 'Close' (Figure 2 Making the signal MUX connection.) to connect the input signal to the scope. The MUX's 'input' field gets the collected input signal index and in the 'output' field we enter the scope index. Clicking 'Set' will validate this connection.

Property 'Close' of the signalMUX causes, in the datastore of BnldspSCOPE, instance 'ppcrf10_EA.SCOPE-A2' (index '2'), the setting of default values for the scope's input signal parameters offset, offsetMax, offsetMin, sensibility, and also the scope's signalMuxStatus field is set to the signal index of '4' (meaning: scope is connected to a signal). Furthermore it notifies the BnldspSCOPE properties 'Sensibility' and 'Offset' of this update, such that OASIS, when subscribed to this scope, is also aware of this.

In the signalMUX we can inspect the result of 'Close' with property 'Status' (Figure 3 The signal MUX output status.) where we see that element of index '2' (the MUX's output index, which is effectively the scope) contains the value of '4' and which is the index of the input signal. Note here that the scope with index '1' (i.e. ppcrf10_EA.SCOPE-A1) already was connected to input signal '5' (i.e. ppcrf10_EA.FGRLGAIN-DS).

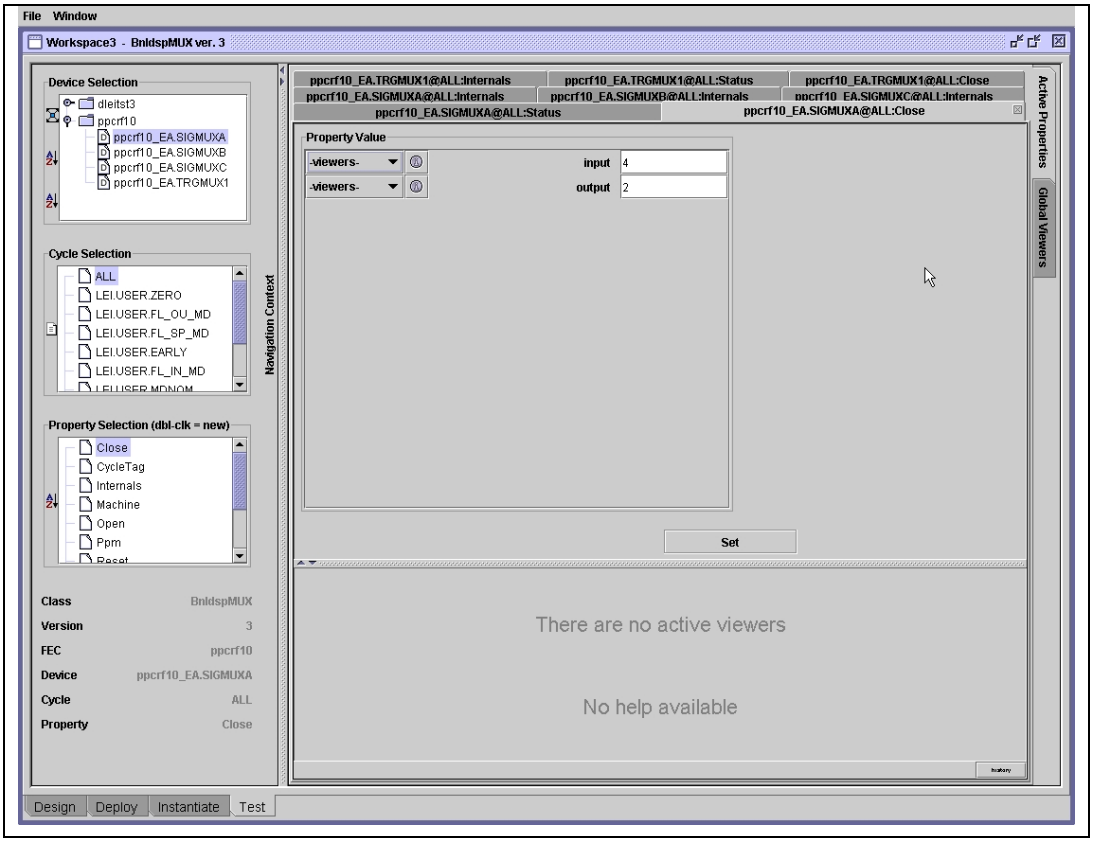

Figure 2 Making the signal MUX connection.

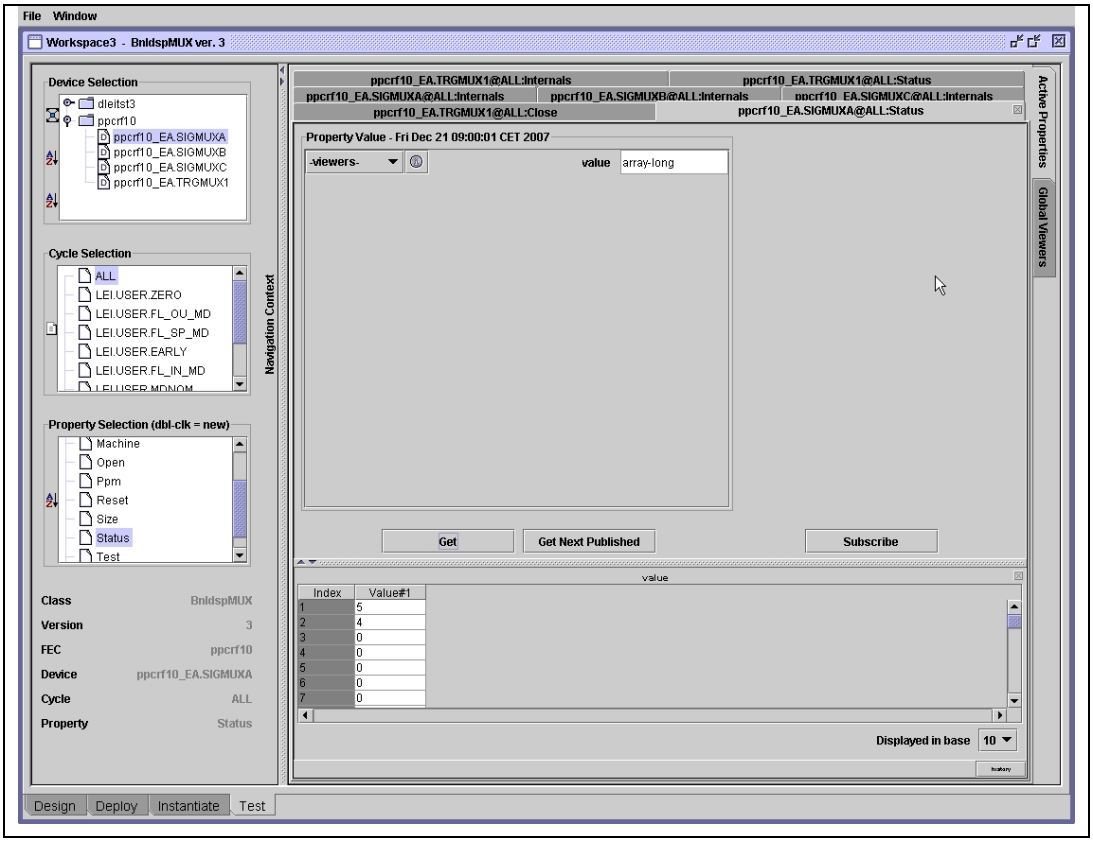

Figure 3 The signal MUX output status.

Trigger selection.

Setting up a scope's 'external' trigger follows a similar pattern like setting up for signals. The possible trigger points for performing a measurement in an accelerator cycle are however the same for all VME boards. We thus find only 1 instance of BnldspMUX in the BNLDSP LEIR OASIS system that is designated the role of 'trigger MUX'.

The trigger points, relative to the start of a LEIR cycle, have been defined for typical, 'interesting' points of time within the cycle. This, in order to avoid that an operator has to calculate himself the required delay (in seconds) to the moment where he should start his measurement. All intricacies of e.g. taking into account the current cycle length etc. are this way 'hidden' from him; he just selects a trigger value from the total available enumeration and the system takes care of defining the delay value that must be programmed into the relevant instance of BnldspTIM that starts his measurement.

Again, we use the property 'Internals' of the BnldspMUX instance that is assigned for multiplexing the triggers to the scopes for finding out about the indexes of these .

From the picture below (Figure 4 Finding indexes for triggers and scopes.) we use for the current setup example e.g. the trigger 'ppcrf10_EAX.SCY+600' and connect it to our already selected signal scope 'ppcrf10_EA.SCOPE-A2'. The index for the MUX's input is thus trigger '5' whilst the MUX output index will have to be '2'.

We can again verify that the start of the measurement in the BNLDSP hardware will be governed by BnldspTIM instance 'ppcrf10_TIM_test29' (the scope's index '2' indicates in the timingNames field the BnldspTIM instance that services the selected scope). Remember that internally, in the Fesa class code, arrays are numbered from '0' onwards, whilst the 'indexes', as used here, start with '1'.

Furthermore we note that in the field 'signalNames' nothing has been defined here. This is correct; the currently selected instance of BnldspMUX is a 'trigger MUX' and therefore the field of signalNames is irrelevant.

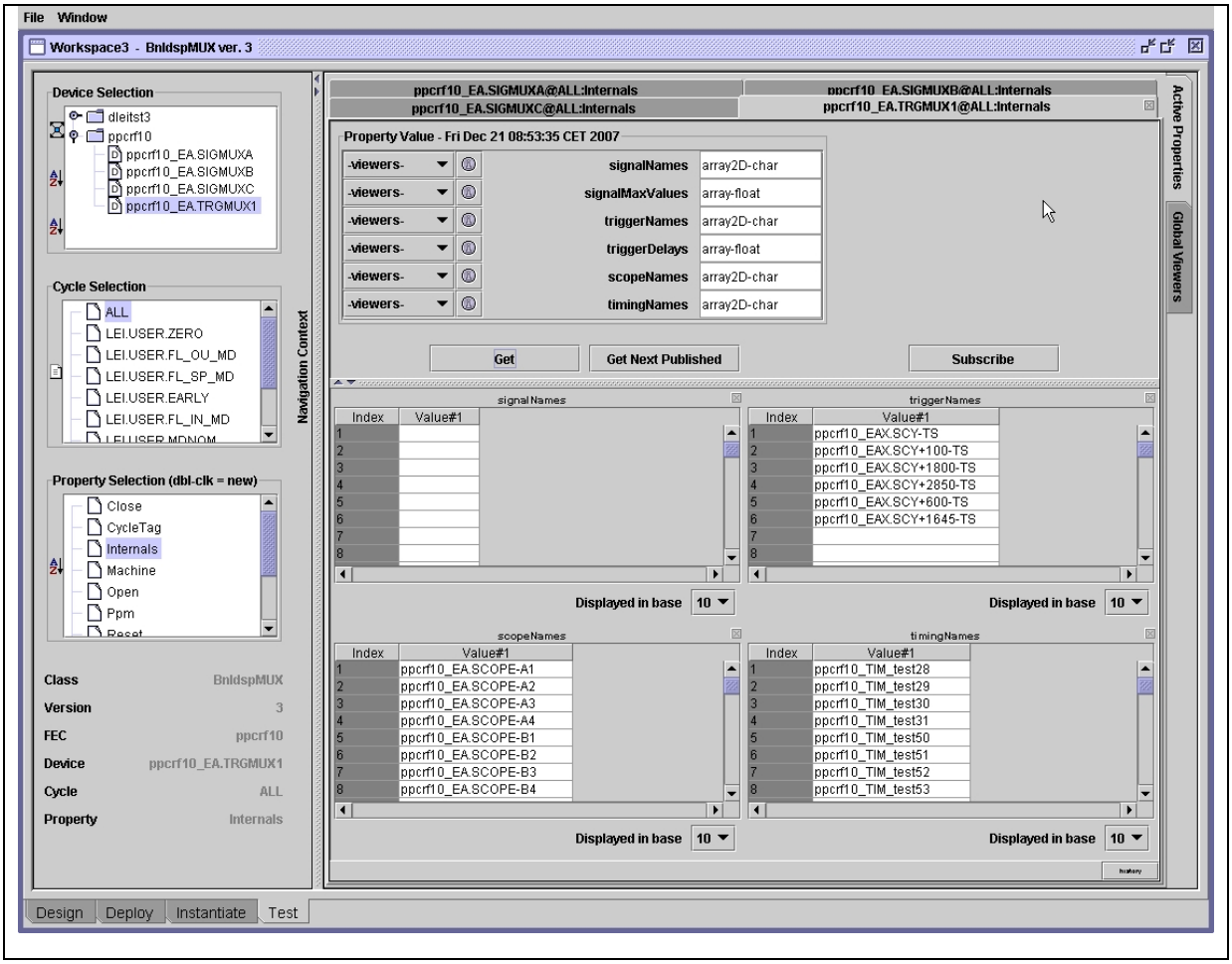

Figure 4 Finding indexes for triggers and scopes.

To finalize the connection, we use property ‘Close’ (Figure 5 Making the trigger MUX connection.) to connect the input trigger to the scope. The MUX’s ‘input’ field gets the collected input trigger index and in the ‘output’ field we enter the scope index. Clicking ‘Set’ will validate this connection.

Property ‘Close’ of the triggerMUX causes, in the datastore of BnldspSCOPE, instance ‘ppcrf10_EA.SCOPE-A2’ (index ‘2’), the scope’s triggerMuxStatus field to be set to the trigger index of ‘5’ (meaning: scope is connected to a trigger). No update, however, to any BnldspSCOPE property is done; OASIS, when subscribed to this scope, doesnot need to be aware of this.

In the triggerMUX we can inspect the result of ‘Close’ with property ‘Status’ (Figure 6 The trigger MUX output status.) where we see that element of index ‘2’ (the MUX’s output index, which is effectively the scope) contains the value of ‘5’ and which is the index of the trigger. Note here that the scope with index ‘1’ (i.e. ppcrf10_EA.SCOPE-A1) already was connected to trigger ‘1’ (i.e. ppcrf10_EAX.SCY-TS).

OASIS is not intended to work in a PPM cycle to cycle multiplexed environment. The signal digitizer hardware, for which the system was initially designed, generally cannot cater for a PPM like environment, but the BNLDSP hardware and software is fundamentally constructed for it. The oscilloscope trigger provider, Fesa class BnldspTIM, could therefore be used to ‘multiplex’ triggers and Fesa class BnldspSCOPE, which accesses the hardware in real-time, caters for de-multiplexing with its data buffers defined on a ‘per PPM User’ basis.

To make use of this, the trigger MUX has the property ‘User’ (Figure 8 The trigger MUX PPM User setting.), where we must program for which PPM User (sub cycle in the LEIR super cycle) data has to be acquired. We need to enter a ‘value’, which is a bitmask (note that value = -1 is valid, meaning: ‘All sub cycles’!), and an ‘output’ which represents the oscilloscope index of the already mentioned connected scope. The PPM User value can be calculated from the ‘User lines’ of the LEIR telegram documentation currently at: http://ab-dep-co-ht.web.cern.ch/ab-dep-co-ht/timing/Seq/tgm.htm.
Select from: ‘Telegram description for operational environment’, row: ‘User lines’, the link ‘LEI’. With the

formula: $2^{<Lnum - 1>}$ we find for PPM user: 'LIN3MEAS' the bitmask value of: $2^{(10 - 1)} = 512$. (Figure 7 Trigger MUX PPM User bitmask value.)

Setting this value '512' (PPM User) for output '2' (ppcrf10_EA.SCOPE-A2) will have the effect of updating the 'Start of Measurement' BnldspTIM instance (ppcrf10_TIM_test29) of our oscilloscope with the correct delay value for PPM user 'LIN3MEAS'; corresponding to the selected trigger 'ppcrf10_EAX.SCY-TS+600'. In this case the trigger delay will therefore be '600' (mSec) because the event EAX.SCY-TS ('Start of CYcle') is the ppcrf10_TIM_test29 'time zero' timing reference.

Some interesting detail on the mechanism of setting the effective measurement delay by BnldspMUX instance 'ppcrf10_EA.TRGMUX1'.

This mechanism, in the code of BnldspMUX instance 'ppcrf10_EA.TRGMUX1', that reprograms the 'Start of Measurement' BnldspTIM instance 'ppcrf10_TIM_test29' consists of first setting-up the Fesa field 'triggerDelay' in scope 'ppcrf10_EA.SCOPE-A2' and then notifying the latter's property 'Delay' that the triggerDelay field has been updated. This work is done by BnldspMUX property 'User'.

In its turn, the BnldspSCOPE class code for its property 'Delay' will then recalculate the sum of the (new) value of 'triggerDelay' and the (already existing) value of 'scopeDelay' (the latter is also a field of BnldspSCOPE). Finally the 'Delay' property's code accesses 'ppcrf10_TIM_test29' property 'Setting' via the property interface with the newly recalculated delay value.

Property 'Setting' of BnldspTIM instance 'ppcrf10_TIM_test29' at last will update the effective delay value in its Fesa field 'delayFLT' and its server code will translate the 'float' value into the required 'long integer' format value for the hardware.

By calling BnldspSCOPE and BnldspTIM by the property interfaces (equipment-link) we make use of the correct 'update' algorithms, which need to be coded only once in the Fesa classes involved, and also that OASIS, who might be subscribed to the scope instance, is thus aware of modified settings.

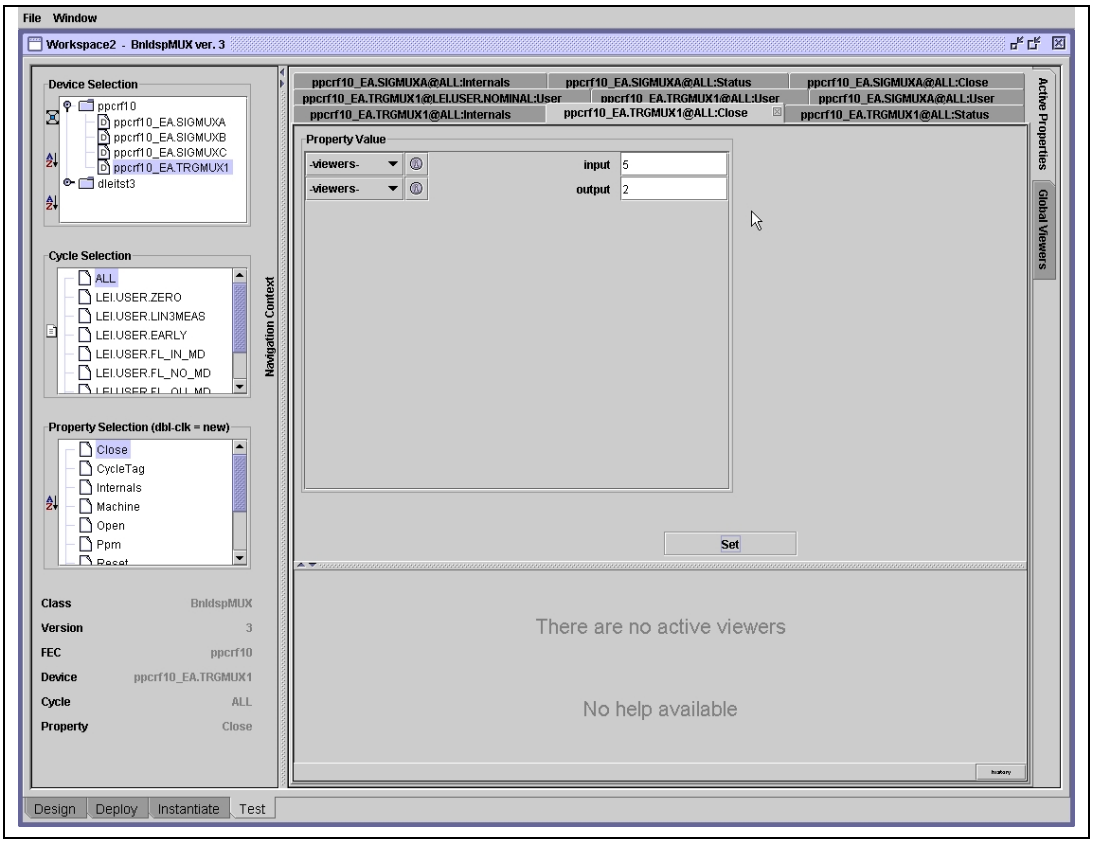

Figure 5 Making the trigger MUX connection.

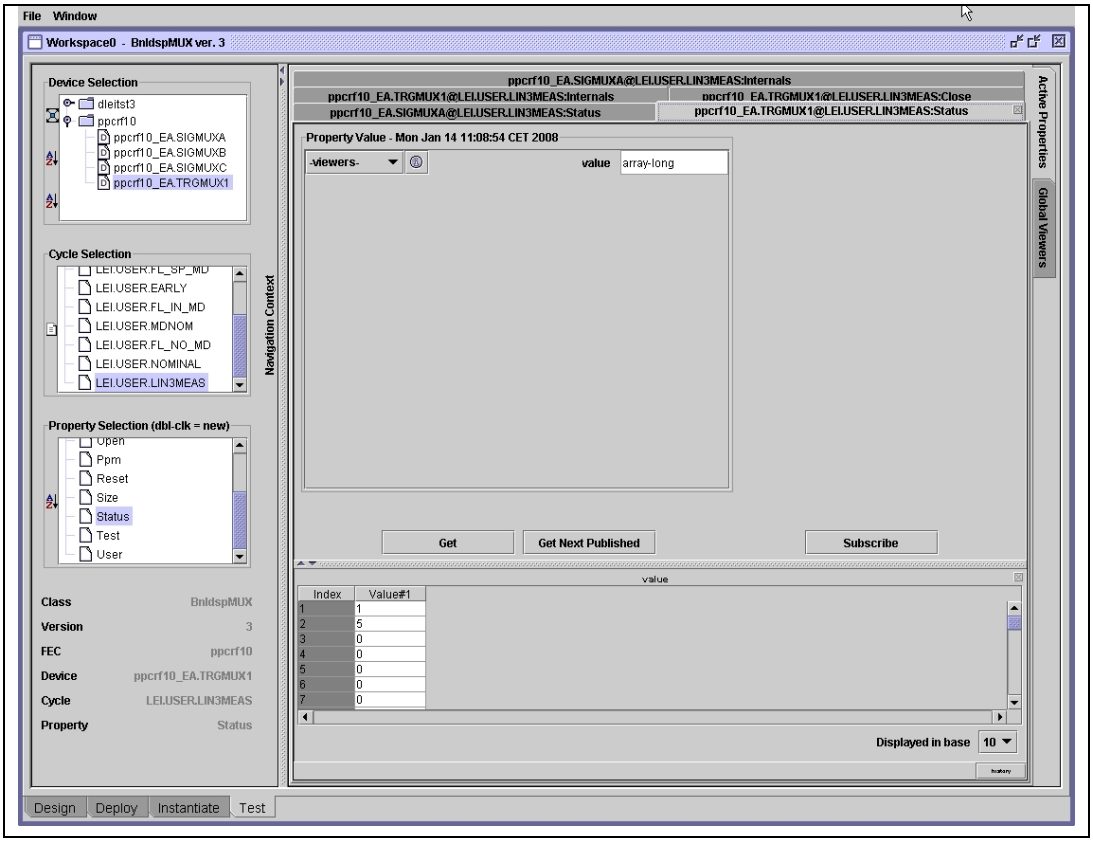

Figure 6 The trigger MUX output status.

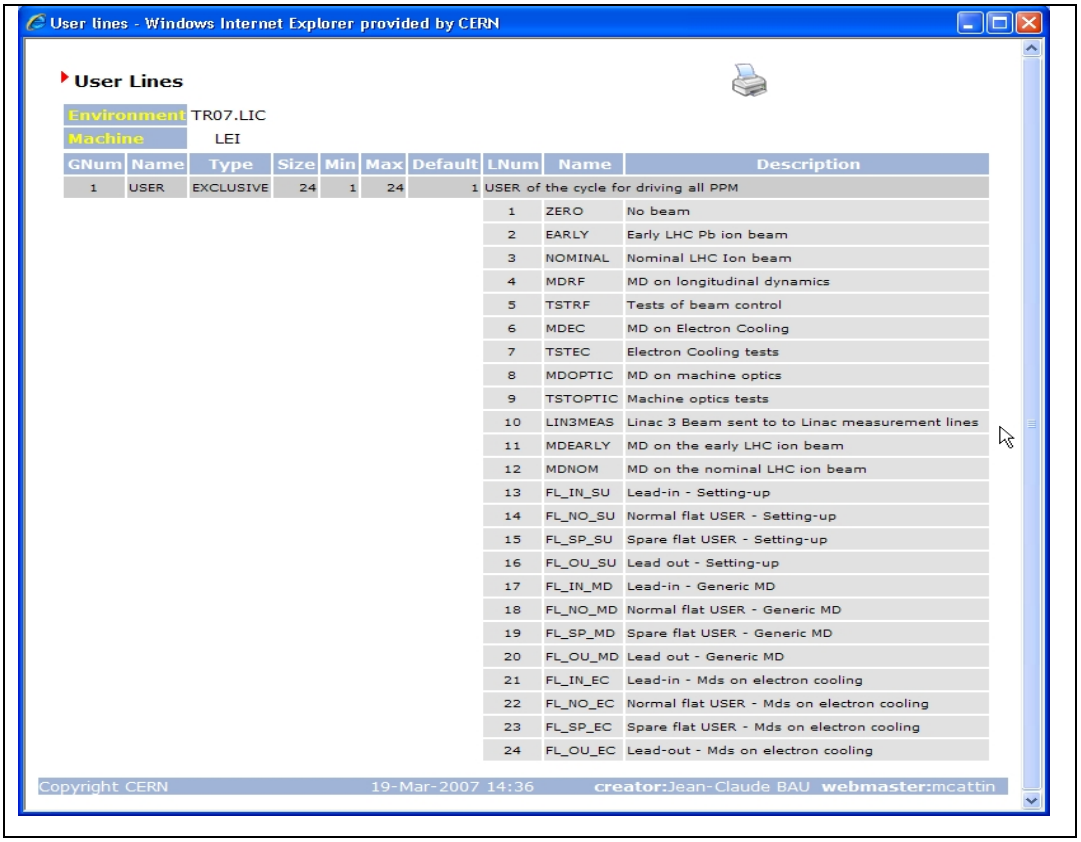

Figure 7 Trigger MUX PPM User bitmask value.

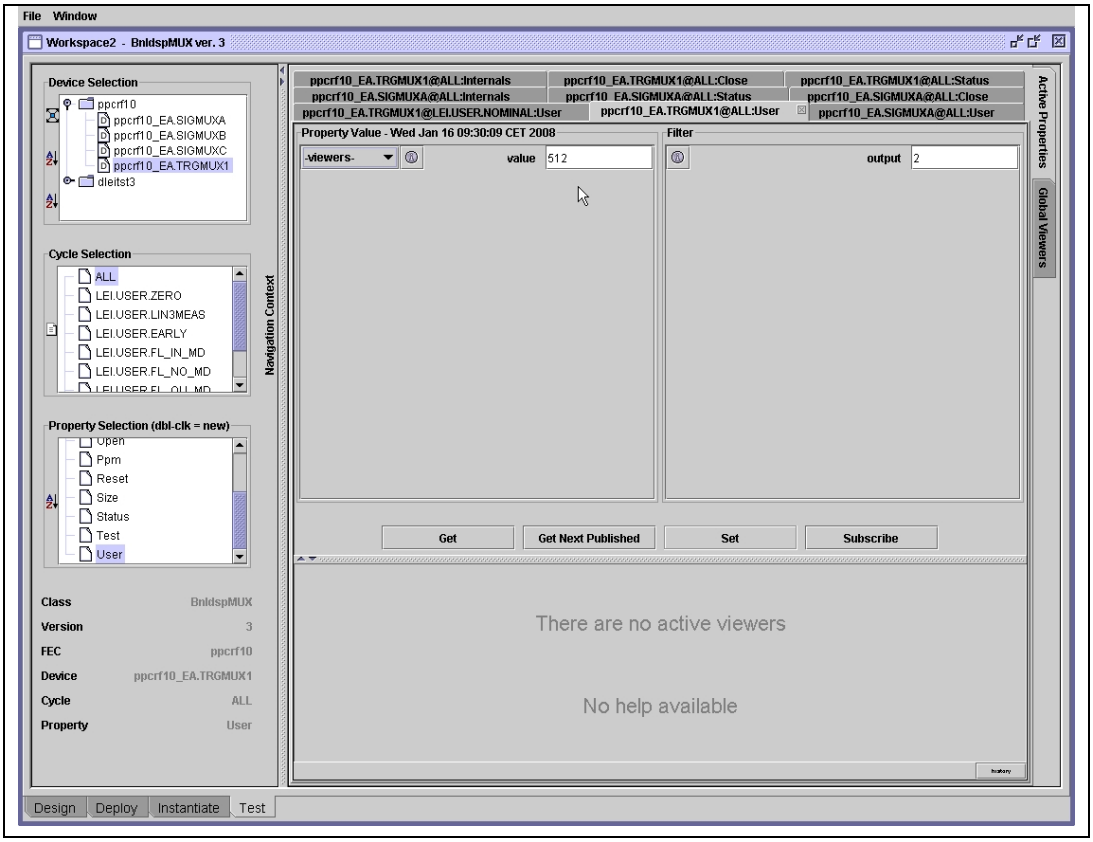

Figure 8 The trigger MUX PPM User setting.

Note that **only** for trigger MUXes the 'User' can be defined. Signal MUXes are non-PPM! If you try:

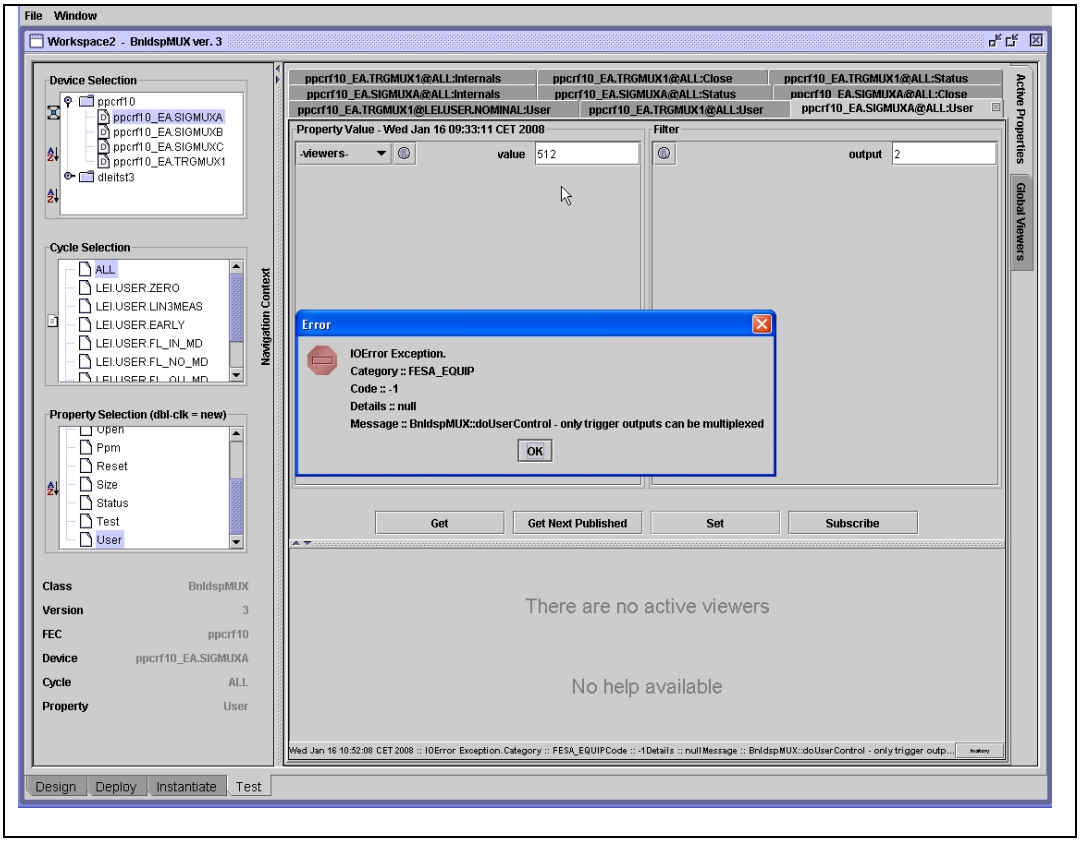

Figure 9 No PPM User setting for signal MUXes!

Setup of the oscilloscope.

Practically all basic scope parameters are now already set inside the scope's data store (Fesa fields) by the Fesa server properties of the signal and trigger instances of Fesa class BnldspMUX. What is left to be done for a basic measurement to be started is 'arming' the oscilloscope. This is done by setting its property 'Standby' to 'true' (Figure 10 Arming the SCOPE for measurement.).

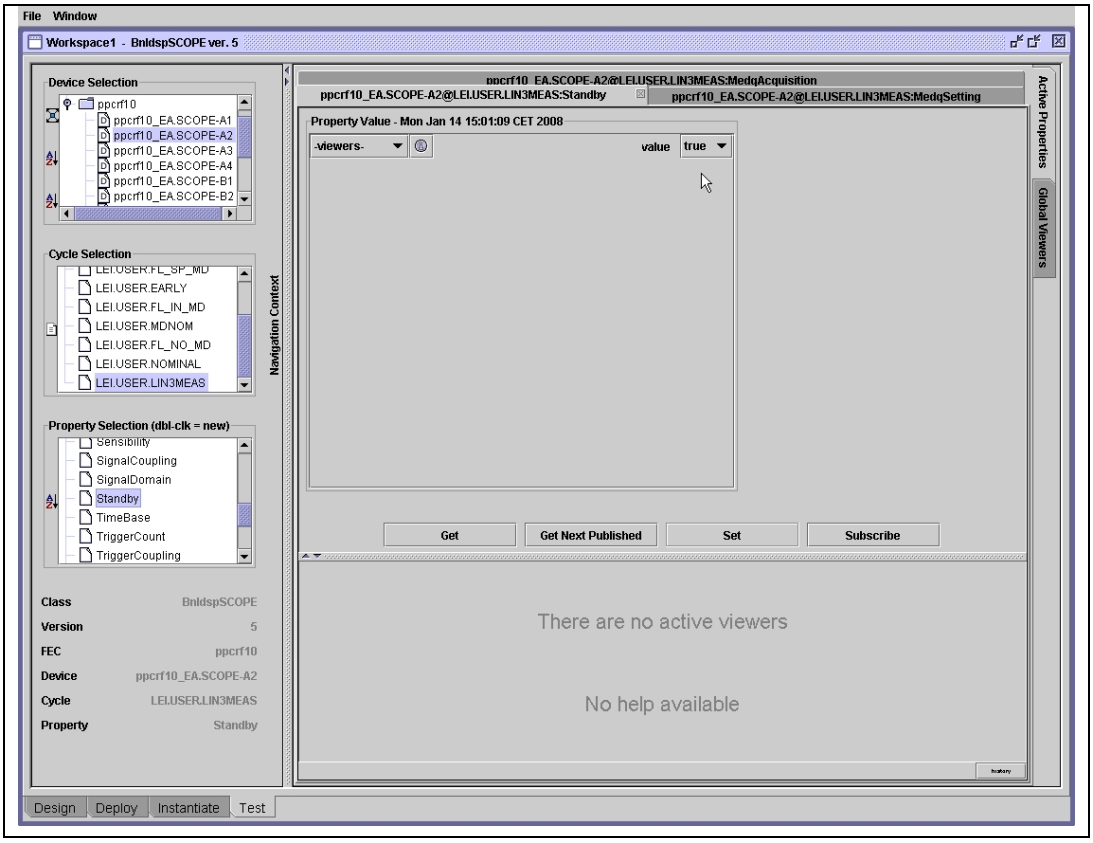

Figure 10 Arming the SCOPE for measurement.

With the basic control properties of 'Medq Settings' (Figure 11 Medq settings overview.) and 'Medq Acquisition' (Figure 12 Medq data acquisition.) it is now possible to see the result of setting, of the acquisition status and acquired data.

In 'Medq Settings' we find:
The acqFieldsUpdateDecrementer is set to -1 meaning the data buffer is continuously, every PPM cycle, refreshed with acquired data. The signal is set to 3 which is the ordinal value of DAQ_FGFREVCORR in the DaqSigsSelectDefs_t enumeration (file: bnldspLU.h). It corresponds to the chosen signal of the 'Input signal selection.' Chapter, namely: ppcrf10_EA.FGFREVCORR-DS (normally the 'signal index' - 1). The samplingTime of 1 is the scope time base multiplier of the Bnldsp board's internal sampling clock of 12.5 microSec.

In 'Medq Acquisition' we find:
The cycleName and acqStamp give information gathered during the RT access to the hardware; acqStamp is in mSec from 'start of cycle'; signal and samplingTime is as for (repeated from) 'Medq Settings'. The value of actSamples gives the total number of valid acquired samples in the buffer (either INTEGERs or FLOATs), the daqStatus of: 'SDAQ_STATUS_DONE' means that the acquisition successfully ended and the dataType of 'SDAQ_DATATYPE_FLOAT' (in this case) tells us that the acquisition was stored by the software in the dataBufFloats array.
Note that selecting 'ALL' for 'Cycle Selection' and then 'Subscribe' shows all currently active cycles in Real Time handled by BnldspSCOPE: the field 'actSamples' gives the actually acquired samples from the hardware!

More explicit information on Medq can be found in the 'documentation' of the BnldspMEDQ Fesa class.

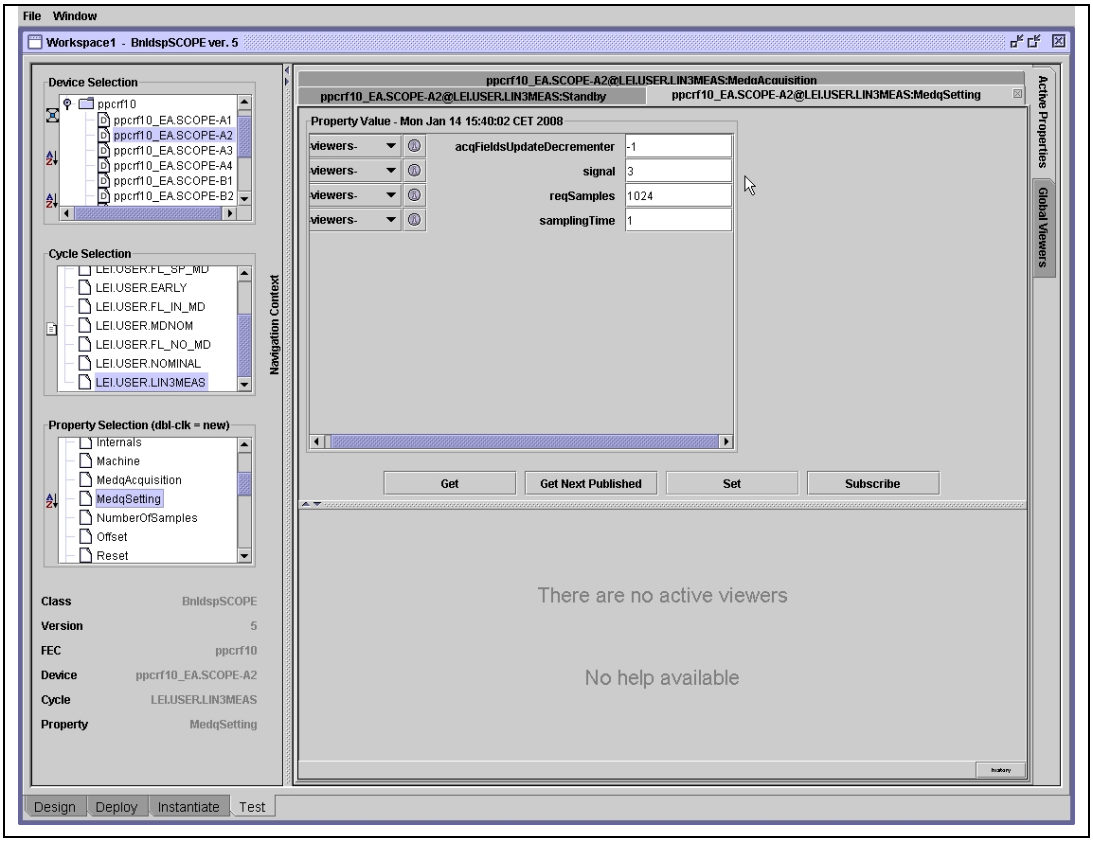

Figure 11 Medq settings overview.

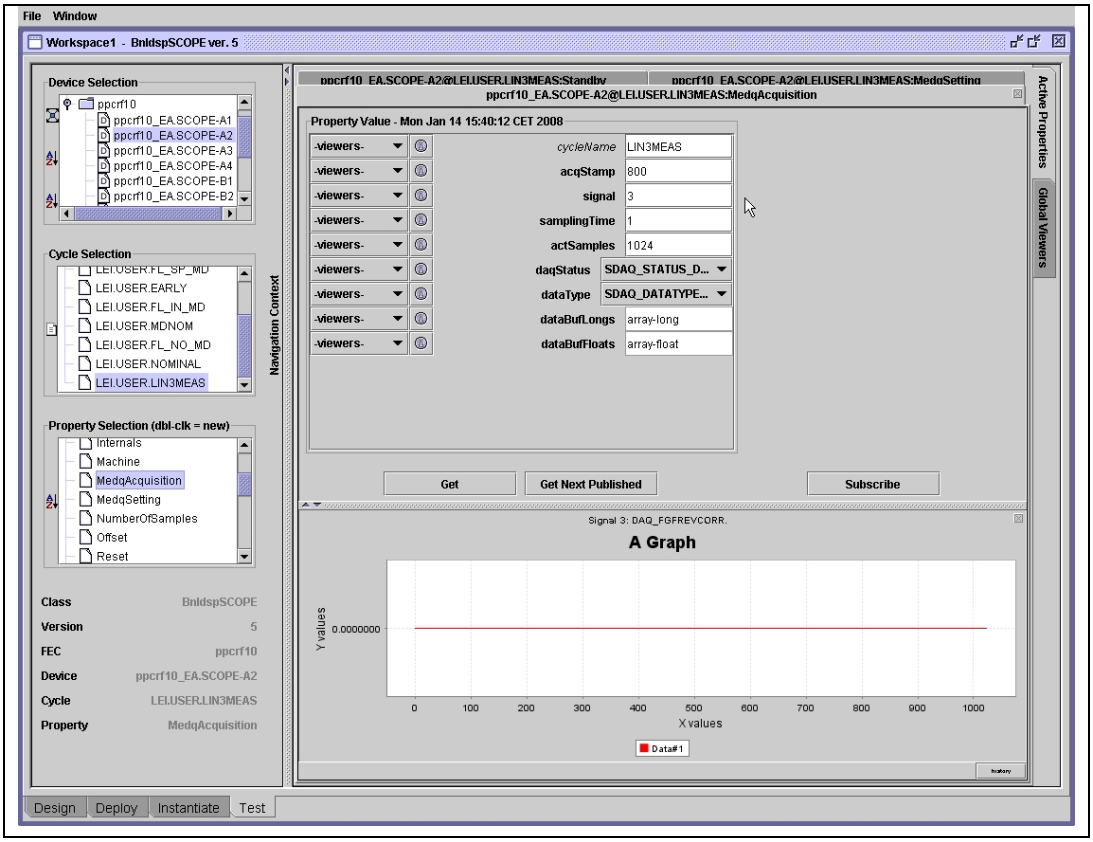

Figure 12 Medq data acquisition.

From within OASIS, of course, things like fine trigger delay settings or time base changes or vertical offset changes can (and will) also be done by the use of other BnldspSCOPE properties. This is however not further discussed in this note.

The class quartet's Fesa design.

The 'close collaboration' of the 4 Fesa classes is implemented by so called 'equipment links'. In our case this means that the Fesa class BnldspMUX will directly access the data store (Fesa fields), for read and for write, of the other 3 classes for getting or setting relevant operational parameters. Another access method is also defined in the Fesa paradigm: that of access via another class's server properties. This access method is also used by BnldspMUX (as it is by BnldspSCOPE, BTW).

In the Fesa class design for BnldspMUX we find therefore a declaration of 'use-friend' that illustrates the 'direct access to the data store' and we also find 'use-interface' for the access to other class's server properties. The classes that permit to BnldspMUX this direct data store access, document this fact by declaring 'used-by-friend' in their design. Special care has to be taken to build a correct software development directory tree to make the collaboration (and its software development) work. More info about the equipment link mechanism in the 'Equipment links manual' at:
http://project-fesa.web.cern.ch/project-fesa/development/notes.htm

The close collaboration is furthermore implemented by putting together all BnldspSIGNAL, BnldspTRIG, BnldspMUX and BnldspSCOPE property servers into one single Fesa shared server. The only Real Time executable is the one for the BnldspSCOPE class, the Fesa class that accesses the hardware, all the others have none. See the 'Equipment links manual' for the details of this type of implementation.

There are 2 batch scripts available for the compilation and the creation of a correct development directory structure, notably for the various symbolic links to directories in the 'friend' trees, required for compilation of the programme code for the collaboration.
These scripts, `remake_fesaQuartet_all.bat` and `remake_fesaQuartet.bat`, form an extension to the code storage in the CVS repository; the first one must be run after the creation of a new CVS development sandbox in order to have the required symbolic links available because CVS does not cater for archiving of these. More on the usage of the scripts in the 'The 'make recipe'.' chapter.

Also 2 scripts, `startOasis.bat` and `stopOasis.bat`, for orderly starting and stopping and, in the latter case, for unlinking of shared memory segments at the same time, of all the 4 classes of the collaboration are available in the …`/BnldspSCOPE/<version>/TEST/<FEC>` directories.

The ‘make recipe’.

The 4 Fesa classes, that form the OASIS Quartet follow, in principle, each individually the procedure for executable code production as described in the note ‘The generation of Bnldsp C object code modules and libraries for the LEIR Fesa BnldspXXX device classes.’ This note is stored in the `BNLDSP/DOCUMENTATION` directory which can be found in the CERN CVS file repository, currently at:
http://isscvs.cern.ch/cgi-bin/viewcvs-all.cgi/?root=abrfcs
(and also available on Erik’s home page at:
http://bracke.home.cern.ch/bracke/HTML/LEIR_Development/LEIR_Development.htm).

Keep in mind however, that **only** the Fesa class BnldspSCOPE does Real Time access to the VME hardware, so this is at the moment (2007) the only class that has C code modules integrated into it.

But, due to the ‘close collaboration’ as implemented by the equipment links, some additional directories need to be added to the development trees of the Fesa class CVS sandboxes before successful compilation can be obtained. Also the order in which compilation of the 4 Fesa classes is done has some importance. Another point would be that symbolic links cannot be restored by CVS and thus requires to be created ‘by hand’. The official recipe of doing all this is the ‘Equipment links manual’ to be found at:
http://project-fesa.web.cern.ch/project-fesa/development/notes.htm

To enhance the development and to document our particular implementation of the ‘close collaboration’ for the OASIS Quartet, 2 scripts have been written which formalize these requirements:

- `remake_fesaQuartet_all.bat`
  Apart from a complete re-compilation of all executable code, this script also first (re-)creates in each of the Fesa classes of the OASIS Quartet all required symbolic links to the locally delivered source files of the other collaborating Fesa classes. Use the script with only the obligatory first argument for the purpose of symbolic link creation; it will yet do a complete compilation afterwards that allows checking whether the links / development CVS sandbox is correctly set-up. It also can, optionally, synchronize your Fesa CVS development sandbox with the Fesa CVS repository and furthermore do a delivery to the operational code repository. With all the command line arguments:
  ‘`<absolute_path_to_the_FESA/BNLDSP_project_directory> SYNC DELIVER`’
  the script will do, in order, the following in the Quartet’s directory trees:
  - Create symbolic links
  - `Fesa Synchronize`
  - `gmake clean all localDeliver`
  - In the `TEST` directory of BnldspSCOPE only: `gmake shared`
  - `Fesa Deliver`
  - In all the `TEST` directories: `gmake rt server`

- `remake_fesaQuartet.bat`
  This script is created to only do the complete re-compilation of all executable code, assuming the symbolic links are in place. Optionally however, it allows doing a synchronization of the CVS development sandbox with the Fesa CVS repository before compilation is achieved. With all the command line arguments:
  ‘`<absolute_path_to_the_FESA/BNLDSP_project_directory> SYNC`’
  the script will do, in order, the following in the Quartet’s directory trees:
  - `Fesa Synchronize`
  - `gmake clean all localDeliver`
  - In all the `TEST` directories: `gmake rt server`
  - In the `TEST` directory of BnldspSCOPE only: `gmake shared`

Because the Fesa CVS repository does not cater for the concept of ‘Projects’ that span several Fesa classes, these 2 scripts are stored in the same `BNLDSP` root module of the aforementioned CVS repository as for the ‘The generation of Bnldsp C object code modules for the LEIR Fesa BnldspXXX device classes’ document.

___