E10-2004-13

A. Yu. Isupov

DAQ SYSTEMS FOR THE HIGH ENERGY AND NUCLOTRON INTERNAL TARGET POLARIMETERS WITH NETWORK ACCESS TO POLARIZATION CALCULATION RESULTS AND RAW DATA

Исупов А. Ю. E10-2004-13

Системы сбора данных с удаленным доступом к результатам вычислений поляризации и «сырым» данным для высокоэнергетического поляриметра и поляриметра на внутренней мишени нуклотрона

Рассматриваются архитектура и реализация системы сбора данных (data acquisition, DAQ) для поляриметра на внутренней мишени нуклотрона (Internal Target Polarimeter, ITP) ЛВЭ ОИЯИ, основанные на распределенной системе сбора и обработки данных *qdpb*. Кратко обсуждаются специфические для данной реализации программные модули, зависящие от аппаратного состава и характера собираемой информации IT-поляриметра, в сравнении с таковыми для высокоэнергетического поляриметра (High Energy Polarimeter, HEP) ЛВЭ ОИЯИ. Описываются предоставляемые пользователям способы доступа к результатам вычислений поляризации и «сырым» данным указанных поляриметров.

Работа выполнена в Лаборатории высоких энергий им. В. И. Векслера и А. М. Балдина ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна, 2004

Isupov A. Yu. E10-2004-13

DAQ Systems for the High Energy and Nuclotron Internal Target Polarimeters with Network Access to Polarization Calculation Results and Raw Data

On-line data acquisition (DAQ) system for the Nuclotron Internal Target Polarimeter (ITP) at the LHE, JINR is explained in respect of design and implementation, based on the distributed data acquisition and processing system *qdpb*. Software modules specific for this implementation (dependent on ITP data contents and hardware layout) are discussed briefly in comparison with those for the High Energy Polarimeter (HEP) at the LHE, JINR. User access methods both to raw data and to results of polarization calculations of the ITP and HEP are discussed.

The investigation has been performed at the Veksler and Baldin Laboratory of High Energies, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna, 2004

INTRODUCTION

At present the data acquisition (DAQ) and slow control systems, based on electronics in the CAMAC standard and computers with the so-called IBM PC compatible architecture (referenced as “Intel i386” below), are still used for physics experiments with a rather low speed of data acquisition and transfer. Such systems survive due to existence of a great assortment of the CAMAC hardware modules, availability of the ISA/PCI adapters for the CAMAC crate controllers, and relatively low cost of mentioned equipment. In particular, on-line electronics of the HE [2] and the IT [1] polarimeters are based on a CAMAC hardware.

The operating system (OS) used on the on-line computer(s) determines what kind of the DAQ system design should be chosen, since the adequate OS selection can essentially simplify implementation, maintenance, and using of the DAQ system.

A UNIX-like OSs are optimal for a wide range of DAQ and slow control tasks. UNIX is a multiprocess and multiuser OS with powerful mechanisms for inter-process and inter-computer communications, support of advanced networking and graphics interfaces, and extended tools for the software design. Resource usage for the OS UNIX working itself is very small and appears negligible for Intel i386 computers with CPUs of i486 class and higher. Such features of UNIX-like OSs as high portability of programming, existence of free distributions with complete sources, and approximately unlimited quantity of the already existing software, are also very attractive. FreeBSD is one of such free UNIX-like OSs, enough stable, reliable, modern, and dynamically developed simultaneously due to its adequate design policy. So, the distributed data acquisition and processing system *qdpb*, described in detail in [3], is implemented under UNIX-like OS FreeBSD.

DAQ system implementations for High Energy and Nuclotron Internal Target Polarimeters are based now on the framework, provided by the *qdpb* system, see [4] for more details about HEP DAQ internals. ITP DAQ is designed in a very similar manner, as we can see in section 1. The present paper concentrates primarily on the user’s point of view to these DAQ systems.

The text below highlights the names of the files and software packages as *italic text*, C language constructions and literals reproduced without changes — as `typewriter text`. The reference to manual page, for example named “qwerty” and situated in the 9th section, is printed as ***qwerty(9)***, the reference to section in this report – as "see section 2". The subjects of substitution by an actual value are enclosed in the angle brackets, for example `<event_kind>`. All the trademarks mentioned in the present paper are properties of their respective owners.

1. HEP AND ITP DAQ SYSTEMS IMPLEMENTATION NOTES

Very compact nature of HEP and ITP CAMAC hardware allows one to concentrate the whole corresponding code in the single C header file for each polarimeter – *azhpol_hardware.h* and *intpol_hardware.h*, which contains:

- `#define`'d constants for numbers of CAMAC stations, functions, subaddresses, etc.;
- macros intended to hide details of work with CAMAC and the raw data format from the user programming level.

HEP and ITP handlers of CAMAC interrupts are based on these constants and macros as well as on the kernel context interfaces ***bpio(9)*** and ***packet(9)***, provided by the *qdpb* system. The handler for each serviced CAMAC interrupt checks its validity, reads CAMAC, encapsulates the obtained data in the *qdpb* packet and puts this packet into the branchpoint buffer.

To manipulate each CAMAC interrupt handler, two utilities are provided – for configuration (`azhpolconf / intpolconf`) and control (`azhpoloper / intpoloper`).

To control the whole HEP / ITP DAQ system, we use a makefile with targets, which represent some operator actions: `load`, `unload`, `pause`, `continue`, etc. The operator can manipulate the DAQ simply by ***make(1)***'ing some target without mediation of any supervisor utility. This solution is chosen due to a very compact nature of HEP / ITP DAQs.

The high-level codes dependent of the polarimeters are grouped together in the *libpol* software library, which contains:

- C declarations and implementation of the programming representation of the polarimeter data content;
- declarations of the packet (event) classification and routines for handling the packet types.

We read information from CAMAC in some raw binary format, which depends on CAMAC modules "geography", and so on. This format needs to be hidden from the user level for many reasons, and it is hidden by *azhpol_hardware.h* / *intpol_hardware.h* macros and format conversion functions. To represent the data at the user level, we define some C structures, named `<polarimeter_name>_<event_kind>`, one per event type. So, polarization calculators can operate with field names instead of offsets, array indices, etc. Conversions of the data from binary to structured format and vise versa are performed by `b2s_<event_kind>()` and `s2b_<event_kind>()` functions, respectively.

The packet (or event) classification was developed initially to implement the SPHERE on-line and off-line software, see [5]. Generally speaking, we introduce some symbolic names for integer numbers, associated with each event type, produced by the DAQ system — for example, beginning of the accelerator burst (`CYC_BEG`), end of the accelerator burst (`CYC_END`), trigger of type 0 (`DAT_0`), etc. For HEP / ITP there is only one event type — `AZHPOL_CYC_END` / `INTPOL_CYC_END`. The event type number is the *qdpb* packet type number: `packet.head.type` (see also section 2).

HEP / ITP DAQ uses at least ***writer(1)*** work module and ***bpget(1)*** and ***netpipes(1)*** service modules from the generic software modules assortment provided by the *qdpb* system. Implementation of the modules mentioned above is independent either of the specific setup hardware layout or experimental data contents. But HEP / ITP specific software modules, namely the polarization values calculators *azhpol* / *intpol* and CGI-style control utility *polar_cgi*, depend on them.

HEP / ITP polarization calculator reads the packet stream and for each obtained packet of type `AZHPOL_CYC_END` / `INTPOL_CYC_END` it calculates the polarization value and error and updates the output files, available through HTTP (see section 3). The *polar_cgi* utility is used to serve these HTTP requests, in particular:

- to start the polarization calculator;
- to stop the polarization calculator;
- to change the user comments;
- to show the history of already completed runs[*] of the polarization calculator.

At the request to change of the HEP / ITP user comments a *polar_cgi* utility updates the corresponding file in the ***usercomm(5)*** format. Such files are read by the HEP / ITP polarization calculators at:

- startup;
- termination;
- arrive each `AZHPOL_CYC_END` / `INTPOL_CYC_END` packet (may be optional).

All HEP / ITP polarization calculator instances are executed on the HTTP server (currently `crab.jinr.ru`) and obtain the input packet stream from the local branch point (in case of HEP) or from the remote one via ***netpipes(1)*** (in case of ITP). Namely, on the machine which hosts the ITP DAQ system (currently `dhe043.jinr.ru`) the server part of the socket connection (***faucet(1)*** utility) is launched and executes ***bpget(1)*** with output into the network socket for each obtained request on connection. These requests are issued by the client parts

[*] Polarization calculations run is a single execution of the polarization calculator `main()` function.

of the socket connection (***hose(1)*** utility), executed by ***polar_cgi(1)*** on the HTTP server `crab.jinr.ru` during IT polarization calculator startup. The same scheme needs to be used for discussed in section 2 obtaining the raw data stream from the HEP / ITP DAQ systems.

2. RAW HEP / ITP DATA OBTAINING

The HEP / ITP DAQ system encapsulates HEP / ITP raw binary data into *qdpb*'s packets (see also [3]), which has the following format:

```
/* packet.h : */
typedef struct {
   header head;            /* packet header */
#define DATA_MAX    (sizeof(header) * 99)
   char data[DATA_MAX];    /* packet body */
} packet;

typedef struct {
   ...
   struct timeval tv; /* packet "originating" time, 4+4 bytes */
   ...
   unsigned short type; /* packet type, 2 bytes */
   ...
} header;

/* sys/time.h : */
struct timeval {
   long tv_sec;  /* seconds since 00:00:00 January 1, 1970 */
   long tv_usec; /* and microseconds */
};
```

The stream of such packets is written on the disk storage as the HEP / ITP raw data files by *writer* work modules (see [3]), which are executed locally on the HEP and ITP DAQ hosts (currently `crab.jinr.ru` and `dhe043.jinr.ru`, respectively). These files are an "official" dataset.

Due to existence of the originating time value within each *qdpb* data packet, a simple method of "tailoring" the HEP / ITP data with some experimental setup data can be recommended. Under each modern UNIX-like OS the standard utilities, for example, ***ntpd(8)***, are available for the system clock synchronization. If the system clocks of the HEP / ITP DAQ hosts, producing HEP / ITP data with "time stamps", and of the some experimental setup DAQ hosts (let name one of them as `extdaqhost` below), are synchronized from the same source, it will be possible to reconstruct correspondence between the HEP / ITP and experimental setup data originating from the same accelerator burst by the simple time stamps comparison at an arbitrary moment in the future. (Of course, DAQ on the `extdaqhost` should produce its own data with some kind of originating time stamps, too.)

Users, who want to apply raw HEP / ITP data in the on-line or off-line analysis, can be provided with the following access methods:

1. (True on-line) Experimental setup DAQ event builder (EVB) executed on the `evbhost` host accepts the packet stream from `crab.jinr.ru` / `dhe043.jinr.ru` through the network socket connection with TCP/IP protocol as the nonstructured bytes sequence without positioning.[*]

   Required: at the programming level: EVB needs to follow
   - *qdpb*'s application program interface (API): *packet.c*, *packet.h*, see [3]
   - HEP / ITP's API: *eazhpol.h* / *eintpol.h*, *fazhpol.c* / *fintpol.c*, *fPOLAR.h*, see [4] for API details

   at the command level:
   - system clocks synchronization by *ntpd*
   - using ***netpipes(1)***

   Advantages:
   - HEP / ITP data transportation as soon as possible
   - critical resources (CPU, disk subsystem) usage on the `evbhost` and `crab.jinr.ru` / `dhe043.jinr.ru` hosts as small as possible
   - adequate programming model

   "Disadvantages":
   - programming efforts required

2. (Semi-on-line) *qdpb*'s utility *writer* executed on the `evbhost` host accepts the packet stream from the `crab.jinr.ru` / `dhe043.jinr.ru` through the network socket with ***netpipes(1)*** mediation and produces HEP / ITP data files on the local filesystem. EVB reads these files.

   Required: at the programming level:
   - EVB needs to follow HEP / ITP's API: *eazhpol.h* / *eintpol.h*, *fazhpol.c* / *fintpol.c*, *fPOLAR.h*, see [4] for API details, and
   - can extract "time stamps" (`packet.head.tv`)

   at the command level:
   - system clocks synchronization by *ntpd*
   - using ***netpipes(1)***, ***writer(1)***

   Advantages:
   - slightly less programming efforts than for method 1

[*] ***netpipes(1)*** package can be used to simplify work with sockets into UNIX streams manner (like `stdin` reading from UNIX pipe).

   - critical resources (CPU, disk subsystem) usage on the `crab.jinr.ru` / `dhe043.jinr.ru` host (not for `evbhost` !) as small as possible

   Disadvantages:
   - nonadequate programming model (EVB accesses HEP / ITP data as files instead of the stream)
   - critical resources (CPU, disk subsystem) usage on the `evbhost` host is increased

3. (Semi-off-line) Filesystem with HEP / ITP “official” data files is exported by NFS mechanism for the `evbhost` from the `crab.jinr.ru` / `dhe043.jinr.ru` and EVB reads these files.

   Required: at the programming level:
   - EVB needs to follow HEP / ITP’s API: *eazhpol.h* / *eintpol.h*, *fazhpol.c* / *fintpol.c*, *fPOLAR.h*, see [4] for API details, and
   - can extract “time stamps” (`packet.head.tv`)

   at the command level:
   - system clocks synchronization by *ntpd*
   - export/mount NFS from HEP / ITP DAQ hosts to `evbhost`

   Disadvantages:
   - nonadequate programming model (EVB accesses HEP / ITP data as files instead of the stream)
   - NFS is neither secure nor reliable enough for “synchronous” access
   - critical resources (CPU, disk subsystem) usage on the `crab.jinr.ru` / `dhe043.jinr.ru` is maximal through all the methods mentioned in the present paper

4. (Off-line only) HEP / ITP whole “official” dataset obtained by arbitrary methods (see below) after the accelerator run termination.

   Required: at the programming level:
   - off-line software needs to follow HEP / ITP’s API: *eazhpol.h* / *eintpol.h*, *fazhpol.c* / *fintpol.c*, *fPOLAR.h*, see [4] for API details, and
   - can extract “time stamps” (`packet.head.tv`)

   at the command level:
   - system clocks synchronization by *ntpd* during all the accelerator run
   - network (*ftp*, *scp*, etc.) or other (floppies, CD–ROMs) access to HEP / ITP datafiles on the `crab.jinr.ru` / `dhe043.jinr.ru` after the run termination

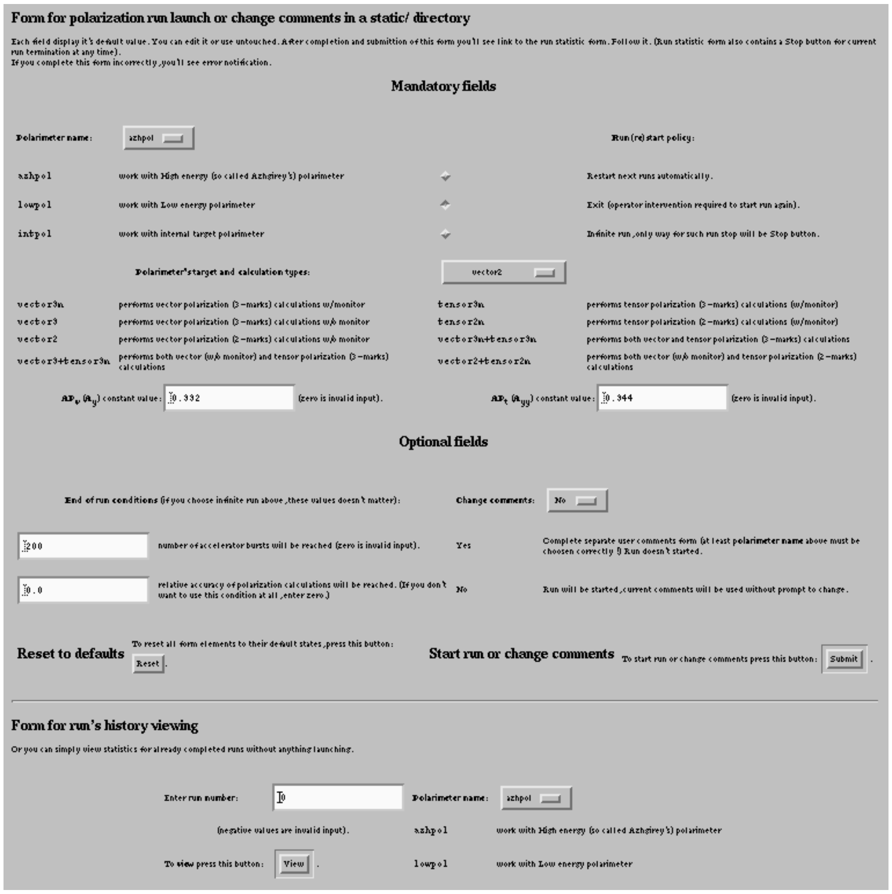
   Advantages:
   - least programming efforts
   - least command level efforts

   Disadvantages:
   - method fits only the off-line.

3. HTTP ACCESS TO POLARIZATION CALCULATION RESULTS

Results of polarization calculations for HE and IT Polarimeters of the LHE, JINR are accumulated on the HTTP server (currently `crab.jinr.ru`, `159.93.18.110`), so the user needs only HTTP client (WWW–browser) of any flavour (*Netscape*, *MSIE*, . . . , without graphics information – *lynx*) to access the results.

According to the functionality of the *azhpol* and *intpol* polarization calculators (see section 1), three working modes are possible: the so-called static



Form for polarization run launch or change comments in a static/ directory

Each field display it's default value. You can edit it or use untouched. After completion and submittion of this form you'll see link to the run statistic form. Follow it. (Run statistic form also contains a Stop button for current run termination at any time).
If you complete this form incorrectly, you'll see error notification.

Mandatory fields

Polarimeter name:	azhpol		Run (re) start policy:
azhpol	work with High energy (so called Azhgirey's) polarimeter	◇	Restart next runs automatically.
lowpol	work with Low energy polarimeter	◆	Exit (operator intervention required to start run again).
intpol	work with internal target polarimeter	◇	Infinite run, only way for such run stop will be Stop button.

Polarimeter's target and calculation types: vector2

vector3n	performs vector polarization (3 –marks) calculations w/monitor	tensor3n	performs tensor polarization (3 –marks) calculations (w/monitor)
vector3	performs vector polarization (3 –marks) calculations w/o monitor	tensor2n	performs tensor polarization (2 –marks) calculations (w/monitor)
vector2	performs vector polarization (2 –marks) calculations w/o monitor	vector3n+tensor3n	performs both vector and tensor polarization (3 –marks) calculations
vector3+tensor3n	performs both vector (w/o monitor) and tensor polarization (3 –marks) calculations	vector2+tensor2n	performs both vector (w/o monitor) and tensor polarization (2 –marks) calculations

AP_v (A_y) constant value: 0.332 (zero is invalid input). AP_t (A_{yy}) constant value: 0.344 (zero is invalid input).

Optional fields

End of run conditions (if you choose infinite run above, these values doesn't matter): Change comments: No

200	number of accelerator bursts will be reached (zero is invalid input).	Yes	Complete separate user comments form (at least **polarimeter name** above must be choosen correctly !) Run doesn't started.
0.0	relative accuracy of polarization calculations will be reached. (If you don't want to use this condition at all, enter zero.)	No	Run will be started, current comments will be used without prompt to change.

Reset to defaults To reset all form elements to their default states, press this button: Reset.

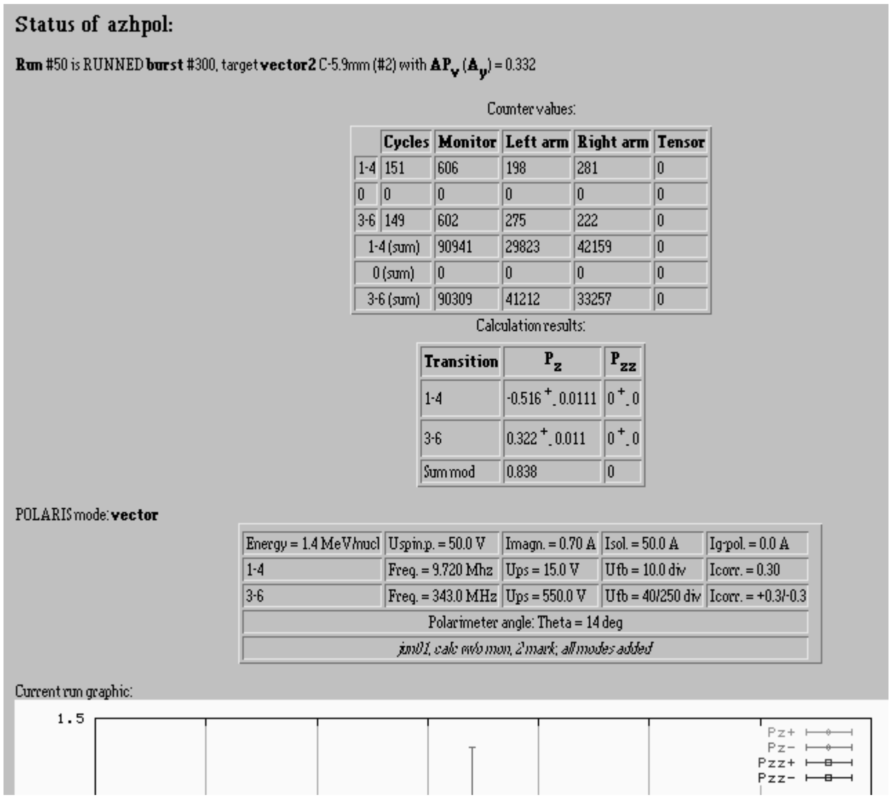
Start run or change comments To start run or change comments press this button: Submit .

Form for run's history viewing

Or you can simply view statistics for already completed runs without anything launching.

Enter run number:	0	Polarimeter name:	azhpol
	(negative values are invalid input).	azhpol	work with High energy (so called Azhgirey's) polarimeter
To view press this button:	View .	lowpol	work with Low energy polarimeter

Fig. 1. Start form for the static run (page `http://crab.jinr.ru/˜camac/static`)



Status of azhpol:

Run #50 is RUNNED **burst** #300, target **vector2** C-5.9mm (#2) with $\mathbf{AP_y}$ ($\mathbf{A_y}$) = 0.332

Counter values:

	Cycles	Monitor	Left arm	Right arm	Tensor
1-4	151	606	198	281	0
0	0	0	0	0	0
3-6	149	602	275	222	0
1-4 (sum)		90941	29823	42159	0
0 (sum)		0	0	0	0
3-6 (sum)		90309	41212	33257	0

Calculation results:

Transition	P_z	P_{zz}
1-4	-0.516 ± 0.0111	0 ± 0
3-6	0.322 ± 0.011	0 ± 0
Sum mod	0.838	0

POLARIS mode: **vector**

Energy = 1.4 MeV/nucl	Uspin.p. = 50.0 V	Imagn. = 0.70 A	Isol. = 50.0 A	Ig-pol. = 0.0 A
1-4	Freq. = 9.720 Mhz	Ups = 15.0 V	Ufb = 10.0 div	Icorr. = 0.30
3-6	Freq. = 343.0 MHz	Ups = 550.0 V	Ufb = 40/250 div	Icorr. = +0.3/-0.3
Polarimeter angle: Theta = 14 deg				
jun01, calc w/o mon, 2'mark; all modes added				

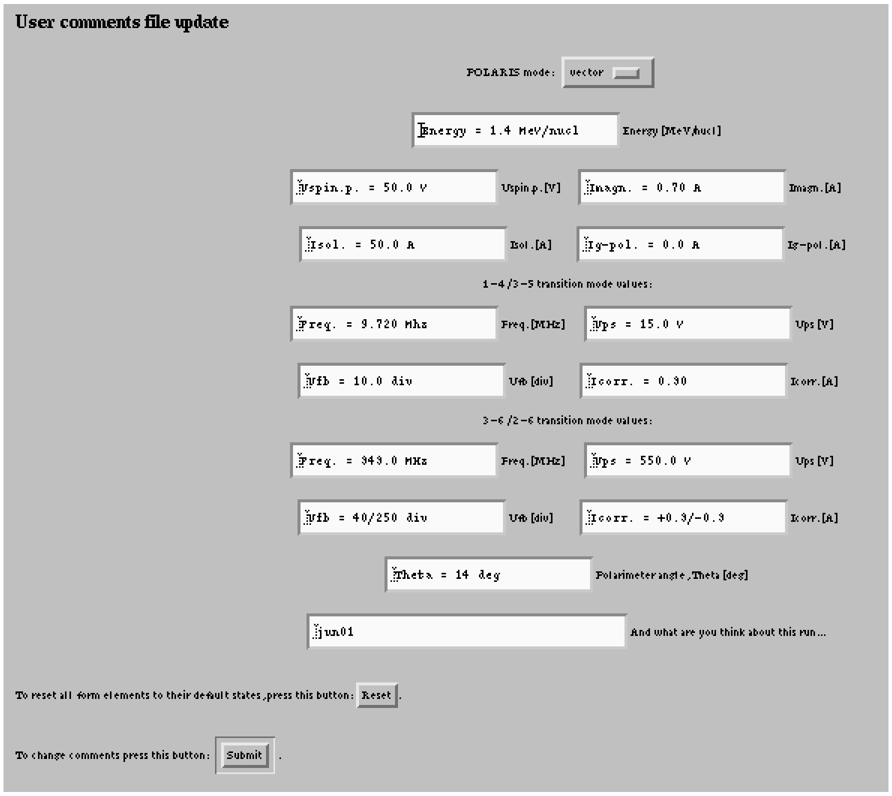
Current run graphic:

Fig. 2. Static run results (page `http://crab.jinr.ru/˜camac/static/azhpol`)

run*, dynamic run, and viewing of the current static run results. History viewing of already terminated static runs is also possible.

1. The purpose of the static run mode is to perform reference polarization calculations by the corresponding polarimeter personnel: in the *static/azhpol* directory — for HEP, in the *static/intpol* directory — for ITP. Only one static run is permitted simultaneously for each polarimeter. Any authorised (by HTTP username/password) user can launch a static run, terminate the already started one, change current comments at any time. The static run may be launched with and without termination condition(s). If these conditions are provided, a request to start the next runs automatically after previous run termination under the same

*Term "run" introduced in section 1.



User comments file update

POLARIS mode: vector

Energy = 1.4 MeV/nucl   Energy [MeV/nucl]

Uspin.p. = 50.0 V   Uspin.p.[V]   Imagn. = 0.70 A   Imagn.[A]

Isol. = 50.0 A   Isol.[A]   Ig-pol. = 0.0 A   Ig-pol.[A]

1-4 /3-5 transition mode values:

Freq. = 9.720 Mhz   Freq.[MHz]   Ups = 15.0 V   Ups [V]

Ufb = 10.0 div   Ufb [div]   Icorr. = 0.30   Icorr.[A]

3-6 /2-6 transition mode values:

Freq. = 343.0 MHz   Freq.[MHz]   Ups = 550.0 V   Ups [V]

Ufb = 40/250 div   Ufb [div]   Icorr. = +0.3/-0.3   Icorr.[A]

Theta = 14 deg   Polarimeter angle ,Theta [deg]

jun01   And what are you think about this run...

To reset all form elements to their default states ,press this button: Reset .

To change comments press this button: Submit .

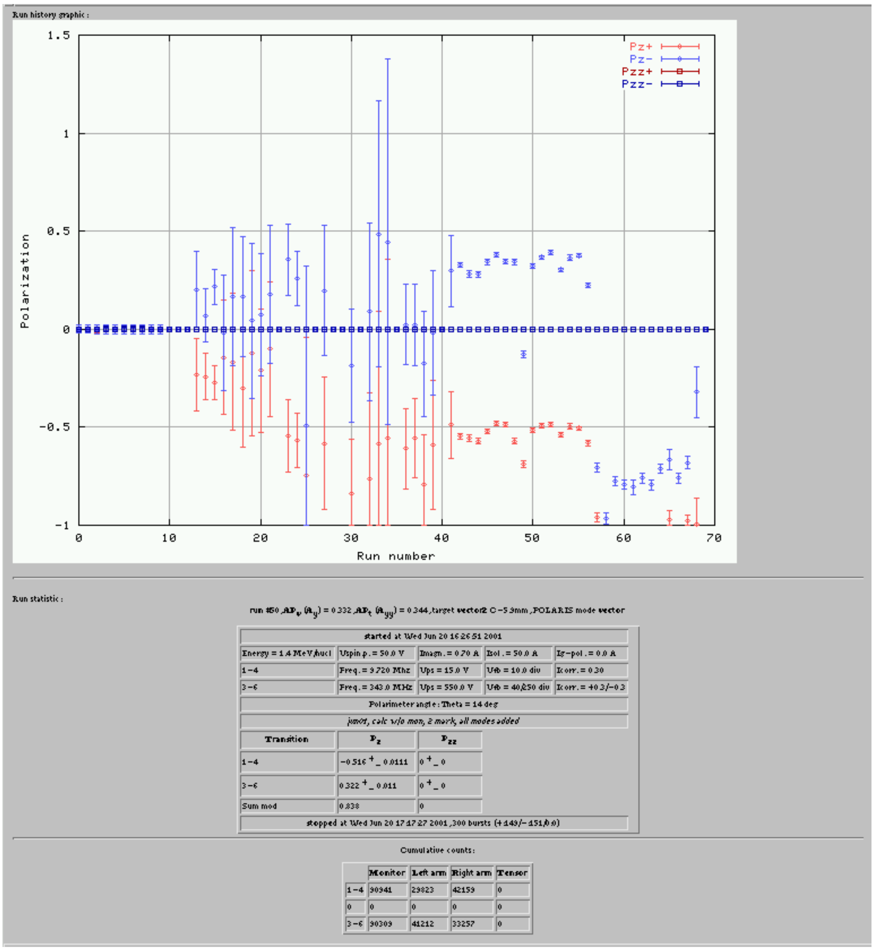
Fig. 3. Change comments form (generated by the *polar_cgi*)

conditions may or may not be given. For all cases an immediate termination (by Stop buttons on the start page and current run page) is permitted for any authorised user. Run termination conditions, which can be requested, are as follows:

1. to reach the required number of accelerator bursts,
2. to reach the required calculation accuracy.

The request for the second condition alone is not permitted. The both conditions can be given but fulfillment of even one of them is enough to terminate the run.

Page `http://crab.jinr.ru/~camac/static` contains a start form for the static run (see Figure 1), after its submission the user obtains in responce a link to the static run results `http://crab.jinr.ru/~camac/static/` `{azhpol,intpol}` (see Figure 2). At change comments request the user obtains a separate page (see



Run statistic :

run #50, AP_v (A_y) = 0.332, AP_t (A_{yy}) = 0.344, target vector2 C−5.3mm, POLARIS mode vector

started at Wed Jun 20 16:26:51 2001				
Energy = 1.4 MeV/nucl	Uspin.p. = 50.0 V	Imagn. = 0.70 A	Isol. = 50.0 A	Ig−pol. = 0.0 A
1−4	Freq. = 9.720 Mhz	Ups = 15.0 V	Ufb = 10.0 div	Icorr. = 0.30
3−6	Freq. = 343.0 MHz	Ups = 550.0 V	Ufb = 40/250 div	Icorr. = +0.3/−0.3
Polarimeter angle : Theta = 14 deg				
jun01, calc w/o mon, 2 mark, all modes added				
Transition	**P_z**	**P_{zz}**		
1−4	−0.516 +_ 0.0111	0 +_ 0		
3−6	0.322 +_ 0.011	0 +_ 0		
Sum mod	0.838	0		
stopped at Wed Jun 20 17:47:27 2001 ,300 bursts (+149/−151/0.0)				

Cumulative counts:

	Monitor	Left arm	Right arm	Tensor
1−4	90941	29823	42159	0
0	0	0	0	0
3−6	90309	41212	33257	0

Fig. 4. History viewing page (generated by the *polar_cgi*), enter from `http://crab.jinr.ru/~camac/static`

Figure 3) and after its submission he will be returned to the start page again. The start page also contains a form for history viewing, after its submission the user obtains a history page (see Figure 4).

2. The purpose of the dynamic run mode is to perform private polarization calculations by the experimental setup personnel. More than one dynamic run is permitted simultaneously (limited by the system implementation) for each polarimeter. Any user (possibly not from any host) can launch the dynamic

run, but only the same user can terminate it (by Stop button in the HTML page corresponding to this dynamic run). The dynamic run may be launched only with termination condition(s) provided and only without the request to start next runs automatically. The termination conditions of the dynamic run and logic of their application are the same as for the static run (see above). The page http://crab.jinr.ru/˜camac/dynamic contains a start form for the dynamic run, after its submission the user obtains in responce a link to the dynamic run results (unpredictable temporary name will be used, so one must not forget this name otherwise you won't be able to see your dynamic run results). Dynamic run results are deleted just after its termination, so be careful to save (by your HTTP client) the dynamic run final page as soon as possible, if you need it. Due to this nature of the dynamic run the change comments and history viewing services are not provided.

3. The purpose of viewing the current static run results is to provide reference results of polarization calculations for the experimental setup personnel.

Any user (possibly not from any host) can view the corresponding pages http://crab.jinr.ru/˜camac/static/{azhpol,intpol}, each of them also contains a form for history viewing.

CONCLUSIONS

The HEP DAQ system with polarization calculation results to be presented via HTTP was used during a number of polarized runs of the Synchrophasotron and Nuclotron — Jun'2001, Oct'2001, Nov'2002 and Dec'2002 without serious problems of stability and functionality. The ITP DAQ system with results presented in a similar scheme was successfully tested during the polarized run of the Nuclotron in December 2002. The presented scheme of polarization calculation distributions is extendible easily to serve more polarimeters, as well as the described design of the HEP and ITP DAQ systems.

ACKNOWLEDGEMENTS

The author has a pleasure to thank Yu. K. Pilipenko for initiation of the presented developments; S. G. Reznikov and S. V. Afanasiev – for useful discussions, help in testing and some code contribution; L. S. Zolin, G. D. Stoletov and V. N. Zhmyrov — for help with polarization calculations; and N. M. Piskunov — for the interest to polarimeters raw data obtaining.

Investigations was supported in part by the RFBR grant N 02–02–16024.

REFERENCES

1. Anisimov Yu.S. et al. Polarimeter for Nuclotron internal beam // Particles and Nuclei Letters (in Russian), 2004, v.1, N.1[118], p.68–79.
2. Azhgirey L.S. et al. // Pribory i Tekhnika Eksperimenta (in Russian), 1997, N.1, p.51.
3. Gritsaj K.I., Isupov A.Yu. A trial of distributed portable data acquisition and processing system implementation: the *qdpb* — Data Processing with Branchpoints. JINR Communication E10–2001–116, Dubna, 2001.
4. Isupov A.Yu. DAQ system for High Energy Polarimeter at the LHE, JINR: implementation based on the *qdpb* (Data Processing with Branchpoints) system. JINR Communication E10–2001–198, Dubna, 2001.
5. Isupov A.Yu. SPHERE DAQ and off-line systems: implementation based on the *qdpb* system. JINR Communication E10–2003–187, Dubna, 2003.

Received on January 27, 2004.

Корректор *Т. Е. Попеко*

Подписано в печать 10.03.2004.
Формат 60 × 90/16. Бумага офсетная. Печать офсетная.
Усл. печ. л. 0,93. Уч.-изд. л. 1,27. Тираж 290 экз. Заказ № 54313.

Издательский отдел Объединенного института ядерных исследований
141980, г. Дубна, Московская обл., ул. Жолио-Кюри, 6.
E-mail: publish@pds.jinr.ru
www.jinr.ru/publish/