УДК: 004.031.4:539.1.075:539.17

СИСТЕМА СБОРА ДАННЫХ НОВОГО ПОКОЛЕНИЯ ДЛЯ СПЕКТРОМЕТРИЧЕСКИХ МНОГОКАНАЛЬНЫХ УСТАНОВОК

А.Ю. Исупов[1], В.Е. Ковтун[2], А.Г. Фощан[2]
[1]Объединённый институт ядерных исследований, Дубна, Россия;
[2]Харьковский национальный университет имени В.Н. Каразина, Харьков, Украина
E-mail: kovtun@pht.univer.kharkov.ua

Предлагаются распределённые системы сбора, транспортировки, обработки и представления данных (DAQ) для ИЯИ НАНУ и установки COMBAS (ЛЯР ОИЯИ) на основе инфраструктуры *ngdp* и пакета *camac* (ЛЯП ОИЯИ). Представление событий в online специальным ROOT-классом позволяет реализовать гистограммный сервер, конфигурируемый во время исполнения. Клиенты последнего с полномасштабным графическим интерфейсом независимы от остальной DAQ и реализуются под любую операционную систему, поддерживающую ROOT.

1. ВВЕДЕНИЕ

В настоящее время в физических установках с числом каналов десятки и сотни часто используются магистрально–модульный стандарт КАМАК и компьютер архитектуры Intel IA-32 (i386) − т.е. совместимый с IBM PC и называемый в обиходе персональным − для съема данных. Зачастую также система сбора данных представляет собой монолитную программу под DOS, чтобы непосредственно работать с аппаратными средствами и отчасти для обеспечения быстродействия. Привязанность к DOS в настоящее время не содержит положительных моментов. Такая программа практически не поддается развитию, равно как и переносу на другую установку, т.е. создание для новой установки собственной системы сбора данных потребует «полного цикла» разработки, что зачастую нереально. Выходом является использование каких-либо готовых решений, например, инфраструктур (framework) *qdpb* [1] и *ngdp* (см. [2] и [3]) для UNIX-подобной операционной системы (ОС) FreeBSD, позволяющих реализовывать модульные распределенные системы сбора, транспортировки, обработки и представления данных для разнообразных ядерно-физических установок с электронным съемом данных. Так, системы сбора данных на их основе успешно эксплуатируются в ЛВЭ ОИЯИ и ХНУ имени В.Н.Каразина, а в качестве подсистемы обслуживания КАМАК используется пакет *camac* (ЛЯП ОИЯИ, г. Дубна) [4] под ОС FreeBSD.

2. ИСПОЛЬЗУЕМЫЕ ПРОГРАММНЫЕ ПАКЕТЫ − NGDP, CAMAC, ROOT

В современном программировании очень важно уметь находить и использовать уже готовые программные пакеты, предназначенные либо точно для решения стоящей задачи, либо для чего-то похожего. Использование таких пакетов на уровне источников позволяет перенимать также готовую продуманную идеологию и стиль программирования.

Как уже отмечено выше, инфраструктуры *qdpb* и *ngdp* предлагают модульный принцип, позволяющий исполнять многие элементы системы одновременно, в том числе на различных процессорах и компьютерах, передавая информацию, оформленную в некие пакеты, между различными программными модулями посредством потоков (а не файлов, т.е. минуя файловую систему). Локальные потоки практически между любыми модулями легко могут быть заменены на межмашинные, что позволяет произвольно распределять систему. Модульность также позволяет изолировать зависимые от конкретной установки части кода и реализовать остальное как набор неспецифических модулей, использующихся в неизменном виде в любых системах DAQ. Инфраструктурный подход минимизирует усилия по разработке и сопровождению многих систем DAQ, поскольку одни и те же отлаженные модули можно применять снова и снова. Инфраструктура *ngdp* избегает принудительного планирования (preemptive scheduling) критичных частей кода, размещая часть модулей в ядре ОС, и использует пакет ***netgraph(4)*** для организации потоков пакетов данных между ними внутри ядра ОС (оригинально ***netgraph(4)*** разработан для реализации сложных многоуровневых сетевых протоколов в виде графов и пересылает сетевые пакеты вдоль ребер графа между трансформирующими их узлами − вершинами графа). Такая техника (оформление модулей контекста ядра в стиле ***netgraph(4)***) повышает быстродействие и скорость реакции на внешние события системы DAQ, не теряя естественную многозадачность UNIX в пользовательском контексте, в котором *ngdp* совместима с *qdpb*.

Пакет *camac* [4] под ОС FreeBSD, применяемый для работы с аппаратными средствами КАМАК, также изолирует в своих драйверах аппаратно-зависимые части кода (поддержку конкретных пар адаптер/контроллер крейта) от остальной, универсальной части пакета. Такой подход позволяет предоставить унифицированный аппаратно-независимый пользовательский программный интерфейс работы с КАМАК и, таким образом, существенно упрощает использование пакета, а также формализует создание под него новых драйверов и модулей пользовательских обработчиков прерываний от КАМАК. Так, авторами реализован драйвер ***kh(4)*** контроллера крейта CC02 [5] с адаптером шины PCI, разработанным в ХНУ им. В.Н. Каразина и предлагаемым к

PROBLEMS OF ATOMIC SCIENCE AND TECHNOLOGY. 2010. № 2.
Series: Nuclear Physics Investigations (53), p.154-158.

использованию в спектрометрах COMBAS и ИЯИ НАНУ. В драйвере ***kh(4)*** реализованы все операции набора ***camac(2)***, кроме установки (**CCEM**) и проверки (**CTEM**) маски LAM'ов, для чего отсутствует аппаратная поддержка в CC02. Для возможности снижения накладных расходов во время исполнения был также написан аппаратно-зависимый набор макросов ***kh(9)*** для контекста ядра (в виде заголовочного файла *ci_kh.h*), реализующих доступ к адаптеру CC02. Таким образом, обработчик прерываний может, жертвуя аппаратной независимостью ради скорости исполнения, использовать макросы ***kh(9)*** вместо функций набора ***camac(9)***. Впрочем, прогресс центральных процессоров неуклонно снижает эту разницу в скорости на фоне неизменной скорости выполнения собственно аппаратного цикла КАМАК.

Поскольку все аспекты гистограммирования достаточно надежно, удобно и эффективно реализованы в пакете ROOT [6], естественно использовать его в качестве библиотеки, предоставляющей классы одномерных (**TH1F**) и двумерных (**TH2F**) гистограмм.

3. СПЕЦИФИКА СИСТЕМЫ DAQ COMBAS

Фрагмент-сепаратор COMBAS [7] предназначен для исследования нестабильных (нейтроно- и протоно-избыточных) ядер, получаемых в реакциях с тяжёлыми ионами низких и средних энергий порядка 100 МэВ/нуклон на циклотроне У-400М ЛЯР ОИЯИ (Рис.1). Сбор данных с 64 каналов стриповых ППД и 9 каналов детекторов CsI(Tl) производится спектрометрической аппаратурой в стандарте КАМАК. В дальнейшем число каналов других плеч спектрометра возрастет до 320 (Рис.2).

Схема построения системы DAQ на основе инфраструктуры *ngdp* с использованием пакета *camac* достаточно подробно описана в [3] на примере установки QUADRO (ФТФ ХНУ имени В.Н. Каразина). Граф, реализующий потоки пакетов в DAQ COMBAS, будет аналогичным (см. Рис.3). При этом зависимость от конкретной аппаратуры КАМАК локализована в обработчике прерываний, а от конкретного состава собираемых данных − в представлении каждого отдельного события (триггера установки) для системы ROOT, реализуемого как ROOT − класс **Ecombas**. Отметим, что видов событий, т.е. триггеров, может быть более одного, возможно различающихся составом читаемых блоков КАМАК, соответственно **Ecombas** может иметь член для представления вида события. Поэтому, строго говоря, все необходимые в нашем случае программные модули уже предоставлены нам инфраструктурой *ngdp* и некоторые лишь требуют компиляции с классом **Ecombas**.

Типовой обработчик прерываний от КАМАК в виде загружаемого (KLD) модуля ядра ОС, сопрягаемый с нодой ***ng_camacsrc(4)*** системы *ngdp*, не содержит в явном виде частей кода, реализующих обслуживание аппаратуры КАМАК. Последние оформлены в виде макросов, осуществляющих инициализацию аппаратуры КАМАК, загрузку конфигураций блоков, выяснение типа триггера, чтение и сброс по триггеру каждого типа и т.д.

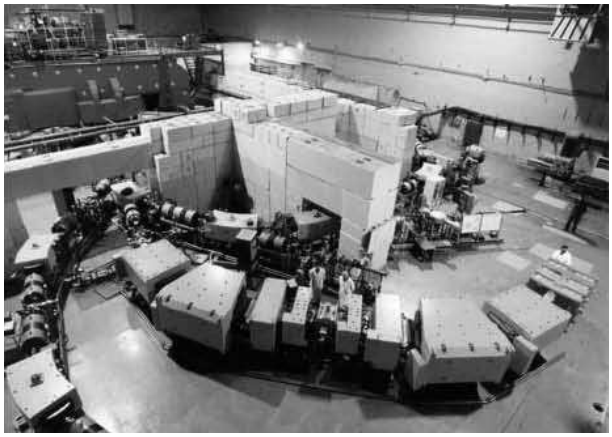

Рис.1. Многоканальный спектрометр COMBAS

В терминах таких аппаратно-зависимых макросов и написан (на C) обработчик прерываний. Это позволяет практически не менять его основной текст от сеанса к сеансу даже при изменениях состава аппаратуры КАМАК. При необходимости работать с различными составами аппаратуры в ходе одного сеанса следует заготовить несколько наборов аппаратных макросов и скомпилировать столько же KLD-модулей обработчика прерываний, для смены которых не требуется даже перезагрузка ОС. Что касается реализации самих аппаратных макросов, то на компактных установках, например, на различных поляриметрах в ЛВЭ ОИЯИ (см., в частности, [8]) или на QUADRO [3] их несложно было написать «вручную». Более обширная аппаратура КАМАК спектрометров COMBAS и ИЯИ НАНУ, видимо, потребует применения так называемого конфигурируемого представления аппаратуры КАМАК [9], успешно эксплуатировавшегося в течение многих лет при сопровождении систем DAQ на установках СФЕРА [10] и СКАН [11] в ЛВЭ ОИЯИ. Конфигурируемое представление позволяет формализовать описание на C аппаратуры КАМАК и автоматизировать генерацию комплекта заголовочных файлов, определяющих аппаратные макросы.

Спектрометры COMBAS и ИЯИ НАНУ предполагают набор и обработку во время online нескольких сотен одномерных гистограмм по ~ 5000 каналов и до сотни двумерных по ~5000×5000 каналов. Предполагая значение float (8 байтов) в каждом канале, получаем более 2 Гигабайт оперативной памяти только для хранения собственно гистограмм. Понятно, что поддерживать их все одновременно слишком ресурсоемко, причем не только по памяти, но и по вычислительной мощности. Тем не менее, сохраняемая в бинарном формате экспериментальная информация позволит построить в offline любые гистограммы в указанном количестве. Что же касается online, то, очевидно, следует накапливать только те гистограммы, которые интересны в данный момент для контроля ключевых параметров эксперимента, т.е. необходимо реализовать управляемый во время исполнения гистограммный сервер.

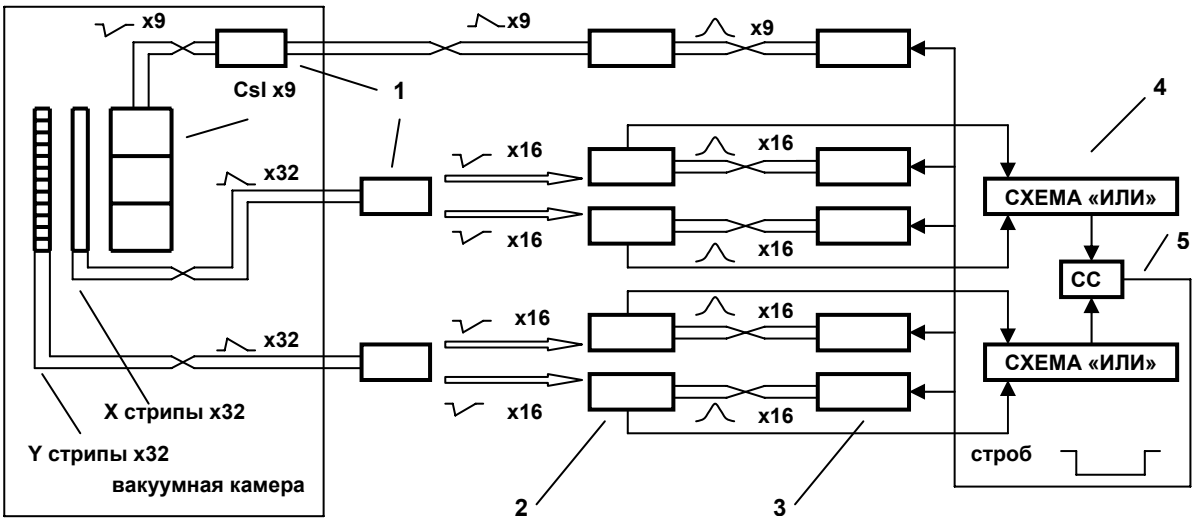

Рис.2. Многоканальная система сбора данных (DAQ) установки COMBAS: 1 – предусилители (ПУ32, ПУ9);2 – спектрометрические усилители (СУ-16); 3 – спектрометрические АЦП16;; 4 – схема «или»; 5 – схема совпадений (СС)

Визуализацией гистограмм должны заниматься клиенты этого сервера, которые будут запрашивать с него − однократно или «непрерывно», т.е. периодически через определенный интервал времени − только те из ранее «заказанных» для накопления гистограмм, которые соответствующие пользователи желают видеть в данный момент, чтобы не расходовать пропускную способность компьютерной сети и вычислительный ресурс клиентских компьютеров и не загромождать графические терминалы.

Такой подход легко реализуем на основе инфраструктуры *ngdp*. Мы организуем поток пакетов сырых данных в формате конкретной установки, как это подробно описано в [2] и [3], и выводим его в пользовательский контекст, так как модуль гистограммного сервера (будем далее называть его ***r2h(1)*** от «ROOT to histograms») реализован в *ngdp* в виде процесса. Это обусловлено невозможностью слинковать в ядро ОС, написанное на C, библиотеки пакета ROOT, написанные на C++. Для удобства работы с данными сначала следует преобразовать сырой формат в пособытийное ROOT−представление **`Ecombas`**, для чего *ngdp* предоставляет конвертер ***b2r(1)*** (от «binary to ROOT»). Класс **`Ecombas`** обязан предоставить для ***b2r(1)*** интерфейс **`pack2r(void *pack)`**. Последний и осуществляет преобразование пакета сырых данных, соответствующего некоторому триггеру установки, в инстанцию класса **`Ecombas`.** Процессы ***b2r(1)*** и ***r2h(1)*** связываются конвейером (pipe), через который сериализированные функцией-членом **`Streamer()`**, унаследованной от класса **`TObject`**, инстанции **`Ecombas`**'а передаются от первого ко второму, для удобства в виде пакетов специального типа.

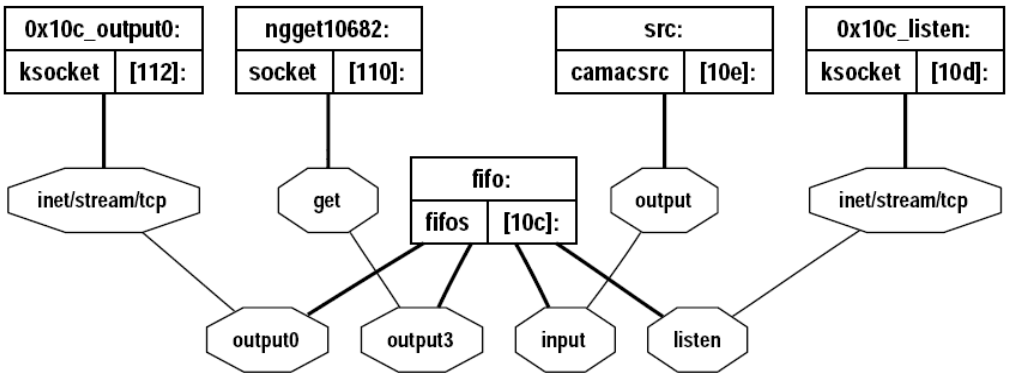

Рис.3. Граф системы ngdp, реализующий DAQ COMBAS: восьмиугольники – hook'и, прямоугольники – вершины графа

Наиболее дешевым способом сериализации нам представляется инкапсуляция в класс **`TBufferFile`**, существующий для ROOT версий старше 5.16. ***b2r(1)*** инстанцирует пишущий вариант класса **`TBufferFile`**, применяет к ROOT-представлению события **`TBufferFile::WriteObject(Ecombas *)`**, затем записывает в поток пакетный заголовок и в качестве тела −собственно сериализированный класс из буфера по указателю **`TBufferFile::Buffer()`**длиной

TBufferFile::Length()байтов. ***r2h(1)*** инстанцирует читающий вариант класса **TBufferFile** с телом пакета в качестве буфера длиной размер пакета минус размер заголовка, после чего применяет **TBuffeFile::ReadObject (TBufferFile::GetClass())**, получая указатель на полноценный объект, **Ecombas***. Как обычно для *ngdp*, поток (данных) между модулями ***b2r(1)*** и ***r2h(1)*** может вместо локального конвейера передаваться посредством межмашинного сокетного соединения, в данном случае **TSocket**, передающего **TMessage**', что позволяет легко распределить систему DAQ COMBAS. Для ***r2h(1)*** класс **Ecombas** обязан предоставить интерфейс **Get_adc(int chan, int mod, int group)**, позволяющий получить величину из события по номеру канала (и, возможно, блока и группы блоков) АЦП для гистограммирования. Этой величиной заполняется (**TH1F::Fill(x)**) одномерная, а парой таких величин (**TH2F::Fill(x,y)**) − двумерная гистограмма.

Гистограммный сервер ***r2h(1)*** открывает сокет на порту TCP 12341 и ждет запросов клиентов на регистрацию, которая может быть ограничена по адресам удаленных машин, на которых исполняются клиенты, двумя списками − полноправных и «только читающих». Список заполняемых гистограмм может быть «заказан» при запуске ***r2h(1)*** посредством файла в формате ***r2h.conf(5)***, указанного в командной строке опцией **-c**, а также позднее по командам от зарегистрированных (в полноправной моде) клиентов. При этом используется единый протокол, команды которого следующие:

Var с параметрами **varname chan mod group** − объявляет переменную с именем **varname**, которая может быть гистограммирована, в терминах триплета номеров: канала **chan**, модуля **mod** и группы модулей **group**. Предполагается, что реализация класса, представляющего событие, позволяет получить величину этой переменной в событии как результат, возвращаемый функцией-членом **Get_adc(chan, mod, group)** этого класса.

Add2var с параметрами **varname chan mod group** − добавляет еще один триплет номеров к уже существующему определению переменной **varname**, что позволяет объединять в одну гистограмму несколько каналов детектора.

Delvar с параметром **varname** − удаляет объявленную ранее переменную **varname** и освобождает соответствующие области памяти.

Book1d с параметрами **hname fullhname varnameX chansX minX maxX** − объявляет одномерную (**TH1F**) гистограмму ROOT с именем **hname**, идентификационной строкой **fullhname**, гистограммируемой переменной **varnameX** (должна быть уже объявлена), числом каналов **chansX**, минимальным **minX** и максимальным **maxX** каналами, которая будет заполняться по приходу каждого события в ROOT-представлении (запрещено в «только читающей» моде).

Book2d с параметрами **hname fullhname varnameX chansX minX maxX varnameY chansY minY maxY** − объявляет двумерную (**TH2F**) гистограмму ROOT с именем **hname**, идентификационной строкой **fullhname**, гистограммируемыми переменными **varnameX** и **varnameY** (должны быть уже объявлены), числами каналов **chansX** и **chansY**, минимальными **minX** и **minY** и максимальными **maxX** и **maxY** каналами, которая будет заполняться по приходу каждого события в ROOT − представлении (запрещено в «только читающей» моде).

Delete с параметром **hname** − удаляет объявленную ранее гистограмму **hname** и освобождает соответствующие области памяти (запрещено в «только читающей» моде).

Connect с параметрами **host [port]** − посылает запрос на присоединение к серверу на порту **port** на компьютере **host**.

Get с параметром **hname** − получает гистограмму с именем **hname** с сервера однократно.

Getcont с параметром **hname** − начинает получение гистограммы с именем **hname** с сервера в «непрерывной» моде.

Stop с параметром **hname** − прекращает получение гистограммы с именем **hname** в «непрерывной» моде.

Stopall (без параметров) − прекращает получение всех гистограмм в «непрерывной» моде.

Reset с параметром **hname** − очищает гистограмму с именем **hname** на сервере (запрещено в «только читающей» моде).

Resetall (без параметров) − очищает все гистограммы на сервере (запрещено в «только читающей» моде).

Quit (без параметров) − отсоединяется от сервера.

Формат ***r2h.conf(5)*** поддерживает также строки комментариев (**#** в первой позиции) и пустые строки, которые игнорируются. Команды **Getcont**, **Stop** и **Stopall** являются внутренними для клиента; команды **Var**, **Add2var** и **Delvar** в настоящий момент являются внутренними для сервера, поскольку объявление всех возможных переменных не ресурсоемко и, следовательно, может не меняться в течение сеанса; внутренние команды клиента, а также **Connect**, **Get**, **Reset**, **Resetall**, **Quit** не поддерживаются в файле.

Минимальный клиент реализован скриптом ROOT *client3.C* , что имеет ограничения (сложности реализации «непрерывной» моды и интерактивной работы с визуализированными гистограммами). Поэтому реализован также полномасштабный клиент ***histGUI(1)***, командная строка которого следующая: **histGUI [-l] [-tgui_sleep] [host [port]]**. После запуска ***histGUI(1)*** соединяется с сервером ***r2h(1)*** на порту **port** на компьютере **host** через **TSocket**, запускает **TTimer** для под-

держки «непрерывной» моды, отображает основное окно и входит в цикл обработки событий (events) ROOT. По получении команды оператора процессирует ее сам или запрашивает сервер, получает ответ и визуализирует гистограмму, либо отображает ответ сервера. Период обновления гистограмм в «непрерывной» моде задается опцией **`-t`** (по умолчанию 5 секунд). Опция **`-l`** означает вывод сообщений об ошибках в системный журнал вместо стандартного потока ошибок. Основное окно содержит как минимум кнопку (**`TButton`**) «Exit», поле ввода команд (**`TGTextEntry`**) и область вывода ответов (**`TGTextView`**). Кроме того, одно и то же окно (**`TCanvas`**) используется для однократного (по команде **`Get`**) вывода любой гистограммы, в «непрерывной» (по команде **`Getcont`**) моде для каждой гистограммы открывается отдельная **`TCanvas`**, закрывающаяся после команды **`Stopall`** или соответствующей команды **`Stop`**. Вероятно, в будущем возможно добавление системы меню в основное окно и дополнительных окон, отражающих специфику установки, для упрощения просмотра гистограмм.

ЛИТЕРАТУРА

1. K.I. Gritsaj, A.Yu. Isupov // *JINR Communications*. Dubna: 2001, E10−2001−116, p.1-19.
2. А.Ю. Исупов, В.Е. Ковтун, А.Г. Фощан. Построение систем сбора данных для многоканальных ядерно-физических установок на основе Unix-подобных операционных систем // *Вестник Харьковского университета. Серия физическая «Ядра, частицы, поля».* 2009, 845(1(41)), p.93-107.
3. А.Ю. Исупов, В.Е. Ковтун, А.Г. Фощан // *«NUCLEUS-2009. Fundamental problems and applications of nuclear physics: from space to nanotechnologies».* Cheboksary, Russia, June, 15-19, 2009.
4. К.И. Грицай, В.Г. Ольшевский // *Сообщения ОИЯИ*. Дубна: 1998, Р10−98−163, с.1.
5. *ООО «РиЭС»*. Харьков: Каталог, 2009, с.1-40.
6. R. Brun and F. Rademakers. ROOT - An object oriented data analysis framework // *Nucl.Instr.and Meth. in Phys.* 1997, v.A389, p.81-86. http://root.cern.ch/.
7. A.G. Artukh, G.F. Gridnev, M. Gruszecki, et al. Wide aperture kinematic separator COMBAS realized on the strong focusing principle // *Nucl. Instr. and Meth. in Phys.* 1999, v.A426, p.605-617.
8. 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.* 2005, v.A55, p.407-414.
9. A.Yu. Isupov // *JINR Communications*. Dubna: 2001, E10−2001−199, p.1-16.
10. A.Yu. Isupov // *JINR Communications*. Dubna: 2003, E10−2003−187, p.1-17.
11. С.В. Афанасьев, А.Ю. Исупов и др. Система сбора данных и триггер установки СКАН // *ПТЭ*. 2008, № 1, с.34-39.

Статья поступила в редакцию 07.09.2009 г.

THE DATA ACQUISITION SYSTEMS OF NEW GENERATION FOR MULTICHANNEL SPECTROMETER SETUPS

A.Yu. Isupov, V.E. Kovtun, A.G. Foshchan

The data acquisition, transportation and processing (DAQ) systems for INR NASU and COMBAS setup (JINR, FLNR) are proposed on base of *ngdp* framework and *camac* package (JINR, DLNP). ROOT dedicated class representing events online allows to implement the runtime configurable histogram server. Ones clients with graphic interface are independent on another DAQ so suitable for any operating system supporting ROOT.

СИСТЕМА ЗБОРУ ДАНИХ НОВОГО ПОКОЛІННЯ ДЛЯ СПЕКТРОМЕТРИЧНИХ МНОГОКАНАЛЬНИХ УСТАНОВОК

О.Ю. Ісупов, В.Є. Ковтун, А.Г. Фощан

Пропонуються розподілені системи збору, транспортування, обробки і зображення даних (DAQ) для ІЯД НАНУ і установки COMBAS (ЛЯР ОІЯД) на основі інфраструктури *ngdp* і пакета *camac* (ЛЯП ОІЯД). Зображення подій в online спеціальним ROOT-класом дозволяє реалізувати гістограмний сервер, що конфігурується під час виконання. Клієнти останнього з повномасштабним графічним інтерфейсом незалежні від інший DAQ і реалізуються під будь-яку операційну систему, що підтримує ROOT.