ГРУППА КОМПАНИЙ ПО ПРОИЗВОДСТВУ
ОБОРУДОВАНИЯ АЗС И НЕФТЕБАЗ

ОАО ПРОМПРИБОР

УНИВЕРСАЛЬНЫЙ ДРАЙВЕР ОБОРУДОВАНИЯ

Руководство программиста

RU.05806720.00002-01 33 01

СОДЕРЖАНИЕ

1 ОБЩИЕ ПОЛОЖЕНИЯ 5

1.1 Состав драйвера 6

2 АППАРАТНЫЕ И ПРОГРАММНЫЕ ТРЕБОВАНИЯ 7

3 УСТАНОВКА И НАСТРОЙКА ДРАЙВЕРА 9

3.1 Порядок установки драйвера на компьютере конечного пользователя 9

3.2 Настройка драйвера 9

3.2.1 Вкладка «Общая настройка» 10
3.2.2 Вкладка «OPC Data Access» 11
3.2.3 Вкладка «Разрешения» 11
3.2.4 Вкладка «COM-порты» 13
3.2.5 Вкладка «О программе» 14

3.3 Порядок удаления драйвера 14

3.4 Действия в случае возникновения проблем в работе драйвера 15

3.4.1 Действия в случае отсутствия связи с оборудованием 15
3.4.2 Действия в случае сбоев драйвера 16
3.4.3 Обращение к разработчику 17
3.4.4 Действия в случае ошибок чтения и записи переменных клиентским приложением 17

4 ВЗАИМОДЕЙСТВИЕ ДРАЙВЕРА С УПРАВЛЯЕМЫМ ОБОРУДОВАНИЕМ 18

5 ВЗАИМОДЕЙСТВИЕ ДРАЙВЕРА С КЛИЕНТСКИМИ ПРИЛОЖЕНИЯМИ ПО СТАНДАРТУ OPC DA 19

5.1 Адресное пространство OPC-сервера для управления драйвером 21

5.2 Адресное пространство OPC-сервера для оборудования отпуска жидкостей 21

5.3 Адресное пространство OPC-сервера для контроллера ЦБУ в пассивном режиме работы 29

5.3.1 Корневая ветвь 29
5.3.2 Ветвь логических входов и силовых выходов 29

5.4 Адресное пространство OPC-сервера для терминалов «ТС-001», «ТС-001Ex», «ТС-002» 30

5.4.1 Корневая ветвь 30
5.4.2 Ветвь Display 30
5.4.3 Ветвь Card 31
5.4.4 Ветвь Key 32
5.4.5 Ветвь MemoDisplay 32
5.4.6 Ветвь InOut 32

5.5 Адресное пространство OPC-сервера для плотномера «Плот-3М» 32

5.6 Адресное пространство OPC-сервера для информационного табло 33

6 ВЗАИМОДЕЙСТВИЕ ДРАЙВЕРА С КЛИЕНТСКИМИ ПРИЛОЖЕНИЯМИ ПО ИНТЕРФЕЙСУ IPROMPRIBOR 33

6.1 Типы данных библиотеки PrompriborDrv.idl 34

6.1.1 TStateEnum 34
6.1.2 TVariableEnum 35
6.1.3 TState 38
6.1.4 TVariableValue 38
6.1.5 TVariableInfo 39

6.2 Интерфейс IPrompribor 40

6.2.1 IPrompribor::StateJumpValid 40
6.2.2 IPrompribor::GetState 40
6.2.3 IPrompribor::SetState 40
6.2.4 IPrompribor::ReReadVarValue 40
6.2.5 IPrompribor::GetVariablesCount 40
6.2.6 IPrompribor::GetVariable 40
6.2.7 IPrompribor::SetVarValue 40
6.2.8 IPrompribor::GetVarValue 40
6.2.9 IPrompribor::TryResult 41
6.2.10 IPrompribor::GetResult 41
6.2.11 IPrompribor::GetVarInterval 41
6.2.12 IPrompribor::SetVarInterval 41
6.2.13 IPrompribor::BandUsed 41
6.2.14 IPrompribor::RegisterSinkInterface 41
6.2.15 IPrompribor::RegisterSinkInterfaceEx 41
6.2.16 IPrompribor::UnRegisterSinkInterface 41

6.3 Интерфейс IPrompriborEvents 42

6.3.1 IPrompriborEvents::PostFound 42
6.3.2 IPrompriborEvents::StateChanged 42
6.3.3 IPrompriborEvents::VarValueChanged 42
6.3.4 IPrompriborEvents::PostDeleted 42

6.4 Интерфейс IPrompriborEventsEx 42

6.4.1 IPrompriborEventsEx::PostWithoutChanges 42
Лист регистрации изменений…… 43

Драйвер предназначен для упрощения процесса создания программного обеспечения (далее называемого клиентским), управляющего оборудованием дозированного отпуска жидкостей, производимым ОАО "Промприбор". Также упрощается обновление существующего программного обеспечения с целью обеспечения поддержки указанного оборудования. Кроме того, драйвер поддерживает подключение SCADA и HMI систем посредством интерфейса OPC DA (OLE for Process Communication Data Access) версии 2.05.

1 Общие положения

В комплекте поставки драйвера включены утилиты-эмуляторы оборудования отпуска жидкостей, что позволяет разрабатывать и тестировать клиентское программное обеспечение (ПО) без необходимости наличия реального оборудования. Разработанное с использованием эмуляторов клиентское ПО переносится на реальное оборудование без изменений.

Драйвер реализует связь с клиентским ПО, также как и со SCADA системами, посредством Windows COM интерфейсов, что позволяет поддерживать следующие возможности:

- клиентское приложение может быть создано в любой среде разработки, поддерживающей COM;
- достаточно просто добавить поддержку нового оборудования в существующее программное обеспечение;
- клиентское приложение может исполняться как на компьютере, к которому непосредственно подключено оборудование, так и использовать встроенные в Windows возможности подключения по локальной сети и через Интернет (в этом случае требуется настройка безопасности DCOM, а также брэндмауэра Windows XP);
- одновременное подключение практически неограниченного количества клиентских приложений разного типа к одному исполняемому экземпляру драйвера;
- в дальнейшем возможна доработка драйвера для поддержки дополнительных стандартов связи клиентских программ с драйвером;
- при обновлении драйвера с целью поддержки нового оборудования существующие интерфейсы прикладного ПО не изменяются (изменяется только реализация интерфейсов), т.е. обновление существующего ПО потребуется только с целью добавления в ПО принципиально новых возможностей оборудования и драйвера.

Поддерживаемые драйвером COM-интерфейсы разделяются на два типа:

- серверные интерфейсы или интерфейсы прямой связи – реализованы драйвером и OPC DA сервером и вызываются клиентским ПО при необходимости чтения/записи параметров управляемого оборудования;
- клиентские интерфейсы или интерфейсы обратной связи (оповещения) – необязательные интерфейсы, которые могут быть реализованы клиентским ПО, и, в случае их реализации, вызываемые драйвером при изменениях состояния и параметров оборудования.

Использование интерфейсов оповещения позволяет избежать необходимости для клиентского ПО вести постоянный циклический опрос драйвера с целью определения изменения состояния оборудования, и, соответственно, уменьшает нагрузку на процессор компьютера, особенно в тех случаях, когда клиентских процессов, подключенных к драйверу одновременно, несколько. При использовании оповещения можно без лишних трудностей составить приложение, ориентированное на события (такие, как окончание отпуска, изменение состояния счетчиков и т.п.). Для этого достаточно выполнить три действия: подключиться к драйверу, зарегистрировать у него интерфейс оповещения и считать начальные значения параметров. Драйвер гарантирует, что после выполнения указанных действий данное клиентское приложение будет постоянно иметь актуальные значения всех параметров оборудования.

1.1 Состав драйвера

Драйвер логически представляет собой два модуля:

- собственно драйвер, поддерживающий постоянную связь с оборудованием, формирующий буфер памяти параметров оборудования, реализующий интерфейс IPrompribor для доступа клиентских приложений и OPC DA сервера к чтению и записи параметров оборудования, а также оповещающий клиентские приложения об изменениях параметров посредством вызова интерфейсов IPrompriborEvents и IPrompriborEventsEx;

- OPC-сервер, взаимодействующий с собственно драйвером для чтения-записи параметров оборудования, реализующий DCOM-интерфейсы согласно спецификации OPC Data Access Custom Interface ver. 2.05 (приложение №1), которые позволяют доступ к оборудованию со стороны SCADA-систем и других OPC DA клиентов, поддерживающих указанную спецификацию.

Указанные модули выполняются в одном процессе в качестве службы Windows. Структурная схема взаимодействия программного обеспечения между собой и оборудованием представлена на рисунке 1.

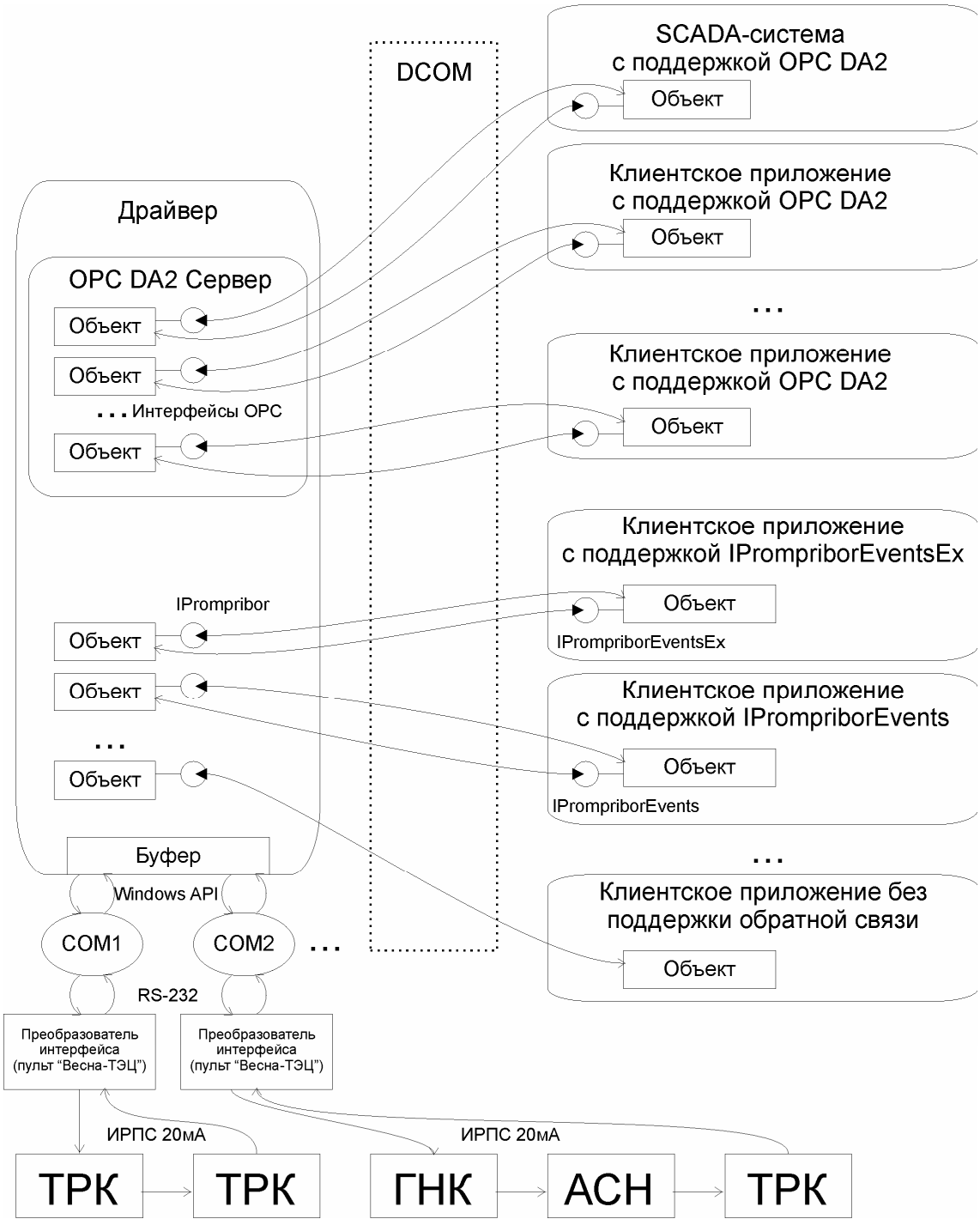

Рисунок 1 - Структурная схема взаимодействия прикладного ПО, драйвера и оборудования

2 Аппаратные и программные требования

Драйвер поддерживает следующее оборудование:

- топливораздаточные колонки «Ливенка», 1КЭД, 2КЭД со встроенной гидравликой, управляемые контроллерами КУП-1, КУП-2 (требуется версия встроенного ПО не ниже 23);

- топливораздаточные колонки «Ливенка», 1КЭД, 2КЭД с выносной гидравликой, управляемые контроллерами КУП-10 (требуется версия встроенного ПО не ниже 41);

- узлы учета и счетчики жидкости, управляемые контроллером КУП-30 (требуется версия встроенного ПО не ниже 0С);

- комплексы налива авто-цистерн и ЖД-цистерн, управляемые контроллерами КУП-40 и/или ЦБУ;

- терминалы «ТС-001», «ТС-001Ex»;

- плотномер «Плот-3М», производства ЗАО «Импульс-Авиа» (только вариант с использованием протокола Modbus RTU);

- информационное табло бегущая строка производства ЗАО "СНИИП-Конвэл" (вариант с использованием протокола СНИИП-КОНВЭЛ).

При наличии устаревшей версии ПО контроллеров возможно его обновление. Информацию по обновлению ПО контроллеров можно получить в конструкторских отделах АО «Промприбор».

Драйвер не поддерживает оборудование, которое разработано позже выпуска драйвера, по причине обратной несовместимости нового оборудования с предыдущими версиями. При наличии нового оборудования требуется обновление драйвера. Для этого обратитесь в отдел АСУ АО «Промприбор».

Демонстрационная версия драйвера ограничена по функциональным возможностям. Она, в отличие от полной версии, не поддерживает команду полного останова отпуска (команда временного останова поддерживается), т.е. при использовании демонстрационной версии драйвера начатый отпуск может быть закончен только по окончании заданной дозы либо путем выключения и включения электропитания контроллера. Использовать демонстрационную версию драйвера для работы на компьютере конечного пользователя не допускается.

На работу с терминалами «ТС-001», «ТС-001Ex» демонстрационная версия накладывает следующие ограничения: для работы с индикатором доступна только верхняя строка, не поддерживается функция персонализации карт и считывателей, а также невозможно сменить группу и тип ключей, по которым происходит аутентификация карты со считывателем.

Подключение оборудования к компьютеру производится посредством последовательных RS-232 или RS-485 (COM) портов. Поддерживается использование от 1 до 32 портов на одном компьютере.

Аппаратные требования к компьютеру:

- процессор не ниже Pentium MMX;

- оперативная память не меньше 32Мб.

Возможно функционирование драйвера также при несоответствии данным требованиям, но с увеличенным временем реакции на события.

Поддерживаемые операционные системы:

- Windows NT 4.0 SP6;
- Windows 2000 SP4;
- Windows XP SP2;
- Windows Vista SP1;
- Windows 7 SP1;
- Windows Server 2003;
- Windows Server 2008.

Операционные системы Windows 3.x, 9x, Me не поддерживаются, т.к., по причине наличия в их архитектуре существенных недостатков, данные ОС применимы только для домашнего использования.

В данном описании далее подразумевается базовое знание читателем основных принципов функционирования технологий COM/DCOM, а в разделах, посвященных OPC DA серверу, кроме того, представление о стандарте OPC Data Access 2.05. Детальную информацию о данном стандарте можно получить на http://www.opcfoundation.org (англоязычный сайт).

3 Установка и настройка драйвера

3.1 Порядок установки драйвера на компьютере конечного пользователя

Установку драйвера можно произвести с использованием прилагаемого установочного пакета. Для этого необходимо запустить на выполнение файл *Setup.EXE* и следовать его дальнейшим инструкциям.

Если планируется осуществить связь клиентского ПО с драйвером по локальной сети, требуется установить на всех клиентских машинах с помощью того же файла *Setup.EXE* клиентские компоненты драйвера, а также при установке на всех машинах, включая сервер, указать опцию «Производить настройку брандмауэра». Для поддержки вызовов по локальной сети COM-интерфейсов клиентского ПО (***IOPCDataCallback***, ***IPrompriborEvents***, ***IPrompriborEventsEx***) необходимо после завершения работы *Setup.EXE* с помощью инструментов Windows настроить запуск службы драйвера «Prompribor hardware monitoring service» под учетной записью, действительной на всех машинах, где выполняется клиентское ПО. Иначе вызов драйвером указанных интерфейсов будет идентифицирован клиентской машиной как анонимный, и поскольку анонимный доступ с COM-объектам в Windows обычно запрещен (настройка безопасности по умолчанию), доступ к интерфейсам обратной связи будет запрещен. По умолчанию, служба драйвера выполняется под учетной записью LocalSystem, который может быть недействительным на других машинах.

Драйвер устанавливается и исполняется в качестве службы Windows. Используется режим запуска «Вручную», Запуск службы выполняется системой DCOM автоматически при запросе связи с драйвером со стороны клиентского ПО. Драйвер автоматически останавливается через заданное время после корректного закрытия всех клиентских процессов (освобождения COM-интерфейсов). При аварийном закрытии клиентских процессов COM освобождает интерфейсы драйвера в течение нескольких минут.

3.2 Настройка драйвера

Настройка сохраняется в реестре Windows в ключе «HKLM\SOFTWARE\Prompribor\Drv» в виде строкового значения «Parameters». Настройка драйвера осуществляется при помощи утилиты *PrompriborDrvCfg.EXE*. Вид утилиты настройки показан на рисунке 2. Чтение и запись конфигурации в реестре происходит автоматически при выборе мышью вкладок «Общая настройка», «Настройка COM-портов», «Разрешения», и при переходе между ними.

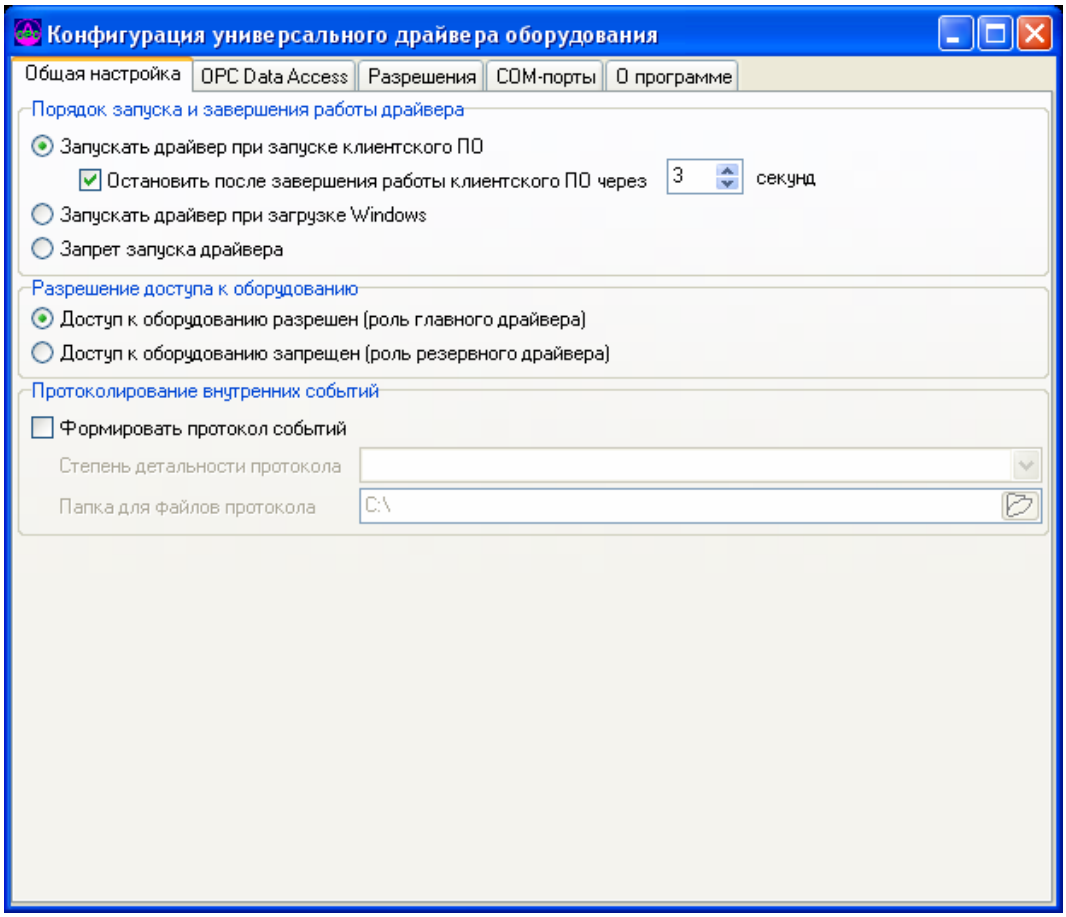

Рисунок 2 - Внешний вид конфигуратора драйвера оборудования

3.2.1 Вкладка «Общая настройка»

На этой вкладке можно настроить следующие параметры:

- разрешение доступа к оборудованию (при использовании в качестве резервного сервера можно запрещать драйверу работать с физическими портами компьютера);

- способ запуска драйвера (автоматический при загрузки Windows или при запуске клиентского ПО);

- уровень протоколирования внутренних событий драйвера, а также путь для файлов отчета. Существуют следующие уровни:

  - 1 – вести запись только внутренних сбоев в драйвере, их появление возможно крайне редко;
  - 2 – вести запись также проблем, возникающих из-за внешних причин – ошибок контроллеров, клиентского ПО, недостаточности системных ресурсов;
  - 3 – вести запись также обычных событий, встречающихся нечасто;
  - 4 – вести запись всех событий, в том числе запросы, передаваемые в контроллеры оборудования, и ответы на них. В данном режиме, в отличие от других, возможно однозначное определение не только вида ошибки, но и причины возникновения. Однако объем файлов протоколов может быть большим – до 5Мб за 1 час работы с одним COM-портом.
  - В случае включении опции протоколирования необходимо указать пустую папку для записи протоколов. Наличие пробелов в имени папки и пути к ней не допускается.

3.2.2 Вкладка «OPC Data Access»

Внешний вид вкладки показан на рисунке 3

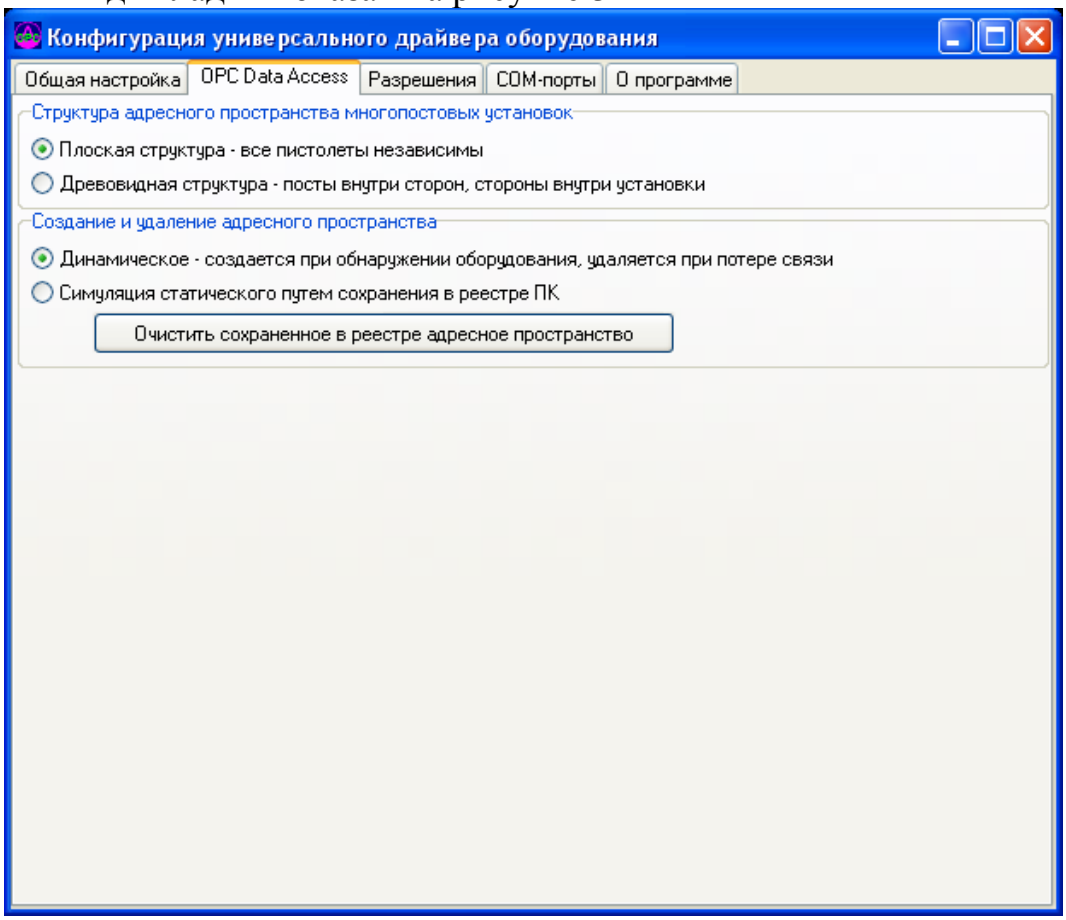

Рисунок 3 - Настройка адресного пространства сервера

Здесь выбирается параметры, связанные с адресным пространством сервера:

-вид структуры переменных OPC DA для ТРК;

-метод создания адресного пространства сервера. Если выбрано динамическое создание адресного пространства, то переменные появляются только при обнаружении оборудования. Если оборудование отсутствует, переменные удаляются из адресного пространства. Если выбран статический метод, переменные хранятся в реестре и добавляются при старте сервера в адресное пространство, даже при отсутствии оборудования.

3.2.3 Вкладка «Разрешения»

Внешний вид вкладки показан на рисунке 4.

Здесь можно произвести настройку безопасности DCOM

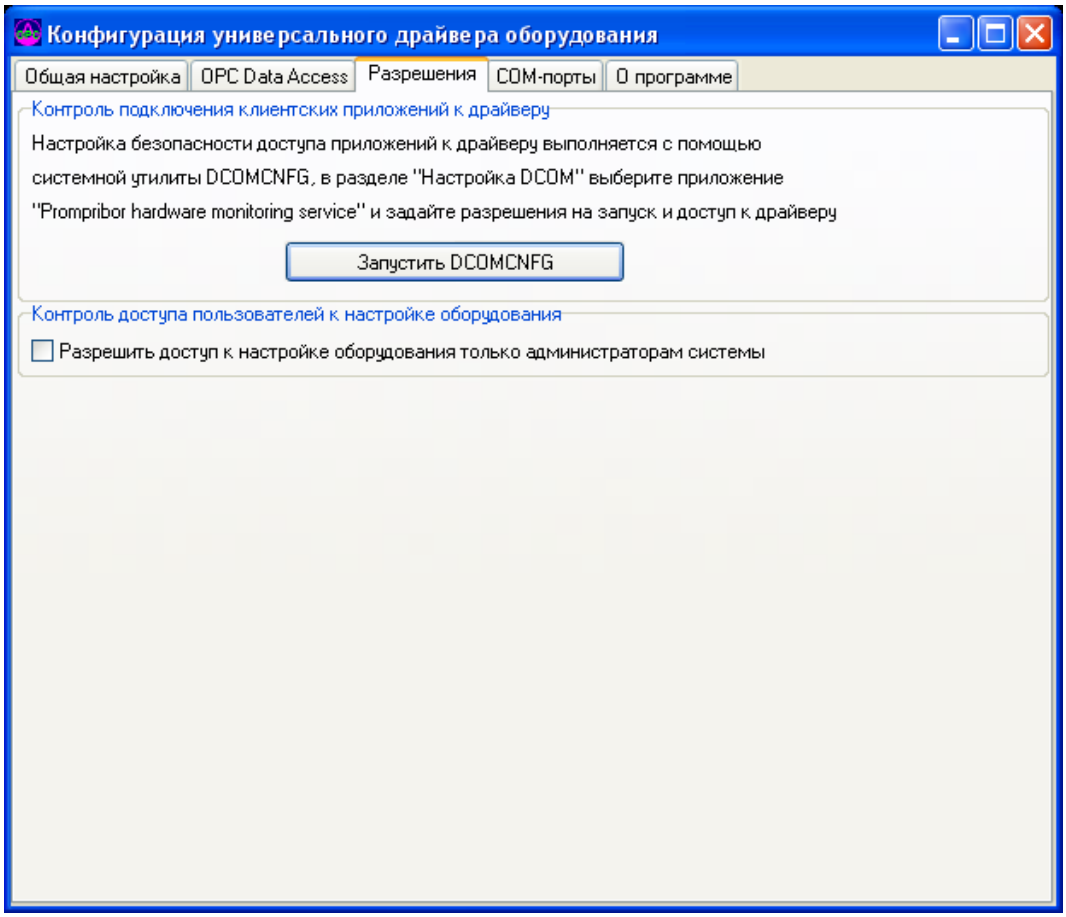

Рисунок 4 - Настройка политики безопасности

Опция «Разрешить доступ к настройке оборудования только администраторам системы» позволяет запретить пользователям ПК, не имеющим административных привилегий Windows, изменять значения свойств оборудования, влияющих на его работоспособность или метрологические характеристики.

3.2.4 Вкладка «COM-порты»

Внешний вид вкладки показан на рисунке 5.

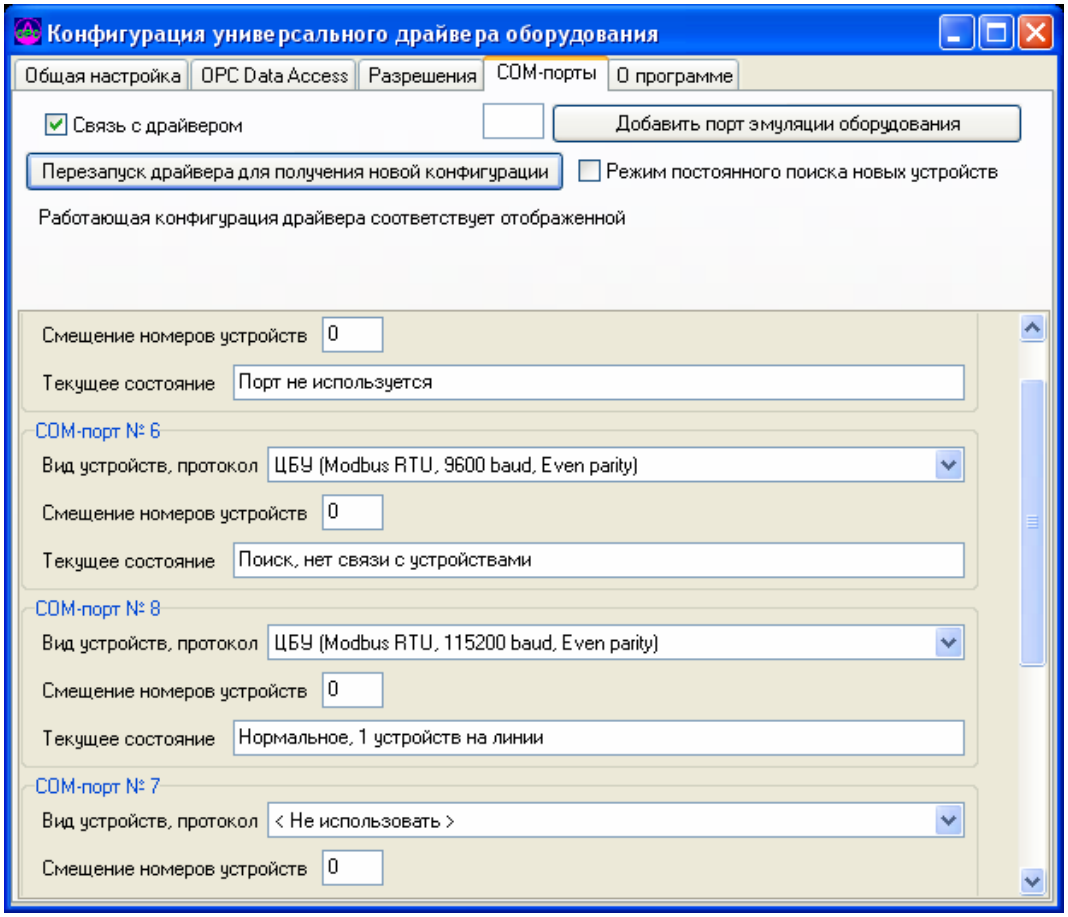

Рисунок 5 - Настройка параметров COM-портов

Здесь выбирается режим работы драйвера по каждому из присутствующих в системе коммуникационных. Каждый порт может иметь смещение, которое прибавляется к физическим адресам оборудования, подключенного к данному порту, после чего формируется номер поста. Это смещение рекомендуется использовать только для предотвращения конфликта имен устройств OPC DA в том случае, когда к двум различным портам подключены устройства одинакового типа с одинаковыми номерами постов. Таким образом, номера постов в таком случае зависят от номера порта.

Драйвер поддерживает работу виртуальных портов, к которым можно подключать виртуальные устройства, эмулируемые с помощью входящих в комплект поставки программ-эмуляторов. Эта возможность используется для тестирования программного обеспечения при отсутствии реального оборудования. Для добавления в конфигурацию виртуального порта необходимо ввести в поле редактирования его номер и нажать кнопку «Добавить порт эмуляции оборудования».

Если установить флажок «Связь с драйвером», программа подключается к драйверу посредством COM. Если драйвер в этот момент не был загружен, Windows автоматически производит его запуск. Это может занять несколько секунд.

В поле «Текущее состояния» для каждого порта отображается количество устройств, обнаруженных драйвером. Можно включить режим постоянного поиска устройств по всем используемым портам. В этом режиме драйвер приостанавливает опрос устройств, с которыми уже установлена связь, и ведет циклический поиск но-

вых устройств. В процессе поиска происходит индикация счетчика циклов сканирования отдельно для каждого COM-порта, который увеличивается, когда драйвер просматривает полный диапазон всех возможных адресов устройств. Таким образом, если при неоднократном увеличении счетчика циклов поиска ни одно устройство не обнаружено, значит либо неверно указан протокол обмена порта, либо нарушена линия связи. Кнопка «Перезапуск драйвера» позволяет дать драйверу команду остановить опрос оборудования, считать конфигурацию из реестра, и снова начать опрос в соответствии с обновленной конфигурацией. Без использования данной кнопки драйвер считывает записанную в реестр конфигурацию только при запуске процесса.

3.2.5 Вкладка «О программе»

На этой вкладке отражена информация о версии драйвера, а также даны ссылки на сайт производителя. Внешний вид вкладки показан на рисунке 6.

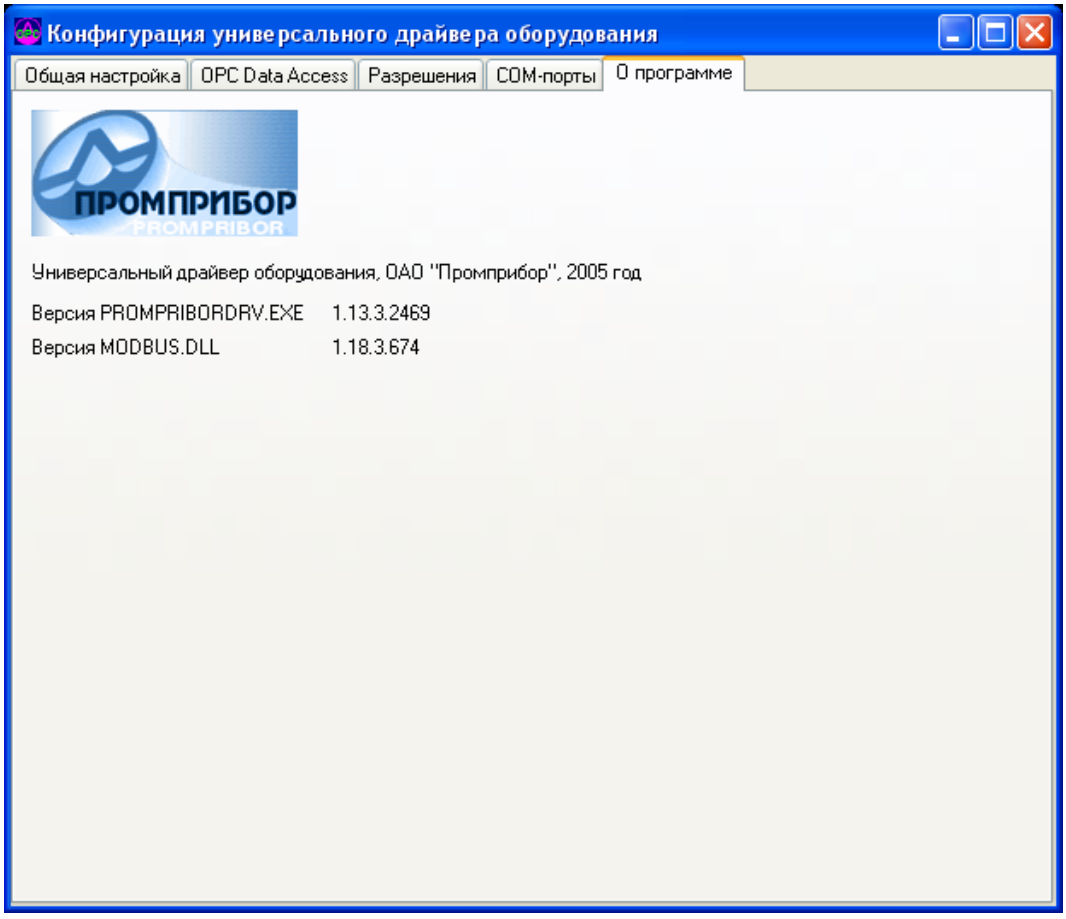

Рисунок 6 - Сведения о версии драйвера

3.3 Порядок удаления драйвера

Удаление драйвера производится с помощью пункта «Установка и удаление программ» панели управления Windows.

При удалении конфигурация драйвера также удаляется из реестра. При переустановке необходимо снова использовать *Prompribor DrvCfg.EXE*.

Компоненты «OPC Core Components», устанавливаемые совместно с драйвером, удаляются независимо. При удалении помните: они могут использоваться другими приложениями, поддерживающими OPC.

В случае невозможности удаления драйвера описанным способом необходимо выполнить следующие шаги:

а) удалить на системном диске папку «Program files\PrompriborDrv». Если удалить невозможно, папку переименовать, произвести перезагрузку операционной системы и выполнить удаление.

б) удалить вручную с помощью системной утилиты regedit.exe ключи реестра:

- «HKLM\SOFTWARE\Prompribor\Drv»
- «HKLM\SOFTWARE\Classes\AppID\PrompriborDrv.EXE»
- «HKLM\SOFTWARE\Classes\CLSID\{40902B39-AC7C-4682-9E76-EC41E0F2A4A4}»
- «HKLM\SOFTWARE\Classes\CLSID\{40902B3A-AC7C-4682-9E76-EC41E0F2A4A4}»
- «HKLM\SOFTWARE\Classes\PrompriborDrv.CUP»
- «HKLM\SOFTWARE\Classes\PrompriborDrv.DA2»
- «HKLM\SOFTWARE\Classes\TypeLib\{F6702C6A-4A90-4873-B43C-C03EA082CA77}».

в) удалить ПО «OPC Core components» с помощью пункта «Установка и удаление программ» панели управления.

г) перезагрузить операционную систему.

3.4 Действия в случае возникновения проблем в работе драйвера

3.4.1 Действия в случае отсутствия связи с оборудованием

При отсутствии связи с контроллерами КУП или ее неустойчивости требуется выполнить следующие шаги:

- проверить конфигурацию драйвера посредством утилиты конфигурирования в соответствии с 3.2;

- если отсутствует связь только с одним контроллером в линии связи, возможно, перепутана полярность подключения данного контроллера к линии связи или контроллер неисправен;

- отсоединить ПДУ «Весна-ТЭЦ» от COM-порта компьютера, проверить работоспособность ПДУ «Весна-ТЭЦ» и контроллеров КУП путем произведения тестовых отпусков с клавиатуры пульта. Если нет связи ПДУ «Весна-ТЭЦ» и контроллеров КУП, см. документацию КУП и ПДУ «Весна-ТЭЦ»;

- выключить питание пульта «Весна-ТЭЦ», подключить его к COM-порту компьютера, включить питание пульта «Весна-ТЭЦ». ВНИМАНИЕ: при работе драйвера ПДУ «Весна-ТЭЦ» должен находиться в неактивном режиме – это режим работы от момента включения питания ПДУ до момента нажатия любой кнопки на пульте. В неактивном режиме на индикаторе ПДУ отображается заставка. После нажатия любой кнопки ПДУ переходит в активный режим работы, при котором связь ПК с контроллерами невозможна. Для перевода ПДУ в неактивный режим выключите и включите его питание;

- если используются нестандартные платы поддержки COM-портов или преобразователи USB-COM, попробуйте подключить пульт «Весна-ТЭЦ» через COM-порт материнской платы компьютера;

- с помощью системной утилиты REGEDIT удалить все подразделы реестра в разделе «HKEY_LOCAL_MACHINE\Software\Prompribor\Drv»;

- перезагрузить компьютер.

Если после указанных действий связь не устанавливается и в случае возникновения других проблем при работе драйвера существует возможность удаленного обнаружения их причин с помощью режима протоколирования в соответствии с 3.2.1.

При отсутствии связи с контроллерами ЦБУ:

- проверить отсутствие подключения к линии связи других устройств, которые работают в качестве активного главного устройства (выполняют опрос подчиненных устройств), например контроллер «Весна-ТЭЦ-2-3К» или другой ПК с установленным драйвером оборудования. В случае подключения к линии связи нескольких главных устройств только одно из них может быть активным. Все другие главные устройства должны быть либо обесточены, либо им должно быть программно запрещено активно работать в линии связи;

- проверить конфигурацию драйвера посредством утилиты конфигурирования, в частности, скорость обмена COM-порта в соответствии с 3.2;

- проверить связь с ЦБУ утилитой конфигурирования ЦБУ. Если утилита конфигурирования ЦБУ устанавливает связь (возможно, после нескольких попыток) на той же скорости обмена, какая указана в конфигурации драйвера, а с драйвером связи нет – конфигурация ЦБУ является ошибочной или ЦБУ неисправен. Если утилита конфигурирования ЦБУ устанавливает связь на скорости, отличной от указанной в конфигурации, измените скорость в конфигурации ЦБУ на соответствующую скорости драйвера;

- проверить полярность подключения линии связи;

- проверить целостность кабеля связи.

Если после указанных действий связь не устанавливается и в случае возникновения других проблем при работе драйвера существует возможность удаленного обнаружения их причин с помощью режима протоколирования по 3.2.1.

3.4.2 Действия в случае сбоев драйвера

В случае обнаружения сбоев или зависаний драйвера и утилиты конфигурирования требуется выполнить следующие шаги:

- запретить доступ к оборудованию в соответствии с 3.2.1 и попробовать повторно запустить драйвер. При повторении ошибки необходимо переустановить драйвер. Если эти действия не помогают, необходимо обратиться к разработчику в соответствии с 3.4.3;

- при использовании нескольких COM-портов – если в режиме запрета доступа к оборудования ошибка не повторилось, необходимо разрешить доступ к оборудованию и проверять работу драйвера при использовании COM-портов поочередно при выключенных контроллерах. Для этого в конфигураторе драйвера поочередно устанавливаются флаги использования порта в соответствии с 3.2.4 и производится перезапуск драйвера нажатием кнопки «Перезапуск драйвера для получения новой конфигурации». В случае повторения ошибки на одном из портов, необходимо переустановить драйвер COM-порта.

- если при отключенном оборудовании ошибок не обнаружено, необходимо произвести анализ ситуации при включенном оборудовании и активном режиме протоколирования (по 3.4.3).

3.4.3 Обращение к разработчику

Для поиска ошибок работы с оборудованием и внутренних сбоев драйвера используется режим протоколирования. Установка этого режима описана в пункте 3.2.1.

Драйвер в процессе своей работы производит запись происходящих событий в текстовые файлы. По содержимому этих файлов разработчики драйвера могут определить причины возникновения ошибок в работе.

Файлы протоколов создаются в папке, выбранной в конфигурации драйвера (в соответствии с 3.2.1) и имеют имена *PrompriborDrv*.LOG*, где вместо * проставляется номер физического или виртуального порта. При запуске драйвера старое содержимое файлов протоколов удаляется и ведется запись заново. Поэтому если требуется сохранить файлы протоколов после прекращения работы драйвера, нужно переместить их в другую папку.

Нужно обратить внимание на размер файлов и наличие свободного места на текущем разделе. При работе в режиме полного протоколирования размер файлов увеличивается на 5-6Мб за час работы каждого COM-порта. Поэтому при непрерывной работе в течение длительного времени, протокол может занять значительный объем памяти на текущем разделе. Для исключения подобной ситуации, использовать режим протоколирования необходимо только в целях поиска неисправностей.

Для определения причин необходимо, кроме ведения драйвером протокола работы, фиксировать на бумаге время возникновения проблемы по часам компьютера, а также смысл проблемы, последовательность действий пользователей (клиентских программ) непосредственно до возникновения проблемы, точный текст индикации управляемых драйвером контроллеров, состояние управляемого оборудования. Следует понимать, что для поиска неисправностей и ошибок программного обеспечения нужна наиболее полная информация о возникающих событиях. Даже, казалось бы, совершенно не относящиеся к проблеме особенности работы ПК и оборудования, могут помочь разработчикам быстро определить проблему в аппаратной части оборудования или устранить ошибку программного обеспечения.

Для обращения к разработчикам необходимо выслать по электронной почте файлы протоколов и указанную выше сопутствующую информацию.

3.4.4 Действия в случае ошибок чтения и записи переменных клиентским приложением

Для проверки наличия переменных драйвера, тестирования работоспособности управляемого оборудования путем чтения и записи переменных, можно использовать утилиту «OPC-клиент», входящую в комплект поставки драйвера. Ее можно использовать как независимо, так и совместно с любым другим клиентским программным обеспечением.

При разработке и отладке клиентского программного обеспечения рекомендуется использовать утилиту «OPC-клиент» для проверки правильности действий программного обеспечения и в качестве дополнительного средства просмотра свойств оборудования.

4 Взаимодействие драйвера с управляемым оборудованием

Драйверу для определения наличия и физических адресов управляемого оборудования со стороны пользователя и клиентского ПО не требуется никакой информации, за исключением указания общих параметров конфигурации. Драйвер определяет по всем адресам наличие подключенного оборудования и считывает в память компьютера информацию в зависимости от его типа оборудования, формируя при этом буфер памяти. Буфер отдельно для каждого поста (в случае оборудования отпуска жидкостей) или устройства содержит набор переменных, каждая из которых содержит текстовое имя, значение переменной, качество значения и время получения значения. В течение работы драйвер циклически опрашивает оборудование с целью определения изменений параметров, вносит соответствующие изменения в буфер и оповещает клиентские приложения об изменениях. По команде клиентских приложений драйвер производит запись (установку) значений параметров. Взаимодействие клиентских приложений с оборудованием посредством драйвера сводится к определению наличия параметров оборудования по их именам, чтению и записи значений.

Во время работы драйвера поддерживается динамическое подключение и отключение дополнительного оборудования, т.к. он периодически опрашивает оборудование по тем адресам, которые не ответили при загрузке.

Структура буфера для оборудования отпуска жидкостей описывается схемой, изображенной на рисунке 7:

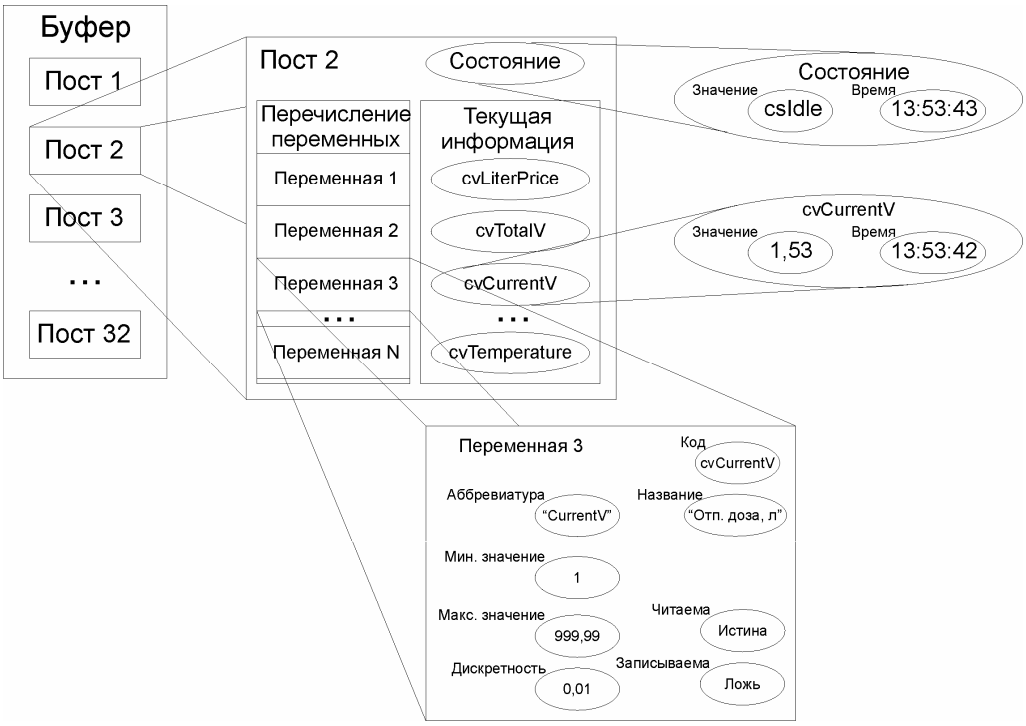

Рисунок 7 - Общая структура буфера памяти драйвера для оборудования отпуска жидкостей

5 Взаимодействие драйвера с клиентскими приложениями по стандарту OPC DA

Драйвер соответствует требованиям *OPC Data Access Custom Interface specification ver. 2.05* (приложение *OPCDA205A_CUST.PDF*).

Согласно указанной спецификации драйвер реализует COM-сервер (OPC-сервер), предоставляющий COM-интерфейсы объектов ***OPCServer*** и ***OPCGroup***:

- ***IOPCServer*** (OPCServer);
- ***IOPCItemProperties*** (OPCServer);
- ***IOPCBrowseServerAddressSpace*** (OPCServer);
- ***IConnectionPointContainer*** (OPCServer);
- ***IOPCCommon*** (OPCServer);
- ***IOPCGroupStateMgt*** (OPCGroup);
- ***IOPCItemMgt*** (OPCGroup);
- ***IOPCSyncIO*** (OPCGroup);
- ***IOPCAsyncIO2*** (OPCGroup);
- ***IConnectionPointContainer*** (OPCGroup);
- ***IOPCAsyncIO*** (OPCGroup, устаревший) ;
- ***IDataObject*** (OPCGroup, устаревший);
- ***IOPCGroupStateMgt*** (OPCGroup).

Указанный набор интерфейсов в значительной степени избыточен. Часть интерфейсов унаследована от первой версии спецификации и не рекомендуется к использованию. Избыточность интерфейсов позволяет при программировании клиентского приложения выбрать наиболее удобный способ взаимодействия с драйвером. Для выбора требуемых интерфейсов нужно ответить на следующие вопросы:

- асинхронно или синхронно должна выполняться чтение-запись переменных;

- нужен ли механизм оповещения приложения-клиента об изменении значений переменных;

- будет ли при чтении клиентом значений переменных выполняться принудительный перезапрос устройства (device read) или использоваться буфер значений (cache read).

Спецификация OPC DA отличается от традиционных интерфейсов, специфичных для конкретного оборудования, прежде всего универсальностью доступа к параметрам оборудования. Доступ к переменным производится независимо от типа оборудования, смысла переменной и типа данных. Это достигается следующим образом:

- при инициализации переменные адресуются по строковому названию, содержащему указание типа, номера устройства и смысла переменной. Строковое название переменной состоит из нескольких частей, разделенных точкой, благодаря чему переменные организованы в виде дерева. В дальнейшем для ускорения доступа нужным переменным назначаются целочисленные идентификаторы (server handle, client handle);

- для обмена значениями переменных используется универсальный тип ***OleVariant***.

Для снижения накладных расходов на вызовы функций между процессами клиента и сервера (тем более, если процессы находятся на различных ПК) переменные произвольно группируются клиентским приложением в группы (***OPCGroup***). Доступ к переменным всегда осуществляется массивами. Коды ошибок возвращаются также массивом, отдельный код для каждой переменной.

Указанные особенности спецификации OPC DA, однако, немного усложняют написание процедур работы с драйвером. Минимальная последовательность действий при инициализации и взаимодействии такова:

1. Получение ***ClassID*** (GUID) драйвера из его ***ProgID*** = **'PrompriborDrv.DA2'** с помощью ***ProgIDToClassID***

2. Создание объекта ***OPCServer*** с помощью ***CreateComObject*** или ***CreateRemoteComObject***. COM автоматически производит запуск драйвера, если драйвер не загружен.

3. Создание одного или нескольких объектов OPCGroup с помощью ***IOPCServer.AddGroup***

4. Просмотр дерева переменных (адресного пространства) драйвера с помощью ***IOPCBrowseServerAddressSpace***. Если клиентское приложение сконфигурировано на наличие некоторого оборудования, но его переменные отсутствуют, данное оборудование в данный момент недоступно. Для появления переменных не требуется перезапуск драйвера, достаточно только подключить оборудование. Клиентское приложение не должно воспринимать адресное пространство драйвера как фиксированное. В некоторых ситуациях следует циклически его перезапрашивать или воспользоваться переменной **System.AddrSpaceVer** (по 5.1).

5. Добавление нужных переменных в группы посредством ***IOPCItemMgt*** с назначением клиентских идентификаторов (client handle) и серверных идентификаторов (server handle).

6. Чтение-запись значений переменных с помощью ***IOPCSyncIO***, ***IOPCAsyncIO*** или ***IOPCAsyncIO2***.

7. Рекомендуется реализовать интерфейс ***IOPCDataCallback*** и зарегистрировать его с помощью ***IConnectionPointContainer***. В таком случае драйвер будет вызывать функции ***IOPCDataCallback*** для оповещения клиента об изменениях значений переменных и для возврата результатов асинхронных операций. Таким образом, устраняется необходимость циклического чтения значений переменных и без лишних усилий обеспечивается своевременная реакция клиентского приложения на самопроизвольные и неожиданные изменения значений переменных. Драйвер специально разработан таким образом, он ведет более частый опрос тех переменных, вероятность изменения которых максимальна. Оценка указанной вероятности достаточно интеллектуальна, поэтому рекомендуется, чтобы клиентское приложение не давало драйверу команд на чтение из устройства. Опрос устройств выполняется даже тогда, когда клиентские приложения не подключены.

Текстовое описание кодов ошибок можно получить с помощью интерфейса ***IOPCCommon*** объекта ***OPCServer***.

По соображениям производительности предпочтительно использовать асинхронный интерфейс второй версии либо синхронное чтение из кэша (CACHE READ). Синхронная запись и чтение из устройства выполняется значительно медленнее.

Для просмотра адресного пространства OPC-сервера и тестирования возможностей драйвера, а также в качестве примера клиентского приложения предоставляется утилита «OPC-клиент» *PrompriborOPCCli.EXE*. Также как вспомогательный инструмент для тестирования возможностей оборудования и драйвера рекомендуется использовать National Instruments Server Explorer (www.ni.com).

5.1 Адресное пространство OPC-сервера для управления драйвером

Для управления (конфигурирования) драйвера в его адресном пространстве, независимо от состава подключенного оборудования, существует ветвь System. Здесь представлены пять переменных:

- **System.AddrSpaceVer** – счетчик изменений адресного пространства OPC в текущей сессии. Увеличивается на единицу при добавлении и удалении переменных и ветвей адресного пространства. Только для чтения. С помощью данной переменной, при реализации клиентским ПО интерфейса ***IOPCDataCallback***, оно будет своевременно оповещаться об обнаружении нового и полной потере связи с существующим оборудованием. В таком случае нет необходимости в циклическом опросе интерфейса ***IOPCBrowseServerAddressSpace***.

- **System.AllParameters** – строка, содержащая полное описание конфигурации драйвера, хранимой в реестре. Доступно: чтение для всех пользователей и запись только администраторам системы. Может быть использована для резервного копирования и восстановления, копирования с главного на резервный сервер. Копировать строку разрешается только целиком, редактировать строку произвольным образом запрещено. Редактирование конфигурации производится с помощью утилиты конфигурирования (по 3.2). В качестве подстроки здесь содержится значение переменной **System.AllPortsEnable**. Т.е. при записи **System.AllParameters** переменная **System.AllPortsEnable** меняет значение, и наоборот. Изменения конфигурации, записанные в системные переменные, вступают в действие немедленно.

- **System.AllPortsEnable** – булевская переменная, разрешение доступа к портам для связи с оборудованием. По умолчанию разрешено. Доступно чтение и запись всем пользователям. Имеет смысл запрещать доступ к оборудованию, если данный экземпляр драйвера выполняется на резервном сервере во избежание конфликта с экземпляром драйвера главного сервера.

- **LastErrorCritical** – строковая переменная, в которой содержится значение последней внутренней ошибки драйвера.

- **LastErrorExternal** - строковая переменная, в которой содержится значение последней ошибки при работе с внешними устройствами.

5.2 Адресное пространство OPC-сервера для оборудования отпуска жидкостей

Адресное пространство является древовидным, динамически определяемым при запуске исходя из состава подключенного оборудования, изменяющимся при его подключении/отключении.

Первый уровень дерева представляет собой список оборудования. Каждый пункт соответствует одному контроллеру оборудования и имеет имя **Post + <номер**

поста контроллера>. Для многопостовых топливораздаточных колонок берется адрес первого поста в данном контроллере. На каждой ветви данного уровня представлены теги, относящиеся к установке (ТРК, ГНК, АСН) в целом. Например, это счетчик инспектора (Inspector), показание датчика температуры (**Temperature**) и т.д. В случае ГНК, АСН и однопостовых ТРК, кроме того, здесь же находятся теги, относящиеся к единственному раздаточному крану (посту).

Второй уровень существует в случае многопостовых ТРК с возможностью одновременного отпуска с двух сторон и представляет собой две стороны – S1 и S2. Внутри каждой из сторон одновременно возможен отпуск только одного поста. Во время отпуска одного поста другие посты внутри той же стороны игнорируют команды начала отпуска. В случае многопостовых, но односторонних ТРК, второй уровень иерархии отсутствует, и третий уровень является непосредственным подуровнем первого. В конфигурации драйвера выбирается, использовать ли указанную 3-уровневую иерархию переменных для ТРК или все ветви постов находятся на первом уровне.

Третий уровень существует в случае многопостовых ТРК и представляет собой список раздаточных кранов (постов, пистолетов ТРК). Имя составляется из ‘P’ + номер поста внутри стороны (P1 – P4). В случае однопостовых установок теги данного уровня находятся на первом уровне. Здесь присутствуют теги, соответствующие параметрам поста, например: код состояния (**State**), заданная доза отпуска в различных единицах (**DozeV**, **DozeM**, **DozeW**), текущее значение отпущенной дозы в различных единицах (**CurrentV**, **CurrentM**, **CurrentW**) и т.д.

Каждый тег (переменная), находящийся в адресном пространстве OPC DA сервера, находится внутри какой-либо ветви (подветви). Тегов на корневом (нулевом) уровне быть не может. Полное имя переменной составляется последовательно из имен содержащих его ветвей, разделяемых символом точки. Местонахождение тега в дереве адресного пространства определяет, к какой именно установке и раздаточному крану относится данный тег, а название определяет, значение какого параметра является значением тега. Например, тег **Post4.S1.P3.State** является состоянием третьего пистолета первой стороны ТРК с базовым адресом «4». Тег **Post2.CurrentV** возвращает текущую отпущенную дозу в литрах АСН, ГНК или однопостовой ТРК с базовым адресом «2».

Список возможных тегов для каждого вида оборудования приведен в таблице 1. **Ошибка! Источник ссылки не найден.**.

Таблица 1 – Список возможных тегов

Название	***Описание***	***Тип данных****	***Поддержка оборудования***
StartStop	Состояние кнопки ПУСК/СТОП: 1 – нажата, 0 – отпущена	VT_BOOL	Все
Earthed	Показание датчика заземления: 1 – заземлено, 0 – не заземлено	VT_BOOL	ЦБУ, КУП40
EarthedEx	Расширенное состояние заземления поста: 0 – Заземление не установлено 1 – Гаражное положение 2 – Рабочее положение 3 – Требуется поместить в гаражное положение	VT_14	ЦБУ
HandleState	Положение наливного стояка: 0 – неопределенное, 1 – рабочее	VT_BOOL	Все
HandleGrState	Гаражное положение наливного стояка: 0 – не гаражное, 1 - гаражное	VT_BOOL	ЦБУ, КУП40
TrapState	Положение перекидного трапа: 0 - неопределенное (рабочее), 1 - гаражное (нерабочее)	VT_BOOL	Для АСН
OverFill	Состояние датчика переполнения наливаемой емкости	VT_BOOL	ЦБУ
DozeV	Заданная доза в литрах	VT_14	Все
DozeM	Заданная доза в деньгах	VT_14	Только ТРК
DozeW	Заданная доза в килограммах. Зарезервировано для будущего использования	VT_14	
CurrentV	Текущее значение отпускаемой дозы в литрах	VT_14	Все
CurrentM	Текущее значение отпускаемой дозы в деньгах	VT_14	Только ТРК
CurrentW	Текущее значение отпускаемой дозы в килограммах	VT_14	ЦБУ, КУП40(при наличии массомера)

Продолжение таблицы 1 – Список возможных тегов

Название	*Описание*	*Тип данных**	*Поддержка оборудования*
TotalV	Значение фискального счетчика в литрах	VT_14	Все КУПы
		VT_R8	ЦБУ
TotalM	Значение фискального счетчика в деньгах.Зарезервировано для будущего использования	VT_14	Все КУПы
TotalW	Значение фискального счетчика в килограммах	VT_14	Все КУПы (при наличии массомера)
		VT_R8	ЦБУ (при наличии массомера)
Temperature	Текущая температура отпускаемого продукта	VT_14 или VT_R4	Все
LiterPrice	Цена 1 литра продукта		Все КУПы
CalibV	Значение 1 импульса счетного устройства, мл/сигнал	VT_R8	Все КУПы
Inspector	Счетчик инспектора (фискальная память, увеличивается при изменении параметров, после увеличения возврат в предыдущее состояние невозможен)	VT_14	Все
FullUp	Количество литров от начала отпуска до разрешения максимального расхода	VT_14	Для АСН
FullDown	Количество сигналов расхода до окончания отпуска для предварительного снижения производительности отпуска	VT_14	Все
USSWaitTime	Время ожидания сигналов расхода во время отпуска до перехода в состояние паузы	VT_14	Все

Продолжение таблицы 1 – Список возможных тегов

Название	*Описание*	*Тип данных**	*Поддержка оборудования*
AirTime	Время, на которое открывается воздушный клапан после окончания отпуска	VT_I4	Для АСН
BasePost	Базовый пост данного контроллера. Переменная присутствует, если контроллер содержит несколько постов. Если два поста имеют одинаковое значение BasePost, они относятся физически к одному контроллеру. Некоторые переменные являются параметрами контроллера, а не поста. Такие переменные располагаются в базовом посту контроллера	VT_I4	ЦБУ, КУП10, КУП1
Side	Если два поста имеют одинаковые **BasePost** и **Side**, они не могут производить отпуск одновременно. Если данная переменная отсутствует, то данный контроллер не поддерживает одновременный отпуск нескольких постов	VT_I4	ЦБУ, КУП10, КУП1
Index	Номер поста внутри данной стороны. Если данная переменная отсутствует, то данный пост единственный на стороне. На каждой стороне в один момент времени возможен отпуск только одного поста	VT_I4	КУП10
FirmwareVersion	Версия ПО контроллера	VT_I4	Все КУПы
		VT_BST	ЦБУ
HardwareVersion	Версия аппаратного исполнения контроллера	VT_I4	Все КУПы
ControllerVersion	Тип контроллера	VT_I4	Все КУПы
Density	Плотность жидкости в счетчике, кг/см^3(если подключен массомер)	VT_R`14	ЦБУ (при наличии массомера)

Продолжение таблицы 1 – Список возможных тегов

Название	***Описание***	***Тип данных****	***Поддержка оборудования***
Productivity	Мгновенный расход жидкости, $см^3/с$	VT_R4	Все
StateWithFlags	Состояние поста в виде набора битовых полей	VT_I4	Все
State	Текущее состояние поста: 0 – начальное состояние(бездействие) 1 – разрешения начала отпуска 7 – процесс налива 80 – состояние останова(пауза) 186 – устаревшая версия контроллера 188 – неподдерживаемая версия оборудования 169 – временное отсутствие связи с постом 177 – контроллер находится в состоянии ошибки	VT_I4	Все
Address	Уникальный номер поста	VT_I4	Все
Enable	Разрешает или запрещает налив (1 – разрешено, 0 – запрещено). Если контроллер находился в состоянии налива, то при записи 0 в эту переменную происходит останов налива	VT_I1	Все
Tank	Описание резервуара. Хранится в драйвере и в контроллер не записывается	VT_BSTR	Все
AdrController	Адрес контроллера по протоколу Modbus	VT_I4	ЦБУ
CountPost	Количество постов в контроллере	VT_I4	ЦБУ
DozeMicro	Доза микрорасхода, литры	VT_I4	ЦБУ

Продолжение таблицы 1 – Список возможных тегов

Название	*Описание*	*Тип данных**	*Поддержка оборудования*
Exception	Код ошибки	VT_I2	ЦБУ
ExceptionParameter	Дополнительные параметры ошибки	VT_I2	ЦБУ
InnerTemperature	Внутренняя температура контроллера	VT_R4	ЦБУ
SnapSensor	Причина останова процесса налива: бит#0 – датчик перелива бит#1 – окончание дозы бит#2 – останов по кнопке Пуск/Стоп бит#3 – отсутствие импульсов расхода бит#4 – датчик заземления бит#5 – датчик рабочего положения стояка бит#6 – датчик гаражного положения стояка бит#7 – датчик гаражного положения трапа	VT_I4	ЦБУ
StatePowerPlate	Состояние силовых выходов поста: бит#0 – соленоид мин. расхода бит#1 – соленоид макс. расхода бит#2 – магнитный пускатель бит#3 – соленоид воздушного клапана бит#4 – сигнал зеленого света бит#5 – сигнал красного света бит#6 – соленоид азотного клапана бит#7 – не используется бит#8 – перегрузка соленоид мин. расхода бит#9 – перегрузка соленоид макс. расхода бит#10 – перегрузка по пускателю	VT_I4	ЦБУ

Окончание таблицы 1 – Список возможных тегов

Название	***Описание***	***Тип данных****	***Поддержка оборудования***
	бит#11 – перегрузка воздушного клапана бит#12 – перегрузка по зеленому свету бит#13 – перегрузка по красному свету бит#14 – перегрузка соленоида азотного клапана бит#15 – не используется		
TopDistance	Расстояние между горловиной и уровнем жидкости в наливаемой емкости, см	VT_I4	ЦБУ с ультразвуковым датчиком уровня
TopDistanceLim	Минимально допустимое расстояние между горловиной и уровнем жидкости в наливаемой емкости, см	VT_I4	ЦБУ с ультразвуковым датчиком уровня

*типы данных:

VT_I4 – четырехбайтовое целое
VT_I2 – двухбайтовое целое
VT_I1 – однобайтовое целое
VT_R4 – число с плавающей запятой одинарной точности
VT_R8 – число с плавающей запятой двойной точности
VT_BSTR – строка
VT_BOOL – булевая переменная

5.3 Адресное пространство OPC-сервера для контроллера ЦБУ в пассивном режиме работы

В пассивном режиме работы ЦБУ его адресное пространство имеет следующий вид. На первом уровне иерархии расположен список контроллеров. Каждый пункт соответствует одному контроллеру и имеет вид ‘CBU’ + адрес устройства по протоколу Modbus в десятичном представлении.

5.3.1 Корневая ветвь

В первом уровне находятся три переменные:

- **Address** - адрес контроллера в линии связи. Тип данных – целый. Права доступа – чтение и запись. Диапазон допустимых значений от 1 до 247. При записи происходит непродолжительный разрыв связи с контроллером из-за необходимости переименования корневой ветки. Для дальнейшего доступа к данному терминалу необходимо обращаться к тегам с указанием в имени нового адреса.
- **Build** - номер сборки ПО контроллера. Тип данных – целый. Права доступа – только чтение.
- **Version** - номер версии ПО контроллера. Тип данных – целый. Права доступа – только чтение.

Второй уровень содержит ветви для каждого выхода силового модуля и для каждого входа модуля ввода. Имена веток создаются следующим образом:

- для выходов - **OutputX_Y,** где X – это номер силового модуля внутри контроллера, Y– порядковый номер силового выхода;
- для входов - **IntputX_Y,** где X – это номер модуля ввода внутри контроллера, Y- порядковый номер входа;

5.3.2 Ветвь логических входов и силовых выходов

В ветви силовых выходов расположены следующие переменные:

- **State** - состояние силового выхода. Тип данных – целый. Права доступа – чтение и запись. Эта переменная позволяет просматривать и управлять состоянием силового выхода. Запись значения 1 приводит к включению выхода, запись 0 – к выключению. При State = 1 может быть обнаружена перегрузка (короткое замыкание) выхода. В таком случае State самопроизвольно принимает значение 0, Overload – значение 1.
- **Overload** - состояние перегрузки силового выхода. Тип данных – булевский. Права доступа – чтение и запись. Эта переменная показывает состояние перегрузки силового выхода. При значении 0 перегрузка отсутствует, при значении не 0 – перегрузка есть. Перегрузку выхода можно сбросить записью значения 0 в переменную.

В ветви входов расположена одна переменная **State**. Тип данных – булевский. Права доступа – только чтение. Эта переменная показывает состояние логического входа.

5.4 Адресное пространство OPC-сервера для терминалов «ТС-001», «ТС-001Ex», «ТС-002»

Первый уровень иерархии представляет собой список терминалов. Каждый пункт соответствует одному терминалу и имеет имя ‘Terminal’ + адрес устройства по протоколу Modbus в десятичном представлении. Например, для терминала с адресом 0Ah имя устройства будет **Terminal10**.

Второй уровень иерархии для каждого устройства содержит следующие ветки:

- **Display** - работа с индикатором терминала;
- **Card** - работа с бесконтактными радиоидентификационными картами;
- **Key** - работа с клавиатурой терминала, буферы ввода;
- **MemoDisplay** - загружаемые строки индикатора.
- **InOut** - дискретные входы и выходы(только для ТС-001)

5.4.1 Корневая ветвь

В корневой ветке, имеющей название терминала, находятся следующие теги:

- **Address** - адрес терминала в линии связи. Тип данных – целый. Права доступа – чтение и запись. Диапазон допустимых значений от 1 до 247. При записи происходит непродолжительный разрыв связи с терминалом из-за необходимости переименования корневой ветки. Для дальнейшего доступа к данному терминалу необходимо обращаться к тегам с указанием в имени нового адреса.
- **CharOutput** - Этот тег предназначен для включения режима отображения знаков на дисплее при посимвольном вводе с клавиатуры. При CharOutput равном 1, режим отображения включен. Имеет булевский тип. Права доступа – чтение и запись.
- **RusChar** - Флаг ошибки русификации сообщения на дисплее. Устанавливается в единицу при попытке одновременно отобразить на экране более 8 различных букв русского алфавита, не совпадающих по написанию с латинскими буквами. Имеет булевский тип. Права доступа – чтение и запись.
- **StartTime** - Время последнего установления связи драйвера и терминала. Устанавливается в случае первоначального запуска драйвера, длительного разрыва и восстановления связи или при откл./вкл. питания терминала. Имеет формат времени. Права доступа – только чтение. Предназначен для использования клиентским приложением при проверки необходимости инициализации энергозависимых параметров терминала.

5.4.2 Ветвь Display

- **Cursor** - Установка позиции курсора на дисплее. Старший байт – номер столбца, младший байт – номер строки. Нумерация начинается с нуля. Двухбайтовое целое.
- **Index1 – Index4** - Индекс для соответствующей строки дисплея, который позволяет выводить на дисплей заранее загруженные строки. Существуют два вида загруженных строк: хранимые в постоянной памяти и динамически загружаемые. Динамически загружаемые строки можно записать,

используя переменные String1 – String20 ветки MemoDisplay (по 5.4.5). Для постоянно хранящихся в ПЗУ индексов отведен диапазон индексов от 100 и далее. Тип данных переменной - двухбайтовое целое.

- **Brightness** - Яркость свечения индикатора. Двухбайтовое целое. Значимыми являются два младших бита. В зависимости от значения в этих битах яркость дисплея будет следующей:
  - 00 - 100%
  - 01 - 75%
  - 10 - 50%
  - 11 - 25%
- **String1 – String4** - Теги для вывода текста на соответствующие строки индикатора. Имеют формат строки. Максимальная длина – 20 символов. Алфавитно-цифровая информация, передаваемая на дисплей, должна иметь кодировку Windows.

Все переменные ветки **Display** предназначены для записи и чтения. Однако, так как непосредственно из устройства невозможно прочитать значение строки, выводимой на индикатор, значение при чтении берется из внутреннего буфера драйвера. В буфере храниться последнее записанное в строку значение. В случае длительной потери связи или отключения терминала качество переменных **String1 – String4** устанавливается плохим.

5.4.3 Ветвь Card

- **CardRequest** - флаг запроса карты. После включения питания флаг установлен в еденицу. При этом терминал ждет поднесения любой карты, после получения номера от карты флаг сбрасывается в 0 и терминал запоминает ее серийный номер в теге CardID. Далее терминал взаимодействует только с данной картой до команды установки флага запроса карты в единицу, все другие карты игнорируются. Карту с серийным номером, равным CardID, при сброшенном флаге запроса можно подносить и относить от терминала произвольное число раз.
- **CardID** - серийный номер карты. Этот номер является уникальным для каждой карты, что гарантируется их производителем. Формат переменной - четырехбайтовое целое. Права доступа – только чтение.
- **TypeKey** - тип криптографических ключей, используемых для чтения и записи памяти карты. По значению этой переменной драйвер определяет, по какой группе ключей будет производиться аутентификация карты с терминалом. Права доступа – чтение и запись. Имеет строковый тип и 6 фиксированных значений:
  - «A0», «B0» - ключи нулевой группы типа A и типа B;
  - «A1», «B1» - ключи первой группы типа A и типа B;
  - «AT» , «BT» - транспортные ключи типа A и типа B.

- **Field1 – Field15** - Энергонезависимая память поднесенной к терминалу карты. Имеют строковый формат. Максимальная длина строки - 47 символов. Права доступа – чтение и запись. Когда карта не поднесена или

выбранный тип ключей не соответствует ключам карты, чтение и запись невозможны (клиентское приложение получает ошибку).

5.4.4 Ветвь Key

- **Buffer1 – Buffer10** - Поля ввода с клавиатуры. Имеют формат строки. Максимальная длина строки - 20 символов. Права доступа – только чтение. Поля ввода на индикаторе нумеруются слева направо сверху вниз в порядке 1, 2, 3,… Значение в этих переменных обновляются в случае завершения ввода с клавиатуры в соответствующее поле ввода индикатора.
- **CharInput –** Флаг посимвольного ввода. Устанавливается в 1 для режима посимвольного ввода. Сбрасывается в 0 для ввода по полям. После ввода каждого символа с клавиатуры в переменную LastKey записывается код нажатой клавиши. Ввод данных в поля ввода в посимвольном режиме не производится. Формат переменной - булевой. Права доступа – чтение и запись.
- **LastKey –** код последней нажатой клавиши. Формат переменной – символ. Права доступа – чтение и запись.

5.4.5 Ветвь MemoDisplay

- **String1 – String20 –** Загружаемые строки. Используются для загрузки в терминал заранее определенных сообщений, которые затем можно выводить на индикатор через запись индексов в переменные **Index1 – Index4**. Сообщение, записанное в переменную **String1**, имеет индекс 1, в **String2** – индекс 2 и т.д. Формат переменной - строка. Максимальная длина строки - 20 символов. Права доступа – только запись. При чтении возвращает плохое качество. При выключении питания терминала информация о строках не сохраняется.

5.4.6 Ветвь InOut

- **In1-In3 –** булевские переменные, показывающие состояние дискретных входов терминала. Права доступа – только чтение.
- **Out1-Out4 -** булевские переменные, позволяющие управлять состоянием дискретных выходов. Права доступа – чтение и запись.

5.5 Адресное пространство OPC-сервера для плотномера «Плот-3М»

Адресное пространство для работы с плотномерами имеет один уровень иерархии. Он представляет собой список плотномеров. Каждый пункт соответствует одному плотномеру и имеет имя ‘DensMeter’ + адрес устройства по протоколу Modbus в десятичном представлении. Например, для плотномера с адресом 01h имя устройства будет **DensMeter1**.

Внутри каждой ветви у плотномера присутствуют следующие теги (переменные):

- **Density** – текущие показания плотности. Размерность значения – кг/литр.;
- **Temperature** – текущая температура. Размерность значения – градусы (^{o}C);

- **Viscosity** – текущие показания вязкости. Размерность значений - мм^2/с;

Формат всех переменной – число с плавающей запятой одинарной точности (float). Права доступа – только чтение;

- **State** – регистр состояния плотномера.

5.6 Адресное пространство OPC-сервера для информационного табло

Адресное пространство для работы с информационным табло имеет один уровень иерархии. Каждый пункт соответствует одному устройству и имеет имя 'RowRun' + адрес устройства по протоколу СНИИП-Конвел в десятичном представлении. Например, для табло с адресом 01h имя устройства будет **RowRun1**. Внутри каждой ветви присутствует одна переменная – **String**. Формат переменной – строка. Максимальное количество символов в строке -170. Права доступа – чтение и запись. При чтении переменной значение берется из буфера драйвера, а не из устройства. Поэтому при первоначальном подключении табло к компьютеру качестве переменной будет плохим.

Если записываемая строка имеет длину меньшую или равную 16 символам, отображение ее на табло будет статическим. При превышении 16 символов, строка будет циклически прокручиваться слева направо.

6 Взаимодействие драйвера с клиентскими приложениями по интерфейсу IPrompribor

Интерфейс IPrompribor специализирован для поддержки оборудования дозированного отпуска жидкостей, управляемого контроллерами КУП. Взаимодействие с плотномерами «Плот-3», терминалами «ТС-001», «ТС-001Ex», АСН, управляемыми контроллерами ЦБУ, и т.д., посредством интерфейса IPrompribor невозможно.

Один исполняющийся экземпляр драйвера поддерживает множество экземпляров клиентского ПО, одновременно обменивающихся информацией с драйвером. Клиентское ПО может исполняться на нескольких компьютерах в пределах одной локальной сети. Внутренняя архитектура драйвера является многопоточной, что позволяет ему обмениваться информацией с оборудованием и отвечать на запросы клиентского ПО без лишних задержек. Операции записи и установки состояния выполняются асинхронно, т.е. вызовы функций всегда завершаются раньше, чем соответствующая команда будет передана драйвером контроллеру оборудования. Клиентское приложение имеет возможность позже запросить результат выполнения той или иной операции записи.

В комплект поставки драйвера включен пример простого клиентского приложения для связи с драйвером, созданного в Borland Delphi 6.0 Update 2 с использованием бесплатной библиотеки компонентов JEDI VCL 2.1 - http://jvcl.sourceforge.net .

Далее приводится описание библиотеки драйвера *PrompriborDrv.idl* . Описание библиотеки не включает описание подсистемы OPC-сервера, т.к. ее компонен-

ты (интерфейсы, типы данных и т.д.) содержатся в библиотеках пакета «OPC Core Components», входящего в комплект поставки драйвера.

6.1 Типы данных библиотеки PrompriborDrv.idl

6.1.1 TStateEnum

Данный тип-перечисление предназначен для определения соответствия всех возможных состояний раздаточного крана (поста) константам:

- **csIdle** (значение 0) – бездействие;
- **csWaitingKey** (значение 1) – возможность начала (продолжения) отпуска после подтверждения водителя (нажатие кнопки ПУСК/СТОП); для некоторых типов оборудования для перехода в состояние отпуска также требуется соблюдение дополнительных условий, например подключение заземления;
- **csWorking** (значение 7) – происходит отпуск, все условия нормального отпуска соблюдаются;
- **csPaused** (значение 80) – в результате несоблюдения условий во время отпуска (водителем нажата кнопка ПУСК/СТОП, перелив, долгое отсутствие сигналов счетного устройства и т.д.) процесс остановлен до достижения заданной дозы; после возврата всех условий (датчиков) в нормальное состояние возможно продолжение (через состояние **csWaitingKey**);
- **csPowerFault** (значение 4) – связь с постом поддерживается, нормальное функционирование невозможно по причине отсутствия внешнего напряжения питания силовых элементов оборудования (контроллер питается от источника бесперебойного питания);
- **csUnknown** (значение 168) – обнаружено наличие какого-то оборудования по указанному адресу поста, для определения точного состояния и списка поддерживаемых тегов требуется время;
- **csOldController(**значение 186) – устаревшая версия контроллера оборудования, требуется его обновление (за информацией обращаться к производителю контроллера);
- **csNewController** (значение 188) – неподдерживаемая версия оборудования, требуется обновление драйвера для поддержки;
- **csSensorError** (значение 178) – значение одного из датчиков оборудования находится вне пределов измерения, что указывает на ошибку датчика;
- **csHardwareError** (значение 176) – ошибка в силовых (электрических или механических) конструктивах оборудования;
- **csControllerError** (значение 177) – контроллер выдает нераспознаваемый ответ вследствие либо ошибки ПО контроллера, либо в электронике контроллера;
- **csCommError** (значение 169) – временное отсутствие связи с постом; после длительного нахождения в этом состоянии пост переходит в состояние **csNone**;
- **csNone** (значение 160) – пост отсутствует либо с постом нет связи в течение длительного времени; никакая дополнительная информация недоступна.

Только в состоянии **csIdle** возможно изменение (запись) любых параметров поста, кроме самого состояния. Т.е. для записи заданной дозы и других свойств оборудования требуется установить **State = 0**.

Состояние процесса («**State**») при безошибочном функционировании оборудования изменяется только по указанным сценариям:

- Начальное состояние **csIdle** (0).
- Переход из **csIdle** в **csWaitingKey** (1) по команде клиентского приложения (команда разрешения начала отпуска).
- Переход из **csWaitingKey** в **csWorking** (7) происходит при соблюдении условий безопасности процесса отпуска и нажатии кнопки «Пуск» на установке (с точки зрения клиентского ПО – самопроизвольно).
- Только для дозирующих контроллеров - переход из **csWorking** в **csIdle** производит контроллер по окончании заданного объема отпуска (с точки зрения клиентского ПО – самопроизвольно).
- Переход из **csWorking** в **csPaused** (80) производит контроллер при нарушении параметров безопасности отпуска (перечень необходимых параметров безопасности программируется в контроллере оборудования) до достижения заданного объема отпуска (с точки зрения клиентского ПО – самопроизвольно).
- Переход из **csPaused** в **csWaitingKey** по команде клиентского приложения (команда продолжения отпуска).
- Переход из **csWorking** в **csPaused** по команде клиентского приложения (команда временной приостановки отпуска).
- Переход из **csPaused**, **csWaitingKey**, **csWorking** в **csIdle** по команде клиентского приложения (команда полного останова отпуска).
- Только для пассивных счетчиков жидкости – самопроизвольный переход из **csIdle** в **csWorking** при протекании жидкости.

6.1.2 TVariableEnum

Данный тип-перечисление определяет константы, каждая из которых соответствует отдельному параметру оборудования, несущему определенный смысл. Каждый пост (раздаточный кран) поддерживает отдельно специфический для него набор параметров, код каждого из которых равен одной из констант. Указанные названия констант соответствуют названиям переменных OPC (без префикса «cv»), относящихся к постам.

- **cvStartStop** – Для ТРК. Состояние кнопки ПУСК/СТОП: 1 – нажата, 0 – отпущена;
- **cvEarthed** – Для АСН. Показание датчика заземления: 1 – заземлено, 0 – незаземлено;
- **cvHandleState** – Для АСН. Положение наливного стояка: 0 – неопределенное, 1 – рабочее;
- **cvDozeV** – Заданная доза в литрах;
- **cvDozeM** – Заданная доза в деньгах;
- **cvDozeW** – Заданная доза в килограммах. Зарезервировано для будущего использования;
- **cvCurrentV** – Текущее значение отпускаемой дозы в литрах;
- **cvCurrentM** – Текущее значение отпускаемой дозы в деньгах;

- **cvCurrentW** – Текущее значение отпускаемой дозы в килограммах. Вычисляется контроллером или драйвером исходя из объема и плотности;
- **cvTotalV** – Значение фискального счетчика в литрах;
- **cvTotalM** – Значение фискального счетчика в деньгах. Зарезервировано для будущего использования;
- **cvTotalW** – Значение фискального счетчика в килограммах;
- **cvTemperature** – Текущая температура отпускаемого продукта;
- **cvLiterPrice** – Цена 1 литра продукта;
- **cvCalibV** – Значение 1 импульса счетного устройства, мл/сигнал;
- **cvInspector** – Счетчик инспектора (фискальная память, увеличивается при изменении тарировочных коэффициентов, после увеличения возврат в предыдущее состояние невозможен);
- **cvFullUp** – Для АСН. Количество импульсов от начала отпуска до открытия клапана макс. расхода;
- **cvFullDown** – Количество сигналов расхода до окончания отпуска для предварительного снижения производительности отпуска (для уменьшения гидроудара при окончании отпуска и увеличения точности отпуска);
- **cvUSSWaitTime** – Время ожидания сигналов расхода во время отпуска до перехода в состояние csPaused (для того, чтобы насос не работал вхолостую при закрытом клапане);
- **cvAirTime** – Для АСН. Время, на которое открывается воздушный клапан после окончания отпуска (для заполнения наливного наконечника воздухом и слива остатков топлива из него в цистерну);
- **cvTrapState** – Для АСН. Положение перекидного трапа - 0 неопределенное (рабочее), 1 гаражное (нерабочее);
- **cvPropan** – Только для газонаполнительного оборудования. Процентное содержание пропана в отпускаемом газе. Задается извне для расчета контроллером плотности отпускаемого газа;
- **cvKGPrice** – Цена 1 кг отпускаемого продукта;
- **cvBasePost** – Базовый пост данного контроллера. Переменная присутствует только если контроллер содержит несколько постов. Если два поста имеют одинаковое значение **cvBasePost**, они относятся физически к одному контроллеру. Некоторые переменные являются параметрами контроллера, а не поста. Такие переменные располагаются в базовом посту контроллера;
- **cvSide** – Сторона (индикаторная плата). Если два поста имеют одинаковые **cvBasePost** и **cvSide**, они не могут производить отпуск одновременно. Если данная переменная отсутствует, то данный контроллер не поддерживает одновременный отпуск нескольких постов;
- **cvIndex** – Номер поста внутри данной стороны. Если данная переменная отсутствует, то данный пост единственный на стороне. На каждой стороне в один момент времени возможен отпуск только одного поста;
- **cvFirmwareVersion** – Версия ПО контроллера;
- **cvHardwareVersion** – Версия аппаратного исполнения контроллера;
- **cvDensity** – Плотность жидкости в счетчике, кг/м^3. Динамически рассчитывается исходя из текущей температуры, записанной извне плотности, измеренной лабораторными методами (**cvLabDensity**), и температуры при

измерении плотности (**cvLabTemperature**). Присутствует только в случае указания при установке драйвера параметра командной строки /WCALC=YES. Используется при расчете массы отпущенной дозы (**cvCurrentW**);

- **cvProductivity** – Мгновенный расход жидкости, $см^3$/с. Вычисляется драйвером исходя из изменения значений переменных CurrentV и TotalV;
- **cvStateWithFlags** – Состояние поста в виде набора битовых полей представлено в таблице 2.

Таблица 2- Состояние поста в виде набора битовых полей

Положение поля: 0 бит – младший, 7 бит - старший	*Маска шестнадцатеричная*	*Значения поля с указанием его назначения. Значение поля получается в результате выполнения операции «И» над значением переменной StateWithFlags и указанной маской. Значение переменной StateWithFlags является суммой значений полей*
0 бит	01h	00h – пост не получал команды на отпуск от драйвера либо остановил выполнение команды, для продолжения остановленного отпуска или начала нового требуется команда; 01h – пост получил команду на отпуск и находится в процессе ее выполнения. Под процессом выполнения понимается не только непосредственно состояние отпуска, но и приготовление к нему (приведение вручную оборудования в соответствующее положение).
1-2 бит	06h	Комбинированное состояние датчиков безопасности и статуса выполнения текущей команды: 00h – все датчики безопасности указывают на невозможность отпуска (например, для АСН – заземление не подключено, положение перекидного трапа гаражное, нет рабочего положения наливного наконечника); 02h – условия безопасности отпуска выполнены частично; 04h – все условия безопасности выполнены, пост полностью готов к отпуску. Если нулевой бит равен единице - требуется только подтверждение водителя (нажатие кнопки ПУСК), если 0 – то сначала команда драйвера, затем подтверждение водителя; 06h – все условия безопасности выполнены, отпуск выполняется (клапаны открыты или открываются, насосы работают или в процессе запуска). Примечание: контроллеры могут быть сконфигурированы таким образом, что некоторые датчики безопасности отсутствуют или их сигнал игнорируется. В таком случае состояние данного датчика не учитывается при вычислении поля. Контроллеры топливораздаточных колонок не имеют датчиков безопасности, соответственно для них до команды отпуска поле имеет значение 0, после команды – 4.
3 бит	08h	Признак начала фактического отпуска по текущей команде отпуска: 00h – по текущей команде фактического отпуска не было (счетчик жидкости не выдал ни одного импульса расхода); 08h – по текущей команде фактический отпуск выполняется или выполнялся. Если значение поля установлено в 8, оно не сбрасывается в 0 при переходе в состояние паузы (сохраняется значение 8, пока не произойдет полный останов – переход в состояние бездействия).
4 бит	10h	Признак наличия расхода в данный момент времени: 00h – расхода жидкости нет; 10h – жидкость течет. Значение поля вычисляется драйвером, исходя из значения переменной Productivity

Окончание таблицы 2 –Состояние поста в виде набора битовых полей

Положение поля: 0 бит – младший, 7 бит - старший	*Маска шест-надцате-ричная*	*Значения поля с указанием его назначения. Значение поля получается в результате выполнения операции «И» над значением переменной StateWithFlags и указанной маской. Значение переменной StateWithFlags является суммой значений полей*
5 бит	20h	Зарезервировано для использования в будущем.
6 бит	40h	Пауза в процессе отпуска. Требуется вмешательство оператора либо для продолжения, либо для полного прекращения отпуска.
7 бит	80h	Признак возможности нормального функционирования поста: 00h – состояние драйвера и оборудования нормальное; 80h – работа драйвера с данным постом невозможна из-за отсутствия связи, поломки оборудования или несовместимости драйвера с данным типом оборудования. **ВНИМАНИЕ:** в этом случае все остальные биты (0-6) изменяют свое назначение – они указывают на характер проблемы. Значение переменной StateWithFlags равно значению состояния, в соответствии с 6.1.1 для значений >= 128

6.1.3 TState

Значение данного типа представляет собой запись с информацией о состоянии запрошенного поста. Список полей представлен в таблице 3.

Таблица 3

Наименование	*Тип данных*	*Описание*
Value	TStateEnum	Значение состояния.
TimeStamp	DATE	Дата и время, когда значение было определено. При нормальном состоянии связи с оборудованием может отличаться на 1-2 секунды от текущего времени. Во время отпуска поста его состояние и переменные считываются чаще, что позволяет быстрее реагировать на события, однако бездействующим постам в таком случае уделяется меньше внимания. Клиентское приложение должно анализировать данное поле с целью определения, насколько актуально значение состояния.

6.1.4 TVariableValue

Значение данного типа представляет собой запись с информацией о значении запрошенного параметра указанного поста. Список полей представлен в таблице 4.

Таблица 4

Наименование	*Тип данных*	*Описание*
Value	Currency	Значение параметра. Тип Currency использован во избежание ошибок при округлении значения. Для значений любых параметров оборудования достаточна поддерживаемая данным типом фиксированная точность четыре знака после точки.
TimeStamp	DATE	Дата и время, когда значение было определено. При нормальном состоянии связи с оборудованием может отличаться на 1-2 секунды от текущего времени. Во время отпуска поста его состояние и переменные считываются чаще, что позволяет быстрее реагировать на события, однако бездействующим постам в таком случае уделяется меньше внимания. Клиентское приложение должно всегда анализировать данное поле. Интервалы времени, которыми руководствуется драйвер при обновлении (считывании заново из контроллера) значения переменной (параметра), задаются отдельно для состояний отпуска (csWorking, csPaused, csWaitingKey) и прочих состояний. При определении наличия поста драйвер устанавливает по умолчанию интервалы обновления каждой переменной в зависимости от ее вида. Клиентское приложение может устанавливать интервалы по своему усмотрению путем вызова метода IPrompribor::SetVarInterval.

6.1.5 TVariableInfo

Значение данного типа представляет собой запись, содержащую справочную информацию о переменной. Эта информация не изменяется во время жизни переменной. Массив записей данного типа, доступный для каждого поста через вызов ***IPrompribor::GetVariable***, описывает поддерживаемые данным постом параметры (переменные). В каждой записи присутствуют поля, представленные в таблице 5.

Таблица 5

Наименование	*Тип данных*	*Описание*
Code	TVariableEnum	Код переменной. Определяет смысл соответствующего параметра в соответствии с 6.1.2
ShortName	BSTR	Сокращенное кодовое наименование соответствующего параметра латинскими буквами. Используется OPC DA сервером в качестве имени тега.
Name	BSTR	Полное название соответствующего параметра по-русски. Кратко описывает смысл переменной.
IsReadable	VARIANT_BOOL	Флаг, показывающий, возможно ли чтение данной переменной (теоретически переменная может быть только для записи). На данный момент все переменные читаемы.
IsWriteable	VARIANT_BOOL	Флаг, показывающий, возможна ли запись данной переменной, либо только чтение.
MinValue	Currency	Минимально возможное значение переменной.
MaxValue	Currency	Максимально возможное значение переменной.
Step	Currency	Шаг (дискретность) значения переменной

6.2 Интерфейс IPrompribor

6.2.1 IPrompribor::StateJumpValid

```
HRESULT _stdcall StateJumpValid([in] byte Post, [in] TVariableEnum StFrom, [in] TVariableEnum StTo, [out, retval] unsigned int * Result );
```

Возвращает S_OK, если поддерживается команда перехода из состояния StFrom в состояние StTo, иначе E_INCOMPATIBLESTATE. Список состояний и возможных переходов указан в разделе 6.1.1.

6.2.2 IPrompribor::GetState

```
HRESULT _stdcall GetState([in] byte Post, [out, retval] TState * State );
```

Возвращает состояние указанного поста

6.2.3 IPrompribor::SetState

```
HRESULT _stdcall SetState([in] byte Post, [in] TStateEnum AState, [out, retval] TCmdID * Result );
```

Установка состояния AState (например csWaitingKey для начала/продолжения отпуска, csPaused/csIdle для паузы/прекращения отпуска) возвращает ID, по которому позже можно узнать результат (код успешного завершения или ошибки) выполнения команды.

6.2.4 IPrompribor::ReReadVarValue

```
HRESULT _stdcall ReReadVarValue([in] byte Post, [in] TVariableEnum Variable, [out, retval] unsigned int * Result );
```

Команда как можно быстрее обновить в буфере значение данной переменной.

6.2.5 IPrompribor::GetVariablesCount

```
HRESULT _stdcall GetVariablesCount([in] byte Post, [out] int * Count, [out, retval] unsigned int * Result );
```

Количество переменных данного поста.

6.2.6 IPrompribor::GetVariable

```
HRESULT _stdcall GetVariable([in] byte Post, [in] int Number, [out] TVariableInfo * Info, [out, retval] unsigned int * Result );
```

Возвращает информацию о переменной по ее номеру (номер и код - разные понятия!).

6.2.7 IPrompribor::SetVarValue

```
HRESULT _stdcall SetVarValue([in] byte Post, [in] TVariableEnum Variable, [in] CURRENCY AValue, [out, retval] TCmdID * Result );
```

Запись значения переменной. Здесь и далее Variable = Info.Code (Info – запись, возвращаемая методом GetVariable). Возвращает ID, по которому позже можно узнать конечный результат выполнения команды, либо, если сразу обнаружена какая-либо проблема, код результата выполнения. Отличить код результата от идентификатора команды можно следующим образом: у идентификатора старшие 16 бит всегда равны 7FFFh, у кода ошибки – никогда.

6.2.8 IPrompribor::GetVarValue

```
HRESULT _stdcall GetVarValue([in] byte Post, [in] TVariableEnum Variable, [out] TVariableValue * Val, [out, retval] unsigned int * Result );
```

Читать значение переменной

6.2.9 IPrompribor::TryResult

```
HRESULT _stdcall TryResult([in] TCmdID CmdID, [out, retval] TCmdID * Result
);
```

Проверить результат выполнения асинхронной операции. Возвратить код результата, если она закончилась, иначе возвратить CmdID (т.е. на выходе то же, что и на входе).

6.2.10 IPrompribor::GetResult

```
HRESULT _stdcall GetResult([in] TCmdID CmdID, [out, retval] TCmdID * Result
);
```

Дождаться окончания асинхронной операции и возвратить код результата выполнения.

6.2.11 IPrompribor::GetVarInterval

```
HRESULT _stdcall GetVarInterval([in] byte Post, [in] TVariableEnum Variable,
[out] unsigned int * IdleInterval, [out] unsigned int * WorkInterval, [out, retval]
unsigned int * Result );
```

Читать интервалы в мсек запросов с контроллера указанной переменной в состоянии бездействия и при отпуске

6.2.12 IPrompribor::SetVarInterval

```
HRESULT _stdcall SetVarInterval([in] byte Post, [in] TVariableEnum Variable,
[in] unsigned int IdleInterval, [in] unsigned int WorkInterval, [out, retval] un-
signed int * Result );
```

Установить интервалы запросов с контроллера указанной переменной в состоянии бездействия и при отпуске. Например, для cvCurrentV интервал при бездействии можно сделать больше (100000) а при отпуске меньше (200)

6.2.13 IPrompribor::BandUsed

```
HRESULT _stdcall BandUsed([out, retval] double * Result );
```

Средняя загруженность токовой петли – оценка способности драйвера выполнять запросы переменных с требуемыми интервалами.

6.2.14 IPrompribor::RegisterSinkInterface

```
HRESULT _stdcall RegisterSinkInterface([in] IPrompriborEvents * Sink );
```

Передать интерфейс для обратной связи. Драйвер будет асинхронно вызывать его методы, оповещая клиента об изменениях состояния и значений переменных по всем постам.

6.2.15 IPrompribor::RegisterSinkInterfaceEx

```
HRESULT _stdcall RegisterSinkInterfaceEx([in] IPrompriborEventsEx * Sink );
```

Передать расширенный интерфейс для обратной связи. Драйвер будет асинхронно вызывать его методы, оповещая клиента об изменениях состояния и значений переменных по всем постам.

6.2.16 IPrompribor::UnRegisterSinkInterface

```
HRESULT _stdcall UnRegisterSinkInterface( void );
```

Отменить обратную связь. Интерфейс клиента освобождается

6.3 Интерфейс IPrompriborEvents

Данный интерфейс, реализуемый клиентом, желающим вместо циклического опроса драйвера получать изменения немедленно тогда, когда они происходят. Драйвер гарантирует, что клиентскому приложению будут доставляться все изменения.

6.3.1 IPrompriborEvents::PostFound

```
HRESULT _stdcall PostFound([in] byte Post );
```

Драйвером обнаружен пост. В параметре Post его адрес

6.3.2 IPrompriborEvents::StateChanged

```
HRESULT _stdcall StateChanged([in] byte Post, [in] TState State );
```

Состояние указанного поста изменилось. Новое состояние в параметре State

6.3.3 IPrompriborEvents::VarValueChanged

```
HRESULT _stdcall VarValueChanged([in] byte Post, [in] TVariableEnum Variable,
[in] TVariableValue Value );
```

Изменилось значение указанной переменной указанного поста. Новое значение в параметре Value.

6.3.4 IPrompriborEvents::PostDeleted

```
HRESULT _stdcall PostDeleted([in] byte Post );
```

Буфер переменных и состояния для данного поста удален из памяти из-за долгого отсутствия связи с постом.

6.4 Интерфейс IPrompriborEventsEx

Данный интерфейс наследуется от IPrompriborEvents и, следовательно, содержит все его методы, и в дополнение:

6.4.1 IPrompriborEventsEx::PostWithoutChanges

```
HRESULT _stdcall PostWithoutChanges([in] byte Post, [in] DATE Stamp );
```

Периодически вызывается при неизменном состоянии и неизменных значениях переменных поста.

Лист регистрации изменений

Изменение	Номера страниц (листов)				Всего листов (страниц) в документе	Номер документа	Входящий номер сопроводительного документа и дата	Подпись (фамилия)	Дата внесения изменения,
	Измененных	Замененных	Новых (дополнительных)	Аннулированных					