E-mail: modus@swman.ru
WWW: http:\\www.swman.ru
Тел/Факс.: (495) 642 89 62,(499) 267 79 59

ActivesXeme

Версия 5.20

г. Москва

Компания Модус

Содержание

Часть 1. Компонент ActivesXeme

1.1 Назначение

Компонент **ActivesXeme –** система отображения графической информации, представленной в виде документа или схемы формата SDE - предназначен для разработчиков прикладного программного обеспечения (ПО). Схемы в фармате SDE и XSDE готовятся с помощью графического редактора Модус. Документ может состоять из нескольких страниц, на которых расположены объекты, с различными характеристиками.

Компонент *ActivesXeme* дает возможность встраивать графический модуль в ПО, созданное с помощью современных средств разработки (Borland Delphi, C++ Builder, Microsoft Visual C, Microsoft Visual Basic и др.) широко используемых в России.

Уважаемые коллеги - разработчики прикладных задач для платформы Windows! Наш опыт показывает, что разработка и сопровождение специализированной графической подсистемы высокого качества является весьма трудоемкой и дорогостоящей задачей. Созданная нами объектно-ориентированная графическая система отвечает высоким требованиям заказчиков и уже используется в качестве основного средства подготовки электронных схем многими энергосистемами, в т.ч Мосэнерго.

Технология COM/ActiveX позволяет нам обеспечить простой и удобный доступ Ваших программ к нашей графической подсистеме. Теперь Вы можете целиком сосредоточиться на решении прикладных задач, а коллеги-технологи смогут использовать привычный интерфейс. Избавьте себя от перерисовки схем для каждой задачи, а нас – от конвертации данных из одного формата в другой! Многие Ваши коллеги уже оценили преимущества такого подхода. В настоящее время нашу графическую подсистему используют около 50 компаний – разработчиков программного обеспечения и предприятий электроэнергетики в России и СНГ.

Имеется успешный опыт использования компонента в приложениях разработанных в средах Visual Studio 6.0 (Visual Basic, Visual C++, Visula FoxPro), Borland Delphi 4-7, Borland C++ Builder 3-6, в Oracle Developer, MS Access (VB for Applications), Visual Studio .Net (C#, VB.Net) и др.

1.1.1 Использование в корпоративных сетях

Так как ActivesXeme реализует технологию ActiveX фирмы Microsoft, он может быть легко интегрирован в Web-страницы для просмотра схем через Интернет.

Уважаемые коллеги - разработчики корпоративных решений! Теперь Вы можете организовать централизованный доступ к альбомам схем. Наш компонент протестирован для использования в MS Internet Explorer.

1.1.2 Спектр решаемых задач.

Опыт автора показывает, что разработка и сопровождение специализированной графической подсистемы высокого качества является весьма трудоемкой и дорогостоящей задачей. В то же время среди разработчиков, занимающихся информационными технологиями в области энергетики, при решении большинства видов технологических задач, требуется обеспечить работу со схемами и данным, связанным с ними и с их элементами. Будем называть такие задачи **схемно-ориентированными**, а соответствующий пользовательский интерфейс – схемно-ориентированным.

Приведем примеры подобных задач:

1. Согласование имеющихся у пользователя баз данных с элементами схем. Каждому элементу схемы можно назначить один или несколько запросов к базе и обеспечить показ их результата по нажатию клавиши мыши на элемент схемы.
2. Обеспечение автоматического отображения нужной схемы и элемента в ней, соответствующего выбранной записи из базы данных.
3. Отображение актуального состояния элементов схемы и параметров режима на участках схемы. Данные о состоянии элементов (положение коммутационных аппаратов, результаты измерений, выраженные численно, выделение выбранных элементов) обрабатываются прикладной программой и отображаются средствами взаимодействия с графическим компонентом. Источниками информации могут быть ОИК, база данных, программы расчета и анализа режима сети и т.п.
4. Создание графического интерфейса расчетных программ. Может быть организован автоматический запуск ПО расчета режима по факту переключения коммутационного аппарата, просмотр параметров расчетной модели через специальные элементы отображения на схемы (в виде текста, в форме показаний макетов приборов).
5. Дистанционное управление объектом. Воздействие оператора программы на элемент схемы может быть преобразовано в команду системы телеуправления объектом.
6. Учет состояния схем в процессе обработки и согласования заявок.
7. Конструирование различных видов тренажеров (коммутационных, коммутационно-режимных).

8. Ведение мнемосхемы (возможно, с одновременным ведением истории ее изменений и диспетчерских журналов).

Очевидно, что полное решение комплекса вышеперечисленных задач нереально силами одной фирмы – разработчика, однако возможно решить задачу обеспечения единого схемного интерфейса для всех видов технологических задач, связанных со схемной графикой.

При выборе платформы для разработки мы будем использовать следующие предположения:

1. Подавляющее большинство персональных компьютеров работает в настоящее время под управлением операционной системой Windows

2. Графическая система может быть разработана силами одной команды программистов, использующих какое – либо средство разработки под Windows. В то же время разработчики прикладных систем используют другие языки и средства разработки, поэтому включение графической системы как части монолитного приложения в большинстве случаев невозможна. В том случае, если разработчики графической и прикладной системы используют одно и то же средство разработки, такое включение осложняется следующими факторами:

- графическая система может содержать много дополнительных библиотек (компонент), что усложняет итоговую сборку прикладной системы (проблемы сложности системы);
- усложнение поддержки версий компонент (конфликт версий). Так, если пользователь использует документ, созданный более поздней версией программного обеспечения, чем та, с которой которая было скомпилировано прикладное приложение, то такой документ может не открыться приложением (проблемы конфликта версий);
- проблемы с возможным неконтролируемым распространением исходных кодов (проблемы лицензирования);

Эти проблемы могут быть разрешены при использовании технологии COM / ActiveX, обеспечивающей простой и удобный доступ прикладных программ к графической подсистеме. При таком подходе разработчики могут целиком сосредоточиться на решении прикладных задач, а технологи - использовать привычный пользовательский интерфейс.

Проектирование и создание COM - компонента требует высокой квалификации команды – разработчика, и меньшей квалификации – у прикладных программистов, использующих его.

В минимальном варианте, графическая система может разрабатываться и поставляться в виде двух составных частей (базовых приложений) – графический редактор + компонент ActiveX. Используя предложенный подход, прикладной разработчик или конечный пользователь может подготовить схему в графическом редакторе, а затем «оживлять» ее в собственном приложении, используя так называемую *объектную модель* схемы (предоставляющую как список элементов с указанием типа, так и топологию схемы, которая строится автоматически при рисовании схемы в графическом редакторе) и набора событий пользовательского интерфейса (выбор элемента на схеме пользователем, изменение масштаба схемы и т.п.).

Такой подход позволяет существенно уменьшить затраты на разработку диспетчерских и других технологических комплексов, исключив необходимость разработки собственной графической системы. Моделирование сущностей (интерактивная модель) осуществляется в графической системе, данные модели также хранятся в графической системе, либо могут быть распределены между графической системой и прикладным приложением. Функциональная модель, оперирующая данными интерактивной модели, реализуется в прикладной программе.

Далее будут также рассмотрены подходы, допускающие повторное использование функциональных моделей в нескольких прикладных приложениях.

Схема использования компонента прикладной программой представлена на рис. 1.

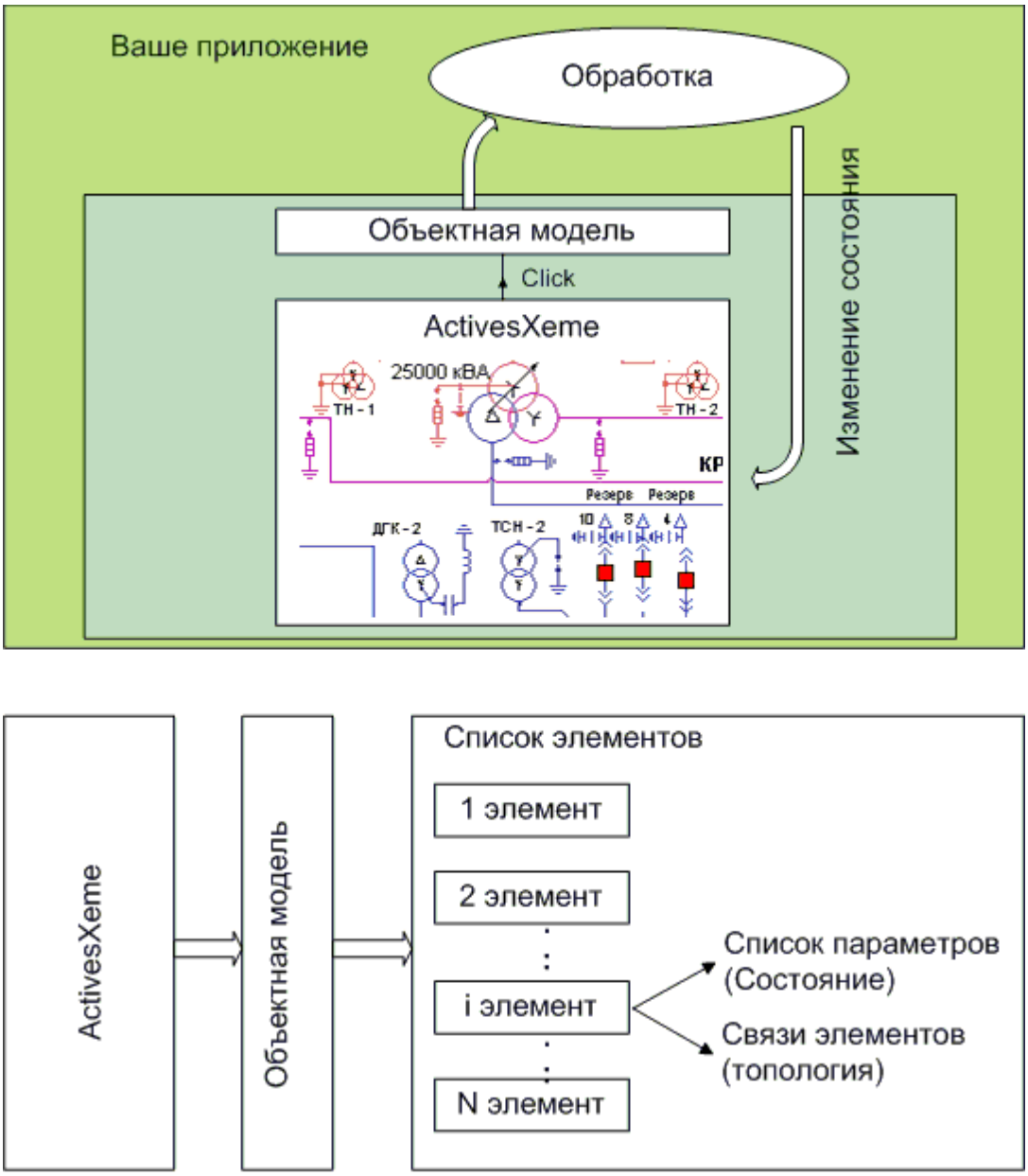


Рис. 1. Использование компонента ActivesXeme

1.1.3 Интерфейс компонента.

Компонент ActivesXeme обеспечивает:

- Загрузку и отображение схемы из файла данных.
- Печать изображенияю
- Переход по связям элементов схем (гиперссылкам).
- Интерфейс управления отображением - масштабирование, прокрутку, детализацию.
- Быстрый поиск участков схемы с использованием окна навигации
- Доступ программы к элементам схемы по их идентификаторам

- Поиск и показ элемента по идентификатору.
- Доступ к элементу под курсором мыши.
- Изменение состояния элементов схемы.
- Обработку реакции приложения на нажатие клавиши мыши в поле схемы и на переход по гиперссылке.
- Отображение обесточенных участков схем в соответствии с состоянием коммутационной модели.
- Настройка вида схемы в соответствии со стандартами, принятыми в организации пользователя.

1.2 Опыт использования

Проблема использования схем, подготовленных в графическом редакторе Модус, возникла фактически сразу же после появления графического редактора (1995-97), быстро ставшего популярным среди пользователей - энергетиков. Первые попытки разработчиков Модус привязки графической системы к другим системам (паспортизации, расчета режима) путем внедрения исходного кода графической системы окончились неудачей из-за чрезмерной сложности получающегося исполняемого кода и проблем с обновлением версий. В то же время были сформулированы основные требования к программному интерфейсу для использования графической системы в технологических приложениях.

Следующей попыткой интеграции, на этот раз удачной, стал компонент ActivesXeme, разработанный с использованием технологии ActiveX и в настоящее время пользующийся популярностью среди разработчиков прикладных решений.

В нем был реализован набор программных интерфейсов (объектная модель), позволяющих прикладному программисту осуществлять контроль над схемой и расположенными на ней объектами.

Опыт использования компонента в решениях компании Модус (редактор связей с OPC, редактор связей с БД) а также общение с разработчиками, использующих графическую систему, позволили сформулировать новые требования к модулю, реализующему открытый доступ к схемной графике. Среди этих требований:

- Корректная работа во всех популярных средах разработки, позволяющих использовать ActiveX - модули.
- Единообразная объектная модель для ActivesXeme и других приложений Модус,

предоставляющая возможность написание модулей, использующихся совместно с разными приложениями Модус.

- Поддержка стандартных соглашений в интерфейсах (например, в реализации коллекций). Использование дуальных интерфейсов, предоставляющих варианты работы как с ранним, так и поздним связываем.
- Доступность настроечных данных (как, например, стилей отображения) для разработчика прикладного комплекса.
- Реализация типовых операций с объектами (например, поиска по имени) в объектной модели, что облегчает использование компонента и увеличивает быстродействие системы.
- Увеличение скорости и «гладкости» работы системы в части отображения графики.

В соответствии с этими новыми требованиями, начиная с 4 версии, в комплекс вошел компонент ActivesXeme2.

Компонент НЕ поддерживает программные интерфейсы ActivesXeme(компонент для версии 3). Для разработчиков, использующих ActivesXeme и планирующих перейти на 4 или 5 версию комплекса, мы рекомендуем в первую очередь ознакомиться с этим руководством. Как показывает опыт, переход ActivesXeme на ActivesXeme2 связан с существенным УПРОЩЕНИЕМ связующего кода - его объем уменьшается в 2-3 раза, читаемость улучшается) . В компоненте ActivesXeme версии 3.5 уже было реализовано расширение объектной модели, похожее на объектную модель ActivesXeme2, переход с этой модели должен произойти без особого труда.

1.3 Установка в среде разработки.

Для использования компонента в среде разработки требуется установить программный комплекс с дистрибутивного диска Модус. Как минимум, при установке должен быть выбран пункт *Средства разработчика\ActivesXeme,* также рекомендуется установить *Графический* редактор.

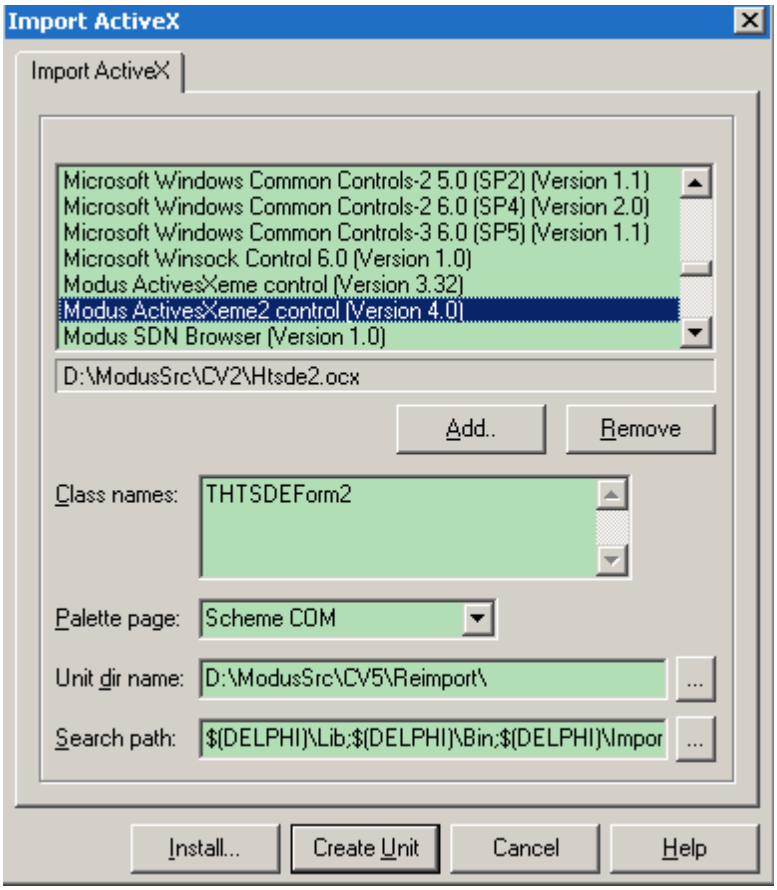
Имейте в виду, что при использовании незарегистрированной (имеется в виду регистрация лицензии) версии периодически будут возникать сообщения об использовании нелицензионной версии, ее возможно бесплатно зарегистрировать в ознакомительных целях, смотреть на сайте раздел *Бесплатная регистрация для разработчиков* http://

www.swman.ru/pricing/free_developers.htm.

После установки общих компонент все необходимые COM / ActiveX компоненты окажутся зарегистрированы в системе, далее их нужно установить для дальнейшего использования в среде разработки, здесь приводятся примеры установки в наиболее популярных средах.

Установка компонента в среде *Borland Delphi*

1. Выполните стандартную процедуру установки ActiveX компонента для Delphi, выбрав в главном меню пункт ***Component / Import ActiveX Control***.



2. В появившемся окне *Import ActiveX*, из выпадающего списка *Registered Control* выберите пункт **Modus ActivesXeme2 control** (если у Вас этот пункт не появился, выполните процедуру регистрации компонента заново, либо обратитесь в Модус – мы проконсультируем), при этом в окне *Class Names* у Вас появится название нового класса *HTSDEForm2*. Нажмите на кнопку установки компонента.
3. По завершении процесса компиляции и установки в среде Delphi на странице, указанной

в поле *Palette Page* (по умолчанию *ActiveX)* появится новый компонент *HTSDEForm2.*

Установка компонента в *C++ Builder*

Все совершенно аналогично установке в Delphi.

Установка компонента в *Microsoft Visual C*

1 a) В главном меню выберите пункт:

Project / Add To Project / Components and Controls…

b) В появившемся окне *Components and Controls Gallery* выберите компонент **Modus ActivesXeme2 control**.

2 a) Нажмите правую клавишу мыши над формой

b) Выберите в появившемся меню пункт **Insert ActiveX Control…**

c) В появившемся окне выберите компонент **Modus ActivesXeme2 control**.

Установка компонента в Microsoft Visual Basic

1 a) В главном меню выберите пункт: ***Project / Components…***

b) В появившемся окне (на вкладке **Controls**) выберите компонент **Modus ActivesXeme2 control**.

2 a) Нажмите правую клавишу мыши над панелью **Toolbox**

b) Выберите в появившемся меню пункт **Components…**

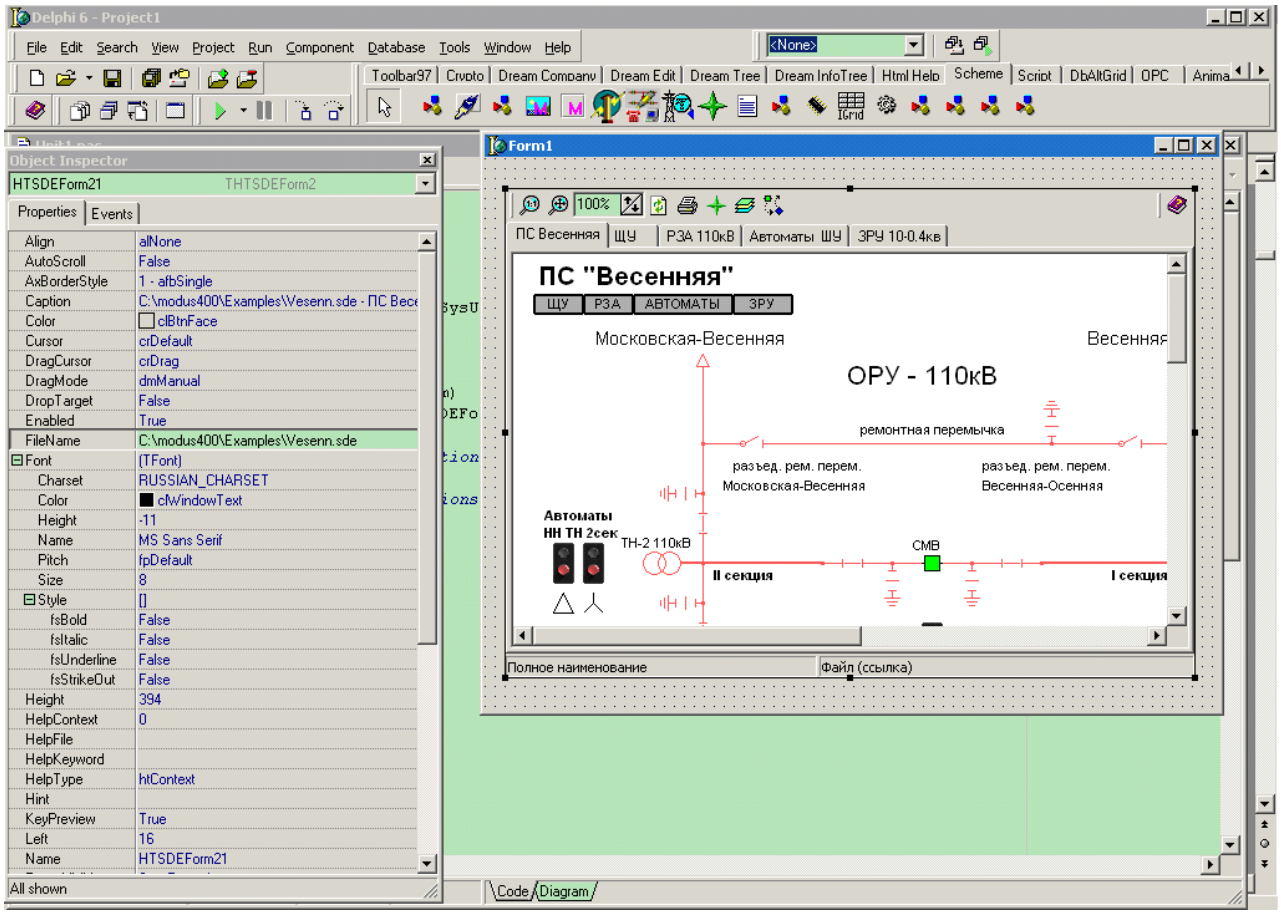
c) В появившемся окне выберите компонент **Modus ActivesXeme2 control.**

Далее Вы можете использовать компонент из палитры компонентов среды разработки на свою форму так, как описано в документации среды разработки. Либо открыть пример из поставки Модус, соответсвующий тому языку программирования, который Вы используете. Все примеры из поставки находятся по умолчанию в директории *C:\Modus420\ActivesXeme* - версия 4.20 и *C:\Program Files\modus500\ActivesXeme* - версия 5.0.

1.4 Начало работы

Проиллюстрируем на примере Delphi создание простого приложения.

1. Создайте свой проект с единственной формой.
2. Поместите на форму компонент ActivesXeme из палитры (в палитре Delphi он называется HTSDEForm2).
3. Выберите на форме компонент, в инспекторе свойств выберите свойство FileName, в него впишите имя файла со схемой SDE, расположенной на диске вашего компьютера. Нажмите Enter.
4. В установленной на форме компонете должна показаться схема.
5. Запустите приложение.
6. Все. Ваше первое приложениеие с использованием ActivesXeme готово.



См. пример Step1

Примечание. В случае использования незарегистрированной версии как при установке на форму, так и при старте приложения с компонентом может происходить задержка в 3-5 сек. Это нормально. В случае использования зарегистрированного компонента такой задержки быть не должно.

1.5 Загрузка документа. Используемые форматы.

Первое чему нужно научиться при использовании компонента – загружать в него подготовленные (с помощью графического редактора) документы. Это можно сделать несколькими способами, вызывая разные программные интерфейсы.

См. пример Step2

```
0: HT.FileName := Opendialog1.FileName;
1: HT.Open(Opendialog1.FileName);
2: HT.Document.SourceFileName := Opendialog1.FileName;
3: HT.Document.Open (Opendialog1.FileName);
```

все эти способы открытия файлов равноправны, можно использовать любой из них. Здесь проиллюстрированы разные способы доступа.

Компонент умеет загружать файлы формата SDE и XSDE (схемы комплекса Модус), подготовленные версиями графического редактора начиная с версии 2.05 (1998 г.). Более ранние версии формата компонентом не поддерживаются. В случае использования схем более ранних версий, необходимо открыть их графическим редактором и сохранить (рекомендуется под другим именем), в этом случае они сохраняются в формате последней версии.

Также компонент умеет загружать файлы формата XSDE. Это вариант схемы SDE, сохраненный с использованием XML формата. Его преимущество – открытость. Недостатки – большой объем файла со схемой и несколько более медленная загрузка и сохранение (то и другое примерно в 2 раз больше) для пятой версии комплекса.

Внимание! В компоненте реализована также возможность сохранения файла в формате SDE.

При этом гарантируется СОВМЕСТИМОСТЬ версий, то есть компонент более поздней версии прочитает файл со схемой, подготовленный более ранней версией графического редактора или компонента. Однако в случае использования компонента более поздней версии, файл при сохранении может быть записан в формате, не совместимом с используемой версией граф редактора. Система не обеспечивает ОБРАТНУЮ СОВМЕСТИМОСТЬ версий документа. На практике, обычно в такой ситуации, при открытии документа пользователь увидит сообщение, что документ подготовлен более поздней версией графической системы и не может быть открыт при сохранении можно указать сохраняемую версию.

1.6 Знакомство с объектной моделью. Документ.

В предыдущем разделе нам встретилось выражение

HT.Document

Здесь HT – это имя компонента HTSDEForm в приложении, а его метод Document предоставляет нам доступ к *объектной модели* схемы, через которую мы в дальнейшем будем получать все остальное содержимое схемы. Document является интерфейсом типа ISDEDocument. Понятие документа соответствует понятию файла со схемой SDE. В один компонент HTSDEForm одновременно может быть загружен только один документ. Если в приложении Вам необходимо иметь несколько загруженных схем, то возможно загрузить в нем несколько компонентов ActivesXeme2.

Понятие документа является одним из основных в графической системе Модус. Важно, что через понятие интерфейс и документу к нему Вы можете с помощью одного и того же программного кода работать как с объектной моделью компонента ActivesXeme2, так и других приложений комплекса (например, с Интегратором схем), получая доступ к объектной модели из плагина (см. отдельный том документации по работе с плагинами).

Заметим, что не все приложения комплекса являются однодокументными, как ActivesXeme2. Например, Интегратор схем является многодокументным приложением. Как работать с коллекциями документов, будет описан позже, а пока перечислим основные свойства и методы интерфейса ISDEDocument.

Наименование	Тип	Описание

свойств		
Flat (read)	ISDEObjects2	Обеспечивает доступ ко всем объектам схемы посредством интерфейса ISDEObjects2
FlatIndex (read / write)		Задает/определяет поле, по которому осуществляется доступ к объекту, через интерфейс ISDEObjects2
Application	IDispath as ISDEApplication	Доступ к интерфейсу SDE - приложения
Author	String	Автор документа
Comments	String	Комментарии к документу
CurrentEntity	IDispatch	Зарезервировано для использования во внутреннем сервере состояний.
CurrentPage (read)	ISDEPage	Доступ к текущей странице
CurrentPageIndex (read / write)	Variant	Номер текущей страницы. При присвоении числового значения открывает текущую страницу. При присвоении строки открывает страницу с указанным именем.
DetailsLevels	IDetailsLevels	Коллекция уровней детализации
Int	integer	Зарезервировано
FullName	String	Полное название документа
KeyWords	String	Ключевые слова

MultiSelect (read / write)	boolean	Включает / отключает режим множественного выделения объектов. Действует не во всех приложениях комплекса.
Name	String	Имя документа
Path	String	Путь к документу (директория, в которой он находится)
Pages	ISDEPages	Обеспечивает доступ к страницам схемы посредством интерфейса ISDEPages, позволяющим получить число страниц в схеме и доступ к конкретной странице схемы
ParamBlock	IDispatch	Зарезервировано для использования во внутреннем сервере состояний.
ReadOnly (read)	boolean	Отвечает, был ли документ открыт только для чтения
RTID	integer	Уникальный в пределах приложения идентификатор документа.
Saved	boolean	Отвечает, был ли документ сохранен
SelectedView	ISDEObject	Должен обеспечивать доступ к выбранному объекту посредством интерфейса ISDEObject2 (Если выбрано несколько, то к одному из них)
SelectedViews	ISDEObjects	Обеспечивает доступ к коллекции выбранных объектов. Необходимо использовать его в режиме множественного выделения (multiselect).

Selections	IHighLights	Коллекция групп выделения (см. Выделения)
SourceFileName (read / write)	String	Имя файла, содержащего схему. При присвоении нового имени открывается схема с указанным именем. Дополнительная возможность: при присвоении пустой строки (‘’) схема очищается.
Subject	String	Тема документа
Title	String	Заголовок документа
UserProperties	IUserProperties	Коллекция введенных пользователем именованных свойств. См. пользовательские именованные свойства.
Activate		Активизирует документ (используется в многодокументных приложениях, в этом случае окно с документом всплывает поверх других)
BeginUpdate		Для массового обновление параметров – начало транзакции.
Close		Закрывает документ (также полезно в основном в многодокументных приложениях)
DocGoToView	Element: SDEObject2	позиционирует схему для показа объекта и выделяет его. procedure DocGoToView(const Element: SDEObject2); Element – показываемый объект
DocHightLight	Element: ISDEObjects2;	Позиционирует схему к элементу и выполняет хорошо заметную подсветку. Color – цвет

	Color: long; Restrict: boolean	подсветки (RGB). Restrict = false – выполняет выделение сходящейся рамкой, затем мигает несколько раз стилем. Restrict = true – только мигание стилем.
EndUpdate		Для массового обновление параметров – конец транзакции.
FindViewIndex	Index: string; Value: string; Element: ISDEObject2	Находит объект по его имени и индексу и обеспечивать к нему доступ посредством интерфейса ISDEObject2. подробнее – см. поиск объектов.
FindViewsIndex	Index: string; Value: string; Element: ISDEObjects2	Находит коллекцию объектов по признаку. Необходимо применять вместо **FindViewIndex** в том случае , если в результате поиска ожидается более, чем один объект. подробнее – см. поиск объектов
Open	FileName: string; Result: boolean	Открывает файл (см. загрузка документа)
Print		Вывести документ на печать
PrintOut		То же, для совместимости с Basic названо иначе
Save	FileName: Variant; Version: Variant	Сохранение документа. В качестве версии можно указать например ‘3.20’ в таком случае документ сохраняется в формате версии 3.20. На полную совместимость версий компонент не тестировался. Если указана пустая строка, сохраняет в формат текущей версий.

свойства и методы интерфейса ISDEDocument введенные в версии 4.2.

Наименование свойств	Тип	Описание
LightChanging (Read\Write)	boolean	
NeedRecreateElectric Model	boolean	
FindDamaged	Mask: TxFindMask; Mode: TxMaskChoice	
RefreshIndex	String	
DeleteIndex	String	
Graphics	TypeFormat: TxTypeFormat; ObjectRTID: integer; DefaultOrient: WordBool; DefaultSize: WordBool	
ProtectionModel		

свойства и методы интерфейса ISDEDocument введенные в версии 5.0.

Наименование свойств	Тип	Описание
Techs		

1.7 Соглашения. Коллекции.

При реализации объектной модели по COM разработчики придерживались рекомендаций Microsoft.

Дуальные интерфейсы

Все интерфейсы доступа выполнены в виде Interface – CoClass. Каждому классу соответствует ко-класс, что позволяет выполнять полный контроль из средств разработки, использующих позднее связывание (в т.ч скриптовых языков).

Реализованы коллекции

Однотипные объекты могут быть доступны из в коллекций. Любая коллекция имеет метод Count, возвращающий число элементов в ней. Элемент из коллекции может быть извлечен с помощью свойства Item (Index: Variant); В качестве числового индекса может выступать число в диапазоне 0.. Count.

Если объект в коллекции может характеризоваться именем (например страница в коллекции страниц), то доступ к ней в коллекции можно получить по имени, передавая его как параметр-строку.

Во всех коллекциях реализован метод _NewEnum, что позволяет организовать перебор коллекций из языков, в которых используется оператор ForEach (Basic, C# и др.).

Пример перебора коллекции из среды Delphi:

```
S : = '';
for i := 0 to HTRootForm1.Document.flat.Count-1 do
  S := S + HTRootForm1.Document.flat.Item[i].SNIdent+ ^M;
//Формирует строку, включающую в себя имена всех элементов в схеме
```

Пример перебора для VB.Net

```
Dim Page As SDECore.SDEPage
ListBox1.Items.Clear()
ListBox1.Items.Add("Страницы:")
For Each Page In AxHTSDEForm21.Document.Pages
```

```
        ListBox1.Items.Add(Page.Name)
    Next
; добавляет в список имена всех страниц в схеме
```

Экспонируемые свойства и методы

Интерфейсы ISDEDocument и ISDEApplication релизованы в соответвии со спецификацией Microsoft на серверы автоматизации, для этого в них реализован ряд обязательных методов:

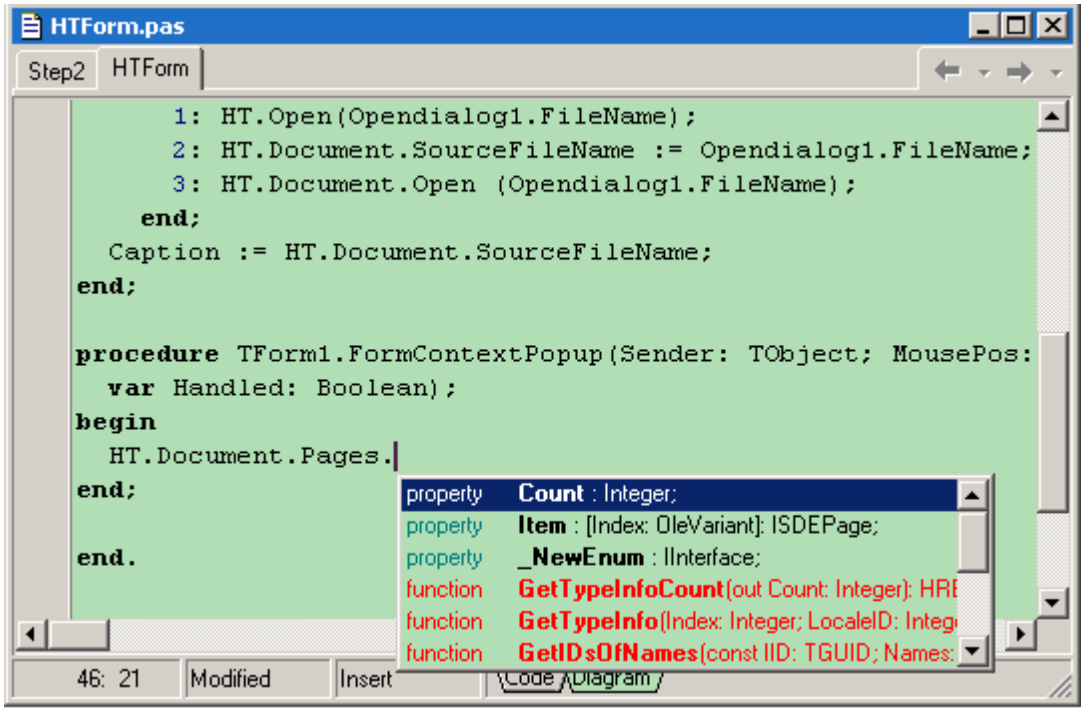
ISDEDocument	**B ISDEApplication**
Свойства	***Свойства***
Application Author Comments FullName Keywords Name Parent Path ReadOnly Saved Subject Title	ActiveDocument Application Caption FullName DefaultFilepath Documents FullName Height Interactive Left Name Parent Path StatusBar Top Visible Width
Методы	***Методы***
Activate Close NewWindow	Help Quit Repeat

Print PrintOut Save SaveAs	Undo

1.8 Исследование программного интерфейса компонента. Библиотеки типов.

Так как компонент постоянно развивается, со временем в новых версиях могут появляться новые интерфейсы. Хотя те интерфесы, которые описаны в этой документации измениться не должны – ингаче это противоречило бы правилам работа с COM – объектами.

Удобно иметь средство для удобного просмотра использующихся программных интерфейсов. Такие средства, как правило, предоставляются средой разработки, например средством Code completion в Delphi (см. рис). Другие средства разработки также предоставляют аналогичные возможности просмотра.



Для того чтобы просмотреть полный список интерфейсов, реализовнных в компоненте, можно использовать просмотр непосредственно файлов OCX и TLB. Для этого, например, в Delphi можно воспользоваться командой Open / File, в появившемся диалоге выбрать с списке типов файлов Type Library (*.tlb;*.dll;*.ocx;*.exe;*.olb) и выбрать нужный файл ocx или tlb.

Внимание! Начиная с версии 4, в отличие от предыдущих версий, описание типов интерфейсов находится не только в модуле htsde2.ocx, но и в отдельных билиотеках TLB. Это файлы

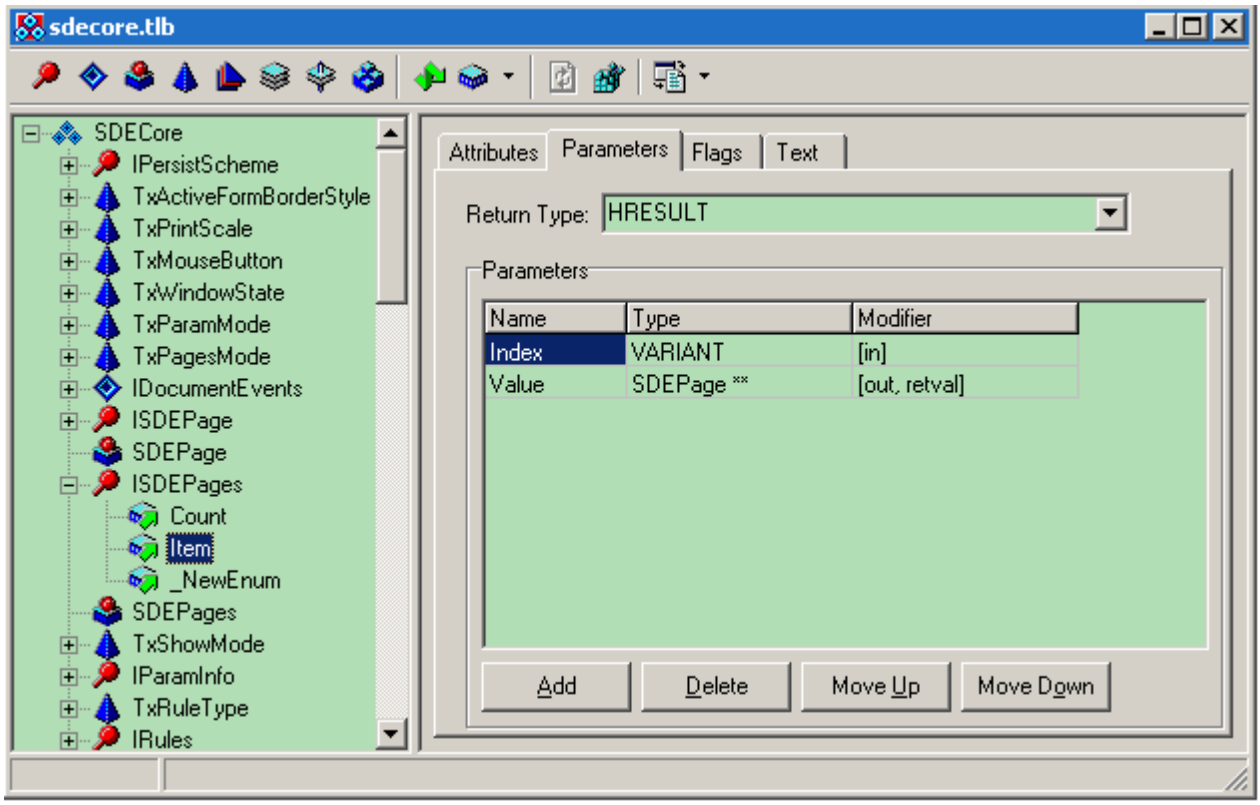
SDECore.tlb – интерфейсы доступа к объектам графического и технологического ядра

SDEApp.tlb - интерфейсы доступа к объектам приложения Модус

SDE_electric.tlb - интерфейсы доступа к технологическим надстройкам Модус (электричсекая модель)

ModusPlugin.tlb – общие интерфейсы модулей расширения (плагинов).

Для использования ActivesXeme наиболее важной является библиотека типов SDECore.tlb. Бо'льшая часть интерфейсов, описанных в руководстве, определяется в этой библиотеке.



Описание интерфейсов вынесено в отдельный модуль, а не описано в модуле htsde2.ocx для того, чтобы использовать идентичный программный код для использования с разными приложениями Модус, предоставляющих доступ к объектной модели, а не только с ActivesXeme2.

В модуле SDEApp.tlb - содержится описание интерфейсов доступа к объектам приложения Модус. В компоненте htsde2.ocx эта библилтека типов не используется, она используется во всех Exe – серверах автоматизации комплекса, ее использование

необходимо в том числе для организации подключения и работы с плагинами.

В модуле SDE_electric.tlb пока находится только интерфейс для включения / отключения и общей настройки раскраски схемы в соответствии с состоянием электрической модели, в дальнейшем здесь будут описываться интерфейсы доступа к самой модели.

В ModusPlugin.tlb описаны общие интерфейсы модулей расширения (плагинов). Более подробную информацию о ее содержимом смотрите в руководстве разработчика плагинов.

Для просмотра содержимого TLB и OCX можно использовать и отдельные утилиты, например TypeLibrary Explorer (http://www.itripoli.com) TLBView, ActiveXExplorer и т.п. Часть из этих утилит является платными.

Для работы комплекса все указанные tlb должны быть установлены и зарегистрированы в системе. По умолчанию они устанавливаются инсталлятором в директорию *C:\Program Files\Modus500\bin* и автоматически регистрируются.

При необходимости компоненты можно зарегситрировать/разрегистрировать вручную. Это можно сделать, например с помощью утилиты trevsvr.exe. В отличие от стандартной regsvr32.exe она позволяет регистрировать бибиотеки типов TLB, а также разрегистрировать библиотеки.

Порядок регистрации имеет значение. Поскольку многие библиотеки ссылаются на sdecore.tlb, эта библиотека должна быть зарегистрирвана вначале. Рекомендуемый порядок регистрации:

Sdecore.tlb

SDEApp.tlb

SDE_Electric.tlb

ModusPlugin.tlb

Далее регистрация всех серверов автоматизации (EXE и DLL).

Разрегистрация компонент производится в обратном порядке.

Еще один способ исследования набора интерфесов – на основе файлов, автоматически формируемых средой разработки при регистрации в ней компонента. Как правило, среда генерирует файлы с описанным в текстовом виде программынм интерфейсом (Delphi – файлы XXX_TLB.pas где XXX- имя библиотеки типов, в директории Delphi\Imports, Visual C++ -файлы PPP.cpp где PPP –название интерфейса и т.п.)

Важно! Виды регистрации

В этом и других разделах упоминается регистрация. Существует по крайней мере 3 вида действий, которые следует различать:

1. **Регистрация COM** – компонент в системе (обычно говорится «регистрация» или регистрация в системе). Помещает информацию о зарегистрированных в библиотеках типов (*.tlb;*.dll;*.ocx;*.exe;*.olb файлах) интрефейсах в системный реестр Windows. Обязательно должна быть выполнена на машинах разработчика и конечного пользователя (обычно производится автоматически дистрибутивом).

2. **Регистрация в среде разработки**. Делает доступным компонент для использования в своих программах на рабочем месте разработчика. Иконка компонента появляется в панели компонентов среды разработки. Выполняется, как описано в документации среды разработки.

3. **Регистрация лицензии**. Выполняется на рабочем мете разработчика либо конечного пользователя после покупки программы (либо получения бесплатной лицензии разработчика). Без регистрации ПО может работать с некоторыми ограничениями, описанными в документации, и периодически выдавать сообщение об использовании нелицензированной версии. Процедура регистрации выполняется при помощи программного, аппаратного или сетевого электронного ключа, описана в документации администратора программного комплекса.

1.9 Работа со страницами схемы.

В документе, подготовленном в графической системе Модус, может быть произвольное количество страниц. В компоненте ActivesXeme2 несколько страниц схемы отображаются в виде закладок, щелкая по которым, можно переключаться с одной страницы на другую.

Режим отображения страниц можно изменить с помощью свойства PagesVisible, которое может принимать значения

txPagesVisible – закладки всегда видны

txPagesHidden – закладки всегда скрыты

txPagesAuto – закладки видны, если их не менее двух в документе (настроено по умолчанию).

С помощью интерфейса SDEDocument.Pages (ISDEPAges) можно получить доступ к коллекции страниц.

Интерфейс ISDEPages

Наименование свойств	Параметры	Описание
Count (read)	integer	Возвращает количество страниц в схеме
Item (read)	Index: Variant; Value: ISDEPage	Обеспечивает доступ к конкретной странице схемы посредством интерфейса ISDEPage ISDEPage – интерфейс для работы со страницей схемы, с объектами, расположенными на странице Доступ к старнице производится по ее порядковому номеру или по имени.

Интерфейс ISDEPage

Наименование свойств	Параметры	Описание
Name (read/write)	string	Задает/определяет имя страницы схемы. Имена страниц в пределах документа должны быть уникальны.
NavPicture (read/ write)	Variant	Позволяет получить (либо переопределить) картинку, сформированную для отображения навигатора схемы, через программный интерфейс в виде ФАЙЛА BMP
Color (read/write)	OLEcolor	Цвет фона схемы

Params		Обеспечивает доступ к параметрам самой страницы посредством интерфейса Params –интерфейс для работы с коллекцией параметров
PosEndX (read) **PosEndY**	integer	Позволяет получить координаты нижнего левого угла скроллера.
PosX **PosY** (read/write)	integer	Текущая позиция скроллера (в пикселях)
RTID	integer	Уникальный числовой идентификатор (уникален в пределах документа)
SizeX **SizeY** (read/write)	integer	Текущая позиция скроллера в текущем масштабе (в пикселях)
Scale (read/write)	double	Текущий масштаб схемы.
SDEObjects		Обеспечивает доступ к колекции объектов страницы посредством интерфейса ISDEObjects2 ISDEObjects2 – интерфейс для работы с объектами
UseTopology	boolean	Использовать ли поддержку топологии для этой страницы (используется при рисовании страницы в граф. редакторе).
WinLeft (read/write) **WinTop** **WinWidth**	integer	Координаты окна на экране (применяется для «плавающих» окон).

WinHeight		

Интерфейс ISDEPage в версии 5.0

Наименование свойств	Параметры	Описание
Elements		
SizeXBase **SizeYBase**	integer	
IntersectElements	X1, Y1, X2, Y2: double	
ContainsElements	X1, Y1, X2, Y2: double	
Reconnect		
GetPictureByCoordinates	X, Y, RigthX,BottomY: SYSINT; const FormatName: WideString; const FileName: WideString; out DataPicture: OleVariant	
GetObjectByCoordinates	X: SYSINT; Y: SYSINT	

У документа есть понятие текущей страницы. Доступ к ней можно получить через свойство *ISDEDocument.CurrentPage.*

Для того, чтобы сменить текущую страницу, можно использовать свойство *ISDEDocument.CurrentPageIndex (Value: Variant*). В качестве Value использовать число (номер страницы) или строку (имя страницы).

Временные ограничения:

Изменять параметры страницы (координаты скроллера, масштаб и т.д.) можно только у текущей страницы.

Планы Планируется добавить интерфейс для добавления / удаления страниц в документе.

Пример: получение картинки навигатора (Delphi) со страницы.

```
//функция взята из библиотеки JCL,
// http://sourceforge.net/project/showfiles.php?group_id=47514
//модуль JCLCOM
function StreamToVariantArray(Stream: TStream): OleVariant;
var
pLocked: Pointer;
begin
{ Use VarIsEmpty to determine the result of this method!
  VarIsEmptry will return True if VarClear was called - indicating major problem! }

if NOT Assigned(Stream) then
  raise EInvalidParam.CreateResRec(@RsComInvalidParam);

if Stream.Size > 0 then
begin
  Result := VarArrayCreate([0, Stream.Size - 1], varByte);
  try
    pLocked := VarArrayLock(Result);
    try
      Stream.Position := 0;
      Stream.Read(pLocked^, Stream.Size);
    finally
      VarArrayUnlock(Result);
    end;
  except
    { If we get an exception, clean up the Variant so as not to return incomplete data! }
    VarClear(Result);

    { Alternative:  Re-Raise this Exception
    raise; }
  end;
```

```
end else
  { Stream has no data! }
  Result := Null;
end;

Var
 Image1: TImage; (не забудьте проинициализировать)

procedure TFSDEDoc.SetPicture;
var
st: TMemoryStream;
begin
if not Assigned(Doc.CurrentPage) then
  exit;
st:= TMemoryStream.Create;
try
  VariantArrayToStream (Doc.CurrentPage.NavPicture, TStream (st));
  Image1.Picture.Bitmap.LoadFromStream(st);
finally
  st.Free;
end;
end;
```

1.10 Перечень объектов на схеме. Интерфейс ISDEObjects.

Для получения списка графических объектов , находящихся на странице схемы, нужно использовать коллекцию *ISDEPage.SDEObjects: ISDEObject2*.

ISDEObjects – коллекция элементов типа *ISDEObject2*. *ISDEObject2* – один из базовых типов графической системы, соответствующий графическому элементу. Через этот интерфейс можно получить полный доступ к элементу, включая его месторасположение, идентификацию и свойства.

Следует иметь в виду, что через ISDEPage.SDEObjects можно получить доступ только к элементам, непосредственно расположенным на схеме. Элементы, находящиеся в контейнере, в коллекцию *ISDEPage.SDEObjects* НЕ попадают.

Для того чтобы получить доступ к элементам, находящимся в контейнере, можно

использовать метод *ISDEObject2.SDEObjects: ISDEObject2*, который вызывается у объекта типа контейнер и выдает коллекцию её подобъектов.

Таким образом, для того чтобы корректно получить полный перечень объектов, находящихся в документе, нужно применять рекурсивный вызов.

Пример:

Составление списка имен объектов, расположенных в схеме (пример Step3).

```
procedure TForm1.Button2Click(Sender: TObject);
var
i,j : integer;
Objs: ISDEObjects2;
begin
Memo1.Lines.Clear;
for i := 0 to HT.Document.Pages.Count-1 do begin
  Objs := HT.Document.Pages.Item [i].SDEObjects;
  for j := 0 to Objs.Count-1 do
    ProcessObject (Objs.Item [j]);
end;
end;

procedure TForm1.ProcessObject(Obj: ISDEObject2);
var i: integer;
begin
Memo1.Lines.Add (Obj.SNIdent);
if Assigned (Obj.Elements) then
  for i := 0 to Obj.Elements.Count-1 do
    ProcessObject(Obj.Elements.Item[i]);
end;
```

Интерфейс ISDEObjects2

Наименование свойств	Параметр	Описание
AddView	Value: SDEObject2	Добавляет существующий на схеме объект к коллекции.
Clear		Очищает коллекцию.
Count (read)	integer	Возвращает количество объектов в коллекции.
CreateView	**(**TypeName: string; Params: string; var Value: SDEObject2**)**	Создает новый объект на схеме. Возвращает интерфейс к вновь созданному объекту.
GetList(read)	**(**QueryType: TxStatQuery; ElemType: string; Param: string; var Value: string**)**	Возвращает информацию о коллекции. QueryType = (sqViewTypes, SqViewParams, SqObjTypes, sqParamValues). Подробное описание см. Доступ к описателям схемы.
Item (read)	Index: variant Var Value ISDEObject2	Возвращает объект из коллекции ISDEObject2 Синтаксис: [ISDEObjects2].Item[Index] В качестве индекса можкт использоваться число (порядковый номер элемента в коллекции)или строка (SNIdent элемента).
IsPresent	Value: SDEObject2; Result: boolean	Проверяет, есть ли данный объект в коллекции.
Name (read)	string	Возвращает имя коллекции.

RemoveView	Value: SDEObject2	Удаляет элемент из коллекции.

Интерфейс ISDEObjects42

Наименование свойств	**Параметр**	**Описание**
UseTopology	Boolean	
NeedRecreateElectricModel	Boolean	
NodeListCount		
GetVListByTyp	integer	
ListType		
RTID		
OwnerView		
OwnerPage		
OwnerDoc		

Важно!

Заметим, что коллекция элементов возвращается многими методами. Среди них получение содержимого страницы (списка объектов), возвращение результата поиска, получение списка выделенных объектов.

В зависимости от того, каким путем получена коллекция, ее методы могут действовать по-разному либо не дейстовать вообще. Например, метод CreateView действует для коллекций ISDEPage.SDEObjects (страница схемы) и SDEObject2.Elements (элементы контейнера) но не действует для коллекции, возвращенной в результатах поиска ISDEDocument.FindViewsIndex.

Метод AddView действует для коллекций, определяющих список стилей группы выделения, но не работает в ISDEPage.SDEObjects. При анализе того, может ли данный метод работать для данной колекции, исходите из здравого смысла.

Интерфейс ISDEObject2.

Интерфейс ISDEObject2

Интерфейс ISDEObject2 предоставляет доступ к элементу схемы.

Этот COM интерфейс позволяет работать с объектом.

Наименование свойств	Тип/ Параметры	Описание
Container	ISDEObject2	То же что Owner.
Elements (read)	ISDEObjects2	Обеспечивает доступ к объектам, находящимся внутри объекта если это контейнер.
LoadStream	Source: variant; Format: TxPersister	Метод для загрузки из потока (десериализация). Формат определяет в каком виде ожидается прочитать объект (коллекция строк, бинарный, XML).
Nodes	INodes2	Доступ к топологии схемы.
NoOfNodes(read)	integer	Число геометрических вершин (не коннекторов). Совпадает с числом значений в массиве координат.
Orient90	integer	Ориентация (кратная 90 град.). 0 – направление вверх, 1 – вправо, 2- вниз, 3 - влево.
Document	ISDEDocument	То же что OwnerDoc.
Owner (read)	ISDEObject2	Владелец - контейнер.
OwnerPage (read)	ISDEPage	Владелец - страница.
OwnerDoc (read)	ISDEDocument	Владелец - документ.

Page	ISDEPage	То же что OwnerPage.
Params (read)	IParams	Обеспечивает доступ к именованным параметрам объекта. Подробнее см. Именованные параметры.
Radius(read)	integer	Радиус (для окружностей, прямоугольников с закругленными углами и т.п.).
RTID	integer	Уникальный числовой идентификатор. Уникальность гарантируется в пределах документа.
SaveStream	Destination: variant; Format: TxPersister	Метод для сохранения в поток (сериализация) . Формат определяет в каком виде сохранять объект (коллекция строк, бинарный, XML).
ShortPath (read)	string	Возвращает путь в числовой нотации (см. Идентификация).
SNIdent (read)	string	Возвращает путь в строковой нотации (уникальный для каждого объекта схемы).
Tag (read)		Возвращает значение номера объекта. Уникальность обеспечена только в пределах типа / контейнера.
Tech	ITechObject	Технологический объект агрегированный элементом схемы. Зарезервировано для будущих версий.
Techs	ITechObjects	Коллекция технологических объектов, обобщенных одим элементом схемы.

		Зарезервировано для будущих версий.
TypeName (read)		Возвращает тип объекта.
VisibleExt (read)	TxVisibility Status	Возвращает статус видимости объекта: Видим / погашен / находится на невидимом уровне детализации.
X, Y (read, write)	Index: integer; Value: double	Координаты элемента. Массив значений. Для центрированных элементов определяет адрес базовой точки (обычно центра или коненктора). Для линий (ломаных), многоугольников и т.п – координаты вершин. Измеряется в единицах сетки (подробнее см. документацию по граф. системе).

Интерфейс ISDEObject42

Наименование свойств	Параметр	Описание
Clone	DestDoc: ISDEDocument; DestPage: ISDEPage; ShiftX, ShiftY: double; internal: WordBool	
Shift	X, Y: double	
ChangeType	NewType: OleVariant	
Typ		
MinimalDistance	V: ISDEObject42	
VInheritsFrom	string	
OwnerList		

Дополнение к интерфейсу ISDEObject в версии 5.0

Наименование свойств	Параметр	Описание
sValue	string	

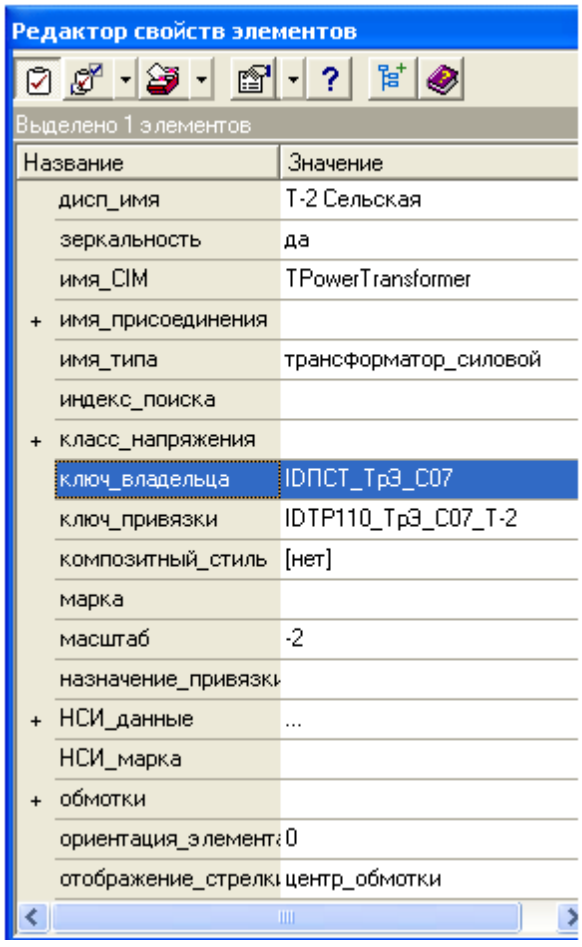
Интерфейс ISDEElectricObject

Наименование свойств	Параметр	Описание
NagrState	TxNagrState	
NoOfExternalConnector	integer	
LineGround	boolean	
UnderVoltage	boolean	
GetViewNodeNum	integer	
NodeState	integer	
DamagedSet		
Closed	boolean	
VoltageLevel	index: integer; Value: integer	
KeyRoleLink	String	
RoleLink	String	
OwnerKeyRoleLink	String	
DispName	String	
EquipType	String	
Locked	boolean	
OperServo	boolean	
OperTok	boolean	
Blinking	boolean	
DOCRTID	integer	
Bounds	SDERect	
GetConnectorPoint	integer	
InternalBounds	SDERect	

1.11 Именованные параметры.

Технологические объекты, размещенные на схеме, могут иметь большое число параметров. Сюда же можно отнести и параметры, влияющие на отрисовку графического объекта. У разных типов графичских объектов набор параметров разный. Для того, чтобы не создавать программный интерфейс с десятками и сотнями параметров, описывающих графический объект, из которых будет использоваться, как правило, только небольшая часть, был разработан механизм именованных параметров.

Пользователи графического редактора хорошо знакомы с ним, при работе в этом приложении при вызове инспектора объетов показывается форма, представляющая список имеющихся у выбранного элемента параметров.



В объектной модели именованному параметру соответствует интерфейс *IParam*. У элемента *SDEObject* определена коллекция параметров *Params : IParams.*

Интерфейс Iparams

Наименование свойств	Параметры	Описание
AsText (read)	string	Возвращает весь набор именованных параметров элемента в виде строки вида параметр=значение;… положение=включен;видимость= видим; …
Count (read)	integer	Возвращает количество параметров в элементе.
Item (read)	Index: Variant; Value: IParam	Обеспечивает доступ к конкретному параметру объекта Доступ к параметру производится по его порядковому номеру в коллекции или по имени параметра.

Дополнения интерфейса Iparams в версии 5.0

Наименование свойств	Параметры	Описание
SParamInfo	String	
SValue	String	
SView	ISDEObject2	
SAddView	ParamNm: WideString; SDEObject: ISDEObject50	

SDeleteParam	ParamNm: WideString	
SSubObject	ParamNm: WideString	
SCreateSubObject	ParamNm: WideString	
ParamsList	Prefix: WideString; RecursionLevel: Integer	
SDisUsed	ParamNm: WideString; Value: WordBool	
SDisUsedIndex	ParamNm: WideString; const Index: WideString	
FindParam	ParamNm: WideString	

Интерфейс Iparam

Вложенные параметры.

	Наименование свойств	**Параметры**	**Описание**
Pr	**Dim** (read)	integer	Размерность параметра (в том случае если параметр индексированный).
Pr	**Category**	TxPararCategory	Категория именованног свойства.

Prw	**IndexedValue** (read, write)	Index: integer; Value: string	Доступ к значению индексированного свойства.
F	**Info**	IParamInfo	Альтернативный доступ к описателям параметра через интерфейс IparamInfo. Добавление пользовательских параметров в схему возможно только с использованием IparamInfo.
Pr	**Mode**	integer	Режим использования параметра и время его жизни.
Pr	**Name** (read)	string	Имя параметра.
Prw	**Value** (read, write)	Value: string	Доступ к значению индексированного свойства.
Pr	**Values**	string	Возможные значения именованного свойства; применимо для перечислимого и строкового типа.
Pr	**ValuesCount**	integer	Число возможных значений для перечислимого типа.

Интерфейс Iparam 4.20

	Наименование свойств	**Параметры**	**Описание**
	PChangeSource (Read\Write)	TxPChangeSource	
	PChangeMode	TxPChangeMode	

	(Read\Write)		
	SDEObjectRef (Read\Write)	ISDEObject2	

Интерфейс Iparam 5.0

	Наименование свойств	Параметры	Описание
	SubObject	IParams50	
	Flags	Index: TxParamFlags	
	DisUsed	Index: OleVariant; Value: WordBool	
	Index2Str	Index: Integer	
	Str2Index	IndexS: WideString	

Параметры идентифицируются по имени. Доступ к параметру элемента можно получить по имени:

```
Var
  Obj: ISDEObject2
  State : string;
State: = Obj.Params.Item [‘положение’];
```

Пример: переключение всех коммутационных аппаратов на схеме (кроме тех, что в контейнерах)

```
procedure TForm1.Button2Click(Sender: TObject);
var
i,j : integer;
Objs: ISDEObjects2;
Param: IParam;
```

```
begin
Objs := HT.Document.CurrentPage.SDEObjects;
for i := 0 to Objs.Count-1 do begin
  Param:= Objs.Item [i].Params.Item ['положение'];
  if Assigned (Param) then
    if param.Value = 'включен' then
      param.Value := ' отключен '
    else
      param.Value := ' включен ';
end;
end;
```

Параметры могут иметь разные типы, но их значение доступно только как строковый параметр через свойство Value.

Параметры бывают следующих типов:

Строчные: пример. дисп_имя = ‘В-1’

Перечислимые.

Параметр может принимать одно из предопределенных положений: положение = (‘включен’, ‘отключен’). Набор возможных значений можно получить через свойство Values.

Числовые

Радиус = 120.

Числовые в заданном диапазоне.

Минимальное возможное значение задается в paramInfo.min, максимальное – в paramInfo.max.

Слой: (1..255) - paramInfo.min = 0; paramInfo.max = 255

Множества:

Повреждение = [КЗ, пожар]

Набор возможных значений можно получить через свойство Values. В отличие от

перечислимых свойств, которые могут принимать ОДНО из множества доступных параметров, причем всегда должно быть определено одно из них, в множестве может быть установлено произвольное число значений из доступных, в том числе и пустое множество.

***Небольшая хитрость*:** на практике часто приходится устанавливать или сбрасывать одно из значений свойства типа множество. При этом, конечно, текущее значение свойства может быть получено и выглядеть примерно так: *композитный_стиль = [выделение,повреждение*]. Пусть в множестве нужно еще и установить флажок *НеКвитирован*. Конечно, просто присвоение значения *[НеКвитирован]* будет некорректно, так как при этом сбросятся те флажки, что уже были установлены. Поэтому корректно разбирать в своей программе строку *композитный_стиль = [выделение,повреждение]* для того чтобы узнать какие флажки были установлены и формировать строку *композитный_стиль = [выделение, повреждение, НеКвитирован]*. Есть и более легкий путь. Достаточно записать в параметр композитный_стиль значение *+НеКвитирован*. При этом уже установленные флажки сохранятся, а новый *НеКвитирован* будет добавлен. Аналогично, для сброса флажка достаточно присвоить значение *–НеКвитирован*. На Delphi это выглядит так :

Obj.Params.Item ['композитный_стиль'] := '–НеКвитирован';

Цвет

Хотя параметры, обозначающие цвет, относятся к строковым параметрам, их обработка делается по-особому. Цвет представляется в виде шестнадцатеричного числа в формате RGB : $FF8844. В том случае, если цвет совпадает с одним из 40 предопределенных цветов, он показывается в виде строки:

черный, коричневый, оливковый, темно_зеленый, тепмно_сизый, темно_синий, индиго, серый_80%, темно_красный, оранжевый, коричнево_зеленый, зеленый, сине_зеленый, синий, сизый, серый_50%, красный, светло_оранжевый, травяной, изумрудный, темно_бирюзовый, темно_голубой, фиолетовый, серый_40%, лиловый, золотистый, желтый, ярко_зеленый, бирюзовый, голубой, вишневый, серый_25%, розовый, светло_коричневый, светло_желтый, бледно_зеленый, светло_бирюзовый, бледно_голубой, сиреневый, белый,

В некоторых случая возможно применение цвета *прозрачный*

Для удобства работы параметры разбиты на несколько групп (категорий). Эти

категории не используются при поиске параметра среди имеющихся у элемента, но используются при их представлении человеку (например, в пользовательском интерфейсе) В инспекторе свойств объекта возможна фильтрация всех возможных параметров по категориям.

Тип / имя	**Примеры**	**Комментарий**
pcLinking Привязки	дисп_имя счетчик вн_номер ключ_владельца	Свойства , идентифицирующие объект или устанавливающие принадлежность объекта
pcPersistent Свойства	класс_напряжения тип_выключателя	Свойства, которые с большой долей вероятности не изменятся за время жизни объекта
pcState Состояния	положение повреждение положение_тележки	Свойства, которые могут изменяться с той или иной периодичностью
pcDrawing Рисование	подпись_выравнивание масштаб слой высота	Свойства, которые влияют на отрисовку самого графического элемента.
ocRegim Режим	м_напряжение нагрузка	Свойства, связанные с электрическим режимом оборудования. Зависят не только от положения оборудования, но и от состояния окружающей сети. Данные поступают извне – из телеизмерений, коммутационной модели и т.п.
pcAnother Остальные		То, что не укладывается в остальные категории

В этом руководстве не ставится цель расшифровки назначени всех именованных

параметров, их проще исследовать в приложениях комплекса, например в граф. редакторе.

Планы

Доступ к значению параметра через тип Variant

Индексированные по имени параметры

Поддержка чисел с плавающей точкой

1.12 Идентификация

При использовании графической системы с другими приложениями обычной задачей является привязка графического элемента к набору внешних для графической системы данных. Для этого у каждого элемента схемы существует несколько идентификаторов. Чрезвычайно важно понимать, какие из идентификаторов в каких случая лучше применять. Рассмотрим набор доступных идентификаторов и их возможную облвсть применения.

Obj.SNIdent

Это полный идентификатор объекта, составленный из имени объекта и пути к нему. Выглядит, например, так:

ПС Центральная\РУ 110 кВ\МВ 110 кВ Т-2

Здесь *ПС Центральная* – имя страницы, *РУ – 110 кВ* – имя контейнера *МВ 110 кВ Т-2* – диспетчерское имя объекта. Если диспетчерское имя не задано, автоматически подставляется имя состоящее из типа и номера, например *выключатель[11]*

Достоинства:

- хорошо воспринимается при чтении.

Недостатки:

- не гарантирована уникальность, так как уникальность диспетчерских имен не отслеживается автоматически, хотя их повторение может детектироваться при верификации схемы в граф. Редакторе, либо другими средствами.

- крайне не рекомендуется производить привязку объектов до тех пор, пока не проставлены все диспетчерские имена объектов.

- имя не устойчиво к перемещениям объектов, так как при перемещении со страницы на страницу, помещении / выносе из контейнера может меняться.

Несмотря на недостатки, применяется сейчас в тренажере для привязки в редакторе упражнений.

Рекомендации:

Использование в качестве идентификатора: возможно при соблюдении определенной дисциплины, но не рекомендуется.

Obj.Tag

Номер объекта. Уникален в пределах одного типа, страницы и контейнера. То есть самостоятельное применение в качестве уникального идентификатора невозможно.

Obj.RTID

32- битовое число.

Внутренний номер объекта. Системой гарантируется уникальность в пределах сеанса загрузки схемы. На самом деле, если проставлено в элементе проставлено поле "счетчик", то значение RTID берется из него. Поле "счетчик" было введено в версии 3.50, однако его корректное поведение (соблюдение простановки для всех элементов и уникальности) было обеспечено не во всех случаях. Начиная с версии 4 работает корректно. Поэтому для гарантии уникальности нужно открыть схему средствами версии 4, 4.20 или 5 (граф. Редактор или ActivesXeme2), сохранить, после этого можно привязывать элементы с использованием тега.

Достоинства

- Лаконичность
- уникальность в пределах документа

Недостатки

- Указание на то что данный объект на разных схемах является одним и тем же, возможно

только внешними средствами.

- Нечитаемость.

Рекомендации:

Использование в качестве идентификатора ГРАФИЧЕСКОГО ОБЪЕКТА: возможно при соблюдении некоторой дисциплины.

Obj.Params.Item [‘дисп_имя’]

Собственное диспетчерское имя элемента.

Достоинства

- читаемость
- В отличие от SNIdent, не зависит от перемещения из контейнера в контейнер, со страницы на страницу.

Недостатки

- В отличие от SNIdent, нельзя обеспечить уникальность за счет иерархии.
- Нельзя производить привязку объектов, пока не проставлены все диспетчерские имена объектов.
- Не обеспечено автоматическое отслеживание уникальности (возможно только с использованием верификации)

Рекомендации:

Можно использовать с соблюдением дисциплины на схемах, имеющих несложную иерархическую организацию.

Obj.Params.Item [‘вн_имя’]

Фрормируется иерархически так же, как SNIdent, только вместо диспетчерских имен используются внутренние номера + указание числа-номера.

Для привязки – все недостатки RTID.

Рекомендации:

Для привязки использовать не рекомендуется.

Obj.Params.Item [‘счетчик’]

То же самое, что RTID.

Obj.Params.Item [‘ключ_привязки’]
Obj.Params.Item [‘назначение_привязки’]

Были введены с перспективой совместного использования. В этой идеологии, "ключ_привязки" является идентификатором основного оборудования, а "назначение_привязки" предоставляет дополнительную информацию (роль). Например, выключатель на схеме имеет определенный "ключ_привязки" и пустое "назначение_привязки", а ключ управления, относящийся к нему, имеет тот же "ключ_привязки" и специальное "назначение_привязки" *КУ*. Более подробно о рекомендациях по правилам формирования и использованию ключей смотрите "Руководство по подготовке схем для электронного журнала" или общую часть документации. Можно считать, что сочетание "ключ_привязки" + "назначение_привязки" однозначно идентифицирует графический объект. Вообще это просто строки, в них можно использовать и идентификацию, принятую в Вашей информационной системе.

Недостатки

Нет встроенных средств гарантирования уникальности, кроме верификации.

Достоинства

Позволяет организовать уникальную идентификацию, действующую не только в пределах одной схемы, но и набора схем. Автоматически производится сопоставление объектов. Автоматически организуется привязка к внешней информационной системе, либо к нескольким системам.
При применении определенного подхода к расстановке – хорошая читаемость.

Рекомендации

Наиболее перспективный подход, позволяющий создавть большие альбомы схем, с разными

вариантами представления схем. Требует предварительного продумывания методики идентификации.

Obj.Params.Item [‘пользовательский_параметр’]

В том случае, если по каким–то причинам Вы не хотите использовать "ключ_привязки" и принимаете решение об идентификации объекта по какому-то особому признаку, например *инвентарный_номер*, то Вы можете воспользоваться возможностью добавить в схему соответвующий параметр. Поскольку целостность именованных параметров поддерживается системой при редактировании в граф. редакторе (при переносе на другую страницу и т.п. он сохраняется), то в целом такой подход аналогичен применению поля "ключ_привязки", за исключением несколько большего потребления ресурсов.

Итак, здесь представлены различные способы идентификации объектов. Выбор системы остается за разработчиком и определяетя уровнем создаваемого комплекса. Возможно и рекомендуется использование разных способов идентификации в разных подсистемах. Например, в подсистемы, работающие в рамках документа, можно использовать RTID, а для интеграции с внешними системами – "ключ_привязки" + "назначение_привязки".

Планы. Формирование строк идентификаторов с помощью гибко настраиваемой маски.

1.13 Флэт - модель. Доступ к описателям схемы.

В разделе "Перечень объектов на схеме" был приведен пример получения полного списка объектов в документе. Поскольку такая операция обхода – часто выполняемая операция (сделать со всеми объектами какую-то операцию), а порядок обхода обычен. Для её облегчения введено понятие флэт-модели. Эта модель предоставляет все элементы документа, представленные в виде коллекции.

Доступ к ней происходит через метод *ISDEDocument.Flat: ISDEObjects* (пример **Step 5**).

Порядок сортировки во флэт – модели в обычно не важен, но, тем не менее, его можно настраивать с помощью свойства *ISDEDocument.FlatIndex*.Это свойство задает поле параметра, по которому производится сортировка, определяющая порядок обхода.

В ходе эсплуатации компонента выявились несколько операций, связанных с обходом элементов схемы. Для упрощения эти операции вынесены в объектную модель и

реализуются методом

ISDEDocument.Flat.GetList(QueryType: TxStatQuery; const ElemType: WideString; const Param: WideString): WideString;

TxStatQuery может принимать значение:

sqViewTypes – список типов на схеме

sqViewParams – список параметров элемента заданного типа.

sqParamValues - – список возможных значений параметра

Примеры:

LTypes : = HT.Document.Flat.GetList (sqViewTypes, '', '');

Lparams : = HT.Document.Flat.GetList (sqViewParams, 'выключатель', '');

Lparams : = HT.Document.Flat.GetList (sqViewParams, '', ''); - для всех типов

Lvalues : = HT.Document.Flat.GetList (sqViewParams, 'выключатель', 'положение');

Поиск объектов. Переход к объектам.

Для поиска объекта на схеме используются методы:

ISDEDocument. FindViewIndex (Index: String; Value: String): ISDEObject2;

В качестве индекса используется название именованного параметра, в качестве значения – искомая величина.

Пример

Obj := HT.Document.FindViewIndex('счетчик', '1017');

См. Примеры **Step6.**

В качестве параметра также можно использовать специальные зарезервированные идентификаторы:

PATH : поиск по вн_номер,

SN : SNIdent,

Stag:счетчик,

Tag : вн_номер (не составной).

Возможен поиск по составным индексам, количество полей в составном индексе не ограничено. При поиске по составному индексу и названия параметров, и значения разделяются знаком ‘+’.

Пример:

Obj := HT.Document.FindViewIndex(‘имя_типа+Tag', ‘разъединитель+78’);

Функция *FindView* предполагает, что в результате поиска будет найден один элемент, удовлетворяющий критерию (либо ни одного). Если условию поиска удовлетворяют несколько объектов, то результат работы функции вернет первый из них, удовлетворяющий критерию. Для поиска всех элементов, удовлетворяющих критерию, используйте функцию

ISDEDocument. FindViewsIndex (Index: String; Value: String): ISDEObjects2;

Она возвращает коллекцию объектов. Конечно, методы AddView, CreateView в такой коллекции не работают.

О быстродействии. Следует иметь в виду, что при поиске элемента в ядре системы строится внутренний индекс. Это означает, что при первый вызов поиска в схеме происходит повыбранному индексу относительно медленно (на очень большой схеме это могут быть сотни миллисекунд и даже секунды на медленной машине). Все последующие поиски с тем же индексом производятся быстро – за микросекунды либо доли микросекунд. В любом случае, поиск встроенными методами производится быстрее, чем перебор элементов и сравнение параметров в пользовательской программе.

Если поиск использовался для того, чтобы показать пользователю найденный элемент, можно воспользоваться методами:

ISDEDocument. DocGoToView (Element: ISDEObject2);

Этот метод активизирует страницу, на которой находится элемент, прокручивает ее к месту расположения элемента и выделяет его.

Если элемент имеет небольшой размер, или малозаметен относительно размера экрана (что особенно характерно для отображения на экранах колелктивного пользования, либо при отображении схемы в мелком масштабе), этого бывает недостаточно, чтобы пользователь нашел элемент. В таком случае лучше применять метод

ISDEDocument. DocHighLight (Element: ISDEObject2; color:Long; Restrict: boolean);

Этот метод не только скроллирует экран к элементу, но и подсвечивает его хорошо заметной сходящейся рамкой, что подволяет хорошо зафиксировать внимание на элементе. Здесь параметр color задает цвет рамки мигания, если установлен параметр Restrict, то операция производится без показа сходящейся рамки.

Важно!

В этом разделе так же отметим, каким образом можно производить проверку, являются ли объекты, представленные разными интерфейсами, одним и те и же. Интерфейсы к объекту, полученные из разных источников при сравнении сравнивать напрямую некорректно.

SelectedView

SDEObject1 := HT.Document.SelectedView;
SDEObject2 := HT.Document.FindViewIndex('счетчик', '1017');
EsEqual := (SDEObject1= SDEObject2);

Такая проверка даст отрицательный результат, даже если интерфейсы указывают на один и тот же объект. Корректно производить такое сравнение :

EsEqual := (SDEObject1.RTID= SDEObject2.RTID);

В этом случае в результате будет возвращен положительный результат, если интерфейсы указывают на один и тот же объект.

Дополнительно, для проверки, содержится ли элемент в коллекции, применяется метод

ISDEObjects.IsPresent (SDEObject).

1.14 Организация пользовательского интерфейса со схемой. События пользовательского интерфейса.

При разработке пользовательского интерфейса технологического приложения, работающего со схемной графикой, важнейшим моментом является организация реакции системы на такие события, как выбор элемента на схеме (обычно производится щелчком левой кнопкой мыши на элементе), выбор пункта контекстного меню, которые в свою очередь, появляется вслед за щелчком пользователем правой кнопки мыши на схеме, и формируется в зависимости от того, на элементе какого типа было произведено действие. Для того, чтобы приложение узнало , в какой момент и с каким элементом пользователь

захочет произвести действие, программный интерфейс организован с помощью событий.

Важнейшими событиями, связанными с работой пользователя со схемой, являются:

Наименование	параметры	Описание
OnDocClick	Doc: ISDEDocument; DocEventInfo: IUIEventInfo	Событие возникает при клике (нажатии и отпускании) левой кнопки мыши;
OnDocRightClick()	Doc: ISDEDocument; DocEventInfo: IUIEventInfo	Событие возникает при клике (нажатии и отпускании) правой кнопки мыши;
OnDocDblClick()	Doc: ISDEDocument; DocEventInfo: IUIEventInfo	Событие возникает при двойном клике левой кнопки мыши. Нужно иметь в виду что даже в этом случае сначала будет вызвано событие OnClick;
OnDockeyPress	Doc: ISDEDocument; DocEventInfo: IUIEventInfo	Событие возникает при нажатии клавиши на клавиатуре, если событие не было перехвачено приложением.
OnPopup	Doc: ISDEDocument; (Var Popup: WordBool)	Событие возникает перед показом контекстного меню переходов (гиперссылок) на элементе. Установка параметра Popup в False запрещает показ этого меню. Для анализа содержания меню (списка ссылок) можно использовать содержимое именованного индексированного параметра “гиперссылка”.
OnDocNavigate	(var FileName: String; PageName: String; View Ident: String; Var Perform: boolean	Событие возникает перед переходом по ссылке на другой объект/файл. Параметры FileName – имя файла (схемы) – допускает изменение PageName – имя страницы в схеме View Ident – имя объекта в схеме Perform – флажок для указания выполнения (True) или отмены (False) перехода.
OnDocObjectEnter (SdeObject: ISDEObject)	Doc: ISDEDocument; const SDEObject: ISDEObject2	Событие возникает если курсор мыши попал на изображение объекта схемы. Параметром является интерфейс ISDEObject2 к объекту схемы, над которым находится курсор.
OnDocObjectLeave	Doc: ISDEDocument;	Событие возникает при выходе курсора мыши за пределы

(SdeObject: ISDEObject)	const SDEObject: ISDEObject2	элемента схемы. Параметром является COM интерфейс ISDEObject2 к объекту схемы, над которым находился курсор.
OnDocObjectShowHint	(SdeObject: ISDEObject2; var HintStr: String; var CanShow : WordBool)	Событие возникает перед показом всплывающей подсказки (Hint'a), если пользователь задержал курсор мыши над объектом более чем на определенный интервал времени. Для данного объекта событие происходит только один раз после попадания курсора на объект (вызова события OnObjectenter). Параметры Первым параметром является COM интерфейс ISDEObject2 для объекта схемы, над которым находится курсор. HintStr – текст который будет показан в окошке Hint; CanShow – разрешает(True) или запрещает(False) показ окошка Hint. Подробнее смотри Формирование всплывающей подсказки
OnDocSelectViews	Doc: ISDEDocument	Возникает при отметке / разотметки элементов.
OnObjectChangeParam		Событие возникает при смене значения какого-либо параметра объекта. Параметры COM интерфейс ISDEObject2 для объекта схемы, в котором произошли изменения. ParamName – имя изменяемого параметра ParamIndex – индекс параметра ParamValue – новое значение параметра Processed – переменная, показывающая, произошло ли изменение параметра

Как вы заметили, все эти события

1. Имеют префикс OnDocXXX
2. В качестве параметра передается интерфейс к документу.
3. В качетсве параметра в большей части событий передается параметр DocEventInfo: IUIEventInfo

Этот интерфейс предоставляет детали, связанные с состоянием клавиатуры и мыши в момент возникновения события.

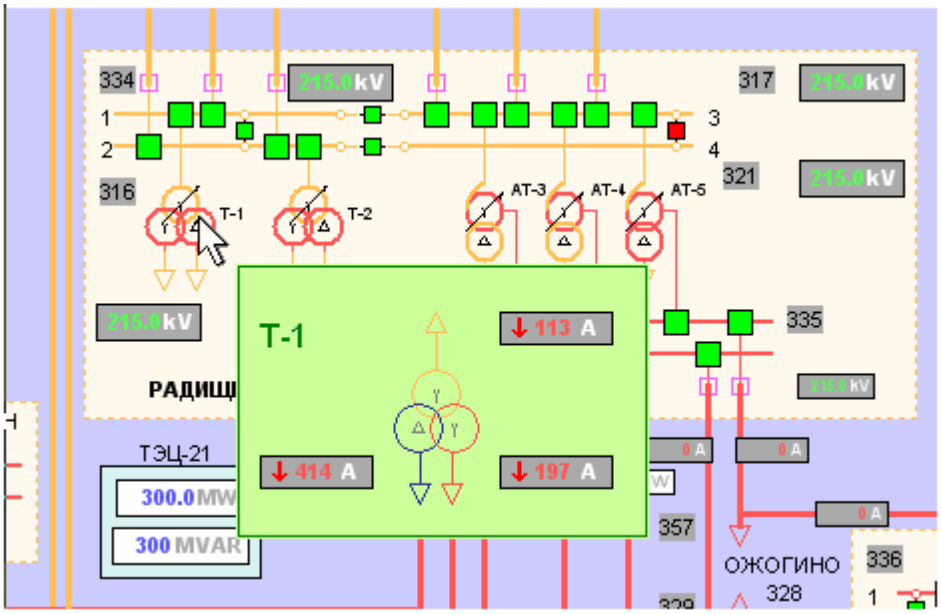
Интерфейс IUIEventInfo

параметр	тип	Описание
AltKey	boolean	Нажата ли клавиша Alt.
Button	long	Код, соответвующий нажатым кнопкам мыши.
CtrlKey	boolean	Нажата ли клавиша Ctrl.
KeyChar	char	Символ, соотвествующий нажатой клавише.
KeyCode	long	Код нажатой клавиши.
KeyState	long	Состояние клавиатуры.
Shifkey	boolean	Нажата ли клавиша Shift.
Touched	ISDEObject2	Элемент на котором произведено действие.
X, Y	integer	Координаты мыши над схемой в момент возникновнения события.
Zone	string	Наименование зоны элемента, на которой сделано действие. Например накладка имеет несколько "горячих" зон, при клике на которые происходит разные действия.

1.15 Формирование всплывающей подсказки

Подробнее рассмотрим событие *OnDocObjectShowHint (SdeObject: ISDEObject2; var HintStr: String; var CanShow: boolean)*. Его назначение – сформировать всплывающую подсказку в зависмости от элемента, над которым находится курсор мышь. Эта функция существовала давно и давала возможность сформировать произвольный текст подсказки. Начиная с версии 4 программного комплекса появилась возможность использовать для

формирования этой подстказки различные стили начертания текста, растровую графику и таблицы.



сформированная с использованием растровой графики и таблиц всплывающая подсказка.

Для того, чтобы сформировать подсказку в виде текста, следует в обработчике назначить нужный текст переменной HintStr. По умолчанию, в этом тексте уже может находиться текст, сформированный по умолчанию компонентом (например, имя SNIdent элемента). В том случае, если вы хотите добавить подсказку к тексте по умолчанию, следует следует добавить сооветсвующий фрагмент текста, либо заместить его.

Возможно использовать и более богатые возможности отображения подсказки (форматированный текст, таблицы, рисунки). Для этого применяется язык QHTML, являющийся ограниченным подмножеством языка HTML. В таком случает в качестве строки нужно передавать строку QHTML. Более подробная информация об испольховании QHTML находится в файле QHTMLSpec.html из документации разработчика.

1.16 События окружения

Здесь кратко рассмотрим события, не связанные непосредственно с действиями пользователя на схеме.

Наименование	параметры	**Описание**

OnActivate	Doc: ISDEDocument	Событие вознкает при активизации документа.
OnBeforeDestroy	Doc: ISDEDocument	Событие вознкает перед уничтожением документа.
OnBeforeDocObject ChangeParam	Doc: ISDEDocument; SDEObject: ISDEObject2; ParamName: WideString; ParamIndex: Integer; ParamValue: WideString; var Allow: WordBool	Событие вознкает перед изменением параметра элемента. Аналогично OnDocObjectChangeParam, однако событие возникает до изменения параметра, и имеется возможность его отменить с помощью флажка Allow.
OnCreate		Событие вознкает при создании документа.
OnDestroy		Событие вознкает при уничтожении документа.
OnDocCreate	const Doc: ISDEDocument	Фактически то же, что OnCreate.
OnDocDeActivate	Doc: ISDEDocument	Событие вознкает при деактивизации документа.
OnDocLevelChange	Doc: ISDEDocument; Level: Integer	Событие вознкает при изменении видимости каких-либо уровней детализации, имейте в виду что переменная Level не отражает полной картины изменения, так в режиме «Выбранные» может измениться видимость более чем одного УД.

OnDocObject ChangeParam	Doc: ISDEDocument; SDEObject: ISDEObject2; ParamName: WideString; ParamIndex: Integer; ParamValue: WideString;	Событие вознкает при изменении параметра элемента. В целях оптимизации быстродействия, событие будет наступать только в случае, если произведена подписка на соответвующий параметр и тип элемента. Подписка производится с помощью интерфейса ICPNotifyFilter документа.
DocObjectsChange Params	Doc: ISDEDocument; const Info: WideString	Событие вознкает аналогично по назначению OnDocObject ChangeParam, однако является более экономичным с точки здения быстродействия, так как оповещения производится пакетами. Промежуток времени между пакетами определяется параметром NotifyTime интерфейса ICPNotifyFilter.
OnDocPageChange	Doc: ISDEDocument; Page: ISDEPage	Событие вознкает перед сменой текущей страницы схемы на экране, и при загрузке новой схемы.
OnDocScaleChange	Doc: ISDEDocument; Scale: Double	Событие вознкает при изменении масштаба страницы схемы.
OnDocScrolling	Doc: ISDEDocument; PosX, PosY: Integer	Событие вознкает при скроллировании документа.
OnEnter		Стандартное событие.
OnExit		Стандартное событие.

OnPaint	Doc: ISDEDocument	Перерисовка схемы или участка.

При изучении библиотек типов Модус Вы обнаружите два интерфейса событий: *ISDEAppEvents* в SDEApp.TLB и *IHTSDEForm2Events* в HTSDE2.tlb. Первый из них используется в приложениях, являющихся сервервами автоматизации (Интегратор, Граф. Редактор и др). Второй используется только для Activesxeme2. Такое дублирование обусловлено тем, что COM не позволяет использовать несколько интерфейсов событий, поэтому приходится создавать дублирующиеся интерфейсы. Тем не менее, при разработке мы стремились, чтобы одинаковые по смыслу методы в разных интерфейсах были идентичны по синтаксису.

1.17 Динамическое создание и удаление объектов.

Такая возможность давно была востребована пользователями, в ActivesXeme2 она, наконец, была реализована. Для вставки объекта на схему используется метод *ISDEPage.SDEObjects.CreateView (TypeName: string; params: string): ISDEObject2;*

В переменную Typename передается тип элемента, например ‘выключатель’. Поскольку набор параметров, необходимых для точной настройки формируемого элемента, велик, он задается в формате, допускающем задание произвольного списка параметров. Как минимум, обычно требуется задать координаты объекта, а также уточняющие базовые параметры, например число обмоток трансорматора.

Список параметров может выглядеть так:

X1=120
Y1=67
класс_напряжения[1]=110кВ
дисп_имя = В-1

Параметры элемента могут быть как заданы при создании, так и скорректированы непосредственно после создания, либо позже с помощью интерфейса именованных параметров – метод *CreateView* возвращает интерфейс *ISDEObject2* к вновь созданному элементу. Те параметры элемента, которые используются как автоинкрементные, можно либо задать при вставке в параметрах, либо, если они не указанны, то они назначаются автоматически.

При добавлении элементов топологическая обработка, как при работе в граф. редакторе (соединениие элементов, разрыв ошиновки при размещении на ней элемента), не производится – для того чтобы получить корректную топологическую схему, рекомендуется сохранить сформированную таким обрахом схему, открыть ее в граф редакторе, выполнить команду *Сервис/ Соединить элементы*.

В этом руководстве не приводится полный список именованных параметров элементов и параметров, необходимых при создании. Лучший способ для изучения этого – посмотреть на готовой схеме элементы в граф. редакторе через инспектор объектов, а также в графическом редакторе сохранить схему в формате XSDE (XML), и изучить список сохраняемых параметров.

Планы: перестроение топологии после формирования схемы.

Для удаления элемента из схемы используйте метод *ISDEPage.SDEObjects.RemoveView (Value: ISDEObject2);*

1.18 Добавление пользовательских именованных свойств

В графической системе имеется возможность добавления пользовательских именованных свойств. Свойство состоит из описателя, аналогичного по семантике именованному параметру, и указания области действия (свойство, действующее на все элементы схемы, на элементы заданных типов, на указанные экземпляры).

Коллекция пользовательских именованных свойств относится к документу, и полностью сохраняется в документе. Для программного доступа используется колекция *ISDEDocument.UserProperies : IUserProperies.*

Интерфейс IUserProperies

Наименование свойств	параметры	Описание
Add	Name: string; var Parametr: IparamInfo	Создание пользовательского именованног о свойствaю при вызове создаетсяIParamInfo, через который можно редактировать шаблон свойства.

Count	integer	Возвращает число именованных свойств в коллекции.
Delete	Param: Variant	Удаляет из коллекции именованное свойство названое по номеру или имени.
Item	Index: Variant; Value: IparamInfo	Возвращает описатель именованного свойства по индексу.
SaveToFile	string	Экспорт в файл набора именованных свойств описанных в документе. Расширение фала – udp.
LoadFromFile	string	Импорт набора именованных свойств из фала – udp.

Интерфейс IParamInfo

Частично пересекается с интерфесом IParam. Дает более подробное описание параметра. Структура необходима при создании пользовательских именованных параметров.

	Наименование свойств	**Параметры**	**Описание**
Prw	**aType**	TxAttributeType	Тип атрибута (строковый, перечислимый, числовой).
	AutoIncVal	integer	Число, равное максимальному значению параметра у элемента, имеющегося в схеме, для параметров у которых Copyflags = TxAutoInc.
Prw	**Category**	TxPararCategory	Категория именованного свойства.
Prw	**Copyflags**	TxCopyFlags	Определяет дополнительную настройку свойства. TxNonCopy – параметр не копируется при копировании элемента в граф. редакторе TxAutoInc – австоинкрементный параметр (только числовой тип) – при добавлении элемента автоматически принимает значение, которого не

			было в документе (на 1 больше максимального имеющегося) cfUnique – должен быть уникальным (производится проверка на уникальность при изменении значения) .
Prw	**DefaultValue**	string	Значение параметра по умолчанию.
Prw	**ElTypes**	string	Список типов элементов, к которым применим данный параметр.
Prw	**Help**	string	Строка справки.
Prw	**Hint**	string	Комментарий к параметру.
Prw	**IsDefault**	integer	Является ли параметр параметром по умолчанию у элемента.
Pr	**MaxIndex** (read)	integer	Размерность параметра (в том случае если параметр индексированный) = Dim у IParam.
Prw	**Max**	integer	Минимальное значение (для ограниченных целых).
Prw	**Min**	integer	Минимальное значение (для ограниченных целых).
Pr	**Name** (read)	string	Имя параметра.
Pr	**ParMode**	integer	Режим использования параметра и время его жизни = mode у IParam.
Prw	**PropertySource**	TxPropertySource	Место определения параметра (зашито в ядро граф. системы, настроечный пользовательский файл, приложение, зашито в схему).
Pr	**Values**	string	Возможные значения именованного свойства; применимо для перечислимого и строкового типа.
Pr	**ValuesNo**	integer	= MaxIndex.
Pr	**ValuesCount**	integer	Число возможных значений для перечислимого типа = ValuesNo у IParam.

Pr	**ValuesVerb**	string	Набор глаголов, соответствующих переходу в данное состояние. Например положение=включен – включить.

Пример:

PI:= HT.Document.UserProperties.Add ('инвентарный_номер");

PI.aType := atInteger;

PI:= HT.Document.UserProperties.Add ('дата_прокладки_кабеля');

PI.aType := atEnumerated;

PI.Values := '@XIXв@XXв@XXIв@';

PI.ElTypes := '@выключатель@';

Планы: возможность сохранения описателей параметров не только в документ, но и в системное хранилище.

1.19 Работа с уровнями детализации

Для доступа к списку уровней детализации документа (описателей) служат интерфейсы:

Интерфейс IDetailslevels

Наименование свойств	Параметры	Описание
Add	Name: string; var Level: DetailsLevel	Создание нового уровня детализации; при вызове создается объект типа Level, через который можно редактировать свойства УД.
Count	integer	Возвращает число УД в схеме.
Delete	Param: Variant	Удаляет из коллекции УД, указанный по номеру или имени.
Item	Index: Variant; Value: DetailsLevel	Возвращает УД по индексу, указанному по номеру или имени.

SaveToFile	параметр: string	Экспорт в файл Списка УД в документе. Расширение фала – .detal.

Интерфейс IDetailslevel

Наименование свойств	Параметры	Описание
Comment	string	Комментарий.
Docked	boolean	Признак блокировки УД для редактирвания.
Name	string	Имя УД.
PasswordShow	string	Пароль на показ УД.
PasswordEdit	string	Пароль на редактирование УД.
Value	integer	Числовой идентификатор УД (0..255). При показе УД они сортируются в порядке нумерации. Нумерация по Value не обязательно сплошная.
Visible	boolean	Видим ли данный УД.
PassOpened	boolean	Признак того, что пароль для УД был введен.

Для доступа к УД конкретного элемента используйте именованное свойство элемента «уровень_детализации».

1.20 Стили выделения. Группы выделения

Как показал опыт, в технологических приложениях очень удобно выделять элементы с использованием подсветки разными цветами рамки или фона.

Удобный механизм для этого давно реализован в графической системе и называется «стили выделения». Стиль выделения – это именованный признак, позволяющий при проставлении в элемент отображать заданным цветом рамку вокруг элемента (с

использованием цвета заполнения или без). Помимо цвета, возможно также менять стиль границы (сплошная, пунктир), а также назначать мигающие стили и стили с отображением геометрических примитивов (фигурок), расположенных в пределах границ элемента. Стили настраиваются с помощью приложения **Редактор стилей выделения**, в этом разделе нам важно, что у каждого из этих стилей есть уникальное в пределах библиотеки стилей имя.

Описания стилей хранятся в специальной системной библиотеке (Default.XSTL).

Для присвоения стиля элементу нужно использовать именованные свойства элемента **стиль_выделения** или **композитный_стиль**. **Стиль_выделения** – это устаревшее свойство, оно позволяет установить в элементе один выбранный стиль из библиотеки, либо ни одного (значение нет). Список имен доступных стилей можно получить через свойство *Values* именованного параметра **стиль_выделения**.

Композитный_стиль позволяет присвоить элементу произвольную комбинацию стилей. Параметр **композитный_стиль** имеет тип «множество». Так, если один из стилей задает штриховку фона, а другой цвет границы элемента, то на элементе покажутся оба стиля. Если стили действуют на один и тот же параметр отрисовки (например граница линии), то будет отрисован стиль, определенный в библиотеке позже.

При установке и снятии стиля удобно устанавливать значение +имя_стиля и -имя_стиля, как это было указано в разделе об именованных параметрах.

Планы.

- Усовершенствование механизма отрисовки стилей для отображения накладывающихся стилей.
- Стили с повторяющимися, накладываемыми на линии фигурами (заполнение).
- возможность разнесения библиотеки стилей на несколько файлов - библиотек.

При практическом использовании стилей часто возникает вопрос о хранении списков элементов, помеченных этим стилем. Например, для того, чтобы показать разным цветом элементы на схеме, привязанные к базе данных и не привязанные, необходимо составить в памяти список тех и других. Первый способ решения – сохранить этот список в собственном приложении и назначать стили выделения в соответствии с ним. Так предлагалось делать при использовании ActivesXeme. В ActivesXeme2 появился механизм, позволяющий не только

подсвечивать элементы, но и хранить списки выделенных.

Эту возможность можно использовать при помощи интерфейса **IHighLights.**

Наименование свойств	**Параметры**	**Описание**
Add	Name: string; StyleName: string; Value: ISDEObjects2	Создает новую группу выделений. В результате возвращает интерфейс к колекции принадежащих к нему элементов.
Clear		Очищает весь список групп выделений.
Comment	string	Комментарий.
Delete	Index: variant	Удаление группы выделений из списка. Индекс – номер в списке или название группы.
Items	Index: variant; Value: ISDEObjects2	Индекс – номер в списке или название группы. Результат – коллекция содержащихся элементов.
StyleName	Index: variant; var Value: string	По номеру или имени группы возвращает имя стиля.
Visible	Index: variant; Value boolean	Использовать ли подсветку для группы выделения.

Пример

Заводим в схеме два списка выделения, в один помещаем включенные, в другой отключенные элементы

```
procedure TForm1.Button2Click(Sender: TObject);
var
i: integer;
HOn, HOff: ISDEObjects2;
LParam: IParam;
begin
HOn := HT.Document.Selections.Add (‘включен', 'Linked');
HOff:= HT.Document.Selections.Add ('отключен', 'NotNormal');
with HT.Document.Flat do
for i := 0 to Count-1 do begin
  Lparam := Item[i].Params.Item ['положение'];
  if Assigned (Lparam) then begin
    if Lparam.Value = ‘включен' then
      HOn.AddView( Item[i] )
    else if Lparam.Value = 'отключен' then
      HOff.AddView( Item[i] );
  end;
end;
end;
```

включаем / отключаем видимость стилей выделений

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
HT.Document.Selections.Visible [‘включен'] := SpeedButton1.Down;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
HT.Document.Selections.Visible [‘отключен'] := SpeedButton2.Down;
end;
```

1.21 Правила отображения

В конкретных проектах часто приходится настраивать отображение элементов в

зависимости от каких – либо условий. Например, в зависимости от балансовой принадлежности ТП раскрашивать ее в соответствующий цвет фона. Для решения подобных задач без программирования может служить механизм правил отображения. Этот механизм – простой и быстрый, хотя, возможно, в некоторых случаях недостаточно гибкий. Подробнее о настройке правил отображения смотрите документацию к Редактору правил отображения.

Программную логику работы правил отображения можно сформировать и в собственной программе, однако поддержка механизма правил отображения в Вашем приложении позволит пользователям настраивать приложение более гибко.

В текущей версии не предусматривается динамическое формирование списка правил отображения, их можно только загрузить из файла (с расширением shr или XRS). В документ можно последовательно загрузить правила из нескольких файлов.

Интерфейс IRules

Наименование свойств	**Параметры**	**Описание**
Clear		Очистить коллекцию правил отображения.
Count	integer	Возвращает число загруженных правил отображения.
Item	Index: Variant; Value: IRule	Возвращает правило по индексу или имени.
LoadFromFile	string	Импорт списка правил из файла.

Интерфейс IRule

Наименование свойств	Параметры	Описание
Name	string	Имя правила.

Планы

- Развитие гибкости правил отображения.

- Возможность формирования описания правил через программный интерфейс.

1.22 Топология

Все большую популярность приобретает использование топологической модели схемы, впервые предоставленной из компонента ActivesXeme. Топологическая модель строится в графическом редакторе при рисовании схемы и доступна в компоненте ActivesXeme через специальный интерфейс. Впервые эта топологическая информация была использована в тренажере по оперативным переключениям, теперь же она доступна и используется для таких задач, как построение расчетной схемы энергосистемы.

Для доступа используется интерфейс *ISDEObject2.Nodes: INodes2*

Интерфейс INodes2

Наименование свойств	Параметры	Описание
Count	integer	Возвращает число узлов элемента, совпадающее с количеством внешних коннекторов.
Item	Index: Variant; Value: INode2	Возвращает узел по индексу.

Интерфейс INode2

Наименование свойств	Параметры	Описание
Count	integer	Возвращает число узлов элементов, присоединенных к этому узлу.
Item	Index: Variant; Value: ISDEObject2	Возвращает узел по индексу.
NodeNum	integer	Возвращает номер топологического узла

Интерфейс INode42

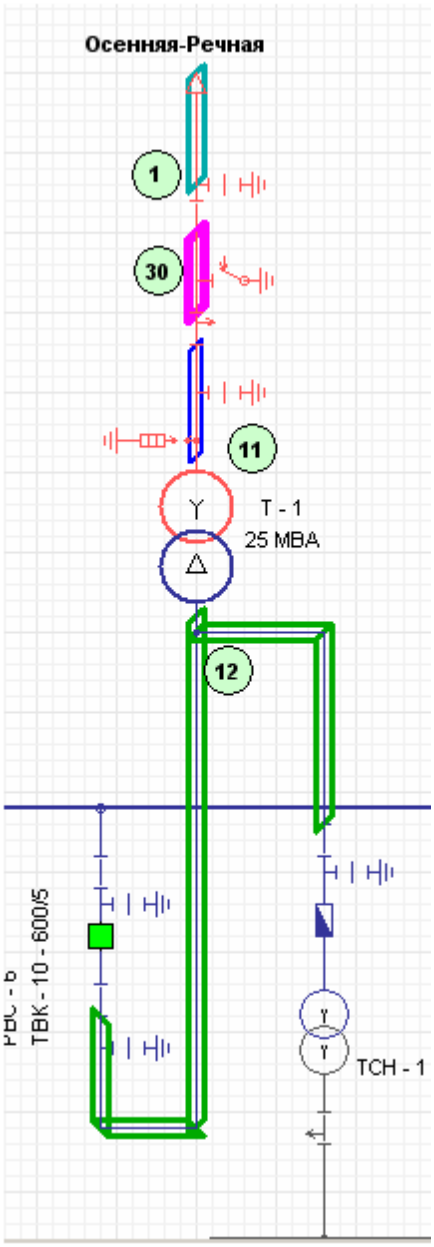
Наименование свойств	Параметры	Описание
NewEnum		
NodeState	TxNodeState	
GetConnector	SDERect	

Как видино из описания программного интерфейса, он не предоставляет возможности редактирования топологической модели, допускается только чтение.

Внимание! В демонстрационном (без регистрации лицензии) режиме доступна информация только по одному топологическому узлу каждого элемента.

Каждый интерфейс *INode* в коллекции *INodes* элемента соответвует коннктору элемента. Интерфейс *INode* сам является коллекцией, в котором перечислены элементы, присоединенные к этому узлу. Топологическая модель не соответвует структуре электрической модели, принятой в программах расчета режима. Для построения такой модели можно использовать существующую модель, упрощая ее, так как существующая

модель несет в себе максимум информации по топологии. В разных задачах требуется разные уровни подробности представления модели, соответствунно алгоритм упрощения будет зависить от задачи.



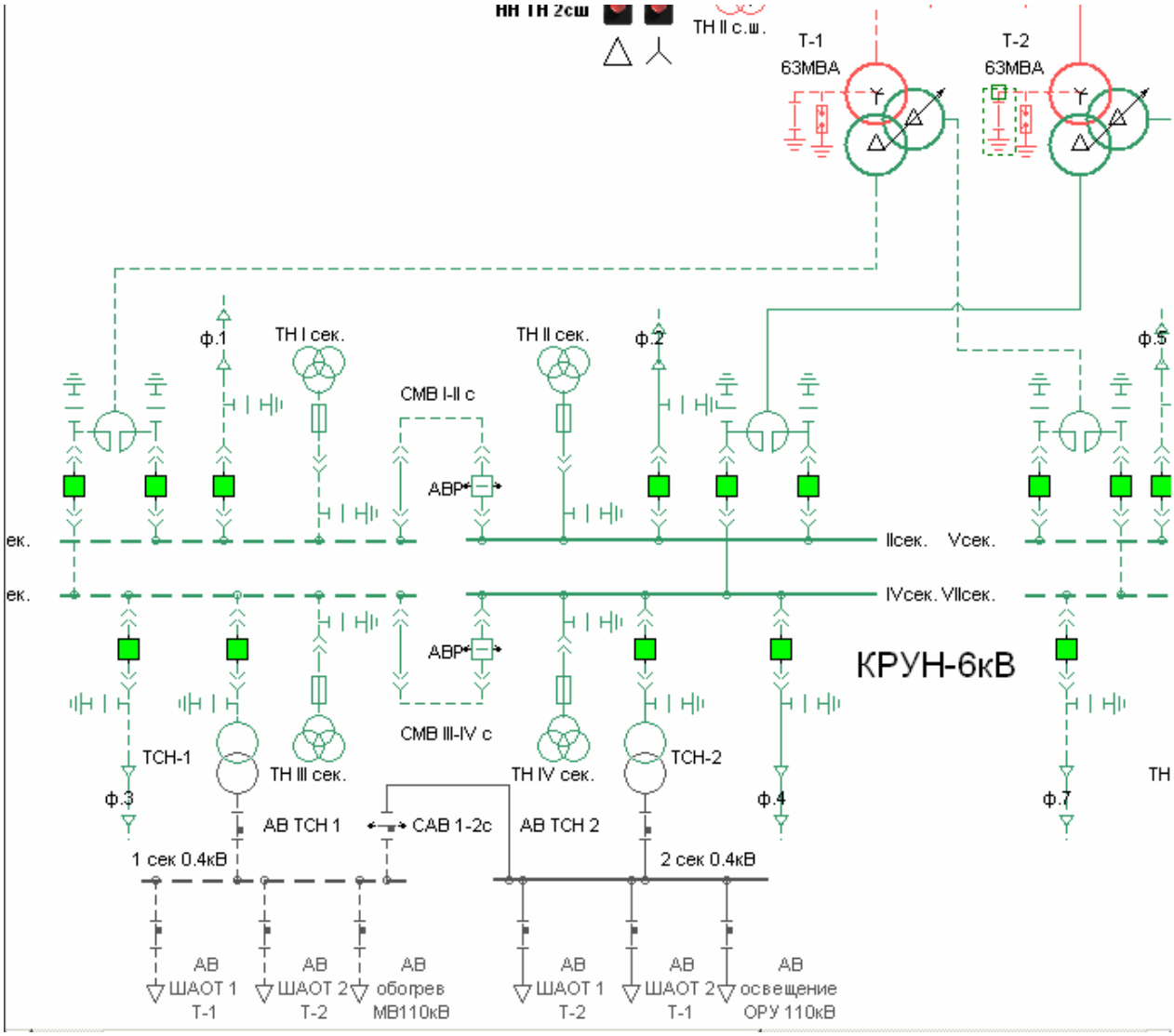
Достаточно нетривальным моментом является прохождения топологической связи через границу контейнера. Это осуществляется с помощью элемента «коннектор». С одной стороны, он является элементом, присоединенным к топологическому узлу внутри контейнера, с другой стороны, он является коннектором контейнера, осуществляющим его связь с внешним миром. Заметим также, что нумерация коннекторов (в интерфейсе она представлена свойством NodeNum) уникальна внутри каждого контейнера и страницы.

На рисунке разным цветом показаны участки схемы, принадлежащие разным узлам,

помеченных числами в кружках.

1.23 Использование электрической модели

В компоненте Activesxeme реализована задача показа обесточенных участков схемы. Входной информацией является топология схемы и состояние ее коммутационных аппаратов. Такая возможность ранее имелась в других приложениях комплекса, например в аниматоре схем.



Обесточенные участки показываются стилем в соответствии с настройками, заданными в форме параметров отображения. Аналогично, специальным стилем отрисовываются зазамленные участки.

Для активизации модели (по умолчанию она отключена) нужно использовать интерфейс *IElectricModel2*, описанный в библиотеке типов sde_electric.tlb. Для получения

интерфейса нужно использовать свойство *IHTSDEForm2.ElectricModel* , приведя его к нужному типу: *IHTSDEForm2.ElectricModel as IElectricModel2.*

Для использования его нужно активизировать, установив свойство *Active*.

Интерфейс IElectricModel2

Наименование свойств	**параметры**	**Описание**
Active	boolean	Активизация модели
BeginUpdate		Временно приостанавливает пересчет модели, например перед большим обновлением состояния
DelayTime	integer	Время в мс задающее минимальное время между пересчетами модели.
EndUpdate		Возобновляет пересчет после BeginUpdate
Locked	boolean	

Пример: Step10.

1.24 Сохранение в потоки

Для сохранения / чтения схемы в хранилища, отличающиеся от файла (memo – поля СУБД, при скачивании схемы через http) используйте интерфейс *IpersistScheme*.

Интерфейс IpersistScheme

Наименование метода	**Параметры**	**Описание**

Load	fStorage:IStorage	Загрузка данных схемы из хранилища
Save	fStorage:Istorage, CompleteSave:wordbool	Запись данных схемы в хранилище
SaveCompleted	fStorage:IStorage	Подтверждение успешной записи в хранилище

Приведенные примеры осуществляют чтение и запись в файл, однако используются потоки

Пример: **Step11**

чтение из потока

```
procedure TForm1.Button1Click(Sender: TObject);
var
FStorage: IStorage;
PerStg: IPersistScheme;
begin
if Opendialog1.Execute then
//откроем файл и получим ссылку на IStorage
  if StgIsStorageFile( StringToOleStr(Opendialog1.FileName) )= S_OK then begin
    OleCheck(StgOpenStorage(StringToOleStr(Opendialog1.FileName),
          nil,STGM_READ or STGM_SHARE_EXCLUSIVE,nil,0,FStorage));
//Получим адрес интерфейса IPersistScheme для компонента HTSDE
    PerStg := HT.ControlInterface as IPersistScheme;
//Загрузка файла в HTSDE
    OleCheck(PerStg.Load(fStorage));
  end else
    ShowMessage (Opendialog1.FileName + M +
          'Файл не является структурированным хранилищем!');
Caption := Opendialog1.FileName;
end;
```

запись в поток

```
procedure TForm1.Button2Click(Sender: TObject);
var
FStorage: IStorage;
PerStg: IPersistScheme;
begin
//если выбран файл сформируем его и получим ссылку на IStorage
  if SaveDialog1.Execute then begin
    //Получим адрес интерфейса IPersistScheme для компонента HTSDE
    PerStg := HT.ControlInterface as IPersistScheme;
    OleCheck( StgCreateDocFile(
          StringToOleStr(SaveDialog1.FileName), STGM_CREATE or
          STGM_READWRITE or STGM_SHARE_EXCLUSIVE,0,FStorage));
    try
    try
    //запись данных схемы в хранилище
     OleCheck(PerStg.Save(fStorage,false));
     fStorage.Commit(STGC_DEFAULT);
    finally
    //Подтверждение успешной записи в хранилище
      OleCheck(PerStg.SaveCompleted(fStorage));
    end;
    except
      Messagebeep(0);
    end;
end;
end;
```

1.25 Установка ПО на рабочих местах пользователя

В случае поставки пользователям вашего ПО с использованием компонента ActivesXeme возникает вопрос, каким образом осущетвлять поставку им ActivesXeme. Поскольку компонент использует ряд внешних библиотек типов, DLL, настроечных файлов, в том числе требующих регистрации, написание собственного дистрибутива, который может установит все необходимые компоненты, требует определенной квалификации и трудозатрат.

Поэтому мы рекомендуем устанавливать на компьютере пользователя Программный комплекс Модус версии 5.0. При установке достаточно выбрать пункт "Примеры использования схем в программах сторонних разработчиков и создания плагинов". Так же может быть полезно установить Графический редактор.

Для использования интергратора и плагинов необходимо установить выбрать пункт Интегратор. Затем нажать кнопку с "**...**" выбрать дополнительные расширения(плагины).

1.26 Как перейти от ActivesXeme

В ходе модернизации системы мы выработали методику перехода от приложений, использующих ActivesXeme (компонент версии 3) к ActivesXeme2. Она очень простая. Перед переходом необходимо выписать список обработчиков событий, назначенных компоненту ActivesXeme. А также свойств, назначенных компоненту, отличающихся от установленных по умолчанию. Далее компонент удаляется с формы. На его место с палитры устанавливается ActivesXeme2. Далее производится компиляция кода, которая выявляет несоответвие старых и новых конструкций. При переходе, как показывает опыт, объем связующего кода уменьшается в 2-3 раза, востанавливает назначение свойств и событий. .

Рекомендуется сразу же рассматривать возможность применения новых возможностей, так как это зачастую упрощает связующий код. Например, использование внутренних структур ActivesXeme2 для списков выделенных элементов и групп выделения избавляет от необходимости заводить и поддерживать собственные структуры (списки).

Возможность хранения ссылок на элементы схемы в виде целых чисел (RTID) при достаточно быстром поиске по ним избавляет от необходимости организации собственных списков интерфейсов и т.д.

Мы надеемся, что проделанная нами работа по модернизации объектной модели компонента облегчит поддержку имеющихся у Вас программных комплексов и откроет перед Вами новые возможности.

1.27 Интерфейс компонента ActivesXeme 2

В предыдущих разделах были рассмотрены в основном особенности внутренней организации объектной модели схемы, здесь познакомимся с работой с компонентом как с обычным ActiveX – контролом.

Компонент **ActivesXeme 2** обеспечивает следующие возможности:

- Показ полного вида страницы схемы (кнопка "Панорама").
- Переход к единичному масштабу отображения (кнопка "Масштаб 100%").
- Изменение масштаба схемы клавишами "+" и "-" на цифровой клавиатуре.
- Если схема не помещается на экран, то ее можно прокручивать с помощью линеек прокрутки, расположенных справа и внизу схемы.
- Если указатель мыши на объекте принимает вид руки, то при нажатии правой кнопки мыши возникает всплывающее меню, с помощью которого можно перейти по ссылке на указанную страницу схемы или другую схему, либо выполнить какое-либо действие, зависящее от приложения.

Внешний вид компонента представлен на рисунке 1.

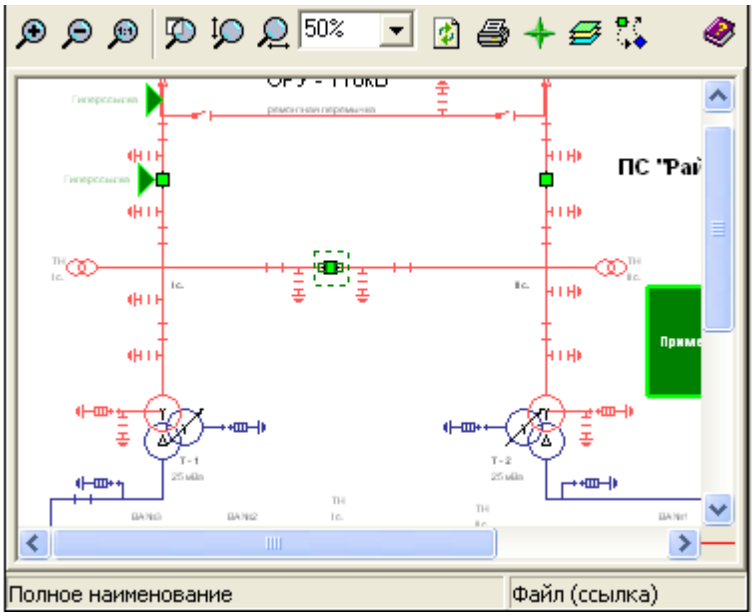


Рисунок 1. Внешний вид компонента.

В **клиентской** части окна отображается сводная схема.

На **панели управления** расположены кнопки управления компонентом.

На **панели статуса** выводится название объекта, на котором находится указатель мыши.

Кнопки управления

Масштаб 100%

Выводит на экран страницу схемы с масштабом 1:1.

Панорама

Изменяет масштаб вывода страницы схемы так, чтобы она поместилась на экран.

Масштабирование

Изменение масштаба страницы схемы при выводе ее на экран.

Перерисовка схемы

Перерисовывает страницу схемы.

Печать схемы

Открывает диалог печати. Возможна распечатка схем на любом принтере или плоттере в выбранном масштабе, с разбиением по листам и пр.

Размер печатного поля выбирается автоматически в зависимости от установок принтера.

Навигатор

Показать/спрятать окно навигатора - окно с уменьшенным представлением схемы.

Уровень детализации

Обычно при просмотре схем разными категориями пользователей (например, диспетчер энергосистемы и специалист по релейной защите и автоматике) необходим разный уровень подробности представления информации. Так, диспетчеру системы не обязательно видеть, в каком положении находятся заземляющие ножи на подстанции, а лицу, производящему переключения – обязательно. Для разрешения этой проблемы были введены уровни детализации. При просмотре схемы достаточно установить видимость на желаемый уровень через специальное диалоговое окно.

Настройки

Вызывает диалоговое окно, в котором можно осуществить настройку параметров отображения схемы, таких как:

- Цвет элементов (в зависимости от классов напряжения);
- Форма (в соответствии с принятым стандартом);
- Параметры отображения всплывающей подсказки.

О программе

Выводит информационное окно о версии компонента и его разработчиках.

1.28 Описание и использование компонента

HTSDEForm

Технология ActivesXeme реализована в виде компонента ActiveX HTSDEForm. Далее в по тексту эти термины можно рассматривать как синонимы, но следует помнить, что ActivesXeme – это название технологии, а HTSDEForm – ее конкретная реализация.

Установка

Для отображения схем с элементами автоматики, приборами и пользовательскими элементами требуется установка библиотек элементов фирмы Модус (см. более подробно в общей документации). Это обеспечивается автоматически при инсталляции комплекса программ Модус.

1.29 Работа с компонентом в среде разработки

Поместив компонент на форму, Вы получите доступ к свойствам, методам и событиям, стандартным для выбранной Вами среды разработки приложения способом.

Обмен данными между графическим модулем и приложением предоставляется самим графическим модулем через его методы (свойства), представляемые интерфейсами. Совокупность интерфейсов назвается объектной моделью.

Объектные модели рассмотрены в пункте **Объектные модели**, сейчас же мы обратимся к другому аспекту взаимодействия с компонентом.

Для более подробного изучения работы с компонентом, в приложениях 2, 3, 4, 5 рассмотрены примеры программ, разработанных в различных средах программирования (Borland Delphi, Visual C++, Borland C++ Builder, Visual Basic).

1.30 Объектные модели

Как уже упоминалось выше, обмен данными между графическим модулем и приложением осуществляется посредством интерфейсов, объединенных в группы

(объектные модели).

Для получения доступа к элементам схемы, в зависимости от задач, можно придерживаться следующего алгоритма:

1. Первоначально, Вы имеете доступ к форме графического модуля (HTSDEForm) через интерфейс IHTSDEForm. С помощью него Вы можете управлять состоянием формы. Для работы с объектом на схеме Вам необходимо получить доступ к объекту SDEDocument, используя свойство Document.
2. Получив доступ к объекту SDEDocument, Вы получаете возможность работать с текущей страницей, получить коллекцию всех страниц схемы и коллекцию всех объектов схемы. Для последовательной работы со схемой необходимо получить доступ к объекту SDEPages, используя свойство GetPages (свойство Pages).
3. Страницы будут представлены в виде коллекции объектов SDEPage.Объект SDEPages позволяет получить доступ к конкретной странице схемы используя метод GetItem (свойство Item).
4. Объект SDEPage позволяет работать со свойствами страницы и получить коллекцию всех объектов, располагающихся на данной странице. Метод GetSDEObjects (свойство SDEObjects) возвращает объект SDEObjects2.
5. Объект SDEObjects2 - коллекция (список) объектов – обеспечивает доступ к определенному объекту из данной коллекции. Метод GetItem (свойство Item) – возвращает объект типа SDEObject2.
6. Объект SDEObject2 позволяет работать со всем свойствами и характеристиками объекта схемы, к которому предоставляет доступ.

Кроме методов, перечисленных выше, достаточно быстрый поиск элемента по SNIdent можно осуществить следующим образом:

MyObject := HTSDEForm.Document.Flat.Item [MySNIdent]

где *MySNIdent* – уникальный идентификатор объекта.

Если необходимо выполнить поиск не по SNIdent, а по другому полю (например ключ_привязки + назначение_привязки), то необходимо выполнить следующие операци:

HTSDEForm.Document.FlatIndex := 'ключ_привязки+назначение_привязки'

(в этот момент в памяти готовится нужный индекс, и после этого можно искать объект)

MyObject := HTS DEForm.Document.Flat.Item ['IDПСТ210+ЛПЧ']

1.31 Описание тестов – учебных примеров компонента ActivesXeme2

В данном разделе приведено описание примеров коипонента ActivesXeme2. После установки программного комплекса Модус примеры можно просмотреть в папке *"C:\Program Files\modus520\ActivesXeme\OCX2\DELPHI\LEARN\"*.

1.31.1 Step 1. Различные способы открытия файла

Примеры разных способов открытия файла (из файла, из потока)

```
if Opendialog1.Execute then
  case RadioGroup1.ItemIndex of
    0: HT.FileName := Opendialog1.FileName;
    1: HT.Open(Opendialog1.FileName);
    2: HT.Document.SourceFileName := Opendialog1.FileName;
    3: HT.Document.Open (Opendialog1.FileName);
  end;
Caption := HT.Document.SourceFileName;
```

1.31.2 Step 2. Получение полного списка элементов имеющихся на схеме.

Рассмотрим получение полного списка элементов имеющихся на схеме на примере заполнения поля Memo1.

Здесь *HT.Document.Pages.Item [i].SDEObjects* - коллекция элементов на странице *Obj.Elements.Item[i]* - коллекция элементов контейнера.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  i,j : integer;
  Objs: ISDEObjects2;
begin
  Memo1.Lines.Clear;
  for i := 0 to HT.Document.Pages.Count-1 do begin
    Objs := HT.Document.Pages.Item [i].SDEObjects;
    for j := 0 to Objs.Count-1 do
```

```
    ProcessObject (Objs.Item [j]);
  end;
end;

procedure TForm1.ProcessObject(Obj: ISDEObject2);
var i: integer;
begin
  Memo1.Lines.Add (Obj.SNIdent);
  if Assigned (Obj.Elements) then
    for i := 0 to Obj.Elements.Count-1 do
      ProcessObject(Obj.Elements.Item[i]);
end;
```

1.31.3 Step 3. Пример работы с именованными свойствами.

Пример переключения всех коммутационных аппаратов. Для каждого элемента схемы, у которого есть свойство "положение", изменить значение на противположное

```
 Objs := HT.Document.CurrentPage.SDEObjects;
  for i := 0 to Objs.Count-1 do begin
    Param:= Objs.Item [i].Params.Item ['положение'];
    if Assigned (Param) then
      if param.Value = 'включен' then
        param.Value := 'отключен'
      else
        param.Value := 'включен';
  end;
```

1.31.4 Step 4. Получение информации об объектах на схеме используя метод Flat.

Пример использования метода flat. Метод Flat позволяет получить информацию обо всех элементах на схеме, о типах элементов, свойствах и значениях свойств.

Пример получения полного списка объектов

```
  Memo1.Lines.Clear;
  for i := 0 to HT.Document.Flat.Count-1 do begin
```

```
  Obj := HT.Document.Flat.Item [i];
  Memo1.Lines.Add (Obj.SNIdent);
end;
```

Пример получения списка типов объектов

```
S := HT.Document.Flat.GetList (sqViewTypes, '', '');
ComboBox1.Items.text := S;
```

Пример получения списка параметров выбранного типа

```
S := HT.Document.Flat.GetList (sqViewParams, ComboBox1.Text, '');
ComboBox2.Items.text := S;
```

Пример получения списка возможных значений параметра выбранного типа

```
S := HT.Document.Flat.GetList (sqParamValues, ComboBox1.Text, ComboBox2.text);
Memo1.Lines.text := S;
```

1.31.5 Step 5. Пример организации переходов к элементу. Поиск элемента. Подсветка.

Данный пример демонстрирует работу с методами *SelectedView* - свойства выделенного элемента, *FindViewIndex* - поиск элемента по индексу(по умолчанию поиск идет по "счетчику"), *DocGoToView* - переход на элемент схемы, *DocHighLight* - переход и подсветка элемента.

Создание "Закладки".(Сохранение "Счетчика" для выделенного элемента).

```
if Assigned (HT.Document.SelectedView) then
   BM := HT.Document.SelectedView.Params.Item ['счетчик'].Value;
```

Переход на "Закладку"

```
Obj := HT.Document.FindViewIndex('счетчик', BM);
if Assigned (Obj) then
  HT.Document.DocGoToView(Obj);
```

Переход на "Закладку" и подсветка элемента.

```
Obj := HT.Document.FindViewIndex('счетчик', BM);
if Assigned (Obj) then
    HT.Document.DocHighLight (Obj, clRed, false);
```

Переход на следующий элемент выбранного типа. ComboBox1 - содержит список типов

элементов схемы.

```
if not Assigned (Els) then begin
  Els := HT.Document.FindViewsIndex('имя_типа', ComboBox1.text);
  ElsPos:= -1;
end;
if not Assigned (Els) then exit;
inc (ElsPos);
if ElsPos >= Els.Count then ElsPos := 0;
Obj := Els.Item [ElsPos];
HT.Document.DocGoToView(Obj);
HT.Document.DocHighLight (Obj, clRed, false);
```

1.31.6 Step 6. Динамическое добавление пользовательских именованных параметров.

Пример работы с пользовательскими свойствами. В примере показано как создать новое свойство для элемента или группы элементов.

Для каждого элемента схемы добавляется свойство _All. А для "выключателя" _Switch.

```
PI:= HT.Document.UserProperties.Add ('_All');
PI:= HT.Document.UserProperties.Add ('_Switch');
  if Assigned (PI) then
     PI.ElTypes := '@выключатель@';
```

Для выделенного элемента выводим список именованных свойств в ListBox1

```
ListBox1.Items.Clear;
if Assigned (Doc.SelectedView) then
with Doc.SelectedView.Params do begin
  for i := 0 to Count-1 do
    ListBox1.Items.Add (Item [i].Name);
end;
```

1.31.7 Step 7. Использование стилей подсветки.

В примере создано две группы стилей подсветки "включен" и "отключен", задан стиль выделения элемента для каждой группы 'Linked', 'NotNormal'. В первую группу добавлены

все включенные элементы, во вторую - отключенные.

```
HOn := HT.Document.Selections.Add ('включен', 'Linked');
HOff:= HT.Document.Selections.Add ('отключен', 'NotNormal');
with HT.Document.Flat do
for i := 0 to Count-1 do begin
  Lparam := Item[i].Params.Item ['положение'];
  if Assigned (Lparam) then begin
    if Lparam.Value = 'включен' then
      HOn.AddView( Item[i] )
    else if Lparam.Value = 'отключен' then
      HOff.AddView( Item[i] );
  end;
end;
```

Если нажата SpeedButton1 подсвечиваются все включенные элементы.

```
HT.Document.Selections.Visible ['включен'] := SpeedButton1.Down;
```

1.31.8 Step 8. Активизация коммутационной модели.

Активизация коммутационной модели.

```
with HT.ElectricModel as ElectricModel2 do
  Mode := emOnUpdate;
```

Переключение элементов по двойному клику.

```
if Assigned (DocEventInfo.Touched) then begin
  Lparam := DocEventInfo.Touched.Params.Item ['положение'];
  if Assigned (Lparam) then begin
    if Lparam.Value = ''включен' then
      Lparam.Value := 'отключен'
    else if Lparam.Value ='отключен' then
      Lparam.Value := ''включен'
  end;
end;
```

1.31.9 Step 9. Запись схемы в файл.

Загрузить схему из файла.

```
if Opendialog1.Execute then
//откроем файл и получим ссылку на IStorage
  if StgIsStorageFile( StringToOleStr(Opendialog1.FileName) )= S_OK then begin
    OleCheck(StgOpenStorage(StringToOleStr(Opendialog1.FileName),
          nil,STGM_READ or STGM_SHARE_EXCLUSIVE,nil,0,FStorage));
//получим адрес интерфейса  IPersistScheme  для компонента HTSDE
    PerStg := HT.ControlInterface as IPersistScheme;
//Загрузка файла в HTSDE
    OleCheck(PerStg.Load(fStorage));
    PerStg := nil; // Delphi release
    fStorage := nil;
  end else
    ShowMessage (Opendialog1.FileName + ^M +
          ' Файл не является структурированным хранилищем!');
Caption := Opendialog1.FileName;
```

Запись схемы в файл .

```
//Если выбран файл сформируем его и получим ссылку на IStorage
  if SaveDialog1.Execute then begin
    //===Формирование правой панели риложения
     //получим адрес интерфейса  IPersistScheme  для компонента HTSDE
    PerStg := HT.ControlInterface as IPersistScheme;
    OleCheck( StgCreateDocFile(
          StringToOleStr(SaveDialog1.FileName), STGM_CREATE or
          STGM_READWRITE or STGM_SHARE_EXCLUSIVE,
          0,
          FStorage) );
    try
    try
    //Запись данных схемы в хранилище
```

```
      OleCheck(PerStg.Save(fStorage,false));
      fStorage.Commit(STGC_DEFAULT);
    finally
    //подтверждение успешной записи в хранилище
      OleCheck(PerStg.SaveCompleted(fStorage));
      PerStg := nil; // Delphi release
      fStorage := nil;
    end;
    except
      Messagebeep(0);
    end;
  end;
```

1.31.10 Step 10. Динамическое добавление гиперссылок.

Добавление гипперсылки

```
  Obj := HT.Document.SelectedView;
  if Assigned (Obj) then
    if ExecuteElem (HT.Document, El) then begin
      HP := Obj.Params.Item ['гипперсылка'];
      HP.Dim := HP.Dim + 1;
      HP.IndexedValue [HP.Dim] := '!!'+El;
      FillLinks(Obj);
    end;
```

Удаление гипперсылки

```
  Obj := HT.Document.SelectedView;
  if Assigned (Obj) then begin
    HP := Obj.Params.Item ['гипперсылка'];
    HP.Dim := 0;
    FillLinks(Obj);
  end;
```

1.31.11 Step 11. События изменения параметров.

В примере происходит изменение значений параметров по таймеру.

```
i:= random (HT.Document.Flat.Count-1);
Obj:= HT.Document.Flat.Item [i];
Par := Obj.Params.Item [CBParam.text];
Cnt := Cnt + 1;
if Assigned (Par) then
  if Par.name = 'положение' then begin
    if Par.Value = 'отключен' then
      Par.Value := 'включен'
    else
      Par.Value := 'отключен';
  end else begin
  if Par.name = 'значени' then begin
    Par := Obj.Params.Item ['значение_базовое'];
    Par.Value := IntToStr (Cnt);
  end;
  end;
```

1.31.12 Step 12. Работа с навигатором: переключение между окнами.

Пример демонстрирует работу с навигатором. При переключениями между схемами изменяется картинка навигатора. Показан пример вставки навигатора на панель.

Пример придочивания к заданной форме.

```
//вставляем навигатор в панель, убираем заголовок и рамку окна
H := HT1.Navigator.Handle;
if H <> 0 then begin
  windows.SetParent(H, PNavigator.Handle);
  WL:= GetWindowLong (H, GWL_STYLE );
  SetWindowLong(H, GWL_STYLE, WL and not (WS_CAPTION + WS_THICKFRAME));
  RefreshChild;
end;
```

```
procedure TForm1.RefreshChild;
var
  H, st, hr: HWND;
begin
  hr := 0;
  H := GetWindow(PNavigator.Handle, GW_CHILD);
  if H <> 0 then
  begin
    GetWindowRgn(Handle, hr);
    SetWindowRgn(H, hr, true);
    SetWindowPos(H, HWND_TOP, -1, -1, PNavigator.ClientWidth, PNavigator.ClientHeight ,
SWP_SHOWWINDOW);
  end;
end;
```

1.31.13 Step 13. Настройка вида и содержания всплывающей подсказки.

По событию *OnObjectShowHint* настраиваем внешний вид подсказки. Т.е. задаем значение переменной *HintStr* соответствующее заданному стилю.

```
procedure TfrmHTForm.HTSDEForm21ObjectShowHint(Sender: TObject;
  const Doc: ISDEDocument; const SDEObject: ISDEObject2;
  var HintStr: WideString; var CanShow: WordBool);
begin
  if chHint.Checked then
    HintStr := mHint.Text
  else
    mHint.Text := HintStr;
end;
```

Значение *mHint.Text* для подсказки в виде таблицы

```
  mHint.Text :=
  '<table>' +
    '<tr>' +
      '<td>текст1</td>' +
      '<td bgcolor="#00FF00">текст2</td>' +
    '</tr>' +
    '<tr>' +
      '<td bgcolor="#C0C0C0">текст3</td>' +
      '<td bgcolor="#FFFF00">текст4</td>' +
    '</tr>'+
  '</table>';
```

Значение *mHint.Text* для подсказки с разными шрифтами

```
  mHint.Text :=
    '<B>Пример</B>'+
    '<FONT size = "12" color="#FF0000"> использования </FONT>' +
    '<I>различных  </I>' +
    '<U>стилей </U>' +
    '<S> шрифта </S>';
```

Значение *mHint.Text* для подсказки в виде картинки

```
if opPDlg.Execute then
begin
  str := '<p><IMG SRC = ' + opPDlg.FileName + '></p>' + '<BR>' +
       '<p>' + opPDlg.FileName + '</p>'
end;
mHint.Text := str;
```

Значение *mHint.Text* для подсказки в виде html-файла.

```
if opDlgHTML.Execute then
begin
  mHint.Text := '<txt File = "' + opDlgHTML.FileName + '">';
end;
```

1.31.14 Step 14. Работа с таблицами.

В примере изменяются свойства ячеек таблицы.

Функция *FindTable* - поиск таблицы в схеме.

```
function TForm1.Findtable: ISDEObject2;
var
  i: integer;
  F: ISDEObjects2;
  E: ISDEObject2;
begin
  result := nil;
  if not Assigned (HTSDEForm21.Document) then exit;
  F := HTSDEForm21.Document.Flat;
  for i := 0 to F.Count-1 do begin
    E := F.Item [i];
    if E.TypeName = 'таблица2' then begin
      Result := E;
      exit;
    end;
  end;
end;
```

Изменение цвета ячейки:

```
procedure TForm1.BlinkColor;
var
  C: ISDEObject2;
  P: IParam;
begin
  C:= RandomCell;
  if not Assigned (C) then exit;
  P := C.Params['цвет_заполнения'];
  if (P.Value <> 'красный')
    then P.Value:= 'красный'
    else P.Value:= 'желтый';
end;
```

Добавление текста в ячейку: текст

```
procedure TForm1.ChangeText;
var
  S: string;
  C: ISDEObject2;
  P: IParam;
begin
  C:= RandomCell;
  if not Assigned (C) then exit;
  P := C.Params['текст'];
  S:= P.Value;
  P.Value:= S{Copy (S, 1, 6)} + ' ' + IntToStr(FCounter);
end;
```

Изменение цвета текста: цвет_текста

```
procedure TForm1.ChangeTextColor;
var
  C: ISDEObject2;
```

```
  P: IParam;
begin
  C:= RandomCell;
  if not Assigned (C) then exit;
  P := C.Params['цвет_текста'];
  if (P.Value <> 'лиловый')
    then P.Value:= 'лиловый'
    else P.Value:= 'зеленый';
end;
```

Выравнивание содержимого ячеек по горизонтали:

```
procedure TForm1.ChangeAlignHor;
var
  C: ISDEObject2;
  P: IParam;
begin
  C:= RandomCell;
  if not Assigned (C) then exit;
  P := C.Params['текст_горизонтально'];
  if (P.Value = 'слева')
    then P.Value:= 'в_центре'
    else
  if (P.Value = 'в_центре')
    then P.Value:= 'справа'
    else
  if (P.Value = 'справа')
    then P.Value:= 'слева';
end;
```

Выравнивание содержимого ячеек по вертикали:

```
procedure TForm1.ChangeAlignVert;
var
  C: ISDEObject2;
```

```
  P: IParam;
begin
  C:= RandomCell;
  if not Assigned (C) then exit;
  P := C.Params['текст_вертикально'];
  if (P.Value = 'сверху')
    then P.Value:= 'в_центре'
    else
  if (P.Value = 'в_центре')
    then P.Value:= 'внизу'
    else
  if (P.Value = 'внизу')
    then P.Value:= 'сверху';
end;
```

1.31.15 Step 15. Пример работы с именованными параметрами.

Откройте схему и выберете один из элементов схемы. В левой части окна появится список свойств для выбранного элемента.

Заполнение элемента *TV:TreeView* (список свойств)

```
procedure TForm1.HTDocClick(ASender: TObject; const Doc: ISDEDocument;
  const DocEventInfo: IUIEventInfo);
begin
  FTouched := DocEventInfo.Touched;
  Fill (FTouched);
end;

procedure TForm1.Fill (V: ISDEObject2);
begin
  Clear;
  if V = nil then exit;
  FillNode (V.Params, nil);
end;
procedure TForm1.FillNode (AParams: IParams; ANode: TTreeNode);
var
```

```
  i, j, MInd : integer;
  LInfo      : IparamInfo;
  NNode      : TTreeNode;
  LItemData  : TItemData;
  PName, SJ,
  ParentName : string;
  ParentData : TItemData;
  P          : IParam50;
begin
  for i := 0 to AParams.Count-1 do begin
    P := AParams.Item [i] as IParam50;
    if not Assigned (P) then continue;
    LInfo := P.Info;
    if LInfo.Name <> '' then begin
      if (TxCanRead and LInfo.ParMode) = 0 then
        continue;
      MInd := 1;
      if (TxIndexed and LInfo.ParMode) <> 0 then
        MInd := LInfo.maxindex;
      for j := 1 to MInd do begin
        if (TxIndexed and LInfo.ParMode) <> 0 then begin
          SJ := P.Index2Str [j];
          if SJ = '' then
            SJ := IntTostr (j);
          PName := LInfo.Name + '[' + SJ + ']';
        end else
          PName := LInfo.Name;
        NNode := TV.Items.AddChild (ANode, PName);
        LItemData := TItemData.Create;
        LItemData.FullName := PName;
        NNode.Data := LItemData;
        ParentName := '';
        ParentData := ItemData (ANode);
```

```
      if Assigned (ParentData) then
        ParentName := ParentData.FullName;
      LItemData.FullName := CondSep (ParentName, '.', PName);
      if LInfo.atype = atAggregate then
        NNode.HasChildren := true;
    end;
  end;
 end;
end;
```

Выберете одино из свойств схемы. В поле *Memo1* появится иформация о свойстве. Полное название свойства будет выведено в поле *EParamFull.* И заполнятся выпадающие списки с возможными значениями свойств.

```
procedure TForm1.TVChange(Sender: TObject; Node: TTreeNode);
var
  LInfo    : IparamInfo;
  P        : IParam;
  PN       : string;
begin
  Memo1.Lines.Clear;
  if not Assigned (Node) then exit;
  if not Assigned (Node.Data) then exit;

  PN := ItemData (Node).FullName;
  P := FTouched.Params.Item [PN];

  if not Assigned (P) then exit;

  LInfo := P.Info;
  AP ('Name', LInfo.Name);
  AP ('Mode', TxParamMode2String (LInfo.ParMode));
  AP ('Category', '???');
```

```
  AP ('ValuesNo', IntToStr (LInfo.ValuesNo));
  AP ('Values', LInfo.Values);
  AP ('aType', '???');
  AP ('ValuesVerb', LInfo.ValuesVerb);
  AP ('min', IntToStr (LInfo.min));
  AP ('max', IntToStr (LInfo.max));
  AP ('maxindex', IntToStr (LInfo.maxindex));
  AP ('Hint', LInfo.Hint);
  AP ('ElTypes', LInfo.ElTypes);
  AP ('PropertySource', '???');
  AP ('CopyFlags', '???');
  AP ('AutoIncVal', IntToStr (LInfo.AutoIncVal));
  AP ('Help', IntToStr (LInfo.Help));
  AP ('IsDefault', IntToStr (LInfo.IsDefault));
  AP ('DefaultValue', LInfo.DefaultValue);

  Memo1.SelStart := 1;

  EParamFull.Text := ItemData (Node).FullName;
  CBValue.Text := FTouched.Params.Item [ItemData (Node).FullName].Value;
  CBSValue.Text := (FTouched.Params as IParams50).SValue [ItemData (Node).FullName];
  CBOValue.Text := (FTouched as ISDEObject50).SValue [ItemData (Node).FullName];

  { ComboBox }
  FillCombo (CBValue,  P);
  FillCombo (CBSValue, P);
  FillCombo (CBOValue, P);

  { DisUsed }
  CBDisUsed.Enabled := ((TxcanDisUse and LInfo.ParMode) <> 0);
end;

procedure TForm1.FillCombo (ACombo: TComboBox; P: IParam);
```

```
var
  i        : integer;
  S        : string;
begin
  ACombo.Items.Clear;
  if P.Info.atype = atEnumerated then begin
    for i := 0 to P.Info.ValuesNo do begin
      S := ExtractEnumValue(P.Info.Values, i);
      ACombo.Items.Add (S);
    end;
  end;
end;

procedure TForm1.AP(Param, Value: string);
begin
  Memo1.Lines.Add (Param + '=' + Value);
end;
```

Чтобы получить доступ к значению параметра можно воспользоваться одним из трех способов. Покажем это на примере изменения значения параметра.

Пример Params.Item.Value:

```
FTouched.Params.Item [EParamFull.Text].Value := CBValue.Text;
```

Пример Params.SValue:

```
(FTouched.Params as IParams50).SValue [EParamFull.Text] := CBSValue.Text;
```

Пример Object.SValue:

```
(FTouched as ISDEObject50).SValue [EParamFull.Text] := CBOValue.Text;
```

1.31.16 Step 16. Пример реализации функции поиска.

Откройте схему задайте свойство (поле Param), по которому будет производиться поиск. Выберете значение свойства из списка Value. Нажмите кнопку Find.

Формирование выпадающего списка параметров CBParam

```
CBParam.Items.Clear;
LTechs := (HT.Document as ISDEDocument50).Techs as INamedPBs50;
for i := 0 to LTechs.Count-1 do begin
  LTech := LTechs.Item [i];
  for j := 0 to LTech.Params.Count-1 do begin
    S := LTech.Params.Item[j].Name;
    if CBParam.Items.IndexOf (S) < 0 then
      CBParam.Items.Add (S);
  end;
end;
```

Формирование выпадающего списка значений параметров CBValue

```
CBValue.Items.Clear;
LTechs := (HT.Document as ISDEDocument50).Techs as INamedPBs50;
if CBparam.Text = '' then exit;
for i := 0 to LTechs.Count-1 do begin
  LTech := LTechs.Item [i];
  for j := 0 to LTech.Params.Count-1 do begin
    S := LTech.Params.Item[CBparam.Text].Value;
    if S <> '' then
      if CBValue.Items.IndexOf (S) < 0 then
        CBValue.Items.Add (S);
  end;
end;
```

Поиск элемента по выбранному значению свойства. И формирование списка найденных элементов (FillTechs).

```
CBValue.Items.Clear;
LTechs := (HT.Document as ISDEDocument50).Techs as INamedPBs50;
if CBparam.Text = '' then exit;
if CBValue.Text = '' then exit;
```

```
  Found := LTechs.FindPBsIndex(CBparam.Text, CBValue.Text) as INamedPBs50;
  if Found <> nil then
    FillTechs(Found);

procedure TForm1.FillTechs(PBs: INamedPBs);
var
  i     : integer;
  S     : string;
  PB    : NamedPB;
begin
  ListBox1.Clear;
  for i := 0 to PBs.Count-1 do begin
    PB    := PBs.Item [i];
    S := DispName(PB) + ';' + PB.Params.Item ['счетчик_тех'].Value;
    ListBox1.Items.Add (S);
  end;
end;
```

1.32 Приложение 1. Перечень методов (свойств) компонента

Наименование свойства	Доступ	Возможные значения	Описание
FileName	**Read/ write**		**Позволяет выбрать файл формата SDE для отображения. Это свойство доступно только при выполнении приложения.**
Scale	**Read/ write**	**-8..+7**	**Задает масштаб отображения текущей страницы схемы; 0 соответствует масштабу 1:1; -2=50%,-1 = 70%, +2=200%**
DetailsLevel	**Read/**	**0..60**	**Задает уровень детализации**

	write		отображения схем.
NEW!!!			
ScrollSizeX	Read/ write		Задает/определяет ширину холста, на котором отображается схема, в пикселах
ScrollSizeY	Read/ write		Задает/определяет высоту холста, на котором отображается схема, в пикселах
ScrollPointX	Read/ write	0.. (ScrollSizeX – Width)	Задает/определяет позицию ползунка в горизонтальном скроллбаре Width – ширина рабочей области отображения схемы в пикселах
ScrollPointY	Read/ write	0.. (ScrollSizeX – Height)	Задает/определяет позицию ползунка в вертикальном скроллбаре Height – высота рабочей области отображения схемы в пикселах
ScrollVisible	Read/ write	True False	Задает/определяет состояние полос прокрутки компонента
ShowMode	Read/ write	0=txShowSxeme 1= txShowTable	Задает/определяет режим отображения схемы

Document	**Read**	**Nil / ISDEDocument**	**COM интерфейс для работы со схемой**

Методы

Наименование метода	Описание
AboutBox()	Выводит окно с информацией о разработчиках программы.
FitToWindow()	Устанавливает такой масштаб страницы схемы, чтобы она целиком помещалась в окне.
ShowOptions	***Показ панели настроек рабочего места.*** В версиях начиная с 3.06 добавлена возможность подробной настройки рабочего места.
Print	**Pocedure** Print ; **Описание** Вызывает диалог печати текущей страницы.

1.33 Приложение 2. Использование компонента в прикладном ПО (на примере Borland Delphi)

В качестве примера использования компонента ACTIVESXEME рассмотрим создание нового приложения. Проектируемое приложение должно выполнять следующие функции:

- просматривать схемы формата SDE;
- просматривать параметры объектов схемы и их возможные и текущие значения;
- изменять параметры объектов схемы;
- сохранять измененную схему;
- изменять масштаб просмотра схемы;
- изменять уровень детализации просмотра схемы;

- скрывать и показывать инструментальную панель и панель статуса компонента;
- переходить на схеме к указанному объекту и выделять этот объект;
- создавать и обрабатывать дополнительный классификатор для объектов схемы

Прежде чем начать работать с компонентом HTSDEFORM, убедитесь в том, что компонент правильно зарегистрирован в системном реестре, установлен в Delphi и зарегистрирована лицензия компании Модус на использование данного компонента сторонними разработчиками.

Разработаем программу, выполнив следующие этапы и реализовав следующие функции:

Разработка формы

Открытие и отображение файла схемы.

Сохранение файла схемы.

Показать или спрятать инструментальную панель и панель статуса компонента.

Показ панели настроек рабочего места.

Изменение масштаба схемы.

Печать страницы схемы.

Окно навигатора.

Изменение уровня детализации просмотра схемы

Информация о компоненте HTSDEFORM и его разработчиках.

Получение списка объектов.

Обработка событий компонента.

Создание новых параметров для объектов схемы и их обработка.

Разработка формы

Создадим новый проект и назовем его TestOCX. Затем создадим новую форму Form1 (модуль Unit1), поместим на проектируемую форму Form1 компонент ActiveX HTSDEFORM (при установке компонента в Delphi он по умолчанию попадает на палитру инструментов ActiveX) с параметром Align := alClient и стандартные компоненты PopupMenu, OpenDialog, MainMenu, SaveDialog.

Создадим MainMenu следующей структуры:

Каждому пункту меню по событию OnClick поставим вызов соответствующих выбранному пункту меню процедур.

Добавим на проектируемую форму дополнительную панель (для вывода параметров объекта, их допустимых и текущих значений и для изменения значения параметров объекта) с параметром Align := alRight.

Открытие и отображение файла схемы

Для загрузки и отображения файла со схемой можно пользоваться свойством FileName, *доступным только во время выполнения*. При изменении этого свойства компонент попытается загрузить соответствующий файл и отобразить его. При отсутствии файла или ошибках во время загрузки будут выдаваться диагностические сообщения. Например, процедура открытия файла тогда выглядела бы так:

```
procedure TForm1.Open1Click(Sender: TObject);
begin
if OpenDialog1.Execute then begin  //если выбран файл
   HTRootForm1.FileName := OpenDialog1.FileName;    //откроем его
   //===формирование правой панели приложения
    ListValueParam.Items.Clear;
    NameSNIdent.Caption := 'Укажите объект кликом мыши на схеме';
    NoOfParamSNIdent.Caption := 'Укажите объект кликом мыши на схеме';
    SchemElems.Text := '';
    SchemElems.Items.Clear;
    Label3.Visible := False;
    ValuesParam.Visible := False;
   //===
end;
end;
```

Для загрузки и отображения файла со схемой можно воспользоваться интерфейсом

компонента IpersistScheme. Пример использования этого интерфейса для загрузки данных приведен ниже:

```
procedure TForm1.Open1Click(Sender: TObject);
var PerStg: IPersistScheme;
  FStorage: IStorage;
begin
with OpenDialog1 do begin;
  if Execute then begin  //если выбран файл
  //откроем его и получим ссылку на IStorage
  //===формирование правой панели приложения
    ListValueParam.Items.Clear;
    NameSNIdent.Caption := 'Укажите объект кликом мыши на схеме';
    NoOfParamSNIdent.Caption := 'Укажите объект кликом мыши на схеме';
    FillItem();
   Label3.Visible := False;
    ValuesParam.Visible := False;
  //===
  // откроем файл и получим ссылку на IStorage
    if IsStorageFile(FileName) then begin
    //
      OleCheck(StgOpenStorage(StringToOleStr(FileName),
            nil,STGM_READ or STGM_SHARE_EXCLUSIVE,nil,0,FStorage));
  //Получим адрес интерфейса IPersistScheme для компонентаHTSDEFORM
      PerStg := HTRootForm1.ControlInterface as IPersistScheme;
  //Загрузка файла в HTSDEFORM
      OleCheck(PerStg.Load(fStorage));
      PerStg := nil; // Delphi release
    end else
    HTRootForm1.FileName := FileName;
  end;
  FileDir := ExtractFilePath(FileName);
end;
```

```
end;
```

Сохранение файла схемы

Получим имя файла для с помощью SaveDialog1: TSaveDialog, создадим новый составной файл и получим ссылку на IStorage для корневого хранилища нового файла, запишем данные схемы в хранилище по этой ссылке, выполним операцию записи на диск из промежуточных буферов хранилища и сообщим компоненту, что он снова может выполнять запись в свое хранилище. Если операция по сохранению данных не была успешно выполнена, сформируем выдачу звукового сигнала.

```
procedure TForm1.Save1Click(Sender: TObject);
var PerStg: IPersistScheme;
  FStorage: IStorage;
begin
with SaveDialog1 do begin;
if Execute then begin// если выбран файл
  //сформируем его и получим ссылку на IStorage

  //===формирование правой панели приложения
ListValueParam.Items.clear;
    NameSNIdent.Caption := 'Укажите объект кликом мыши на схеме';
    NoOfParamSNIdent.Caption := 'Укажите объект кликом мыши на схеме';
    Label3.Visible := False;
    ValuesParam.Visible := False;
    //===

    //Получим адрес интерфейса IPersistScheme для компонента HTSDEFORM
    PerStg := HTRootForm1.ControlInterface as IPersistScheme;
    //Создадим новый составной файл и получим ссылку на IStorage
    //для корневого хранилища нового файла
    OleCheck( StgCreateDocFile(
          StringToOleStr(SaveDialog1.FileName),
```

```
            STGM_WRITE or STGM_SHARE_EXCLUSIVE or STGM_CREATE,
            0,
            FStorage) );
        try
        try
        //Запись данных схемы в хранилище
        OleCheck(PerStg.Save(fStorage,False));
          //Подтверждение всех изменений для хранилища.
          //Запись на диск из промежуточных буферов хранилища
          fStorage.Commit(STGC_DEFAULT);
        finally
          //Подтверждение успешной записи в хранилище. HTSDE возвращается в режим
корректировки.
          OleCheck(PerStg.SaveCompleted(fStorage));
          PerStg := nil; // Delphi release
        end;
        except
        //При неудачной записи – звуковой сигнал
          Messagebeep(0);
        end;
      end;
    end;
    end;
```

Показать или спрятать инструментальную панель и панель статуса компонента

В приведенном ниже коде ToolBar1 и StatusBar1 – это пункты меню View.

```
procedure TForm1.Status1Click(Sender: TObject);
begin
Status1.Checked := not Status1.Checked;
HTRootForm1.StatusVisible := Status1.Checked;
end;
```

```
procedure TForm1.Toolbar1Click(Sender: TObject);
begin
Toolbar1.Checked := not Toolbar1.Checked;
HTRootForm1.ToolbarVisible := Toolbar1.Checked;
end;
```

При создании проектируемой формы следует установить текущее состояние и видимость панелей компонента и видимость окна навигатора:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
Toolbar1.Checked := HTRootForm1.ToolbarVisible;
Status1.Checked := HTRootForm1.StatusVisible;
Navigator.Checked := HTRootForm1.NavigatorVisible;
end;
```

Показ панели настроек рабочего места

В версиях начиная с 3.06 добавлена возможность подробной настройки рабочего места. Для программного показа диалога настройки следует использовать метод ShowOptions.

Изменение масштаба схемы

Изменение масштаба просмотра страницы схемы выполняется с помощью метода компонента FitToWindow или свойства Scale, доступного для записи.

```
procedure TForm1.FitToWinClick(Sender: TObject);
// Изменяет масштаб вывода страницы схемы так, чтобы она поместилась на экран.
begin
HTRootForm1.FitToWindow;
end;

procedure TForm1.ScaleClick(Sender: TObject);
```

```
//Каждый пункт меню имеет значение TMenuItem(Sender).Tag
//в соответствии с указанным масштабом в поле TMenuItem(Sender).Caption
//Например, если поле TMenuItem(Sender).Caption =’100%’, для вывода на экран
//страницы схемы с масштабом 1:1, то в поле TMenuItem(Sender).Tag следует указать 0.
begin
HTRootForm1.Scale:=TMenuItem(Sender).Tag;
end;
```

Печать страницы схемы

Вызов диалога печати текущей страницы схемы.

```
procedure TForm1.PrintClick(Sender: TObject);
begin
HTRootForm1.Print;
end;
```

Окно навигатора

Выводит или скрывает окно навигатора.

```
procedure TForm1.Navigator1Click(Sender: TObject);
begin
Navigator.Checked := not Navigator.Checked;
HTRootForm1.NavigatorVisible := Navigator.Checked;
end;
```

Изменение уровня детализации просмотра схемы

Изменение уровня детализации просмотра страницы схемы выполняется с помощью свойства компонента DetailsLevel, доступного для записи.

```
procedure TForm1.LevelsClick(Sender: TObject);
```

```
//Каждый пункт меню имеет значение TMenuItem(Sender).Tag
//в соответствии с указанным уровнем детализации в поле TMenuItem(Sender).Caption
//Например, если поле TMenuItem(Sender).Caption =’Коннекторы (самый подробный)’,
//для вывода на экран страницы схемы с самым подробным уровнем
//детализации, то в поле TMenuItem(Sender).Tag следует указать 60.

begin
HTRootForm1.DetailsLevel:=TMenuItem(Sender).Tag;
end;
```

Информация о компоненте HTSDEFORM и его разработчиках

```
procedure TForm1.About1Click(Sender: TObject);
begin
HTRootForm1.AboutBox;
End;
```

Получение списка объектов

Рассмотрим получение списка всех объектов схемы на примере заполнения выпадающего списка SchemElems :

```
Procedure Tform1.FillItem()
Var i: integer;
Begin
SchemElems.Items.Clear;
SchemElems.Text := '';
For (i := 0) to HTRootForm1.Document.Flat.Count - 1 do
  SchemElems.Items.Add(HTRootForm1.Document.Flat.Item[i]. SNIdent);
End;
```

Обработка событий компонента

К событиям, которые генерирует HTSDEFORM, относятся следующие:

OnClick()		Событие, возникающее при нажатии левой кнопки мыши (переопределено)
OnRightClick()		Событие, возникающее при нажатии правой кнопки мыши
OnNavigate(Var FName: WideString; Wname: WideString; Vname: WideString; Perform: WordBool)		Событие вызывается перед переходом по ссылке на другой объект/файл. Параметры FName – имя файла (схемы) WName – имя страницы в схеме VName – имя объекта в схеме Perform –флажок для указания выполнения (True) или отмены (False) перехода.

Обмен данными между графическим модулем и приложением осуществляется через свойства и методы ***текущего объекта*** компонента. Для определения текущего объекта его нужно выбрать, указав его на экране курсором мыши и нажав одну из ее кнопок. А определить, имеется ли текущий элемент можно с помощью свойства ***TouchedView.*** Если *TouchedView* не равен 0, то существует текущий элемент (Текущий элемент можно также установить в результате успешного вызова функций ***FindView*** и ***GotoView***). У каждого объекта схемы есть параметр “***выбран***”, доступный для чтения и записи, принимающий значение “да” или ”нет”. Этот параметр удобен для выделения объекта на схеме и может быть использован в разрабатываемом приложении.

1. ***OnClick()*** - Событие, возникающее при нажатии и отпускании левой кнопки мыши. Разрабатываемое нами приложение будет обрабатывать событие *OnClick()* для формирования данных правой панели (*Panel1*) проектируемой формы. Рассмотрим назначение объектов этой панели более подробно.

NameSNIdent:TRXLabel	Всегда показывает идентификацию текущего элемента (текущий элемент будет выбираться либо нажатием курсора мыши на объекте схемы, либо из списка *SchemElems*) или идентификацию последнего текущего элемента, если курсор мыши был нажат на “пустое место”. Если нет ни одного выбранного объекта, то будет выведен текст 'Укажите объект кликом мыши на схеме'.
SchemElems:TComboBox	Обнуляется при загрузке нового файла схемы. В список заносится идентификация объекта схемы. В список попадают все объекты схемы, на которых был нажат указатель мыши. При выборе элемента списка этот объект становится текущим (осуществляется переход на схеме к выбранному элементу, его выделение и формирование данных на правой панели проектируемой формы).
ListValueParam:TTextListBox	Показывает для текущего элемента все параметры, доступные для чтения и/или записи. При двойном нажатии указателя мыши на параметр из списка будет сформирован список возможных значений для данного параметра *ValuesParam*
NoOfParamSNIdentt:TRXLabel	Всегда показывает общее количество параметров (как доступных для чтения и/или записи, так и внутренних) текущего элемента (текущий элемент будет выбираться либо нажатием курсора мыши на объекте схемы либо из списка *SchemElems*) или общее количество параметров последнего текущего элемента, если курсор мыши был нажат на “пустое место”. Если нет ни одного выбранного объекта, то будет выведен текст 'Укажите объект кликом мыши на схеме'.

ValuesParam:TTextListBox	Формирует список возможных значений для выбранного двойным щелчком указателя мыши параметра из списка *ListValueParam*. При двойном нажатии указателя мыши на некотором значении параметра из списка это значение будет установлено для данного параметра текущего элемента.

Приведем код процедур:

```
procedure TForm1.HTRootForm1Click(Sender: TObject);
//Обработка события нажатия левой кнопки мыши на схеме
begin
Label3.Visible := False; //Пока не выбрали конкретный параметр, нечего показывать
ValuesParam.Visible := False; //Пока не выбрали конкретный параметр, нечего
показывать
//Если есть текущий элемент – сформируем данные правой панели Panel1
if HTRootForm1.TouchedView <> 0 then MakeParamPanel;
end;

procedure TForm1. MakeParamPanel;
//Процедура формирования данных для Panel1
var i,n: Integer;
  S: string;
begin;
with HTRootForm1 do if HTRootForm1.TouchedView <> 0 then begin;
  // Для формирования всех возможных параметров текущего элемента и
  // их возможных значений – используем PopupMenu
  MakePopupMenu; // Собираем параметры и их значения для текущего элемента
  n := NoOfParam; // Общее количество параметров текущего элемента
  ListValueParam.Items.Clear; // Чистим список параметров
  NameSNIdent.Caption := SNIdent; // Выводим идентификацию текущего элемента
  if SchemElems.Items.IndexOf(NameSNIdent.Caption)=-1 then
```

```
    // Если нет такого элемента в списке - добавим
    SchemElems.Items.Add(NameSNIdent.Caption);
  SchemElems.Text := NameSNIdent.Caption;
  SchemElems.Hint := NameSNIdent.Caption;
  // Выведем общее количество параметров текущего элемента
  NoOfParamSNIdent.Caption := Format('%d ',[n]);
  // Сформируем список параметров, доступных для чтения и/или записи,
  // укажем способ доступа к параметру и его текущее значение
  for i:= 0 to pred(n) do begin;
   case GetParamMode(i) of // Метод доступа
     1: S := '(R)';       // Read
     2: S := '(W)';       // Write
     3: S := '(R,W)';     // Read and Write
     else S := '';       // Параметр не доступен извне
   end;
   if S <> '' then
   ListValueParam.Items.Append(GetParamName(i)+' '+S+' '+ //Параметр и метод доступа
            ' = '+GetParamCurrentValue(i));                  // Текущее значение параметра
  end;
end;
end;

procedure TForm1.ListValueParamDblClick(Sender: TObject);
// Обрабатывает двойное нажатие кнопки мыши при выборе
// некоторого параметра текущего элемента из списка ListValueParam
var i,k: Integer;
  S : String;
begin
with HTRootForm1, ListValueParam do begin;
   if FindView(NameSNIdent.Caption) <> S_OK then exit;
   ValuesParam.Items.Clear; // Очистим список значений
   S := Items[ItemIndex];      // Строка из списка параметров
   S := Copy(S,1,Pos(' (',S)-1); // Название параметра
```

```
    i := FindParam(S);              // Номер параметра
    if i < 0 then exit;
    for k :=0 to GetParamValuesNo(i)-1 do begin; // Цикл по количеству возможных
значений
      ValuesParam.Items.Append(GetParamValue(i,k)); // Добавим очередное значение
параметра
    end;
    Label3.Visible := True;
    ValuesParam.Visible := True; // Покажем возможные значения параметра
end;
end;

procedure TForm1.ValuesParamDblClick(Sender: TObject);
// Обрабатывает двойное нажатие кнопки мыши при выборе
// некоторого значения для параметра текущего элемента
var S:String;i:Integer;
begin
with HTRootForm1, ListValueParam do begin;
  if FindView(NameSNIdent.Caption) <> S_OK then exit;
  S := Items[ItemIndex]; // Строка из списка параметров
  i := FindParam(Copy(S,1,Pos(' (',S)-1)); // Номер параметра
  if i < 0 then exit;
  SetParamCurrentValue(i,ValuesParam.Items[ValuesParam.ItemIndex]); // Устанавливает
значение
  // Обновляет текущее значение в списке
  Items[ItemIndex] := Copy(S,1,Pos('=',S)) + ' '+ GetParamCurrentValue(i);
end;
end;

procedure TForm1.NameSNIdentDblClick(Sender: TObject);
begin
HTRootForm1.GotoView(NameSNIdent.Caption);
end;
```

```
procedure TForm1.SchemElemsChange(Sender: TObject);
// Изменение текущего элемента
var
i: integer;
begin
SchemElems.Hint := SchemElems.Text;
if SchemElems.Text = '' then exit; // Нет элемента – нет изменений
NameSNIdent.Caption := SchemElems.Text;
Label3.Visible := False; // Параметр еще не выбран
ValuesParam.Visible := False;
with HTRootForm1 do begin;
  if HTRootForm1.GotoView(SchemElems.Text)<> S_OK then exit; // Изменение текущего
элемента
  HTRootForm1.FindView(SchemElems.Text);
  if TouchedView <> 0 then begin;
    i := FindParam('выбран');          // Присвоим значение параметру ВЫБРАН – ДА,
                                       // чтобы он был выделен на схеме
    if i > -1 then
      SetParamCurrentValue(i,'да'); // Выделим этот элемент на схеме
    MakeParamPanel;      // Сформируем данные на панели для него
  end;
end;
end;
```

2. ***OnRightClick()*** - Событие, возникающее при нажатии правой кнопки мыши. Разрабатываемое нами приложение будет обрабатывать событие *OnRightClick()* для формирования PopupMenu для объекта схемы. Напомним, что если указатель мыши на объекте схемы принимает вид руки, то при нажатии правой кнопки мыши возникает всплывающее меню, по которому можно перейти по ссылке на указанную страницу схемы или другую схему, за выполнение перехода отвечает сам компонент. Если же с текущим элементом схемы не связано никаких переходов, то мы сформируем для

текущего элемента PopupMenu, где пунктами меню будут выступать параметры данного элемента, а пунктами подменю – все возможные значения данного параметра. При выборе пункта подменю, значение параметра будет установлено соответственно выбранному пункту.

```
procedure TForm1.HTRootForm1RightClick(Sender: TObject);
// Процедура обработки события, возникающее при нажатии правой кнопки мыши
var i,k: Integer;
   P: TPoint;
  Item, Item2: TMenuItem;
begin
with HTRootForm1 do begin;
     MakePopupMenu;
     GetCursorPos(P);
     PopupMenu1.PopUp(P.X,P.Y);
   end;
end;


procedure TForm1. MakePopupMenu;
//Формирование PopupMenu для текущего элемента
var i,k: Integer;
   P: TPoint;
  Item, Item2: TMenuItem;
begin
with HTRootForm1 do
if TouchedView <> 0 then // Если имеется текущий элемент
  with PopupMenu1.Items do begin;
    for i:=pred(Count) downto 0 do begin // Очистим PopupMenu
      Item := Items[i];
      Delete(i);
      Item.Free;
    end;
```

```
  for i:=0 to NoOfParam-1 do begin; // Формирем пункты меню
  // Количество пунктов меню - общее количество параметров элемента
    Item := TMenuItem.Create(self);
    Item.Caption := GetParamName(i); // Название параметра
    Item.Visible := GetParamMode(i) <> 0; // Не будем показывать внутренние параметры
                    // Покажем только те, которые можно обрабатывать
    if GetParamValuesNo(i) > 0 then // Если количество значений параметра конечно
(>0),
                                    // то сформируем подменю со всеми возможными
                                    // значениями для данного параметра
      for k :=0 to GetParamValuesNo(i)-1 do begin;
        Item2 := TMenuItem.Create(self);
        Item2.Caption := GetParamValue(i,k); // Возможное значение параметра
        Item.Visible := GetParamMode(i) <> 0;
        if (GetParamMode(i) and TxCanWrite) <> 0 then begin;
          Item2.OnClick := PMItemClick; // Обработчик события выбора пункта меню
                                        // (кол-во значений параметра более 0)
          Item2.Checked := (Item2.Caption = GetParamCurrentValue(i)); // Отметим текущее
значение
          Item.Add(Item2);
        end else
         Item.Caption := GetParamName(i)+' ('+GetParamCurrentValue(i)+')';
      end
    else
```

{Возвращает 0 в случае, если параметр может принимать любые числовые значения, ограниченные конкретным типом текущего элемента. При попытке присвоить значение параметру вне диапазона действительных значений для данного конкретного типа текущего элемента его значение будет установлено на ближайшую границу диапазона. Например, для объекта "вольтметр" параметр под названием "значение" задает величину, отображающую показания прибора в вольтах, а диапазон возможных значений определяется конкретным прибором (текущим элементом).}

```
      if (GetParamMode(i) and TxCanWrite) <> 0 then
        Item.OnClick := PMItemClick2 // Обработчик события выбора пункта меню
                                     // для параметра с неограниченным количеством
значений
      else
        Item.Caption := GetParamName(i)+' ('+GetParamCurrentValue(i)+')';
      Add(Item);
    end;
  end;
end;

procedure TForm1.PMItemClick(Sender: TObject);
// Обработка события выбора пункта меню для параметра с конечным числом его
значений
var i: Integer;
begin
with HTRootForm1 do begin;
  i := FindParam(TMenuItem(TMenuItem(Sender).Parent).Caption);
  SetParamCurrentValue(i,TMenuItem(Sender).Caption); // Присвоить выбранное значение
параметру
end;
HTRootForm1Click(self); // to rebuild values list
end;

procedure TForm1.PMItemClick2(Sender: TObject);
// Обработка события выбора пункта меню для параметра с неограниченным
количеством значений
var i: Integer;
begin
with HTRootForm1 do begin;
  i := FindParam(TMenuItem(Sender).Caption);
  Form2.Edit1.Text := GetParamName(i); // Название параметра
  Form2.Edit2.Text := GetParamCurrentValue(i); // Текущее значение
```

```
  if Form2.ShowModal = mrOK then // Показать окно для ввода значения параметра
    SetParamCurrentValue(i,Form2.Edit2.Text); // Присвоить введенное значение
параметру
  HTRootForm1Click(self); // to rebuild values list
end;
end;
```

Форма *Form2* – это простая форма для ввода значений параметра, который может принимать любые числовые значения, ограниченные конкретным типом текущего элемента *(При попытке присвоить значение параметру вне диапазона действительных значений для данного конкретного типа текущего элемента его значение будет установлено на ближайшую границу диапазона. Проверку правильности значения осуществляет сам компонент HTSDEFORM).*

3. ***Navigate()*** - Событие вызывается перед переходом по ссылке на другой объект/файл..
   Если указатель мыши на объекте схемы принимает вид руки, то при нажатии правой кнопки мыши возникает всплывающее меню, по которому можно перейти по ссылке на указанную страницу схемы или другую схему, за выполнение перехода отвечает сам компонент. Однако, в приложении можно, например, перед переходом проверить наличие файла и отменить попытку перехода и выдать диагностическое сообщение при его отсутствии.

```
procedure TForm1.HTRootForm1Navigate(Sender: TObject;
var FileName: WideString; const PageName, ViewIdent: WideString;
var Perform: WordBool);
begin
if FileName = '' then exit; // локальный переход в файле
// В связи с тем, что ссылки хранятся относительно текущей схемы,
// необходимо добавить к имени файла путь в текущую директорию
FileName:= fileDir+FileName;
if FileExists(FileName) then begin; // Проверка существования файла
  FileDir := ExtractFilePath(FileName);
  Perform := True;
```

```
// Выдача диагностики при отсутствии файла
end else MessageDlg('File is not exist:'#10+FileName,mtError,[mbOK],0);
end;
```

Общий вид главного окна, созданного нами приложения:

1.34 Приложение 3. Использование компонента в прикладном ПО (на примере Borland C++ Builder)

Рассмотрим применение объектной модели2 на примере разработки программы для тестирования ActivesXeme. Мы рассмотрим только основные аспекты, не уделяя внимание таким функциям как:

- Вызов окна навигатора,
- Вызов окна настроек рабочего места
- И др.

Реализуем следующий набор функций:

Загрузка схемы

Получение списка всех объектов схемы

Переход к указанному объекту схемы

Получение списка всех параметров конкретного объекта схемы

Изменение значения параметров объекта схемы

Перед началом работы необходимо разместить на форме компонент ActivesXeme. Таким образом, в программе вы получите объект HTSDEForm1 класса HTSDEForm.

Загрузка схемы

Для удобства выполнения функции чтения, возможно применение стандартного компонента OpenDialog. Функцию «Загрузка схемы» целесообразно выполнять при выборе пункта главного меню программы «Открыть»:

```
void __fastcall TForm1::Open1Click(TObject *Sender)
{
```

```
if (OpenDialog1->Execute())
{
  HTSDEForm1->FileName = OpenDialog1->FileName;
  FillTree();
}
}
```

Получение списка всех объектов схемы

Получение списка всех объектов схемы реализовано в функции FillTree, которая заполняет компонент TreeView списком всех объектов схемы (разумеется перед реализацией этой функции необходимо разместить на форме компонент TreeView):

```
void TForm1::FillTree()
{
ISDEPagePtr Page;
ISDEObjects2Ptr Objects2;
ISDEObject2Ptr Object2;
TTreeNode *ParentNode, *TypeNode,*Temp;
int i,j;
int Count;
String Type;
bool flag;

TreeView1->Items->Clear();
for (i = 0; i < HTSDEForm1->Document->Pages->Count; i++)
{
  Page = HTSDEForm1->Document->Pages->get_Item(i);
  ParentNode = TreeView1->Items->AddChild(0, Page->get_Name());
  Objects2 = Page->get_SDEObjects();
  Count = Objects2->get_Count();
  for (j = 0; j < Count; j++ )
  {
   Object2 = Objects2->get_Item(j);
```

```
    Type = Object2->get_TypeName();
    TypeNode = TreeView1->Items->AddChild(ParentNode, Type);
    Temp = TypeNode;
    flag = true;
    do
    {
      Temp = Temp->getPrevSibling();
      if (Temp != NULL)
        if (Temp->Text == Type)
          flag = false;
    }
    while((Temp != NULL) && (flag));
    if (flag == false)
    {
      TypeNode->Delete();
      TypeNode = Temp;
    }
    Temp = TreeView1->Items->AddChild(TypeNode, Object2->get_SNIdent());
  }
}
}
```

Переход к указанному объекту схемы

Переход к указанному объекту схемы будем осуществлять при выборе конкретного объекта в построенном дереве объектов:

```
void __fastcall TForm1::ChangeNode(TObject *Sender, TTreeNode *Node,
bool &AllowChange)
{
WideString S;
if (Node != NULL)
{
  S = Node->Text;
```

```
  HTSDEForm1->GotoView(S);
}
}
```

Получение списка всех параметров конкретного объекта схемы

Данную функцию реализуем в качестве построения всплывающего меню, при нажатии правой клавиши мыши над определенным объектом:

```
void __fastcall TForm1::HTSDEForm1RightClick(TObject *Sender)
{
if (HTSDEForm1->TouchedView != 0)
  MakePopUpMenu();
}

TMenuItem* AddMenuItem (String Caption, TMenuItem *Items, Boolean Check)
{
TMenuItem *Item;
  Item = new TMenuItem(Form1);
  Item->Caption = Caption;
  Item->Checked = Check;
  Items->Add(Item);
  return Item;
}

void TForm1::MakePopUpMenu()
{
int i, j, k;
int Mode;
String ItemCaption;
POINT P;
TMenuItem *Item1, *Item2;
```

```
PopupMenu1->Items->Clear();
Item1 = new TMenuItem(this);
Item2 = new TMenuItem(this);

for (i = 0; i < HTSDEForm1->NoOfParam; i++)
{
  Mode = HTSDEForm1->GetParamMode(i);
  if ((Mode & (TxCanRead | TxCanWrite)) > 0)
  {
   for (j = 1; j <= HTSDEForm1->GetParamDim(i); j++)
   {
    ItemCaption = HTSDEForm1->GetParamName(i);
    if ((Mode & TxIndexed) > 0)
     ItemCaption = ItemCaption + '[' + j + ']';
    ItemCaption = ItemCaption +
                  '(' + HTSDEForm1->GetParamIndexedValue(i, j) + ')';
    Item1 = AddMenuItem(ItemCaption, PopupMenu1->Items, false);
    if (HTSDEForm1->GetParamValuesNo(i) > 0)
    {
     String CurValue = HTSDEForm1->GetParamIndexedValue(i, j);
     for (k = 0; k < HTSDEForm1->GetParamValuesNo(i); k++)
     {
      ItemCaption = HTSDEForm1->GetParamValue(i, k);
      bool Checked = (ItemCaption == CurValue);
      Item2 = AddMenuItem(ItemCaption, Item1, Checked);
      Item2->OnClick = SetParamObject;
      Item2->Tag = j;
      Item2->Tag = Item2->Tag + (i << 16);
     }
    }
   }
  }
}
```

```
GetCursorPos(&P);
PopupMenu1->Popup(P.x, P.y);
}
```

Изменение значения параметров объекта схемы

Изменение значения параметров объекта схемы будет происходить при выборе конкретного пункта всплывающего меню. Для успешной реализации этой функции мы выполнили ряд подготовительных этапов, а именно заполнили свойство Tag объектов TMenuItem необходимой информацией, и установили возможность обрабатывать выбор пунктов меню с функцией SetParamObject:

```
void __fastcall TForm1::SetParamObject(TObject *Sender)
{
TMenuItem *Item;
int i, j;
WideString S;
Item = dynamic_cast <TMenuItem*> (Sender);
if (Item)
{
  j = LOWORD (Item->Tag);
  i = HIWORD (Item->Tag);
  S = Item->Caption;
  HTSDEForm1->SetParamIndexedValue(i, j, (wchar_t*)S);
}
}
```

1.35 Приложение 4. Использование компонента в прикладном ПО (на примере Visual C++)

Рассмотрим применение объектной модели2 на примере разработки программы для тестирования ActivesXeme. Мы рассмотрим только основные аспекты, не уделяя внимание таким функциям как:

- Вызов окна навигатора,
- Вызов окна настроек рабочего места
- И др.

Реализуем следующий набор функций:

Загрузка схемы

Получение списка всех объектов схемы

Переход к указанному объекту схемы

Получение списка всех параметров конкретного объекта схемы

Изменение значения параметров объекта схемы

Перед началом работы необходимо разместить на форме компонент ActivesXeme (это возможно, если базовый класс объекта виды выбрать CFormView). Таким образом, Вы получите реализацию всех классов Объектной модели2, т.к. всю работу реализации выполнит Wizard (мастер).

Загрузка схемы

Функцию «Загрузка схемы» целесообразно выполнять при выборе пункта главного меню программы «Открыть»:

```
void CTestOCXView::GetFile(char *FileName)
{
OPENFILENAME fn;

strcpy(FileName, "");
fn.lStructSize = sizeof(tagOFNA);
fn.hwndOwner = m_hWnd;
fn.hInstance = (HINSTANCE) GetWindowLong(m_hWnd, GWL_HINSTANCE);
fn.lpstrFilter = "Файл SDE\0*.sde\0Все файлы\0*.*\0\0";
fn.lpstrCustomFilter = NULL;
fn.nMaxCustFilter = 0;
fn.nFilterIndex = 1;
fn.lpstrFile = (LPSTR) FileName;
fn.nMaxFile = 256;
fn.lpstrFileTitle = NULL;
```

```
fn.nMaxFileTitle = 1;
fn.lpstrInitialDir = NULL;
fn.lpstrTitle = NULL;
fn.nFileOffset = 0;
fn.nFileExtension = 0;
fn.lpstrDefExt = "sde";
fn.lCustData = 0;
fn.lpfnHook = NULL;
fn.lpTemplateName = 0;
fn.Flags=OFN_HIDEREADONLY | OFN_FILEMUSTEXIST | OFN_PATHMUSTEXIST;

GetOpenFileName(&fn);
}

void CTestOCXView::OnFileOpen()
{
CTestOCXDoc *pDoc;
pDoc = GetDocument();
GetFile(pDoc->File);
m_HTSDEForm.SetFileName(pDoc->File);
FillTree();
}
```

Получение списка всех объектов схемы

Получение списка всех объектов схемы реализовано в функции FillTree, которая работает с классом CTreeCtrl, и заполняет дерево списком всех объектов схемы (разумеется перед реализацией этой функции необходимо разместить на форме контрол TreeControl):

```
void CTestOCXView::FillTree()
{
CSDEDocument SDEDocument;
CSDEPages SDEPages;
```

```
CSDEPage SDEPage;
CSDEObjects2 SDEObjects2ofPage;
CSDEObject2 SDEObject2;
HTREEITEM hParentItem, hItem, hTempItem;

long i, j;
char str[250], type[250];
_variant_t Var;

m_tree.DeleteItem(TVI_ROOT);
SDEDocument = m_HTSDEForm.GetDocument();
SDEPages = SDEDocument.GetPages();
for (j = 0; j < SDEPages.GetCount(); j++)
{
  Var = (long) j;
  SDEPage = SDEPages.GetItem(Var);
  hParentItem = m_tree.InsertItem(SDEPage.GetName(), 0, 0);//insert to ROOT
  SDEObjects2ofPage = SDEPage.GetSDEObjects();
  strcpy(type, "");
  for (i = 0; i < SDEObjects2ofPage.GetCount(); i++)
  {
    Var = (long) i;
    SDEObject2 = SDEObjects2ofPage.GetItem(Var);
    wsprintf (str, "%s", SDEObject2.GetTypeName());//name of TYPE
      if (strcmp(str, type) != 0) // not equal
      {
         hItem = m_tree.InsertItem(str, 0, 0, hParentItem);//insert to PAGE
         hTempItem = hItem;
         do
         hTempItem = m_tree.GetNextItem(hTempItem, TVGN_PREVIOUS);//on the same level
         while ((hTempItem != NULL) && (strcmp(m_tree.GetItemText(hTempItem), str) != 0));
         if (hTempItem != NULL)
         {
```

```
        m_tree.DeleteItem(hItem);
        hItem = hTempItem;
        }
        strcpy(type, str);
    }
 m_tree.InsertItem(SDEObject2.GetSNIdent(), 0, 0, hItem);
  }
}
}
```

Переход к указанному объекту схемы

Переход к указанному объекту схемы будем осуществлять при выборе конкретного объекта в построенном дереве объектов:

```
void CTestOCXView::OnSelchangingTree1(NMHDR* pNMHDR, LRESULT* pResult)
{
long id;
NM_TREEVIEW* pNMTreeView = (NM_TREEVIEW*)pNMHDR;
If (pNMTreeView->itemNew.hItem != NULL)
  m_HTSDEForm.GotoView(m_tree.GetItemText(pNMTreeView->itemNew.hItem));
*pResult = 0;
}
```

Получение списка всех параметров конкретного объекта схемы

Данную функцию реализуем в качестве построения всплывающего меню, при нажатии правой клавиши мыши над определенным объектом:

```
const  IDPopUpMenu = 3333

HMENU CTestOCXView::CreatePopUpMenu()
{
HMENU hMenu = CreatePopupMenu(), hMenuLevel2;
int i, k, j;
```

```
int NumberParameters, Mode;
char str[250];
int uniqueID = 0;
int select;

NumberParameters = m_HTSDEForm.GetNoOfParam();
for (i = 0; i < NumberParameters; i++)
{
    Mode = m_HTSDEForm.GetParamMode(i);
    //if (Mode & (TxCanRead | TxCanWrite))
    for (j = 1; j <= m_HTSDEForm.GetParamDim(i); j++)
    {
      if (m_HTSDEForm.GetParamDim(i) == 1)
        wsprintf(str, "%s = (%s)", m_HTSDEForm.GetParamName(i),
m_HTSDEForm.GetParamIndexedValue(i, j));
     else
       sprintf(str, "%s[%d] = (%s)", m_HTSDEForm.GetParamName(i), j,
m_HTSDEForm.GetParamIndexedValue(i, j));

      if (m_HTSDEForm.GetParamValuesNo(i) > 0)
     {
        hMenuLevel2 = CreatePopupMenu();
        for (k = 0; k < m_HTSDEForm.GetParamValuesNo(i); k++)
       {
         if (strcmp(m_HTSDEForm.GetParamValue(i, k), m_HTSDEForm.GetParamIndexedValue(i,
j)) == 0)
             select = MF_CHECKED;
         else
           select = 0;
         AppendMenu(hMenuLevel2, MF_STRING|(MF_CHECKED      & select),
IDPopUpMenu + uniqueID, m_HTSDEForm.GetParamValue(i, k));
          uniqueID++;
       }
```

```
            AppendMenu(hMenu, MF_POPUP, (UINT) hMenuLevel2, str);
        }
        else
        {
          AppendMenu(hMenu, MF_STRING, IDPopUpMenu + uniqueID, str);
          uniqueID++;
        }
      }
}
return hMenu;
}

void CTestOCXView::OnRightClickHtsdeform1()
{
POINT Point;
GetCursorPos(&Point);
TrackPopupMenu(CreatePopUpMenu(), 0, Point.x, Point.y, 0, m_hWnd, 0);
}
```

Изменение значения параметров объекта схемы

Изменение значения параметров объекта схемы будет происходить при выборе конкретного пункта всплывающего меню. Для успешной реализации этой функции мы выполнили ряд подготовительных этапов, а именно: разработали систему уникальных идентификаторов для каждого пункта меню и в карте обработки событий указали функцию, отвечающую за обработку выбора пунктов меню

```
ON_COMMAND_RANGE(IDPopUpMenu, IDPopUpMenu + 500, OnPopUpMenu)
void CTestOCXView::OnPopUpMenu(UINT nID)
{
int i, j, k;
int Mode, ValuesNo, NumberParameters;
UINT uniqueID = IDPopUpMenu;
bool flag=false;
```

```
CString Value;

NumberParameters = m_HTSDEForm.GetNoOfParam();
//Расшифруем параметр и его значение по ID
for (i = 0; i < NumberParameters; i++)
{
Mode = m_HTSDEForm.GetParamMode(i);
if (Mode & (TxCanRead | TxCanWrite))
for (j = 1; j <= m_HTSDEForm.GetParamDim(i); j++)
{
  k = 0;
  ValuesNo = m_HTSDEForm.GetParamValuesNo(i);
  do
  {
    if(uniqueID == nID)
    {
      flag = true;
      break;
    }
    uniqueID++;
    k++;
  }
  while (k < ValuesNo);
  if (flag)
  break;
}
if (flag)
break;
}

Value = m_HTSDEForm.GetParamValue(i, k);
m_HTSDEForm.SetParamIndexedValue(i, j, Value);
}
```

1.36 Приложение 5. Использование компонента в прикладном ПО (на примере Visual Basic)

Рассмотрим применение объектной модели2 на примере разработки программы для тестирования ActivesXeme. Мы рассмотрим только основные аспекты, не уделяя внимание таким функциям как:

- Вызов окна навигатора,
- Вызов окна настроек рабочего места
- И др.

Реализуем следующий набор функций:

Загрузка схемы

Получение списка всех объектов схемы

Переход к указанному объекту схемы

Получение списка всех параметров конкретного объекта схемы

Изменение значения параметров объекта схемы

Применение коллекций на примере функции FindFire

Перед началом работы необходимо разместить на форме компонент ActivesXeme. Таким образом, Вы получите доступ к объекту HTSDEForm1 класса HTSDEForm.

Загрузка схемы

Для удобства выполнения функции чтения, возможно применение стандартного компонента CommonDialog (объект CommonDialog мы назвали CD). Функцию «Загрузка схемы» целесообразно выполнять при выборе пункта главного меню программы «Открыть»:

```
Private Sub Open_Click()
CD.Filter = "Файл SDE (*.sde)|*.sde"
CD.ShowOpen
HTSDEForm1.FileName = CD.FileName
FillTree
End Sub
```

Получение списка всех объектов схемы

Получение списка всех объектов схемы реализовано в функции FillTree, которая заполняет компонент TreeView (объект TreeView мы назвали TV) списком всех объектов схемы (разумеется перед реализацией этой функции необходимо разместить на форме компонент TreeView из библиотеки Common Controls – Comctl32.ocx):

```
Private Sub FillTree()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim str As String
Dim TypeOfObject As String
Dim TempType As String
Dim UniqueType As Boolean
Dim UniqueObject As Boolean
Dim ObjectName As String

TV.Nodes.Clear
For i = 0 To HTSDEForm1.Document.Pages.Count - 1
str = HTSDEForm1.Document.Pages.Item(i).Name
TV.Nodes.Add , , str, str
TempType = HTSDEForm1.Document.Pages.Item(i).SDEObjects.Item(0).TypeName
For j = 0 To HTSDEForm1.Document.Pages.Item(i).SDEObjects.Count - 1
  TypeOfObject = HTSDEForm1.Document.Pages.Item(i).SDEObjects.Item(j).TypeName
  UniqueType = True
  UniqueObject = True

  ObjectName = HTSDEForm1.Document.Pages.Item(i).SDEObjects.Item(j).SNIdent

  For k = 1 To TV.Nodes.Count
    If TV.Nodes.Item(k).Key = str + TypeOfObject Then
```

```
      UniqueType = False
    End If
    If TV.Nodes.Item(k).Key = ObjectName Then
      UniqueObject = False
    End If
  Next k
  If UniqueType = True Then
    TV.Nodes.Add str, tvwChild, str + TypeOfObject, TypeOfObject
  End If
  If UniqueObject = True Then
    TV.Nodes.Add str + TypeOfObject, tvwChild, ObjectName, ObjectName
    End If
Next j
Next i
End Sub
```

Переход к указанному объекту схемы

Переход к указанному объекту схемы будем осуществлять при выборе конкретного объекта в построенном дереве объектов:

```
Private Sub TV_NodeClick(ByVal Node As ComctlLib.Node)
HTSDEForm1.GotoView Node.Text
End Sub
```

Получение списка всех параметров конкретного объекта схемы

Получение списка всех параметров конкретного объекта схемы мы реализуем в виде заполнения компонентов ListBox (ListBox1 и ListBox2) при назатии левой клавишей мыши на объекте:

```
Private Sub HTSDEForm1_OnClick()
Dim i As Integer
```

```
Dim j As Integer
Dim str As String
Dim mode As Long
If (HTSDEForm1.TouchedView <> 0) Then
  List1.Clear
  For i = 0 To HTSDEForm1.NoOfParam - 1
    mode = HTSDEForm1.GetParamMode(i)
    If HTSDEForm1.GetParamDim(i) > 1 Then
      For j = 1 To HTSDEForm1.GetParamDim(i)
        str = HTSDEForm1.GetParamName(i)
        If ((mode And TxIndexed) > 0) Then
          str = str + "[" + Format(j) + "]"
        End If
        str = str + GetMode(mode)
        If (HTSDEForm1.GetParamCurrentValue(i) <> "") Then
         str = str + "=" + HTSDEForm1.GetParamCurrentValue(i)
        End If
        List1.AddItem str
      Next j
    Else
      str = HTSDEForm1.GetParamName(i)
      List1.AddItem str + " " + GetMode(mode) + "=" + HTSDEForm1.GetParamCurrentValue(i)
    End If
  Next i
End If
End Sub

Private Sub List1_Click()
  Dim i As Integer
  Index = 0
For i = 0 To List1.ListIndex
  If HTSDEForm1.GetParamDim(Index) > 1 Then
   i = i + HTSDEForm1.GetParamDim(Index) - 1
```

```
  End If
  Index = Index + 1
Next i
Index = Index - 1
  List2.Clear
For i = 0 To HTSDEForm1.GetParamValuesNo(Index) - 1
  List2.AddItem HTSDEForm1.GetParamValue(Index, i)
Next i
End Sub
```

Изменение значения параметров объекта схемы

Изменение значения параметров объекта схемы будет происходить при выборе пункта в окне ListBox2:

```
Private Sub List2_Click()
Dim ListIndex As Integer
HTSDEForm1.SetParamCurrentValue Index, List2.Text
ListIndex = List1.ListIndex
HTSDEForm1_OnClick
If (ListIndex > -1 And ListIndex < List1.ListCount) Then
  List1.ListIndex = ListIndex
End If
End Sub
```

Применение коллекций на примере функции FindFire

Функция FindFire ищет горящие объекты, и в случае нахождение таковых сообщает о пожаре:

```
Private Sub FindFire()
Dim Object2 As SDEObject2
Dim parameter As Param
Dim find As Boolean
find = False
```

```
For Each Object2 In HTSDEForm1.Document.Flat
  For Each parameter In Object2.Params
    If (parameter.Name = "пожар" And parameter.Value = "горит") Then
      find = True
      Exit For
    End If
  Next
  If find = True Then
    MsgBox ("Внимание, пожар на объекте " + Object2.SNIdent + "!")
  End If
Next
End Sub
```

1.37 Приложение 6. Использование ActiveX в Internet/Intranet.

Так как ActivesXeme реализует технологию ActiveX фирмы Microsoft, он может быть легко интегрирован в Web-страницы для просмотра схем через Internet/Intranet с использованием браузера Internet Explorer фирмы Microsoft или Netscape Navigator. Возможны несколько способов кодирования HTML-документов для показа схем:

Использование тега <OBJECT>.

Это стандартный для Internet Explorer способ загрузки OCX на страницах HTML.

```
<OBJECT       CLASSID = "clsid:91021E34-6222-11D1-A8DF-000001325083"
              CODEBASE = "//www.swman.ru/htsde/htsde.ocx"
              HEIGHT = 300 WIDTH = 400 >
    <PARAM NAME = "SRC" VALUE = "a10kv.sde">
</OBJECT>
```

Значение параметра SRC задает URL к файлу, содержащему схему для просмотра. Параметр CLASSID идентифицирует компонент HTSDEFORM. Эта форма вызова необходима, если OCX еще не установлен на машине клиента. Параметр CODEBASE задает

расположение (URL) файла, содержащего OCX. При первом обращении к такой странице, IExplorer загрузит OCX из сети и установит его на локальном компьютере. В дальнейшем (либо после установки комплекса программ MODUS) можно использовать и другие модификации тега <OBJECT>, например:

```
<OBJECT       TYPE="application/sde" HEIGHT=300 WIDTH=400 >
        <PARAM NAME="SRC" VALUE="a10kv.sde">
</OBJECT>
```

или тег <EMBED> вида:

```
<EMBED HEIGHT=300 WIDTH=400 SRC="a10kv.sde”>
</EMBED>
```

Если компонент уже зарегистрирован в системе, то IExplorer также способен сам загрузить требуемый OCX при указании в URL файла с расширением SDE, например:

```
<a HREF = "a10kv.sde”>Подробная схема</a>
```

К сожалению, это не всегда правильно работает для локальных файлов из-за ошибки в IExplorer. Для обхода этой ошибки имя локального файла следует указывать с использованием прямых, а не обратных слешей для отделения каталогов от имени файла, например:

C:\SHEMES\a10kv.sde – неправильно

C:/SHEMES/a10kv.sde – правильно

Доступ к свойствам и событиям из JavaScript сценариев.

В программах на языке JavaScript можно использовать все доступные методы и свойства HTSDEFORM (методы, свойства и события перечислены в приложении 1). Также можно определить реакцию на события, происходящие в HTSDEFORM, например, на нажатие клавиш мыши. В приведенном ниже примере при нажатии правой кнопки мыши на объекте, имеющем два положения, он изменит свое состояние.

```
<html>
```

```
<head>   <title>Пример</title> </head>
<body>
  <OBJECT id=TestEvt TYPE="application/sde" HEIGHT=500 WIDTH=100%">
    <PARAM NAME="SRC" VALUE=" a10kv.sde ">
  </OBJECT>
  <script language=javascript for="TestEvt" event="onRightClick()">
    if (TestEvt.FTouched != 0)
    {
      var Param = TestEvt.FindParam('положение');
      if (Param >= 0)
      {
        if (TestEvt.GetParamCurrentValue(Param) == 'включен')
          TestEvt.SetParamCurrentValue(Param,'отключен')
        else
          TestEvt.SetParamCurrentValue(Param,'включен');
      }
    }
  </script>
</body>
</html>
```

Показ схем в Netscape Navigator.

К сожалению, непосредственно сам браузер Netscape Navigator не поддерживает технологию ActiveX. Однако существует ряд фирм, разработавших PLUG-IN'ы Netscape Navigator для работы с ActiveX. Вот применение HTSDEFORM с использованием одного из таких PLUG-IN'ов фирмы "Esker":

```
<html>
<body>
    <embed type="application/x-eskerplus"
        classid="clsid:91021E34-6222-11D1-A8DF-000001325083"
        id="NfsdlgX"
        width="100%" height="100%"
```

```
        src="ee11.sde">
</body>
</html>
```

Ключевые слова