

Руководство по проектированию и стилю программирования в среде пакета CRW-DAQ

Оглавление

| | |
|---|----|
| Аннотация..... | 2 |
| Немного философии: о важности эргономики..... | 3 |
| Соглашения о наименовании (присвоении имен)..... | 4 |
| Общие принципы наименования объектов..... | 4 |
| Имя прикладной DAQ системы..... | 5 |
| Имена кривых, тегов, устройств и окон DAQ систем..... | 6 |
| Имена файлов DAQ систем..... | 7 |
| Имена переменных в программном коде DaqPascal..... | 7 |
| Рекомендуемый стиль программирования в DaqPascal..... | 9 |
| Стиль отступа при оформлении логических блоков..... | 9 |
| Стиль оформления логических блоков..... | 9 |
| Список обозначений..... | 16 |
| Список источников..... | 17 |

Аннотация

В данном документе содержатся рекомендации для членов команды (группы) **DaqGroup** - разработчиков прикладных автоматизированных систем управления (**АСУ**) в среде инструментального пакета **CRW-DAQ**[1,2,3]. Приведенные рекомендации служат для унификации и стандартизации программного кода и интерфейса пользователя.

Данный документ составлен с учетом длительного опыта разработки прикладного кода **АСУ** и многолетней практики программирования в среде пакета **CRW-DAQ**, а также на основании советов, взятых из классической книги «Практика программирования» [4], которую настоятельно рекомендуется прочитать вместе с данным документом.

Разработка и техническая поддержка **АСУ** в группе (**DaqGroup**) ведется коллективно. Способность группы вести разработку и техническую поддержку разработанных **АСУ** не должна критически зависеть от занятости и доступности (т. е. отпусков, командировок или больничных) её участников. Поэтому каждый участник группы должен стремиться к взаимозаменяемости, т. е. способность «подхватить» разработку коллеги или передать коллеге свой код для продолжения его разработки или поддержки. Поэтому каждый разработчик должен быть готов к работе с кодом, написанным другим разработчиком или передаче своего кода коллеге. Именно поэтому очень важно придерживаться принятых стандартов и рекомендаций по разработке. Это позволит сократить время, затрачиваемое на проектирование, разработку и поддержку кода, повысит его читабельность, а также снизит вероятность ошибок.

Применяйте рекомендуемый стиль написания кода и соглашения о наименовании (присвоении имен) при разработке прикладного кода. В этом случае вы получите единообразный вид кода, и тогда все участники группы смогут без труда работать с кодом, написанным коллегами, а сам код станет более простым и удобным в обслуживании.

Немного философии: о важности эргономики

Программы, управляющие работой **АСУ** физических установок, читают не только компьютеры, но и люди — программисты, инженеры, физики. Программы и алгоритмы управления проектируют, пишут, читают, дополняют, редактируют и обсуждают люди, специалисты с разной подготовкой и квалификацией. На заре развития компьютеров главным свойством программ казалось их быстрое действие. Но сегодня, когда компьютеры имеют высокое быстрое действие, зачастую важнее сосредоточить внимание на других качествах программ — их ясности, простоте, читаемости, гибкости, возможности технической поддержки и развития. Очень важно, чтобы людям было удобно эти программы читать, понимать и обсуждать.

Пользуются созданными программами тоже люди с самой разной подготовкой и квалификацией. Порой они работают с программами длительное время, в напряженном режиме. Очень важно, чтобы управляющие программы были удобны в использовании людьми — лаборантами, инженерами, физиками, операторами **АСУ**.

Поэтому очень важно учитывать при разработке соображения **эргономики** — науки о «человеческом факторе» которая определяется как *«Научная дисциплина, изучающая взаимодействие человека и других элементов системы, а также сфера деятельности по применению теории, принципов, данных и методов этой науки для обеспечения благополучия человека и оптимизации общей производительности системы»* [6]. По классификатору **ВАК** — это специальность 19.00.03 «Психология труда, инженерная психология, эргономика». Эта наука учитывает физиологические и психические свойства и особенности человека при его взаимодействии с техническими средствами.

Например, людям легче двигать глазами вверх-вниз, чем вправо-влево. Поэтому большие программы, которые приходится часто перелистывать, лучше делать не очень «широкими» (т. е. избегать длинных строк текста), чтобы не приходилось слишком часто «бегать глазами» вправо-влево. По этой же причине следует внимательно относиться к правилам форматирования текста — отступам, комментариям, стандартным конструкциям.

Кроме того, людям легче получать информацию небольшими порциями, чем одним большим куском. Поэтому хорошо структурированные программы, разбитые на небольшие, четко оформленные блоки (процедуры, функции) с интуитивно понятными именами, читаются и понимаются легче, чем огромные «простыни» неструктурированного текста.

Хорошее форматирование и структурирование программного кода — это далеко не мелочь. При большом объеме кода и дефиците времени и людей, часто сопровождающим создание **АСУ**, очень важно, чтобы разработчики могли быстро прочитать код программы и понять его. Этого невозможно добиться без учета эргономики. Хорошее форматирование и структурирование также резко снижает риск ошибок и трудоемкость их поиска, а потому повышает надежность программ. Поэтому так важно выработать для себя верный стиль кодирования (программирования) и привычку им постоянно пользоваться. Став привычкой, хороший стиль кодирования поможет вам работать успешнее и эффективнее.

Соглашения о наименовании (присвоении имен)

Общие принципы наименования объектов

При именовании программных объектов (прикладных систем, файлов, тегов, кривых, окон, переменных, процедур и функций) следует придерживаться некоторых общих принципов, облегчающих создание и поддержку АСУ:

- Имена объектов должны быть (по возможности) само-документированными, т. е. отражающими назначение и смысл именуемых объектов. Например, переменная `BufferSize` является само-документированной, т. к. её имя отражает назначение (размер буфера), а переменная `b123` - не является, т. к. её имя ничего не сообщает о назначении переменной. Само-документированные программы (в которых имена типов, переменных, процедур и функций само-документированы) легче читать и сопровождать, они требуют минимального количества комментариев и пояснений.
- Имена должны быть (по возможности) краткими. Например, переменную в предыдущем примере можно сократить до `BufSize`. Для уменьшения длины имен можно использовать сокращения или аббревиатуры. Для сокращений используют разные способы, например, начальные фрагменты слов (`Buf` вместо `Buffer`), либо слова без согласных букв (`Bfr` вместо `Buffer`). В некоторых типичных случаях (например, переменные цикла `i,j,k,...`) краткость имеет приоритет над другими соображениями.
- Имена должны быть (по возможности) легко произносимыми, т. к. иначе будет трудно обсуждать программный код с коллегами. Попробуйте произнести что-нибудь вроде `Bfszkqw_1p2`.
- Для конструирования имен можно использовать английские и русские слова или термины, а также распространенные аббревиатуры — лишь бы они были понятны разработчикам. При конструировании имен из слов следует использовать правильное (без ошибок) написание слов, т. к. грамматические ошибки создают избыточную неопределенность интерпретации и мешают пониманию кода.
- В именах, составленные из слов, рекомендуется выделять слова регистром (например, `BufferSize`) или разделителем (например, `buffer_size`). Это улучшает читабельность текста.
- Имена могут содержать латинские буквы, цифры и (возможно) разрешенные в текущем контексте знаки пунктуации (знак подчеркивания, точка, прямой или обратный слэш). Символы национальных алфавитов могут находиться в комментариях, но не рекомендуются в именах из-за их неоднозначной интерпретации в различных ОС (во избежание проблемы символьных кодировок).

В силу противоречивости требований к наименованию (трудно сделать имя само-документированным, легко произносимым и кратким одновременно), соглашения по

наименованию в прикладном коде обычно являются компромиссными. Тем не менее, разработчик должен держать в голове все перечисленные принципы наименования, сознательно отдавая предпочтение тому или иному критерию в зависимости от ситуации.

Имя прикладной DAQ системы

Имя прикладной **DAQ** системы должно быть (по возможности) предельно кратким (лучше всего не длиннее 3-4 символов), и (желательно) легко произносимым. Краткость имени нужна потому, что имя системы часто используется как префикс в составных именах, поэтому слишком длинное имя системы будет создавать проблемы.

Для именования драйверов какого-либо устройства обычно используется сокращенная форма названия предприятия (фирмы) производителя, номер модели устройства либо серии устройств, например:

- НПП ИНСИТЕК (INSITEK) → **INTEK**, AEROTECH → **ATECH**, MERADAT → **MERA**,
- TRINAMIC TMC-610 → **TMC610**, TDK-Lambda GENESYS series – **GENDC**,
- TDK-Lambda ZUP series → **ZUPDC**

Для именования систем управления для физических установок лучше всего использовать аббревиатуры, отражающие назначение этих установок, например:

- **FAHT** → Fabric of Automated Hydrogen Targets,
- **VACS** → Vacuum Control System,
- **TRS** → Tritium Retention System.

Аббревиатуры можно строить с использованием английских переводов, включая выражения "Система Управления" (Control System), например:

- **RTCS** → Система Управления Поворотным Столом (Rotating Table Control System),
- **ISCS** → Система Управления Источником Ионов (Ion Source Control System).

Также, для задания имени системы, допустимо использование транслитерации русских слов или аббревиатур латинскими буквами:

- **SPVZK** → СПВЗК,
- **TIR** → ТИР,
- **EGP** → ЭГП.

Следует избегать трудно произносимых аббревиатур, т. к. названия систем часто приходится обсуждать с коллегами вслух (например, по телефону), а трудно произносимые аббревиатуры затрудняют общение.

Имена кривых, тегов, устройств и окон DAQ систем

Обычно имена программных объектов DAQ системы, таких как кривые, теги, устройства и окна имеют общий вид **составного** имени, разделенного точками:

FFF.UUU.PPP...

где **FFF**, **UUU**, **PPP** — компоненты составного имени, а именно: название физической установки (Facility), функционального узла (Unit) и параметра (Parameter). Число компонент составного имени может быть (при надобности) увеличено. Составные имена удобны в работе, т. к. они легко интерпретируются и сочетают в себе краткость и емкость.

Название установки (**FFF**) — это фактически имя **DAQ** системы, рекомендации к выбору которого приведены выше.

Имя узла (**UUU**) обычно характеризует принадлежность к какой-либо функциональной подсистеме, например:

- **MAIN** — подсистема главного графического окна системы,
- **GAS** — подсистема, отвечающая за управление аппаратурой газовой арматуры (вентили, клапаны, насосы и пр.).

Имена устройств должны носить функциональный характер в зависимости от того, для чего это устройство предназначено, например:

- **GAS.MVTOTC** — программа перевода величин из одной единицы в другую (в данном случае, из милливольт в температуру) для газовой подсистемы;
- **MAIN.CTRL** — управляющая программа для главного графического окна системы.

Для имен окон графиков кривых (plot) или таблиц (table), параметр **PPP** указывает на тип данных, которые отображаются в этом окне, а в конце имени добавляется слово **PLOT** или **TAB**, например:

- **GAS.PRES.PLOT** — окно графика кривых давления (Pressure) газовой подсистемы;
- **GAS.TEMP.TAB** — окно таблицы температур (Temperature) газовой подсистемы.

В именах тегов и кривых, однородные параметры могут объединяться в группы с номерами, например датчики температуры:

- **GAS.SEN.TC1** — датчик (Sensor) термопары (ThermoCouple) №1;
- **GAS.SEN.TC2** — датчик (Sensor) термопары (ThermoCouple) №2.

Кроме того, часто в имени кривых и тегов указываются единицы измерения, особенно, если используются калибровочные преобразования из одной единицы в другую. Это позволяет отличать показания датчиков, выраженные в разных единицах измерения. Например, для датчика давления это может выглядеть так:

- **GAS.SEN.PS1.MV** — Pressure Sensor 1, millivolts

- **GAS.SEN.PS1.MA** – Pressure Sensor 1, milliampere
- **GAS.SEN.PS1.MBAR** – Pressure Sensor 1, millibar

Имена кривых, тегов, устройств и окон в конфигурационных файлах обычно пишутся в верхнем регистре (заглавными буквами), за редким исключением, например AdamTraffic. Строго говоря, это неважно, т. к. обычно имена не чувствительны к регистру. Однако по многим соображениям (например, поиск и/или замена имен в файлах) лучше использовать верхний регистр.

Для драйверов устройств, название узла **UUU** (Unit) в именах не обязательно, если этот драйвер работает автономно и не встраивается в качестве подсистемы в другую систему. Однако, если такой драйвер работает с несколькими устройствами, например драйвер **Modbus**, то узел **UUU** используется в качестве идентификатора для каждого устройства. Если драйвер будет включен в состав системы управления как подсистема, то имя основной системы будет **FFF**, а имя драйвера **UUU**. Конечно, можно и не добавлять имя системы, в составе которой будет работать драйвер, а включить его как есть. Но при реализации системы управления как распределенная сетевая система, например по протоколу **DIM** [5], принадлежность драйвера к конкретной системе нужно определять обязательно, чтобы не было конфликтов имен в сети **DIM**. Хорошей практикой является наличие генератора конфигурации для драйвера, в котором предусмотрена возможность добавлять префикс имен кривых, тегов и прочих объектов. Такой генератор обычно реализуется в виде пакетного (batch) командного файла (*.bat, *.cmd). Программный код **DaqPascal** при этом должен быть реализован без привязки к каким-либо конкретным именам (конкретизировать имена можно через секцию в файле конфигурации).

Имена файлов DAQ систем

Для имен файлов принимаются те же правила именования, но имеются некоторые различия: файлы именуются в нижнем регистре, вместо точек используется символ подчеркивания:

fff_uuu_ppp.ext

В некоторых случаях, например для программ-серверов или их конфигураций, допустимо использование таких имен:

UniHeat.cfg

AdamTraffic.cfg

Имена переменных в программном коде DaqPascal

Имена переменных должны быть краткими, понятными и простыми.

Переменные, которые хранят ссылку на тег, то есть связанные с тегом, или на другие программные объекты, необходимо записывать в верхнем регистре, чтобы отличать их от обычных переменных. Такие переменные должны сохранять структуру имени связанного с

ними объекта. Это же правило необходимо применять для имен констант, которые хранят номер цифрового или аналогового входа/выхода, объявленного в конфигурации устройства. Это важное требование для таких переменных. Для остальных есть лишь некоторые рекомендации:

- Не следует давать переменной слишком длинное имя;
- Не следует указывать в имени тип переменной;
- Не следует писать имена, состоящие из нескольких слов, строчными буквами, лучше выделить первые буквы заглавными, например AlarmList;
- В качестве разделителей слов в именах переменных можно использовать символ подчеркивания, например cmd_AlarmTooltip.

Рекомендуемый стиль программирования в DaqPascal

Стиль отступа при оформлении логических блоков

В качестве символа отступа используется одиночный пробел (single space) – « ». Использовать более длинный отступ (например, 2 или 4 пробела) не рекомендуется. При длинных отступах строки кода сложной программы получаются избыточно длинными (широкими) и код становится трудно читать, т. к. приходится часто прокручивать текст вправо-влево. Кроме того, в компиляторе **DaqPascal**, имеется ограничение на длину строки программного кода, при превышении которого компилятор выдает ошибку (текущее ограничение - до 161 символа).

Не следует использовать в качестве отступов символы табуляции, так как ширина табуляции зависит от текущих настроек редактора. Из-за этого на разных компьютерах или при использовании разных редакторов выравнивание текста может меняться и выглядеть неправильно.

При форматировании вложенных блоков надо следить за тем, чтобы отступы соответствовали вложенности блоков (**begin...end**), то есть чтобы каждый открывающий оператор (**begin**, **if**, **while** и т. д.) имел такой же отступ, что и закрывающий его оператор (**end**). Например:

```
procedure Print(Message:String);
begin
  if (Message<>'') then begin
    Writeln(Message);
  end;
end;
```

Приведенное выше правило отступов улучшает читабельность и понимание структуры программы, т. к. сразу видно, где начинается и кончается каждый блок кода. Это также способствует снижению риска ошибок в структуре алгоритмов из-за нарушений порядка вложенности блоков кода. Такие ошибки бывает очень трудно обнаружить, если код отформатирован с нарушением отступов.

Стиль оформления логических блоков

При оформлении условных операторов ветвления и логических блоков, рекомендуется придерживаться следующего стиля:

`if · (condition) · then · (operator1)`

`else · (operator2);`

или:

`if · (condition1) · then · (operator1)`

`else · if · (condition2) · then · (operator2);`

или:

```
if · (condition1) · then · (operator1)
else · if · (condition2) · then · (operator2)
else · (operator3);
```

Примеры оформления логических блоков, ограниченных ключевыми словами begin и end:

```
if · (condition) · then · begin
· (body);
end;
```

или:

```
if · (condition) · then · begin
· (body);
end · else · begin
· (body);
end;
```

или:

```
if · (condition1) · then · begin
· (body);
end · else
if · (condition2) · then · begin
· (body);
end · else · if · (condition3) · then · begin
· (body);
end · else · begin
· (body);
end;
```

В таких конструкциях следует выравнивать все ключевые слова else по одной вертикальной линии, а не ставить их в соответствие операторам if. Такое вертикальное выравнивание подчеркивает, что выполняется последовательная проверка условий.

Стиль оформления комментариев

```
{
```

Комментарий для глобальных или значимых процедур/функций

```
}
```

или:

```
//
```

```
// · Комментарий для вложенных процедур/функций
```

```
//
```

или:

```
{ · Обычный комментарий... }
```

```
// · Обычный комментарий...
```

Комментарии могут быть как однострочные, так и многострочные.

Прочие рекомендации по улучшению читаемости кода

- Не следует использовать пробелы при оформлении логических и арифметических операций. Так строка выглядит компактней, цельной и не создает впечатление разорванного текста, например:

```
if (a>b) then c:=a+b;
```

вместо такой записи:

```
if ( a > b ) then c := a + b;
```

- Не следует объявлять глобальные переменные для использования их в качестве временного хранения значений или для организации циклов в процедурах и функциях. Делайте это локально в секции объявления переменных той процедуры или функции, где они требуются.
- Тип переменных при объявлении рекомендуется записывать с заглавной буквы, например:

```
AlarmList::Integer;  
AlarmTime::Real;
```

- Для лучшей читаемости кода, при множественных вызовах одной процедуры или функции с разными аргументами, следует делать выравнивание текста по аргументам, например:

```
InitTag(first.tag, 'first', 1); // First tag
```

```
InitTag(second.tag, 'second', 2); // Second tag
```

```
InitTag(third.tag, 'third', 3); // Third tag
```

Рекомендации по проектированию графического интерфейса пользователя

Использование стандартных элементов интерфейса, шрифтов и палитры цветов позволяет унифицировать внешний вид мнемосхем, придает им единообразный стиль. Применяя описанные рекомендации, пользователю не придется переучиваться и привыкать к интерфейсу систем, написанных разными разработчиками.

Типографика

ParaType (PT) – это основной шрифт в пакете CRW-DAQ. Использование основного шрифта обеспечивает удобочитаемость, четкость и согласованность текста на мнемосхемах прикладных систем. Семейство шрифтов PT входит в состав пакета, а также доступно для инсталляции в виде установочного файла `install-daqgroup-fonts.exe`, входящего в состав дистрибутива.

При проектировании интерфейсов придерживайтесь следующих рекомендаций по использованию шрифтов PT:

PT Mono используйте для числовых полей (LED) измеренных данных, а также для всех полей ввода и вывода.

PT Sans используйте для написания поясняющих надписей.

PT Sans Narrow используйте для длинных надписей и для экономии места на мнемосхемах и в строковых полях.

Подчеркнуть наиболее важную информацию на мнемосхеме можно при помощи веса, размера и цвета шрифта.

Шрифт PT Mono, кроме того, рекомендован для редактирования исходного программного кода.

Цвет

Цвет — отличный способ передать информацию о состоянии, дать обратную связь в ответ на действия пользователя, а также помогает визуализировать данные.

Основные цвета. В CRW-DAQ имеется ряд стандартных цветов, которые рекомендуется использовать в прикладных системах. Применение стандартных цветов позволяет точно и однозначно определить предназначение каждого поля на мнемосхеме и дает понять текущее состояние системы.

Silver (RGB: C0C0C0). Серебристый или светло-серый цвет является основным цветом для фона мнемосхем. Также он применяется в следующих случаях:

- Неактивное поле ввода или вывода, например, при выключенном опросе или измерении;
- Поле вывода информационных данных или данных, не содержащих потенциально важную информацию;
- Кнопка в обычном или выключенном состоянии.

Gray (RGB: 808080). Серый цвет является акцентным цветом для серебристого и применяется для разделителей или границ элементов, например:

- Разделитель заголовка системы от основного поля мнемосхемы;
- Разделитель любых данных на мнемосхеме;
- Выделение нескольких элементов интерфейса в один блок или группу;
- Границы полей ввода/вывода или любых других полей.

Aqua (RGB: 00FFFF). Цвет морской волны или голубой цвет применяется к полям вывода измеренных данных, например: ток, напряжение, температура, давление и прочее. Следует отметить, что цвет Aqua применяется только в том случае, если измеренные данные имеют важное значение для пользователя. В остальных случаях для таких полей рекомендуется использовать цвет Silver.

Yellow (RGB: FFFF00). Сценарий использования желтого цвета достаточно разнообразен. Если рассматривать его в качестве сигнального цвета, то он прежде всего обозначает возможную опасность, опасную ситуацию или предупреждает о возможной опасности. Его следует применять в следующих случаях:

- Превышение предупредительной пороговой уставки измеренного сигнала, например объемной активности или давления;
- Промежуточное состояние какого-либо элемента, например промежуточное состояние вентиля, когда он открывается или закрывается;
- Промежуточное значение сигнала при его установке по линейному изменению (ramp), например плавное изменение значения тока на выходе источника питания;
- Ошибки приемо-передачи, происходящие в настоящий момент, например таймауты при опросе устройства драйвером;
- Низкая частота опроса устройства программой-драйвером.

Lime (RGB: 00FF00). Зеленый или лимонный цвет применяется в следующих случаях:

- Нормальная работа оборудования, например состояние опроса с хорошей частотой и без ошибок связи;

- Нормальное или допустимое значение измерения, например значение объемной активности в допустимых пределах, ниже предупредительной пороговой уставки;
- Активное состояние какого-либо элемента, например: открытый клапан, включенный выход источника питания и прочее.

Red (RGB: FF0000). Красный цвет применяется в основном для информирования об ошибках, аварийных или опасных ситуациях. Использовать красный цвет не по назначению недопустимо. Применяйте красный цвет в следующих случаях:

- Действия при аварийных ситуациях, например: кнопка аварийной остановки, аварийного выключения и прочее;
- Превышение аварийной пороговой уставки измеренного сигнала, например объемной активности или давления;
- Ошибки в работе оборудования. Некоторые устройства имеют возможность регистрировать ошибки во время работы, которые можно фиксировать драйвером этого устройства.
- Разрыв связи (refuse) при опросе какого-либо устройства драйвером.

Maroon (RGB: 800000). Бордовый цвет применяется к цвету шрифта для имени системы при ее наличии на мнемосхеме.

White (RGB: FFFFFFFF). Белый цвет применяется к полям ввода данных.

Дополнительные цвета

Кроме стандартных цветов, при проектировании пользовательского интерфейса можно использовать нестандартные цвета. Для задания таких цветов рекомендуется использовать именованные цветовые константы палитры Windows, Web (HTML) и X11 (Linux), встроенных в пакет CRW-DAQ.

Пошаговое руководство по созданию Daq-систем в пакете CRW-DAQ

Разработку прикладной Daq-системы можно вести по-разному, например, можно выделить два способа, назовем их так: разработка методом сверху вниз и методом снизу вверх. Метод сверху вниз подразумевает разработку системы в следующем порядке: сначала прорабатывается мнемосхема, рисуется bmp файл изображения, затем описываются сенсоры этой мнемосхемы в src-файле, разрабатываются алгоритмы управления элементами мнемосхемы на языке DaqPascal, после чего добавляются конфигурационные cfg-файлы с описанием аппаратной части системы и симулятор для неё.

При разработке методом снизу вверх, порядок обратный: сначала описывается аппаратная часть системы, симулятор аппаратуры, затем пишутся алгоритмы управления и только потом все это визуализируется на мнемосхеме.

Бывает и так, что исходные данные не полные или часто меняются, в таком случае, разработка прикладной системы принимает смешанный вид, когда приходится постоянно возвращаться к конфигурированию аппаратной части, редактированию алгоритмов управления или перерисовыванию мнемосхемы.

Для драйверов отличие лишь в том, что на конфигурирование аппаратуры требуется значительно меньше усилий, и чаще все сводится к написанию Batch файла – генератора конфигурации. Кроме того, драйвер исполнительного устройства может и не иметь пользовательского графического интерфейса (GUI) либо GUI может быть реализован другими средствами, не CRW-DAQ, например DieselPascal.

В качестве примера рассмотрим создание простой системы для измерения температур с использованием модуля аналогового ввода ADAM I-7018.

...

Список обозначений

| Обозначение | Расшифровка |
|------------------------|--|
| ОС OS | Операционная система Operation system |
| АСУ | Автоматизированная система управления |
| DAQ | Data AcQuisition (сбор данных) |
| | |
| | |

Список источников

1. А.В.Курякин, Ю.И.Виноградов. Программа для автоматизации физических измерений и экспериментальных установок (CRW-DAQ). // Свидетельство РФ об официальной регистрации программы для ЭВМ № 2006612848 от 10.08.2006 г. Сайт программы: <http://crw-daq.su>.
2. А.В.Курякин. Автоматизация физических экспериментов на тритиевых комплексах исследовательских установок «ТРИТОН», «АКУЛИНА» и «ПРОМЕТЕЙ». Автореферат диссертации на соискание степени кандидата физико-математических наук, Саров, 2010. <http://ftp.jinr.ru/dissertation/Kuryakin.pdf>.
3. А.В.Курякин, Ю.И.Виноградов. Программное обеспечение автоматизированных измерительных систем в области тритиевых технологий. // ВАНТ, серия «Термоядерный синтез», 2008 г., выпуск 2, с.80-90.
4. Б.Керниган, Р.Пайк. Практика программирования.
<http://bioinformatics.ru/Books/kernigan.html>
5. Distributed Information Management system (DIM).
<https://dim.web.cern.ch/>
6. Что такое эргономика. <https://iea.cc/what-is-ergonomics/>