XXIV International Baldin Seminar on High Energy Physics Problems “Relativistic Nuclear Physics and Quantum Chromodynamics” ISHEPP–2018
September 17–22, Dubna, Russia

# VME–based DAQ system for the Deuteron Spin Structure setup at the Nuclotron internal target station

**A.Yu.Isupov**

## Markup

Lets note the following highlighting conventions:

• file and software package names are represented by *italic text*,
• C and other languages constructions — by typewriter text,
• reference to the online manual page named “qwerty” in the 9th section is printed as ***qwerty (9)***.
• Subjects of substitution by actual values are enclosed in the angle brackets, while optional parameters are given in the square brackets: [-D<flag>] .

All mentioned trademarks are properties of their respective owners.

# Contents

## Introduction

The new powerful VME–based data acquisition (DAQ) system has been designed for the Deuteron Spin Structure (DSS) setup
[1. V.P. Ladygin et al. Few-body Studies at Nuclotron-JINR. Few Body Syst., **55**, 709–712, (2014)],
[2. P. K. Kurilkin et al. The 270 MeV deuteron beam polarimeter at the Nuclotron Internal Target Station. Nucl.Instr.and Meth.in Phys.Res., **A**(642), 45–51, (2011)],
[3. Yu.V. Gurchin et al. Detection equipment for investigating dp elastic scattering at internal target of Nuclotron in the framework of DSS project. Phys.Part.Nucl.Lett., **8**, 950–958, (2011)],
[4. S.M. Piyadin et al. ΔE-E detector for proton registration in nonmesonic deuteron breakup at the Nuclotron internal target. Phys.Part.Nucl.Lett., **8**, 107–113, (2011)]
placed at the Nuclotron Internal Target Station (ITS)
[5. A. Yu. Isupov, V. A. Krasnov, V. P. Ladygin, S. M. Piyadin, and S. G. Reznikov. The Nuclotron Internal Target Control and Data Acquisition System. Nucl. Instr. and Meth. in Phys. Res., **A**(698), 127–134, (2013)]
studies the polarization observables in the $dp$ elastic scattering for a several years
[6. P. K. Kurilkin et al. Measurements of the vector and tensor analyzing powers for $dp$-elastic scattering at 880 MeV. Phys.Lett.B, **B**(715), 61–65, (2012)],
[7. V. P. Ladygin et al. First results on the energy scan of the vector $A_y$ and tensor $A_{yy}$ and $A_{xx}$ analyzing powers in deuteron–proton elastic scattering at Nuclotron. Journal of Physics: Conference Series, **938**(012007), 1–6, (2017)],
[8. V. P. Ladygin, Yu. V. Gurchin, A. Yu. Isupov, et al. First results on the measurements of the proton beam polarization at the internal target at Nuclotron. Journal of Physics: Conference Series, **938**(012008), 1–4, (2017)],

[9. M. Janek, V. P. Ladygin, Yu. V. Gurchin, A. Yu. Isupov, et al. Dp breakup reaction investigation using polarized and unpolarized deuteron beam. Journal of Physics: Conference Series, **938**(012005), 1–6, (2017)].

Vector and tensor polarizations of the deuteron and proton beams from the LHEP–JINR polarized ions source (SPI)

[10. V. V. Fimushkin et al. Development of polarized ion source for the JINR accelerator complex. Journal of Physics: Conference Series, **678**(1), 012058–012062, (2016)],

as well as $A_y$, $A_{yy}$, $A_{xx}$ analyzing powers of the reactions have been successfully measured in the beam energy range from 135 MeV/nucleon to 900 MeV/nucleon. The DAQ system which will be described in my talk is built using the *netgraph*–based data acquisition and processing framework *ngdp*

[11. A. Yu. Isupov. The *ngdp* framework for data acquisition systems. arXiv:1004.4474 [physics.ins-det], (2010)],

[12. A. Yu. Isupov. CAMAC subsystem and user context utilities in *ngdp* framework. arXiv:1004.4482 [physics.ins-det], (2010)].

The software dealing with VME hardware is a set of *netgraph* nodes in form of the loadable kernel modules, so works in the operating system kernel context. The user context utilities implement Graphical User Interfaces (GUI) for interaction with human operator as well as data representation converters both are based on C++ classes derived from *ROOT* framework

[13. R. Brun and F. Rademakers. ROOT – An Object Oriented Data Analysis Framework. In *Proc. of the AIHENP'96 Workshop*, volume **A**(389) of Nucl.Instr.and Meth.in Phys.Res. (1997), pages 81–86, Lausanne, Switzerland, (1996). See also `http://root.cern.ch/`]

ones.

## The simplified *netgraph* graph to deal with VME hardware and manage the produced data streams

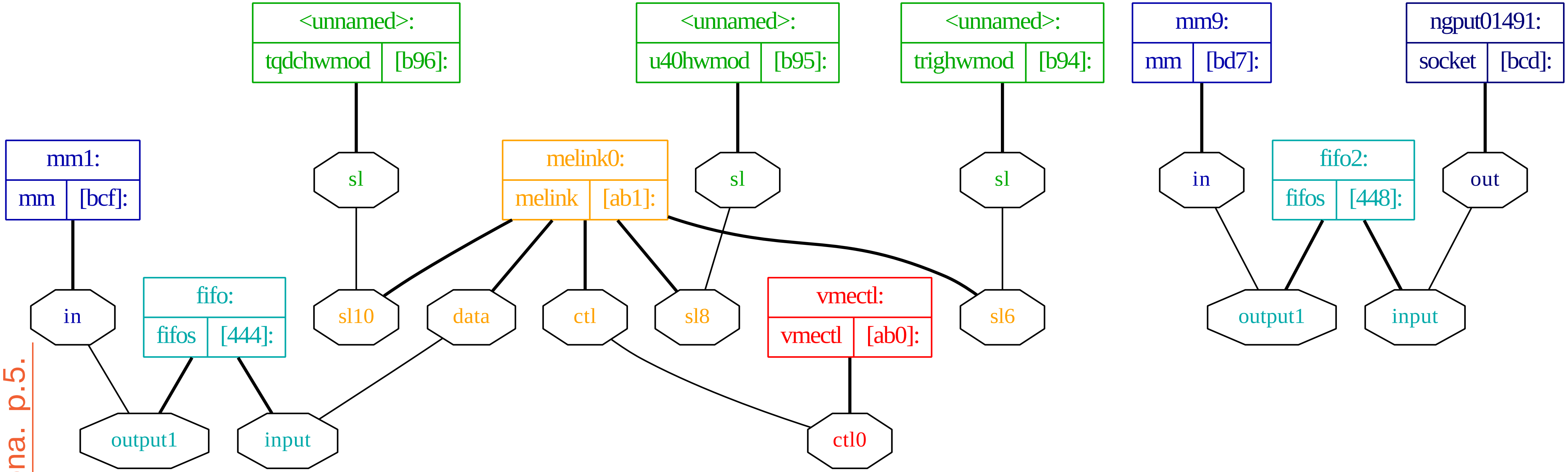



Fig.1.

The heart of DSS DAQ system — *ngdp* graph — we can see in Fig.1. The *ngdp* graph works in the operating system (OS) kernel context. It consists of nodes (rectangles) interconnected by the graph edges produced by pairs of so called hooks (octagons). The data messages flow along the edges. The nodes are implemented in form of the loadable kernel modules (KLD). Code execution in the kernel context allows us to handle VME data in the fastest possible way [11]. The VME hardware handling scheme: each VME master type is represented by corresponding *netgraph* node type (VME driver), while each VME hardware module type — by corresponding handler (VME node). In Fig.1 the VME driver is `melink0`, the VME nodes (only 3 instances for simplicity) are `trighwmod`, `u40hwmod`, and `tqdchwmod`. Interconnections in the DAQ graph are following: VME drivers are connected to special control node ***ng_vmectl(4)***, while VME nodes — to VME driver(s).

**Full scheme of the data flows in the DSS DAQ system**

$\nearrow$ `ng_mm` $\not\rightarrow$ *ngget* | *writer*

`ng_melink melink0:` $\rightarrow$ `ng_fifos fifo:` $\rightarrow$ `ng_mm mm1:` $\not\rightarrow$ *ngget* | *b2r* | *ngput* $\not\rightarrow$ `ng_socket` $\rightarrow$

$\searrow$ `ng_mm` $\not\rightarrow$ *ngget* | *b2r_dump* (EoB counts)

$\nearrow$ `ng_mm mm9:` $\not\rightarrow$ *ngget* |*r2h* $- - - \rightarrow$ *histGUI*

$\rightarrow$ `ng_fifos fifo2:` $\quad\quad$ \ $- - - \rightarrow$ *histGUI*

$\searrow$ `ng_mm` $\not\rightarrow$ *ngget* |*r2h_calc* (polar. calc.) $\cdots$

Fig.2, where:

- $\rightarrow$ ***packet (9 )*** [14. K. I. Gritsaj and A. Yu. Isupov. A Trial of Distributed Portable Data Acquisition and Processing System Implementation: the *qdpb* – Data Processing with Branchpoints. JINR Communications, **E10–2001–116**, 1–19, (2001)] data stream in the kernel context (by *netgraph* data messages).
- $\not\rightarrow$ context boundary crossing by the data stream.
- | ***packet (3 )*** [14] data stream in the user context (by so called pipe – standard UNIX IPC method).
- $- \rightarrow$ socket connection (here `TServerSocket` / `TSocket` from *ROOT* framework [13]).
- $\cdots$ number of the same elements.

The `melink0` VME driver node is the source of data packets stream. This stream is supplied to the `fifo` node [12]. The `fifo` provides own copy of the obtained data stream for each of its consumers. In our case there are ***writer (1 )***, ***b2r (1 )*** for *ROOT* events production [12], and ***b2r (1 )*** for EoB counts viewing. The ***b2r (1 )*** directs *ROOT* events to `fifo2` node. The `fifo2` feeds: the ***r2h (1 )*** histogramming server [12] and ***r2h (1 )*** for polarization calculations. The ***b2r (1 )*** and ***r2h (1 )*** are user context utilities because C++ code and *ROOT* libraries could not be linked into OS kernel. So data streams cross the context boundaries. Nevertheless the fifo implementation in the kernel context is more reliable and provides higher performance then in the user context.

## *ng_vmectl(4)*: VME control node

High level control over VME DAQ graph is performed by special *ng_vmectl* control node type, which intended to issue commands to VME drivers and VME nodes. The commands have form of *netgraph* control messages which are transferred between nodes directly. Instantiation of this node (which is singleton) leads to building of the graph like pictured in Fig.1.

As a first step the *ng_vmectl* should be `mkpeer`ed (using `creat` or `cfg` hooks). During such procedure the *ng_vmectl* loads KLD modules of all known VME driver types (if not yet loaded). Latters perform own `probe()` and `attach()` methods like usual UNIX hardware drivers do. If corresponding VME master hardware exists, required number of device units (and *netgraph* nodes) is instantiated and each `connect`s own `ctl` hook to `ctl<M>` hook of *ng_vmectl*. After that *ng_vmectl* can issue explicit `LoadHwMod <hwmod_type> <N>` control message to corresponding VME driver(s) for each required `<hwmod_type>` and `<N>` pairs. By explicit command the addressed VME driver loads (if not yet loaded) the KLD of the mentioned `<hwmod_type>` VME nodes, `mkpeer`s and `connect`s node of `<hwmod_type>` by its `sl` hook to own `sl<N>` hook. Also so called autoprobe could be used, if it is supported by hardware. It has the following forms:

`LoadHwMod auto` – all slots are involved,
`LoadHwMod auto N` – N$^{th}$ slot only,
`LoadHwMod auto N n` – `n` slots from N$^{th}$,
`LoadHwMod auto N-M` – slots from N$^{th}$ up to M$^{th}$ inclusively,
`LoadHwMod auto N,M[,K[...]]` – only enumerated slots.

At first step of the autoprobe procedure the special ***ng_genhwmod(4)*** node type is `mkpeer`ed and `connect`ed by its `sl` hook to the `tmp` hook of VME driver. After that VME driver issues to it the `AutoProbe` specific control message, and ***ng_genhwmod(4)*** makes a rest of job.

## *ng_melink (4)*: driver for M–Link VME master / PCI–E adapter card

The ***ng_melink (4)*** currently able to:

- perform basic duties of any VME driver implemented by *ng_vmedrv.c* , in particular, it contains interface to handle ***queue (3)*** of requests, main function of ***kthread (9)*** kernel threads, KLD method `modevent()`, generic parts of UNIX driver methods (`probe()`, `attach()`, `detach()`) and *netgraph* methods;
- support realistic specific interfaces: interrupt handler (`melink_hand()`), thread helpers `dma_done()`, `send_resp()`, hardware frames handling;
- support specific parts of UNIX driver methods (`attach()`, `detach()`, `release_resources()`) and *netgraph* methods (`rcvmsg()`, `rcvdata()`);
- obtain commands as *netgraph* control messages, translate commands into requests and store ones into STAILQ ***queue (3)***;
- handle requests STAILQ by $0^{th}$ *ng_melink* kernel thread (see ***kthread (9)***);
- handle the full set of VME node's responses;
- process data (decode from frames, assemble events, encapsulate their into packets (see [14]), and buffer latters for optimal transportation) by the number of ***kthread (9)***s executed in parallel;
- deal with M–Link VME master hardware (could be compiled for FVME [15. `http://afi.jinr.ru/FVME`] or FVME2 [16. `http://afi.jinr.ru/FVME2`] hardware variant) through PCI–E adapter card — send hardware frames, perform realistic `drv_init()`, `drv_reset()`, `run_mode_change()`, `start_run()`, `stop_run()` — as well as proceed master's responses;
- proceed hardware frame arrival, EoB "interrupt" events, and DMA completeness using the interrupt handler `melink_hand()`.

## VME nodes

Currently we support:

• the trigger TTCM/FVME2TMWR modules [17. http://afi.jinr.ru/TTCM], [18. http://afi.jinr.ru/action/show/FVME2TM] by the ***ng_trighwmod(4)*** node type;

• the trigger U40 modules [19. http://afi.jinr.ru/action/show/U40VE] by the ***ng_u40hwmod(4)*** node type;

• the TQDC16 time– and charge–to–digital converter modules [20. http://afi.jinr.ru/TQDC-16] by the ***ng_tqdchwmod(4)*** node type.

These node types currently are able to:

- perform basic duties of any VME node implemented by *ng_vmehwmod.c* , in particular, these nodes understand `clrstats`, `getstats` and `getclrstats` control messages, contain context handling interface, `send_req()`, and generic parts of `eob_func()`, `analyse_context()` and *netgraph* methods;
- support realistic `probe()` and `init()`, `fix_acc()`, and specific parts of `eob_func()`, `analyse_context()` and *netgraph* methods (`ctor()`, `dtor()`, `rcvmsg()`).
- provide access to corresponding specific VME module hardware.

The VME nodes support the one `sl` hook to connect to VME driver's `sl<N>` hook, and `opt` hook to be able to form chain of nodes. Such approach should mimic OOP derived classes to separate more and less specific code and avoid code duplication. Less specific module is connected to VME driver, and specialization is increased along chain. When less specific module obtains control messages with unknown command, it forwards one through own `opt` hook, if any, or discards otherwise.

The VME node shuts down upon receipt of a `NGM_SHUTDOWN` control message, or when all hooks have been disconnected.

## User context utilities

The *ngdp* generic software utilities used by the DSS DAQ system are:

- ***writer (1)*** [14] of packet stream to files on HDD,
- ***ngget (1)*** packet stream extractor from *netgraph* graph,
- ***ngput (1)*** packet stream injector to *netgraph* graph,
- ***b2r (1)*** (binary–to–*ROOT*) converter,
- ***r2h (1)*** (*ROOT*–to–histograms) converter, and
- ***histGUI (1)*** standalone client for ***r2h (1)*** (see [12]).

According to scheme of events representation by *ROOT* classes [12] the specific classes for trigger, Begin–of–Burst, End–of–Burst events as well as required helper classes are designed. These classes are linked into ***b2r (1)*** and ***r2h (1)*** utilities as well as provided in the *libElinpol.so* standalone shared library form to be used by the express–offline *ROOT* scripts and offline processing utilities.

The DSS DAQ specific software utilities are:

- special form of ***b2r (1)*** to online view the EoB event counts,
- special form of ***r2h (1)*** for online polarization calculations, and
- ***sv (1)*** DAQ supervisor utility.

## *b2r(1)* converter (abbreviation of "binary–to–*ROOT* (events representation)")

```
b2r -O [-m{<mmname>|-}] [-d] [-f{<outfile>|-}] [-r{<outrate>|-} [-v]]
  [-s{<filesize>|-}] [-S{<splitlevel>|-}] [-p{<pidfile>|-<template>XXXXX}]
  [-c{<calibfile>|-} [-B] [-n{<hwmodtype>|-}[ ...]]] [-C{<modcfgfile>|-}]
  [-D{<flag>|-}] [-x{<downscale>|-}] [-z{<compression>|-}]
```

The *b2r* reads raw data packets from VME driver and for each of them produces *ROOT* event representation in form of some compiled-in *ROOT* class. This *ROOT* representation is serialized using `TBufferFile` and encapsulated into packet of other type with the same number.

## *b2r_dump* EoB events viewer

```
b2r_dump -O -c<calibfile> -B -nTQDC -C<modcfgfile> -D0x13
```


The EoB events viewer (so called dumper *b2r_dump*) is a specially compiled form of the ***b2r(1)*** converter. The purpose flag `-D0x13` requires to produce textual output instead of *ROOT* events representation. The ***ng_fifos(4)*** feeds dumper by the End–of–Burst events only. In Fig.3 we can see screenshot of the dumper's output for some EoB event.

Fig.3. Textual EoB output of the ***b2r(1)*** compiled as dumper.

### Histogramming server *r2h(1)* (abbreviation of “*ROOT*–to–histograms (representation converter)”)

```
r2h [-d] [-c{<cfgfile>|-}] [-f{<outfile>|-} [-s{<saverate>|-}]]
  [-p{<pidfile>|-<template>XXXXX}] [-r{<outrate>|-} [-v]] [-a<addr>[ ...]]
  [-A<addr>[ ...]] [{-x|-X}{<downscale>|-}] {[-I [-P]]|[<peerhost> [<peerport>]]}
```

The *r2h* reads the data packets with events in the *ROOT* representation produced by ***b2r(1)***, extracts their (by deserialization using `TBufferFile`), fills some histograms configured by `<cfgfile>` in the ***r2h.conf(5)*** format [12], sends requested histogram(s) to already registered client(s) by *ROOT* `TMessage`(s), and optionally writes all configured histograms to *ROOT* `TFile` `<outfile>` on HDD.

### The *r2h(1)*⟷*histGUI(1)* protocol *r2h.conf(5)*

The ***r2h.conf(5)*** protocol [12] allows us to configure the ***r2h(1)*** and ***histGUI(1)*** as well as to establish the conversation between them online. This means we need not to recompile ***r2h(1)*** each time we need to calculate yet another values and to book and fill yet another histograms. Instead we edit the configuration file and restart ***r2h(1)***, which anyway should be restarted at each run if we need to save histograms into per–run `<outfile>` files.

The part of configuration file used at polarization measurements we can see here. The Var objects declare involved raw data values in terms of VME hardware channel, module, and group numbers. Lets note 3 conditions (cpm_x) calculated by universal cells Cvar and used for separate filling histograms for 3 polarization modes (here x means p for "+", m for "−", and 0 for "0"), and 3 histograms (Book1d) of the TOF differences (tLD4RD4_x) of e.g. LD4 (tdc10) and RD4 (tdc26) counters. These histograms in some measurement run at March'2017 we can see in Fig.4. (The Book2d objects declare 2D histograms for QDC correlations.)

```
Var adc10 10 0 0
Var adc26 10 1 0
Var tdc10 10 0 1
Var tdc26 10 1 1
Var L30 37 0 3
Var L31 38 0 3
Var L32 39 0 3
Cvar cpm_0 type_UChar LINPOL_DAT_0 ((!L30) && (L32) && (L31))
Cvar cpm_p type_UChar LINPOL_DAT_0 ((!L31) && (L30) && (L32))
Cvar cpm_m type_UChar LINPOL_DAT_0 ((!L32) && (L30) && (L31))
Cvar t10_26 type_Float LINPOL_DAT_0 ((tdc10 && tdc26) \
   ? (tdc10 - tdc26 + 1024) : -1000)
Book1d tLD4RD4_0 TLD4RD4_0 t10_26 2048 0 2048 cpm_0
Book1d tLD4RD4_p TLD4RD4_p t10_26 2048 0 2048 cpm_p
Book1d tLD4RD4_m TLD4RD4_m t10_26 2048 0 2048 cpm_m
Book2d LD4RD4_0 Ld4Rd4_0 adc10 1024 1 65536 adc26 1024 1 65536 cpm_0
Book2d LD4RD4_p Ld4Rd4_p adc10 1024 1 65536 adc26 1024 1 65536 cpm_p
Book2d LD4RD4_m Ld4Rd4_m adc10 1024 1 65536 adc26 1024 1 65536 cpm_m
```

XXIV ISHEPP–2018, Sept.17–22, Dubna, Russia. p.14

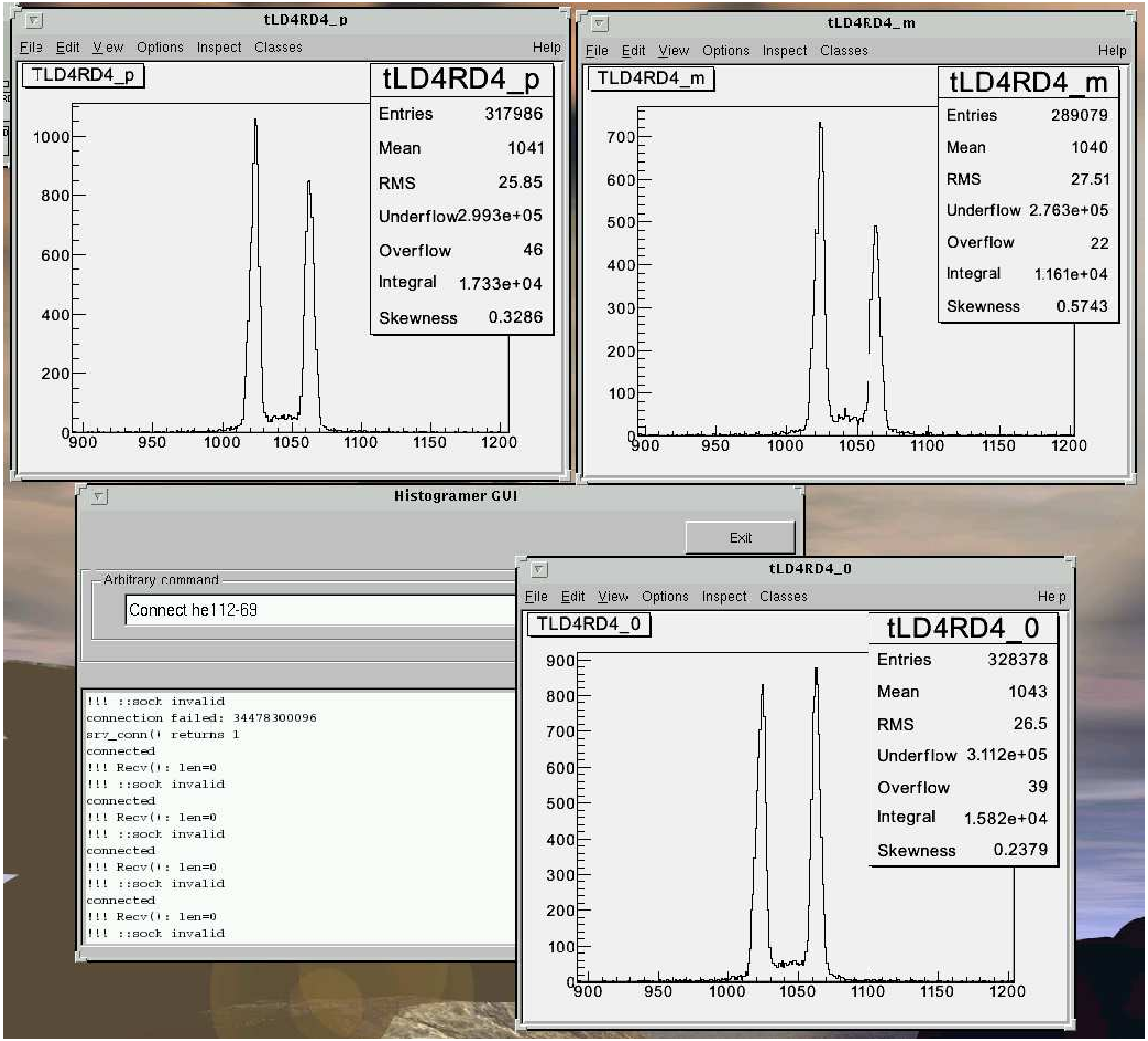



Fig.4. Screenshot of the ***histGUI(1)*** utility. We can see three histogram canvas (each produced by ‘`GetCont`’ command) and the main window.

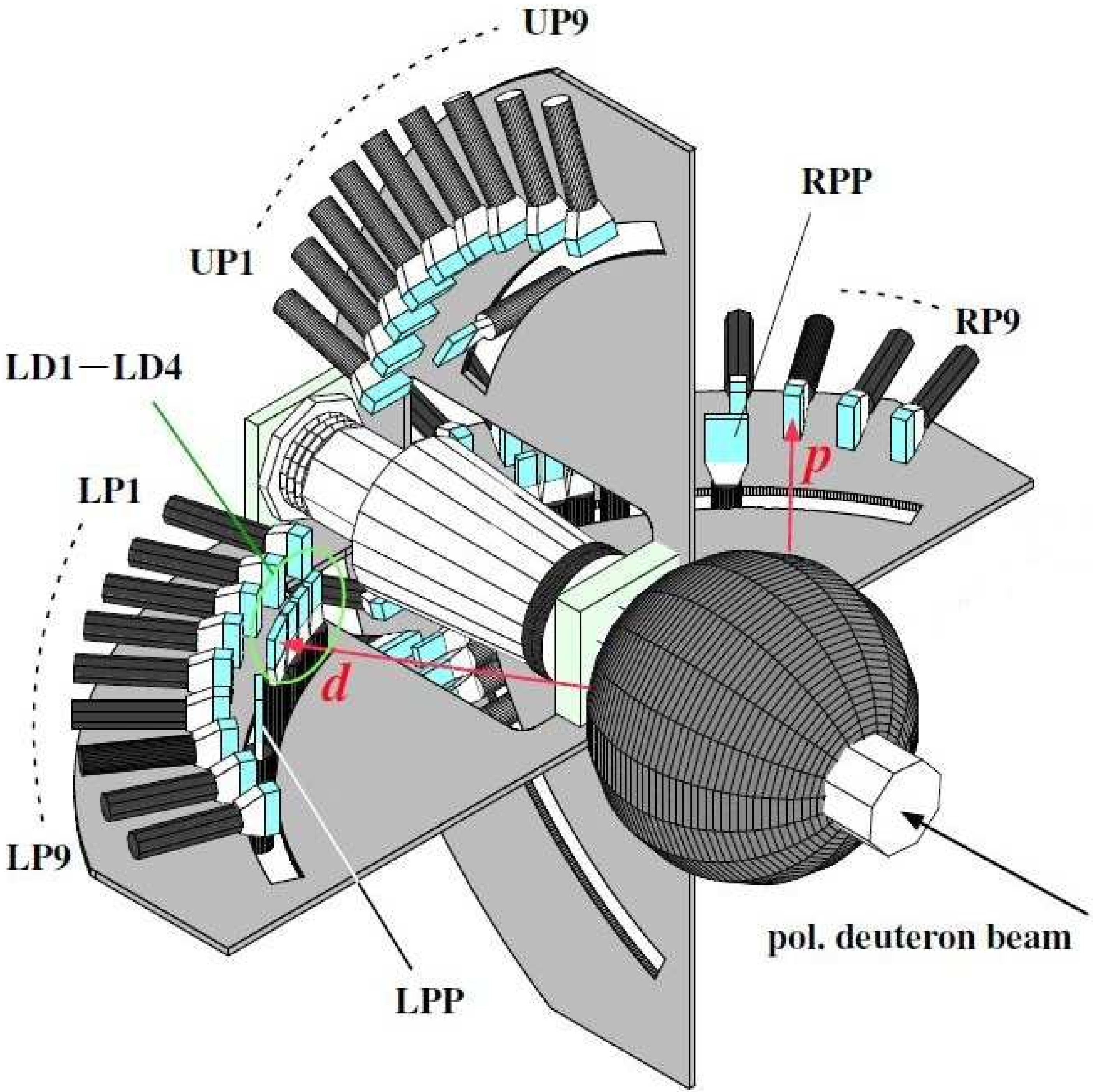
## Polarization calculator

The specially compiled (with `POLAR_CALC #define`'d) ***r2h(1)*** converter (so called *r2h_calc*) supports additional entities `Calcvp` and `Calctp` in the configuration file format ***r2h.conf(5)***. This allows it to calculate vector and tensor polarizations instead of *ROOT* histograms filling. Currently the polarization calculator produces textual output of calculation results at each EoB event arrival. In the future these results could be output in the HTML form to be included into Web–based representation scheme described in
[21. A. Yu. Isupov. Data acquisition systems for the high energy and Nuclotron internal target polarimeters with network access to polarization calculation results and raw data. Czech. J. Phys. Suppl., **A55**, A407–A414, (2005)]
and
[22. A. Yu. Isupov. Upgrade of the DAQ systems for the LHE polarimeters to support Vector–Tensor Polarimeter on the Nuclotron internal target. Czech. J. Phys. Suppl., **C56**, C385–C392, (2006)].

## Polarization calculations

The LHEP–JINR polarized ions source SPI [10] can produce beam with two polarization modes (so called "+" and "−"), as well as nonpolarized beam (so called "0" polarization mode). Each accelerator burst has one of polarization mentioned above, from burst to burst beam polarization changes, and corresponding polarization marks are distributed to polarimeters by SPI electronics.

XXIV ISHEPP–2018, Sept.17–22, Dubna, Russia. p.17




The DSS setup allows us to use coincidences of "proton" and "deuteron" scintillation counters in the complementary arms — LD-RP, RD-LP, UD-DP, DD-UP. Lets name these coincidences as Left, Right, Up, and Down arm scaler counts ($\boldsymbol{X}$). We have 9 (4) "proton" ("deuteron") counters in Left, Right, and Up arms, and 4 (1) – in Down arm (due to lack of space). So we can kinematically cover the 18°–60° angle range in the lab. system for wide range of the beam energies. For polarimetry we use counts of 4 or 3 coincidence pairs. These pairs could be chosen for each beam energy by kinematics and statistics conditions. The LPP–RPP coincidences of the dedicated counters for quasi–elastic $pp$ scattering at 90° in the c.m. are used as monitor ($\boldsymbol{M}$) counts.

Fig.5. The polarimetry part [6] of the DSS setup.

Using the following designations:

$\boldsymbol{L^{+,-,0}}$, $\boldsymbol{R^{+,-,0}}$, $\boldsymbol{U^{+,-,0}}$, $\boldsymbol{D^{+,-,0}}$ — polarimeter's Left, Right, Up, and Down arm scaler counts for some accelerator burst with "+", "−", "0" polarization mark, respectively;

$\boldsymbol{M^{+,-,0}}$ — polarimeter's beam monitor scaler counts for some accelerator burst with "+", "−", "0" polarization mark, respectively;

$\boldsymbol{A_y}$, $\boldsymbol{A_{yy}}$, $\boldsymbol{A_{xx}}$ — the vector and tensor analyzing powers are assumed known from the previous measurements;

we can calculate average (over burst numbers $\boldsymbol{n^{+,-,0}}$) values of mentioned counts ($\boldsymbol{X}$ could be $\boldsymbol{L}$, $\boldsymbol{R}$, $\boldsymbol{U}$, $\boldsymbol{D}$, $\boldsymbol{M}$):

$$\boldsymbol{x^{+,-,0}} := \left\langle \boldsymbol{X_n^{+,-,0}} \right\rangle = \frac{(\boldsymbol{X_1^{+,-,0}} + \boldsymbol{X_2^{+,-,0}} + \ldots + \boldsymbol{X_n^{+,-,0}})}{\boldsymbol{n^{+,-,0}}} = \frac{\boldsymbol{X_{\text{cum}}^{+,-,0}}}{\boldsymbol{n^{+,-,0}}} \, .$$

All the following calculations done in terms of such average count values $\boldsymbol{l^{+,-,0}}$, $\boldsymbol{r^{+,-,0}}$, $\boldsymbol{u^{+,-,0}}$, $\boldsymbol{d^{+,-,0}}$, $\boldsymbol{m^{+,-,0}}$ as well as burst numbers $\boldsymbol{n^{+,-,0}}$. Lets introduce also two ratii, where $\boldsymbol{x}$ means $\boldsymbol{l}$, $\boldsymbol{r}$, $\boldsymbol{u}$ or $\boldsymbol{d}$:

$$\boldsymbol{r_x^{+,-}} := \frac{\boldsymbol{x^{+,-}}}{\boldsymbol{x^0}} \, ,$$

$$\boldsymbol{r_m^{+,-}} := \frac{\boldsymbol{m^0}}{\boldsymbol{m^{+,-}}} \, .$$

XXIV ISHEPP–2018, Sept.17–22, Dubna, Russia. p.18

**Vector polarizations** $P_y^{+,-}$ and their statistic errors $\Delta P_y^{+,-}$ are calculated as follows:

$$P_y^{+,-} = \frac{(r_l^{+,-} - r_r^{+,-})r_m^{+,-}}{3A_y},$$

$$\Delta P_y^{+,-} = \sqrt{\frac{r_l^{+,-}}{l^0} + \frac{r_r^{+,-}}{r^0}} \cdot \frac{r_m^{+,-}}{3A_y\sqrt{n^{+,-}}}.$$

**Tensor polarizations** $P_{yy}^{+,-}$, $P_{xx}^{+,-}$ and their statistic errors $\Delta P_{yy}^{+,-}$, $\Delta P_{xx}^{+,-}$ are calculated as follows:

$$P_{yy}^{+,-} = \frac{((r_l^{+,-} + r_r^{+,-})r_m^{+,-} - 2)(A_{xx} + 2A_{yy}) - 2((r_u^{+,-} + r_d^{+,-})r_m^{+,-} - 2)(A_{yy} + 2A_{xx})}{2((A_{yy})^2 - (A_{xx})^2)},$$

$$\Delta P_{yy}^{+,-} = \left(\sqrt{\frac{r_l^{+,-}}{l^0} + \frac{r_r^{+,-}}{r^0}}(A_{xx} + 2A_{yy}) + 2\sqrt{\frac{r_u^{+,-}}{u^0} + \frac{r_d^{+,-}}{d^0}}(A_{yy} + 2A_{xx})\right)$$

$$\times \frac{r_m^{+,-}}{2((A_{yy})^2 - (A_{xx})^2)\sqrt{n^{+,-}}}.$$

$$P_{xx}^{+,-} = \frac{2((r_u^{+,-} + r_d^{+,-})r_m^{+,-} - 2)(A_{xx} + 2A_{yy}) - ((r_l^{+,-} + r_r^{+,-})r_m^{+,-} - 2)(A_{yy} + 2A_{xx})}{2((A_{yy})^2 - (A_{xx})^2)},$$

$$\Delta P_{xx}^{+,-} = \left(\sqrt{\frac{r_l^{+,-}}{l^0} + \frac{r_r^{+,-}}{r^0}}(A_{yy} + 2A_{xx}) + 2\sqrt{\frac{r_u^{+,-}}{u^0} + \frac{r_d^{+,-}}{d^0}}(A_{xx} + 2A_{yy})\right)$$

$$\times \frac{r_m^{+,-}}{2((A_{yy})^2 - (A_{xx})^2)\sqrt{n^{+,-}}}.$$

In **case of three counts** ($\boldsymbol{v}$ is $\boldsymbol{u}$ or $\boldsymbol{d}$) we use the following:

$$\boldsymbol{P_{yy}^{+,-} = \frac{((r_l^{+,-} + r_r^{+,-})r_m^{+,-} - 2)(A_{xx} + 2A_{yy}) - 2(r_v^{+,-}r_m^{+,-} - 1)(A_{yy} + 2A_{xx})}{2((A_{yy})^2 - (A_{xx})^2)}},$$

$$\boldsymbol{\Delta P_{yy}^{+,-} = \left(\sqrt{\frac{r_l^{+,-}}{l^0} + \frac{r_r^{+,-}}{r^0}}(|A_{xx}| + 2|A_{yy}|) + 2\sqrt{\frac{r_v^{+,-}}{v^0}}(|A_{yy}| + 2|A_{xx}|)\right)}$$

$$\boldsymbol{\times \frac{r_m^{+,-}}{2|((A_{yy})^2 - (A_{xx})^2)|\sqrt{n^{+,-}}}}\,.$$

$$\boldsymbol{P_{xx}^{+,-} = \frac{2(r_v^{+,-}r_m^{+,-} - 1)(A_{xx} + 2A_{yy}) - ((r_l^{+,-} + r_r^{+,-})r_m^{+,-} - 2)(A_{yy} + 2A_{xx})}{2((A_{yy})^2 - (A_{xx})^2)}},$$

$$\boldsymbol{\Delta P_{xx}^{+,-} = \left(\sqrt{\frac{r_l^{+,-}}{l^0} + \frac{r_r^{+,-}}{r^0}}(|A_{yy}| + 2|A_{xx}|) + 2\sqrt{\frac{r_v^{+,-}}{v^0}}(|A_{xx}| + 2|A_{yy}|)\right)}$$

$$\boldsymbol{\times \frac{r_m^{+,-}}{2|((A_{yy})^2 - (A_{xx})^2)|\sqrt{n^{+,-}}}}\,.$$

Above we assume that statistic errors for $\boldsymbol{M^{+,-,0}}$ are negligible small in comparison with ones for scalers $\boldsymbol{X^{+,-,0}}$.

XXIV ISHEPP–2018, Sept.17–22, Dubna, Russia. p.20

## Polarization calculation commands of *r2h.conf(5)* protocol

Above formulas are implemented by `Calcvp` and `Calctp` objects of the ***r2h.conf(5)*** protocol. For example, the following four `Calcvp` objects compute the **vector polarizations** $P_y^{+,-}$ and errors $\Delta P_y^{+,-}$ for 3-mode beam (two work with polarized bursts while other two – with nonpolarized ones):

```
Cvar py1m type_Double LINPOL_CYC_END py1m
Cvar dpy1m type_Double LINPOL_CYC_END dpy1m
Cvar py1p type_Double LINPOL_CYC_END py1p
Cvar dpy1p type_Double LINPOL_CYC_END dpy1p
Calcvp vp1m py1m LINPOL_CYC_END dpy1m -0.392 L1mn R1mn Mmn L10n R10n M0n Nbm cpm_m
Calcvp vp1p py1p LINPOL_CYC_END dpy1p -0.392 L1pn R1pn Mpn L10n R10n M0n Nbp cpm_p
Calcvp vp1m0 py1m LINPOL_CYC_END dpy1m -0.392 L1mn R1mn Mmn L10n R10n M0n Nbm cpm_0
Calcvp vp1p0 py1p LINPOL_CYC_END dpy1p -0.392 L1pn R1pn Mpn L10n R10n M0n Nbp cpm_0
```

(the `Cvar` objects above simply declare calculation cells to store $P_y^{+,-}$ and $\Delta P_y^{+,-}$ values.)

`Calcvp` and `Calctp` uses the 3 mode counts normalized by number of corresponding accelerator bursts. These counts are derived from raw data by so called universal calculation mechanism (***cell(3)***, see [12]). So we can organize polarization calculations in a very flexible way. Instead of recompilation of the ***r2h(1)*** we simply edit its configuration file. For example, at beam energy change we replace analyzing power values and possibly the combination of scintillation counters to be used. We can also apply 1D data cuts during counts preparation.

For example, the following four `Calctp` objects compute the **tensor polarizations** $\boldsymbol{P_{yy}^{+,-}}$, $\boldsymbol{P_{xx}^{+,-}}$ and errors $\boldsymbol{\Delta P_{yy}^{+,-}}$, $\boldsymbol{\Delta P_{xx}^{+,-}}$ for 3-mode beam (two work with polarized bursts while other two – with nonpolarized ones):

```
Cvar pyy1m type_Double LINPOL_CYC_END pyy1m
Cvar dpyy1m type_Double LINPOL_CYC_END dpyy1m
Cvar pyy1p type_Double LINPOL_CYC_END pyy1p
Cvar dpyy1p type_Double LINPOL_CYC_END dpyy1p
Cvar pxx1m type_Double LINPOL_CYC_END pxx1m
Cvar dpxx1m type_Double LINPOL_CYC_END dpxx1m
Cvar pxx1p type_Double LINPOL_CYC_END pxx1p
Cvar dpxx1p type_Double LINPOL_CYC_END dpxx1p
Calctp tp1m pyy1m pxx1m LINPOL_CYC_END dpyy1m dpxx1m 0.445 -0.471 L1mn R1mn U1mn D1mn Mmn \
   L10n R10n U10n D10n M0n Nbm cpm_m
Calctp tp1p pyy1p pxx1p LINPOL_CYC_END dpyy1p dpxx1p 0.445 -0.471 L1pn R1pn U1pn D1pn Mpn \
   L10n R10n U10n D10n M0n Nbp cpm_p
Calctp tp1m0 pyy1m pxx1m LINPOL_CYC_END dpyy1m dpxx1m 0.445 -0.471 L1mn R1mn U1mn D1mn Mmn \
   L10n R10n U10n D10n M0n Nbm cpm_0
Calctp tp1p0 pyy1p pxx1p LINPOL_CYC_END dpyy1p dpxx1p 0.445 -0.471 L1pn R1pn U1pn D1pn Mpn \
   L10n R10n U10n D10n M0n Nbp cpm_0
```

(the `Cvar` objects above simply declare calculation cells to store $\boldsymbol{P_{yy}^{+,-}}$, $\boldsymbol{P_{xx}^{+,-}}$, $\boldsymbol{\Delta P_{yy}^{+,-}}$, $\boldsymbol{\Delta P_{xx}^{+,-}}$ values.)

Lets note `Calctp` uses 4 arms counts if available (non-zero), or only 3 (one of Up and Down arms) otherwise.

The VME hardware used in $53^{rd}$ Nuclotron run consists of:
- 1 FVME [15] VME master,
- 1 TTCM [17] trigger module, and
- 4 TQDC16 [20] modules;

while in $54^{th}$ ($55^{th}$) runs —
- 1 FVME2 [16] VME master,
- 1 FVME2TMWR [18] and
- 1 U40 [19] trigger modules, as well as
- 4(5) TQDC16 [20] modules.

## *sv (1 )* supervisor utility

The *sv* supervisor utility with GUI is provided for the human operator convenience and intended to perform the high–level control over the DSS DAQ system. Supervisor's GUI is expected to be self–explanatory.

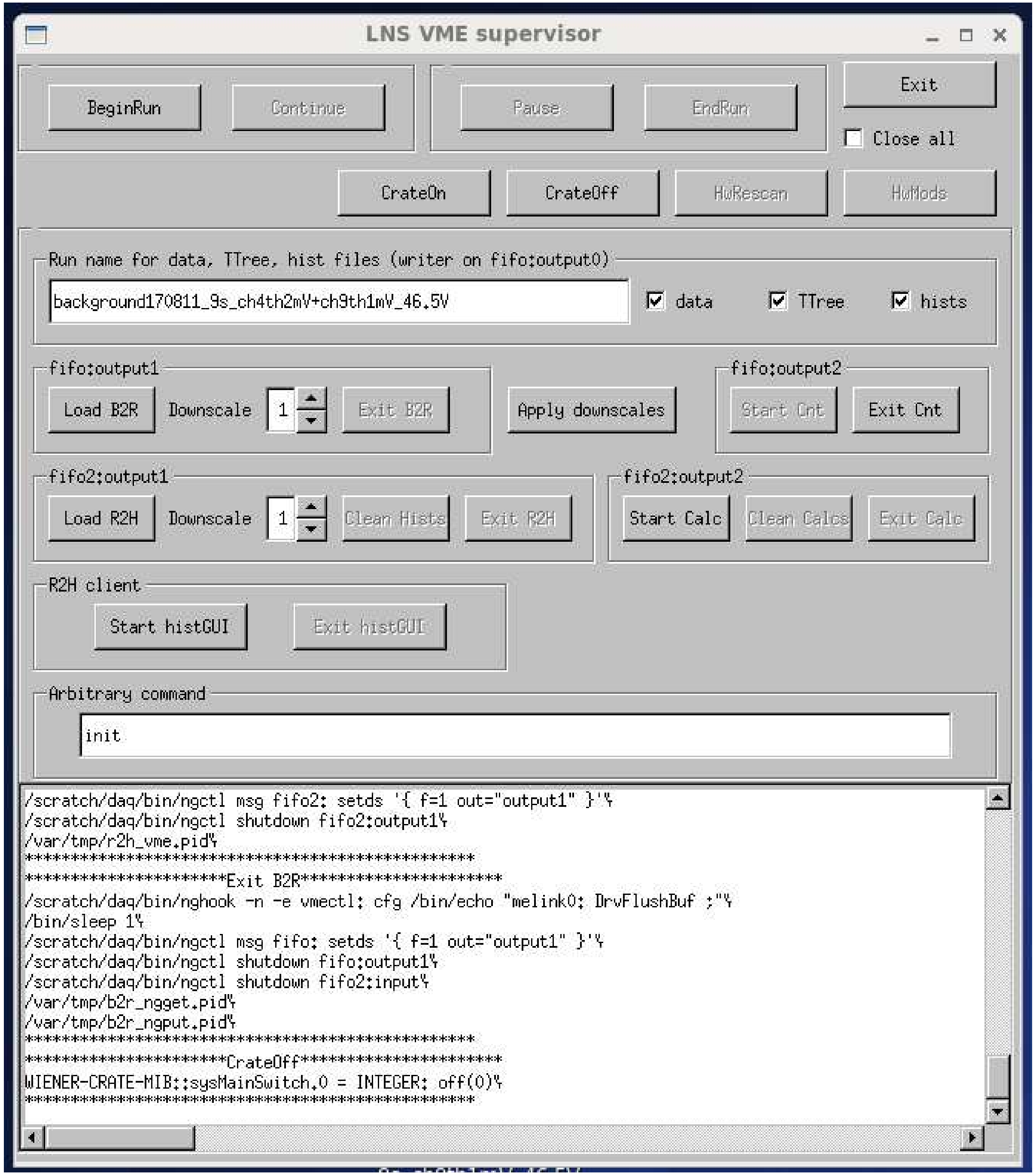



Fig.6. Main window of the *sv(1)* supervisor's GUI.

The main window allows us to perform base DAQ operations like start/stop run, change output filenames, load/unload software components, etc.

The VME hardware configuration is represented for human operator by additional windows (see Fig.7, Fig.8, Fig.9).

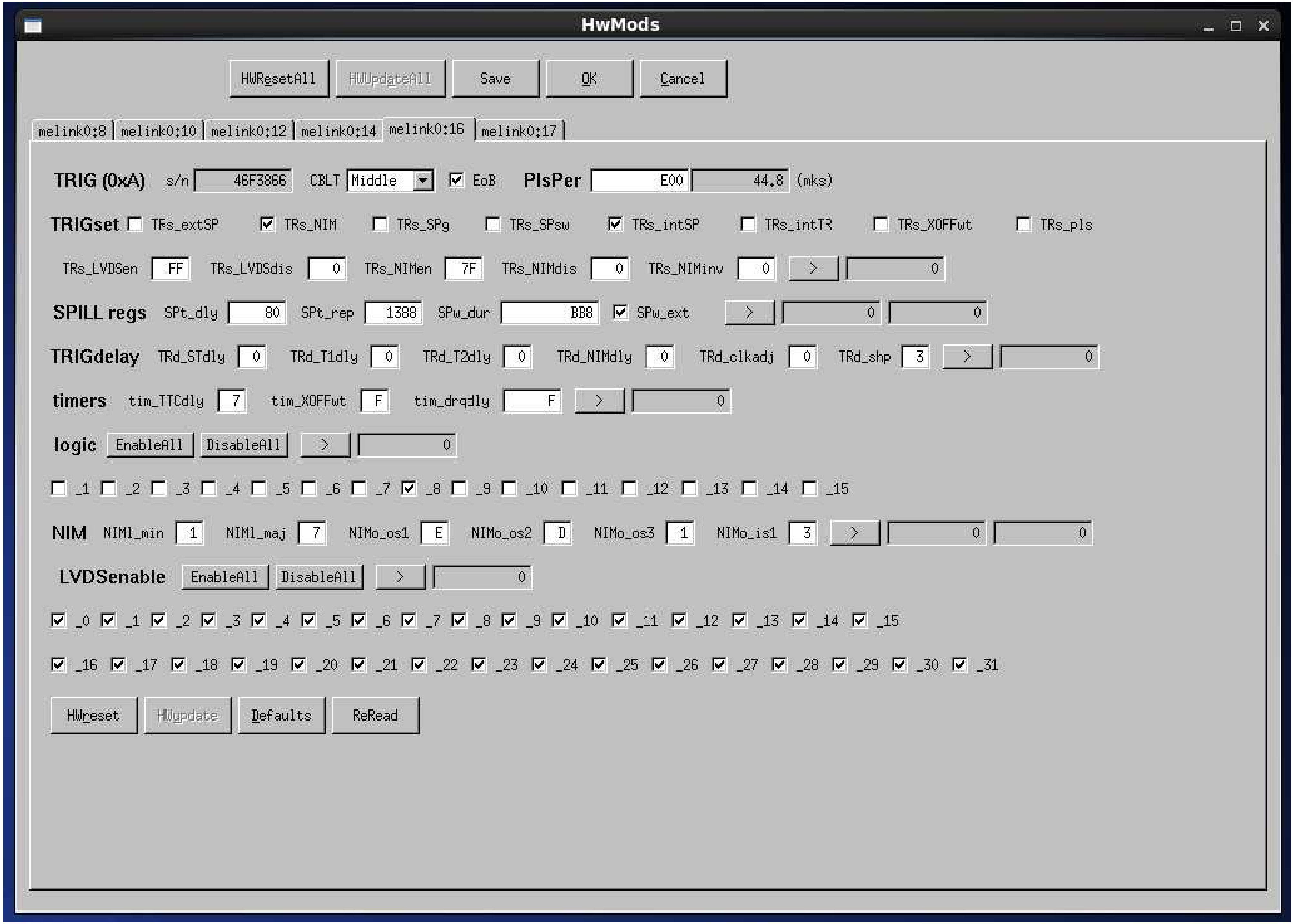



Fig.7. TTCM/FVME2TMWR trigger module configuration window of the *sv(1)* supervisor's GUI.

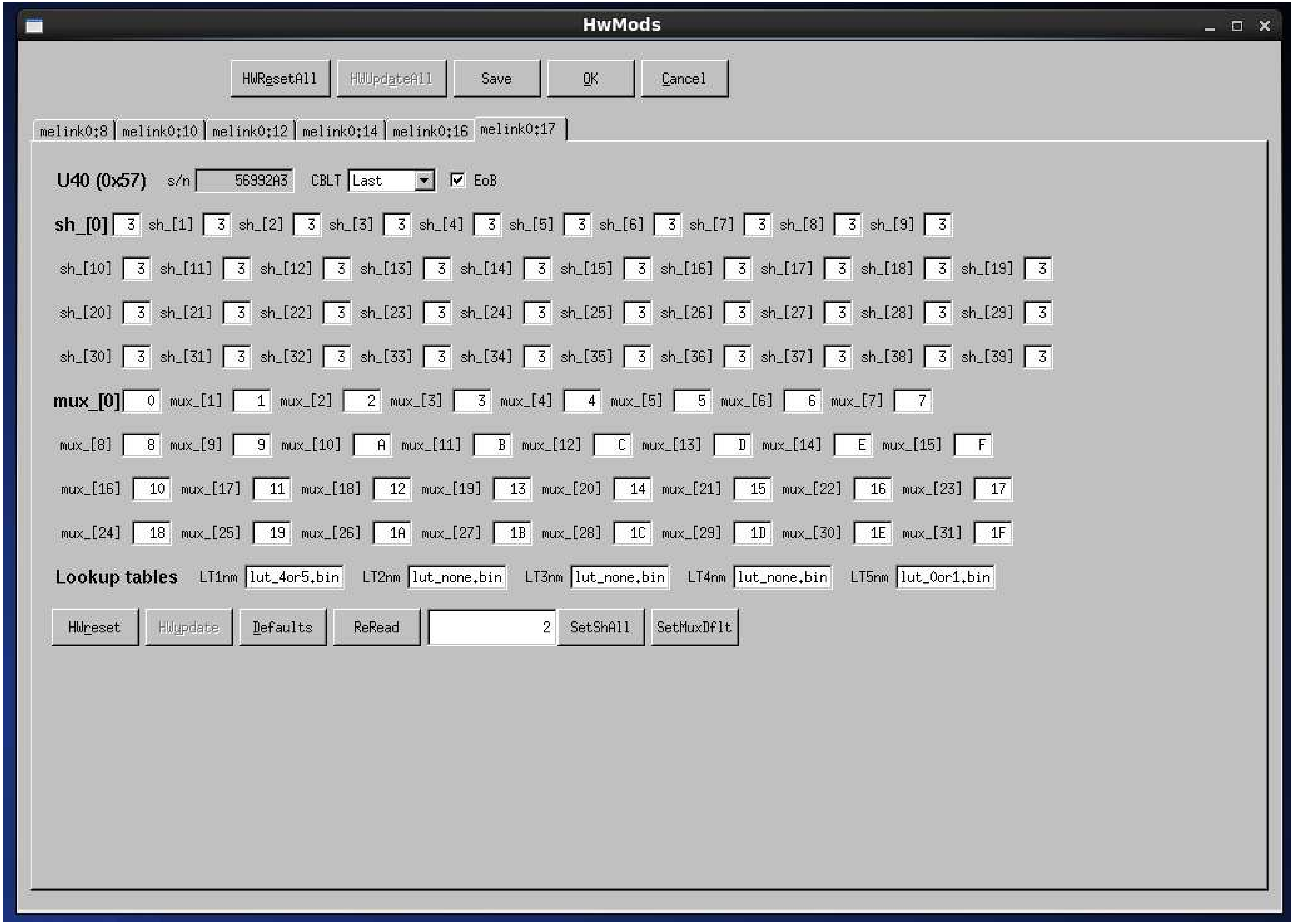



Fig.8. U40 trigger module configuration window of the *sv(1)* supervisor's GUI.

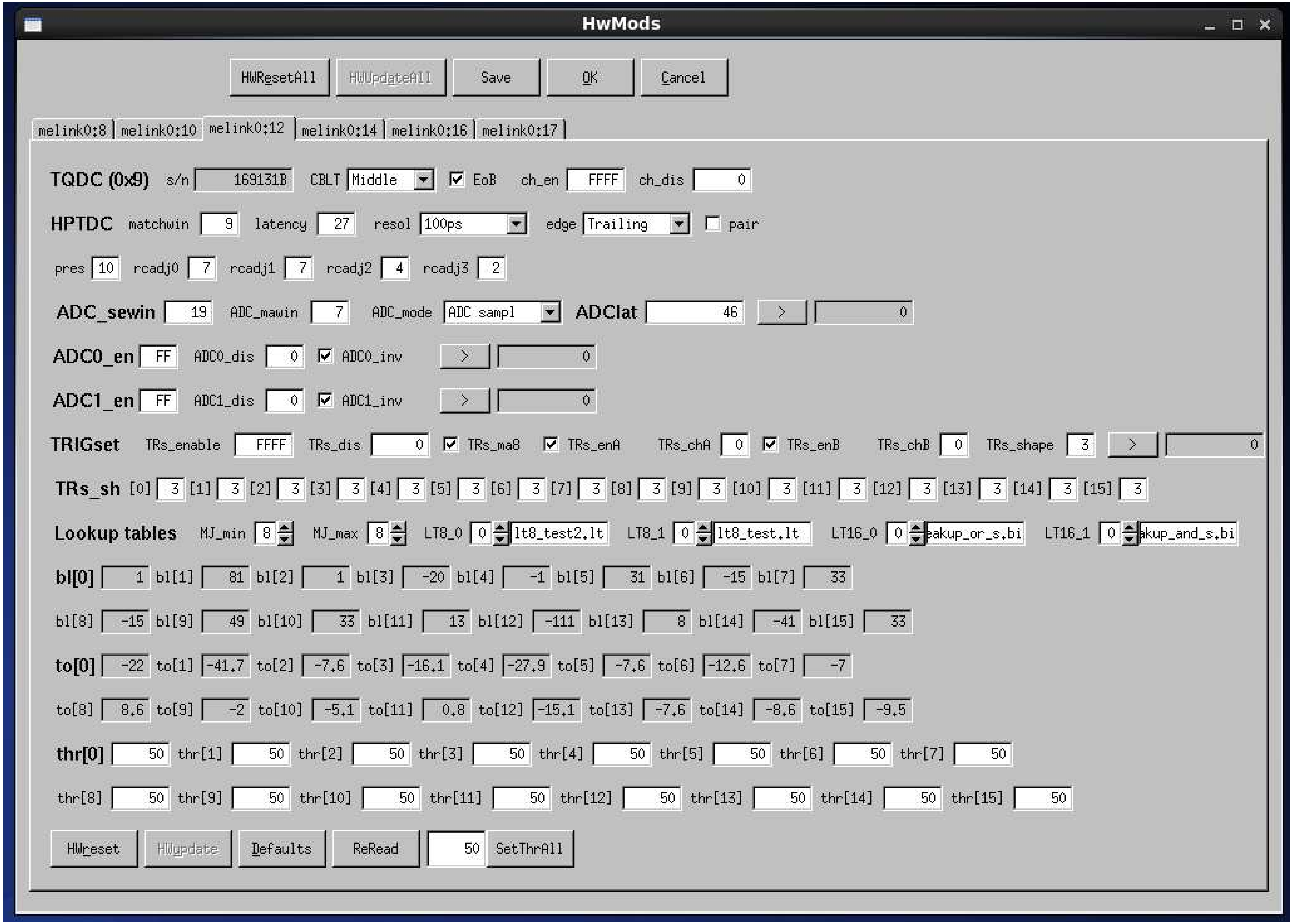



Fig.9. One of TQDC16 module configuration windows of the *sv(1)* supervisor's GUI.

## Conclusions

• The explained DSS DAQ system was successfully used during 53rd (December 2016), 54th (March 2017), and 55th (March 2018) Nuclotron runs.

• The $A_y$, $A_{yy}$, $A_{xx}$ analyzing powers of the $dp$ elastic scattering and vector and tensor polarizations of the deuteron beam from the SPI [10] have been successfully obtained using the designed express–offline software in the *ROOT* scripts form.

• For the SPI mode interesting for DSS project
($P_z^+ = -1/3$, $P_z^- = -1/3$, $P_{zz}^+ = +1$, $P_{zz}^- = -1$) the deuteron polarizations for 135 MeV/nucleon beam at the typical measurement run were following:
$P_z^+ = -0.222 \pm 0.0021$, $P_z^- = -0.256 \pm 0.016$, $P_{zz}^+ = 0.640 \pm 0.039$, $P_{zz}^- = 0.668 \pm 0.030$,
so measured values are $\sim$ **70%** of theoretically possible.

• As SPI methodic studies the polarizations were measured also for other SPI modes:
($P_z^+ = 0$, $P_z^- = +1$, $P_{zz}^+ = 0$, $P_{zz}^- = -2$),
($P_z^+ = +1$, $P_z^- = -2/3$, $P_{zz}^+ = +1$, $P_{zz}^- = 0$),
($P_z^+ = +1$, $P_z^- = +1$, $P_{zz}^+ = -1$, $P_{zz}^- = +1$),
($P_z^+ = -1$, $P_z^- = +1$, $P_{zz}^+ = +2/3$, $P_{zz}^- = 0$)
with satisfactory agreement of theoretic and experimental values.

• In the future the polarizations could be determined online by the polarization calculator utility using the already known analyzing powers and data cuts in terms of 1D–ranges on the time and amplitude 1D– and 2D–histograms. The polarization calculation results could be integrated into Web–based representation scheme described in [21] and [22]. So the DSS DAQ system is suitable for online polarimetry.

• The first measurements of the 500 MeV proton beam polarization were performed during 54th run also [8].

## Acknowledgements

The author has the pleasure to thank V.P.Ladygin for initiation of the polarization calculations development for DSS setup, S.G.Reznikov — for fruitful discussions about VME hardware was used, and whole DSS collaboration — for permanent support and assistance. The author is grateful to V.V.Fimushkin, A.S.Belov, A.V.Butenko, and whole SPI and Nuclotron teams for cooperation during runs. The present work could not be done without background provided by LHEP NOAFI team.

**Thank you for your attention !**

## References


[1] V. P. Ladygin, Yu. V. Gurchin, S. M. Piyadin, A. A. Terekhin, A. Yu. Isupov, et al. Few-body Studies at Nuclotron-JINR. Few Body Syst., **55**, 709–712, (2014).

[2] P. K. Kurilkin, V. P. Ladygin, T. Uesaka, et al. The 270 MeV deuteron beam polarimeter at the Nuclotron Internal Target Station. Nucl.Instr.and Meth.in Phys.Res., **A**(642), 45–51, (2011).

[3] Yu. V. Gurchin et al. Detection equipment for investigating dp elastic scattering at internal target of Nuclotron in the framework of DSS project. Phys.Part.Nucl.Lett., **8**, 950–958, (2011).

[4] S. M. Piyadin et al. ΔE-E detector for proton registration in nonmesonic deuteron breakup at the Nuclotron internal target. Phys.Part.Nucl.Lett., **8**, 107–113, (2011).

[5] A. Yu. Isupov, V. A. Krasnov, V. P. Ladygin, S. M. Piyadin, and S. G. Reznikov. The Nuclotron Internal Target Control and Data Acquisition System. Nucl. Instr. and Meth. in Phys. Res., **A**(698), 127–134, (2013).

[6] P. K. Kurilkin et al. Measurements of the vector and tensor analyzing powers for $dp$-elastic scattering at 880 MeV. Phys.Lett.B, **B**(715), 61–65, (2012).

[7] V. P. Ladygin et al. First results on the energy scan of the vector $A_y$ and tensor $A_{yy}$ and $A_{xx}$ analyzing powers in deuteron–proton elastic scattering at Nuclotron. Journal of Physics: Conference Series, **938**(012007), 1–6, (2017).

[8] V. P. Ladygin, Yu. V. Gurchin, A. Yu. Isupov, et al. First results on the measurements of the proton beam polarization at the internal target at Nuclotron. Journal of Physics: Conference Series, **938**(012008), 1–4, (2017).

[9] M. Janek, V. P. Ladygin, Yu. V. Gurchin, A. Yu. Isupov, et al. Dp breakup reaction investigation using polarized and unpolarized deuteron beam. Journal of Physics: Conference Series, **938**(012005), 1–6, (2017).

[10] V. V. Fimushkin et al. Development of polarized ion source for the JINR accelerator complex. Journal of Physics: Conference Series, **678**(1), 012058–012062, (2016).

[11] A. Yu. Isupov. The *ngdp* framework for data acquisition systems. arXiv:1004.4474 [physics.ins-det], (2010).

[12] A. Yu. Isupov. CAMAC subsystem and user context utilities in *ngdp* framework. arXiv:1004.4482 [physics.ins-det], (2010).

[13] R. Brun and F. Rademakers. ROOT – An Object Oriented Data Analysis Framework. In *Proc. of the AIHENP'96 Workshop*, volume **A**(389) of Nucl.Instr.and Meth.in Phys.Res. (1997), pages 81–86, Lausanne, Switzerland, (1996). See also `http://root.cern.ch/`.

[14] K. I. Gritsaj and A. Yu. Isupov. A Trial of Distributed Portable Data Acquisition and Processing System Implementation: the *qdpb* – Data Processing with Branchpoints. JINR Communications, **E10–2001–116**, 1–19, (2001).

[15] `http://afi.jinr.ru/FVME` .

[16] `http://afi.jinr.ru/FVME2` .

[17] `http://afi.jinr.ru/TTCM` .

[18] `http://afi.jinr.ru/action/show/FVME2TM` .

[19] `http://afi.jinr.ru/action/show/U40VE` .

[20] `http://afi.jinr.ru/TQDC-16` .

[21] A. Yu. Isupov. Data acquisition systems for the high energy and Nuclotron internal target polarimeters with network access to polarization calculation results and raw data. Czech. J. Phys. Suppl., **A55**, A407–A414, (2005).

[22] A. Yu. Isupov. Upgrade of the DAQ systems for the LHE polarimeters to support Vector–Tensor Polarimeter on the Nuclotron internal target. Czech. J. Phys. Suppl., **C56**, C385–C392, (2006).

backup slides

Figure 1:

Figure 2:

Figure 3:

Figure 4:

Figure 5:

Figure 6:

Figure 7:

Figure 8:

Figure 9: