Компоненты WinCC

Графический редактор

Графический редактор

- **Создание кадров процесса**
- Кадры , изображающие процесс в режиме исполнения , создаются с помощью графической системы. Чтобы создать новый кадр процесса и открыть Graphics Designer [Графический дизайнер], выполните следующие действия:
- В левой части окна WinCC Explorer щелкните правой кнопкой мыши на "Graphics Designer". При этом откроется всплывающее меню. Во всплывающем меню выберите пункт "New Picture [Новый кадр]. При этом будет создан и отображен в правой части окна WinCC Explorer файл кадра (".pdl" = "Picture Description File [Файл описания кадра]") с именем "NewPdl0.pdl".

Графический редактор

- Чтобы переименовать кадр, в правой части окна WinCC Explorer щелкните правой кнопкой мыши на "NewPdl0.pdl". Во всплывающем меню выберите пункт "Rename Picture [Переименовать кадр]". В открывшемся диалоговом окне введите новое имя кадра.
- Чтобы посмотреть кадр и открыть Graphics Designer, дважды щелкните на имени кадра в правой части окна WinCC Explorer. Кроме этого , открыть кадр можно , щелкнув правой кнопкой мыши на его имени и выбрав пункт "Open Picture" во всплывающем меню.

Графический редактор

- **Палитра цветов**
- Используется для определения цвета для выбранного объекта. В дополнение к 16 стандартным цветам можно определять свои собственные цвета.
- **Палитра объектов**
- Содержит Standard Objects [Стандартные объекты] (Polygon [Многоугольник], Ellipse [Эллипс], Rectangle [Прямоугольник], и т .д.), Smart Objects [Интеллектуальные объекты] (OLE Control [Элемент управления OLE], OLE Element [Элемент OLE], I/O Field [Поле ввода /вывода], и т .д .) и Windows Objects [Объекты Windows] (Button [Кнопка], Check-Box [Поле - флажок], и т .д .).

Графический редактор

- **Палитра стилей**
- Позволяет изменить внешний вид выбранного объекта. В зависимости от типа объекта можно изменять тип линии или границы, толщину линии или границы, стиль конца линии или узор заливки.
- **Палитра выравнивания**
- Позволяет изменять абсолютное положение одного или более объектов, положение выбранных объектов относительно друг друга или выравнивать высоту и ширину нескольких объектов.

Графический редактор

- **Палитра масштабирования**
- Устанавливает коэффициент масштабирования (в процентах) для активного окна. Можно использовать кнопки для задания нескольких коэффициентов масштабирования, соответствующих данным кнопкам по умолчанию.
- **Панель меню**
- Содержит все команды меню Graphics Designer [Графического дизайнера]. Команды, которые в данный момент выполнить нельзя, отображаются серым цветом.
- **Панель инструментов**
- Содержит кнопки для быстрого вызова команд .

Графический редактор

- **Палитра шрифтов**
- Изменяет тип, размер и цвет шрифта текстовых объектов, а также цвет линий стандартных объектов.
- **Панель слоев**
- Позволяет определить, какие из 32 возможных слоев будут видимы. Слой 0 является видимым по умолчанию.

Графический редактор

- **Замечание.** Для определения панелей или палитр, которые будут отображаться в Graphics Designer, щелкните на командах панели меню "View" "Toolbars". В диалоговом окне "Toolbars" отметьте, какие панели/палитры должны отображаться, и затем щелкните на кнопке "OK".
- Используя палитру объектов, разместим в кадре какой-либо из них. Доступ к параметрам объекта можно получить, сделав по нему 2 клика левой кнопкой мыши, или через Properties в меню правой кнопки мыши, или через кнопку Properties в панели инструментов. К параметрам относятся: геометрия, цвет границы/фона, стиль линий/заливки, уровень заливки и прочее. Все эти свойства можно менять не только вручную, но из различных сценариев, по событиям и по изменению значений тегов.

Запуск проекта, определение свойств окна Runtime

- **Настройка параметров режима исполнения**
- Следующим этапом определяются свойства и параметры режима исполнения для запуска проекта. В числе прочего эти параметры определяют вид экрана в режиме исполнения. В левой части окна WinCC Explorer щелкните на элементе "Computer”. В правой части окна WinCC Explorer щелкните правой кнопкой мыши на имени вашего компьютера. Во всплывающем меню выберите пункт "Properties”. Щелкните на закладке "Graphics Runtime”. Здесь вы можете определить вид экрана в режиме исполнения.
- Для выбора начального кадра щелкните на кнопке "Browse" и выберите нужный кадр. Нажмите на кнопку "OK". В окне "Window Attributes” установите флажки "Title”, "Maximize”, "Minimize” и "Adapt Picture”.

Запуск проекта, определение свойств окна Runtime

- **Запуск проекта**
- Чтобы увидеть, как выглядит проект в режиме исполнения, щелкните на командах меню в WinCC Explorer "File" "Activate". Галочка около "Activate" означает, что режим исполнения активирован.
- Альтернативно можно использовать кнопку "Activate” на панели инструментов WinCC Explorer.
- **Замечание.** Щелкните на кнопке "Activate" панели инструментов Graphics Designer, и вы сразу же увидите изменения, внесенные в кадр .

Запуск проекта, определение свойств окна Runtime

- **Использование имитатора**
- Если к WinCC не подключен ПЛК, для тестирования проекта можно использовать имитатор. Чтобы запустить имитатор, перейдите на панель задач Windows и щелкните на пункте меню "Start" "SIMATIC" "WinCC" "Tools" "WinCC Tag Simulator".
- **Замечание.** Для того чтобы имитатор функционировал правильно, проект должен быть активизирован (в режиме исполнения).
- В диалоговом окне имитатора выберите тег, который вы хотите смоделировать. Для этого щелкните на "Edit" "New Tag". В диалоговом окне "Tags – Project" выберите внутренний тег "Position" и щелкните на кнопке "OK".
- На закладке "Properties [Свойства]" щелкните на режиме имитации "Inc".
- Введите начальное значение "0" и конечное "100". Установите флажок "active [активный]".

Запуск проекта, определение свойств окна Runtime

- **Использование имитатора**
- На закладке "Properties” щелкните на режиме имитации "Inc".
- Введите начальное значение "0" и конечное "100". Установите флажок "active".
- На закладке "Tags” нажмите на кнопку "Start Simulation”. В таблице "Tags” будут отображаться изменяющиеся значения выбранного тега.
- Перейдя к окну режима исполнения, вы сможете увидеть, как имитатор поставляет "реальные " значения кадру.
- Деактивируйте проект WinCC, выбрав команду меню "File" "Activate" WinCC Explorer. Галочка в поле флажка "Activate" исчезнет.

Придание объектам динамических свойств

- Рассмотрим способы, которыми объекту в кадре могут быть приданы динамические свойства. Добавим в кадр возле его левой границы любой стандартный объект. Откроем его свойства, геометрию, позицию по X.
- **Прямая привязка**
- Кликнув правой кнопкой мыши в столбце Dynamic против имени изменяемого свойства, выберем в меню WinCC Tag, в списке тегов - Internal Tags и в правой части окна - тег Position. Клик правой кнопки мыши по столбцу Current позволяет выбрать цикл опроса. Выберем Upon Change - по изменению значения. Флаг в столбце Indirect указывает, что выбранный тег (имя его отображено в столбце Dynamic), содержит строку - имя другого тега, значение которого и будет менять свойство объекта.

Придание объектам динамических свойств

- Активировав проект, используем имитатор для изменения значения тега, определяющего положение объекта на экране.
- Для изменения координаты можно также использовать объект IO Field из панели Smart Objects. Разместив его в кадре, выберем свойство Output/Input и привяжем к тому же тегу Position, настроив цикл обновления Upon Change.
- Теперь при запуске проекта имитацию можно не включать, достаточно вводить с клавиатуры значение в поле ввода. При этом введенное значение соответствует положению левого края объекта на экране в пикселях.

Придание объектам динамических свойств

- **Косвенная адресация**
- Для применения косвенной адресации необходимо использовать внутренний тег типа text 8 bit (ValveName) и задать в качестве его стартового значения имя другого тега, в нашем случае тега Position.
- Применим косвенную адресацию для изменения степени заливки какого-либо объекта. Расположив объект в кадре, отобразим его свойства, выберем Filling, Dynamic Filling и сделаем 2 клика левой кнопкой мыши в столбце Static, установив его в "yes", чтобы разрешить динамическое изменение заливки. Для свойства Fill Level кликнем правой кнопкой мыши в столбце Dynamic, выберем Tag и среди внутренних тегов - тег ValveName. Установим флажок в столбце “Indirect”.

Придание объектам динамических свойств

- Переактивируем проект. Вводя значения от 0 до 100 в поле ввода, будем наблюдать изменение уровня заливки объекта с помощью косвенной адресации.
- **Dynamic Dialog**
- Для динамизации свойств объекта может использоваться динамический диалог (англ. Dynamic dialog). В динамическом диалоге вы определяете выражение, содержащее теги, функции и арифметические операторы. Значение выражения и состояния тегов, используемых в выражении, определяют значение свойства объекта в системе исполнения.

Придание объектам динамических свойств

- Динамический диалог может, например, использоваться для:
  - представления диапазона значений тега с помощью различных цветов;
  - для контроля отдельных битов тега и представления значений бита с помощью цвета или в виде текста;
  - для контроля двоичного тега и представления значений тега с помощью различных цветов или текстов;
  - для контроля состояния тега.
- **Примечание.** При использовании нескольких тегов или операторов производительность динамического диалога резко снижается.

Придание объектам динамических свойств

- **Преобразование в С макрос**
- Макрос , созданный с помощью динамического диалога, отмечается в диалоговом окне "Object Properties [Свойства объекта]" значком красного цвета.
- Вы можете просмотреть код этого макроса, щелкнув правой кнопкой значок в диалоговом окне "Object Properties [Свойства объекта]", а затем выбрав в контекстном меню команду "C Action [С-макрос]". Тогда динамика, созданная с помощью динамического диалога, будет преобразована в С макрос.
- После изменения кода макроса или сохранения макроса этот макрос больше нельзя будет редактировать с помощью динамического диалога.

Придание объектам динамических свойств

- Dynamic dialog используется, чтобы сформулировать выражение, использующее теги, функции и арифметические операторы. Значение выражения используется в качестве значения свойства объекта во время выполнения проекта.
- Возможности по формированию выражения зависят от выбранного типа данных результата.
- **Редактирование триггеров.**
- Если вы не установили триггер, используется событие по умолчанию. Умолчания зависят от содержимого выражения. Если выражение содержит один или более тегов, то в качестве триггера будет использован теговый триггер с циклом опроса, установленным в Graphics Designer.

Придание объектам динамических свойств

- Если выражение не содержит тегов, то в качестве триггера будет использован циклический триггер с циклом опроса, установленным в Graphics Designer.
- События, применяемые в качестве триггеров:
- **Тег.** Событие происходит при изменении тега или циклически, с заданным периодом. Можно применять периоды, заданные пользователем.
- **Стандартный цикл.** Событие происходит циклически, с заданным периодом. Можно применять периоды, заданные пользователем.
- **Цикл кадра.** Событие происходит циклически. Период определяется свойством Update Cycle объекта picture. Содержит опцию централизованного определения циклов всех действий в кадре.

Придание объектам динамических свойств

- **Цикл окна.** Событие происходит циклически. Период определяется свойством Update Cycle объекта Picture Window. Содержит опцию централизованного определения циклов всех действий в окне.
- **Создание выражений для типов результата "Analog", "Boolean", и "Direct".**
- Тип "Analog" позволяет сформировать значение свойства, если результат находится в одном из заданных диапазонов. Тип "Boolean" позволяет задать значение свойства, когда результат выражения есть "Истина" или "Ложь". Тип "Direct" формирует значение свойства непосредственно из значения выражения.
- Теги могут быть введены непосредственно (имя в одиночных кавычках) или с помощью диалога выбора тегов.

Придание объектам динамических свойств

- Если имя введенного тега не найдено, открывается диалог "Missing tags.
- Функции могут быть введены непосредственно или используя кнопку вызова браузера функций. В выражении вы можете использовать любые C функции из вашего проекта, стандартных или внутренних функций, включая функции из Global Script.
- Операторы могут быть введены непосредственно или используя кнопку. Можно использовать операторы сложения, вычитания, умножения, деления.
- Десятичные значения могут быть введены непосредственно. Только точка является разделителем целой и дробной частей.

Придание объектам динамических свойств

- При нажатии кнопки "Check" или "Accept" список значений диапазонов ("Result of Expression /Formula") сортируется по возрастанию верхнего предела диапазона.
- **Задание значений диапазонов для типа "Analog".**
- Можно определить несколько диапазонов значений выражения для указанного типа результатов. Создание нового диапазона производится кнопкой "Add". Новый диапазон значений всегда создается между последним определенным диапазоном и диапазоном "other".
- Можно изменить верхнее значение диапазона, сделав двойной щелчок левой кнопкой мыши в строке диапазона в колонке "to".

Придание объектам динамических свойств

- Можно назначить значение свойства объекта при вхождении значения выражения в указанный диапазон, сделав двойной щелчок левой кнопкой мыши в строке диапазона в колонке свойства объекта.
- **Создание выражений для типа данных "Bit"**
- Теги вводятся, как и для других типов данных.
- Маскирование значащих битов. При работе с 8, 16 и 32х битными тегами можно вызвать диалог для выбора значащих битов. Выбрав значащие биты, можно задать значения свойства, когда эти биты установлены или сброшены.
- Проверка синтаксиса выражения. Если вы нажмете кнопку "Check" или "Accept", то синтаксис выражения будет проверен. Если будет найдена ошибка, вы увидите сообщение об этом.

Придание объектам динамических свойств

- **C-Action**
- С-макросы можно использовать для динамизации свойств объекта и для программирования реакций на события. При динамизации свойств объекта значение свойства объекта определяется по возвращаемому значению С-функции. Используйте С-макросы, если, например, вы хотите обработать в одном макросе несколько входных параметров или выполнить условную конструкцию (if ... then ...). Особенно рекомендуется использовать С-макросы, если необходимо одновременно получить доступ к нескольким тегам. С-макросы следует использовать в том случае, если возможности, предоставляемые соединением с тегом или динамическим диалогом, не достаточны для решения задачи.

Придание объектам динамических свойств

- **С-макрос для программирования реакции на событие**
- Использование макросов для определения реакции на изменение свойства объекта влияет на производительность системы исполнения.
- Событие происходит, если изменяется значение свойства объекта. В этом случае, макрос, связанный с этим событием, начинает выполняться. Когда кадр закрывается, выполнение всех макросов по очереди завершается. Это может привести к большой нагрузке на систему.
- Рассмотрим применение С-макросов для изменения свойств объекта.

Придание объектам динамических свойств

- Разместим в кадре какой-либо стандартный объект, откроем его свойства, Geometry, и кликнем правой кнопкой в столбце Dynamic против свойства Position X. В выпадающем меню выберем C-Action. Выделим и сотрем комментарий в теле функции и вставим оператор return, за которым следует выбранная в правой части окна функция GetTagWord из раздела Internal functions/tag/get. После выбора функции откроется окно выбора ее параметров, которые могут быт тегом, графическим объектом или картинкой. Выберем тег Position, нажмем OK и изменим частоту опроса события, нажав кнопку Trigger - самую правую в панели инструментов редактора макросов. Выберем Tag, укажем тег, значение которого отслеживается и установим частоту опроса Upon Change.

Придание объектам динамических свойств

- Закроем окно редактора кнопкой ОК - макрос будет сохранен и откомпилирован. В случае ошибки она будет описана в нижней части окна и редактор макроса не будет закрыт.
- Добавленный макрос обозначается в столбце Dynamic значком зеленого цвета. Сохраним кадр и перезапустим RunTime. Вводя различные значения в поле ввода, пронаблюдаем изменение положения объекта в кадре.
- **VBS-Action**
- Кроме использования прямого соединения с тегом, С-макросов и тегов, в WinCC можно задавать динамику графических объектов в системе исполнения с помощью VBS-макросов.

Придание объектам динамических свойств

- VBS-макросы следует использовать в следующих случаях
- • если вы хотите в макросе обработать несколько входных параметров,
- • если вы хотите использовать условную конструкцию (if ... then ...),
- • если вы хотите изменить в макросе несколько свойств объекта,
- • если вы хотите обратиться к диалоговым окнам операционной системы, как, например, диалоговому окну выбора файла или цвета
- VBS-макросы создаются в редакторе VBS-макросов в Graphics Designer. Редактор макросов предлагает такой же набор функций, как и редактор VBS в "Global Script".

Придание объектам динамических свойств

- В редакторе Graphics Designer вы можете работать с процедурами, созданными в Global Script.
- Макросы, созданные в Graphics Designer всегда сохраняются вместе в кадром, в котором они были созданы. Документация по созданным VBS-макросам добавляется в проектную документацию Graphics Designer вместе со свойствами всех сконфигурированных объектов. Все VBS-макросы, созданные в кадре, можно посмотреть в WinCC Explorer в диалоговом окне Properties. Это диалоговое окно вызывается из контекстного меню для этого кадра.

Придание объектам динамических свойств

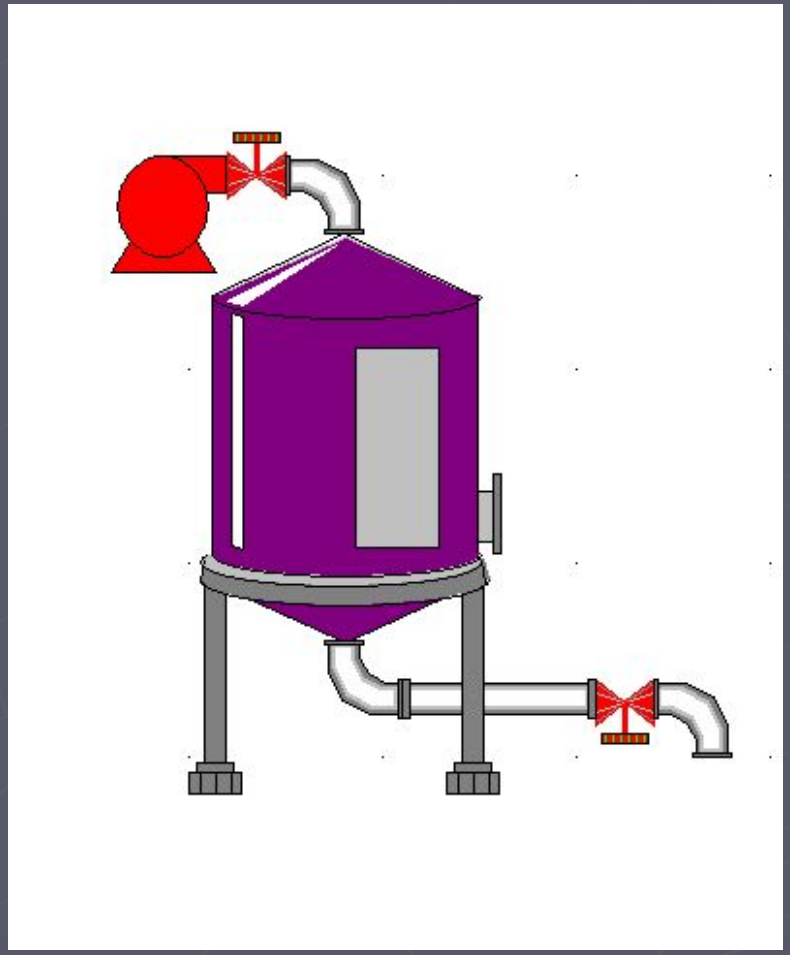
- **VBS-макрос для динамизации свойств объекта**
- VBS-макросы могут использоваться для динамизации свойств объекта. Вы можете определить динамику для свойства объекта в системе исполнения в зависимости от триггера, тега или состояния другого объекта. VBS-макрос следует использовать, если возможности динамизации, предоставляемые соединением с тегом или динамическим диалогом, не достаточны для решения вашей задачи.
- **VBS-макросы для событий**
- VBS-макрос можно использовать для программирования реакции на событие, которое произошло с графическим объектом.

Придание объектам динамических свойств

- Использование макросов для формирования реакции на изменение свойств объекта влияет на производительность системы исполнения.
- Событие происходит, если изменилось значение свойства объекта. В этом случае начинает выполняться макрос, связанный с этим событием. Когда кадр закрывается, то выполнение всех макросов по очереди останавливается. Это может привести к большой системной нагрузке.
- В данном курсе применение VBS-макросов не рассматривается.

Учебный проект

- Создадим новый кадр под именем Start, разместим на нем резервуар, две задвижки и насос.
- **Техническое задание на проект.**
- Цель проекта: разработать подсистему визуализации системы управления водоснабжением, состоящую из:
- Насоса
- Входной задвижки
- Резервуара
- Выходной задвижки.



Учебный проект

- Требования к подсистеме визуализации:
- Насос:
  - Состояние насоса (тег Pump) должно отображаться его цветом;
  - Состояние насоса должно контролироваться по всплывающим подсказкам (элемент Tooltip);
  - Включение/выключение насоса производится выбором состояния из элемента TextList;
  - Включение насоса блокируется при закрытой входной задвижке.

Учебный проект

- Входная/Выходная задвижка:
  - Состояние задвижки (теги Value_1, теги Value_2) должно отображаться ее цветом и значением элемента I/O Field;
  - Состояние задвижек должно контролироваться по всплывающим подсказкам (элемент Tooltip);
  - Открытие/закрытие задвижки производится двумя кнопками (Вкл/Выкл), меняющими свой цвет при нажатии и изменении состояния задвижки.

Учебный проект

- Резервуар:
  - Уровень в резервуаре (тег Level) может увеличиваться только при открытой входной задвижке и включенном насосе
  - Уровень в резервуаре может уменьшаться только при открытой выходной задвижке
  - Уровень в резервуаре отображается визуально уровнем его заливки, меняющей цвет:
    - При уровне от 0 до 75 - синий
    - При уровне от 75 до 90 - желтый
    - При уровне от 91 до 100 - красный

Учебный проект

- При уровне в резервуаре свыше 90 должен визуализироваться элемент StaticText с текстом «Переполнение»;
- Уровень в резервуаре на этапе отладки вводится через элемент I/O Field;
- Уровень в резервуаре на этапе функционирования вычисляется в фоновом действии.

- Кадр процесса должен иметь кнопку «Завершение работы (выход из Runtime), которая активна лишь при выключенном насосе, а при ее нажатии должен отображаться запрос на подтверждение.

Учебный проект

- При уровне в резервуаре свыше 90 должен визуализироваться элемент StaticText с текстом «Переполнение»;
- Уровень в резервуаре на этапе отладки вводится через элемент I/O Field;
- Уровень в резервуаре на этапе функционирования вычисляется в фоновом действии.

- Кадр процесса должен иметь кнопку «Завершение работы (выход из Runtime), которая активна лишь при выключенном насосе, а при ее нажатии должен отображаться запрос на подтверждение.

Учебный проект

- **Создание изображения резервуара с водой**
- На панели меню Graphics Designer щелкните на пункте меню "View" "Library" или на пиктограмме панели инструментов . Будет отображена библиотека объектов (англ. Object Library) со своей собственной панелью инструментов и папками объектов.
- Дважды щелкните на папке "Global Library” и затем - в правой части окна – на папке "Plant Elements". Дважды щелкните на папке "Tanks". Щелкните на пиктограмме панели инструментов библиотеки для предварительного просмотра имеющихся в библиотеке резервуаров.
- **Замечание.** С помощью кнопок и на панели инструментов библиотеки можно изменять размер отображаемых элементов.

Учебный проект

- **Создание изображения трубопровода**
- Используйте требуемые сегменты труб из "Global Library" "Plant Elements" "Pipes - Smart Objects" и разместите их в области рисования. Используйте требуемые клапаны/задвижки из "Global Library" "Plant Elements" "Valves - Smart Objects" и разместите их в области рисования.
- **Замечание.** Объекты типа согнутых сегментов труб не обязательно извлекать из библиотеки каждый раз. Такой объект можно дублировать с помощью команд "Copy", "Paste" и "Duplicate" (меню "Edit" или всплывающее меню). Такой объект также можно создать с помощью команд "Rotate" и "Mirror", меню "Arrange".

Учебный проект

- Добавим также две кнопки из Windows objects, под названиями "Вкл." и "Выкл.", и IO Field из Smart Object. Свяжем поле ввода с тегом Level.
- **Изменение цвета (BackColor) от битовой переменной.**
- Нажатие кнопок должно изменять цвет и состояние (теги Valve_1 и Valve_2) задвижек. Для начала используем Dynamic Dialog (тип данных BOOL) для связи цвета задвижек (зеленый/красный) с их состоянием. Реакцию задвижек на нажатие на кнопки реализуем позже. Убедимся, что изменение стартового значения тега приводит к изменению цвета задвижек.

Учебный проект

- **Изменение уровня в резервуаре (DynFilling), изменение цвета.**
- Будем изменять цвет заливки (Fill Level Color) объекта Tank1 в зависимости от значения тега Level: в диапазоне 0 -75 заливка будет зеленой, 75 - 90 - желтой, 90 - 100 - красной. Вызовем Dynamic Dialog для этого свойства, в качестве триггера выберем стандартный цикл 250 мс, выражение - тег Level, тип данных - Analog, и зададим эти три диапазона.
- Активизируем проект, и, вводя различные значения в поле ввода, убедимся в изменении цвета заливки.
- Для того, чтобы вводимое значение учитывало состояние насоса и задвижек, будем вводить значение тега Level с помощью сценария (IOField/Properties/OutputValue):

Учебный проект

```
double new, cur;
new=GetInputValueDouble(lpszPictureName,"IOField1");
cur=GetTagDouble("Level");  //Return-Type: double
if (new > cur)
{
  if(GetTagBit("Pump") &&GetTagBit("Valve_1"))
    {
      SetTagDouble("Level",new); return new;
     }
}
  else if (GetTagBit("Valve_2"))
  {
   SetTagDouble("Level",new); return new;
}
return cur;
```

Учебный проект

- **Индикация состояния текстом (StaticText).**
- Добавим Smart Object TextList, в котором будем выводить текст "Вкл." и "Выкл.", относящийся к насосу (Pump). Свяжем список с тегом Pump через C-Action (функция tag/set/SetTagBit), а у насоса (тоже через C-Action, функция graphics/set/SetBackColor)будем менять цвет в зависимости от значения тега. Активизируем проект, и, выбирая значения из списка, убедимся в изменении цвета заливки(а значит, и состояния) насоса.

Учебный проект

- **Динамическое отображение объектов (Display).**
- Добавим в кадр стандартный объект StaticText с текстом "ВНИМАНИЕ!". При уровне в резервуаре выше 90 отобразим его на экране. Изначально Свойства/Styles/Weight/0 и FillPattern - Transparent и Miscellaneous/Display/No. У резервуара берем свойства/Events/FillLevel и вставляем C-Action:
- If (value >= 90) {/Internal/Graphics/Set/Misc/SetVisible
- У этой функции4 аргумента: первый не меняем, в качестве второго выберем наш кадр и объект StaticText, третий аргумент - то же самое, но укажем Property, имя же свойства не задаем, если оно входит в имя функции. Четвертый аргумент - значение TRUE.
- Else - все то же самое, но 4й аргумент - FALSE.

Учебный проект

- Заметим, что если С-макросы содержат большое число операций, и вызываются часто, то их лучше оформить в виде библиотеки (dll), и подключать, а не интерпретировать, как макрос.
- **Обработка событий.**
- В свойствах всех объектов есть закладка "Events", среди которых нас интересует "Mouse". Возможно отслеживание пяти событий мыши.
- **Direct Connection.**
- Используем клик мышью на кнопке для включения/выключения насосов. Для этого будем обрабатывать "Mouse Action". Выделив это событие, вызовем меню правой кнопки в столбце "Action" и выберем "Direct Connection".

Учебный проект

- Из источника в приемник можно передать константу (имя кадра, объекта), введя ее или выбрав объект, свойство объекта или тег, либо напрямую, либо косвенно. У приемника есть свойства: текущее окно, объект в кадре, переменная.
- **Контроль по всплывающим подсказкам.**
- Всплывающие подсказки (Tooltip) задаются в свойствах объекта Miscellaneous/Tooltip Text и отображаются в RunTime при наведении курсора мыши на объект. Их можно менять динамически. Рассмотрим пример - отображение состояния задвижки.
- Привяжем значение подсказки динамически через C-Action к состоянию задвижки. Будем анализировать состояние тега Valve_1 и изменять текст подсказки:

Учебный проект

- if (GetTagBit("Valve_1")) return "Включено";
- else return "Выключено";
- В качестве триггера, вызывающего C-Action, выберем состояние тега с циклом опроса Upon Change.
- **Контроль по виду кнопку.**
- По нажатию на кнопку можно изменить ее внешний вид. Как правило, меняют цвет заливки и текст, но можно изменить фоновую картинку: Properties/Miscellaneous/Picture Status On - когда она нажата, и Picture Status Off - когда не нажата. Изменим цвета кнопок, управляющих задвижками - когда они открыты, то кнопка "Выкл." имеет красный фон, кнопка "Вкл." - серый. Когда задвижки закрыты, кнопка "Вкл." будет зеленой, кнопка "Выкл." - серой.

Учебный проект

- Для этого свяжем свойство Colors/BackGround с тегом, используя Dynamic Dialog, результат Boolean: если тег равен TRUE, то цвет - по умолчанию, а если FALSE - зеленый. Для кнопки "Выкл.": если тег равен TRUE, то цвет - красный, а если FALSE - по умолчанию.
- **Кнопка «Выход из Runtime». Использование мастера динамики.**
- Рисуем кнопку с текстом "Выход". В панели инструментов есть мастер динамики (Dynamic Wizard), с его помощью выберем Exit WinCC Runtime. К другим объектам (не объектам управления) мастер динамики не применим, но клик по ним можно обработать с помощью C-Action, вызвав соответствующую функцию из Internal/wincc/system.

Учебный проект

- **Запрос подтверждения на событие (MessageBox).**
- При обработке кликов мыши можно вставить в C-Action код запроса на подтверждение, при положительном ответе на который и выполняется действие. Например, запросим подтверждение на выход из Runtime:
- `HWND Handle;` //Описание переменной - хендла окна Runtime
- `Handle=FindWindow("PDLRTisAliveAndWaitsForYou","WinCC-Runtime- ");`//Поиск окна
- `If (MessageBox(Handle, "Вы уверены?", "Внимание", MB_YESNO |MB_ICONREQUESTION |MB_SETFOREGROUND) == IDNO) return;`
- `DeactivateRTProject();` // деактивация Runtime

Учебный проект

- **Использование свойства Operator Control Enable, функции SetOperation.**
- У всех элементов управления есть свойство Operator Control Enable, указывающее, доступен ли элемент в Runtime. Рассмотрим, как сделать объект доступным с помощью функции. Например, обусловим возможность выхода из Runtime лишь при выключенном двигателе. Изначально ее свойство Operator Control Enable - в состоянии "NO", объект виден, но не активен. Состояние двигателя (тег Pump) отображается его цветлм. При возникновении события смены цвета (Events/Group/Property Topics/Colors/BackgroundColor/ Change) вызовем C-Action, которая проверит состояние тега и сделает доступной кнопку выхода из Runtime:

Учебный проект

- // вставка функции Internal/Get/Tag/GetTagBit/Pump
- `if (!GetTagBit("Pump"))`
- // вставка функции Internal/Graphics/Set/Misc/SetOperation, выбор кадра, объекта, ввод TRUE
- `SetOperation("Start1.Pdl","Button1",TRUE);`
- // в противном случае
- `else SetOperation("Start1.Pdl","Button1",FALSE);`

Учебный проект

- **Текстовый список TextList.**
- Относится к Smart Objects. Имеет особенности по созданию элементов списка и направлению передачи информации. При его создании открывается Configuration Dialog, в котором задается привязка к тегу, частота опроса, тип - для вывода, ввода или того и другого, а также параметры шрифта.
- Для ввода элементов списка используется свойство Output/Input - Assignments. Два клика по нему открывают окно, в котором можно задать, при каком значении тега (диапазоне значений, превышении значения, понижении значения ниже указанного уровня) какой текст будет выводиться.

Учебный проект

- Существующие назначения можно удалять, передвигать выше и ниже, а вновь заданные добавлять. Кроме того, свойство Number of visible Lines указывает число отображаемых строк списка.
- Применим TextList для отображения состояния двигателя. Свяжем его с тегом Pump с интервалом опроса Upon Change, назначим значению 0 текст "Выкл.", а значению 1 - "Вкл.". Установим число отображаемых строк: 1. Сохраним кадр, перезапустим Runtime и убедимся, что значение списка меняется при изменении состояния двигателя.

Учебный проект

- **Windows-объекты.**
- К ним, кроме кнопок, относятся флажки, радиокнопки, круглые кнопки и движки (sliders). Обсудим флажки и радиокнопки, ибо их конфигурирование имеет некоторые особенности.
- У флажков с свойствах/Geometry/Number of boxes модно задать количество флажков. Текст возле флажков задается для каждого отдельно, для чего нужно изменить свойство Font/Index и для каждого значения индекса изменить текст. Свойство Output/Input/Selected boxes устанавливает, какие флажки будут показаны, как отмеченные по умолчанию.

Учебный проект

- Для анализа установленных флажков рекомендуется связать объект с тегом типа unsigned 8, 16, 32 bit и анализировать его значение с помощью Dynamic Dialog, тип результата – Bit, или с помощью сценария, выделяющего отдельные биты из значения тега.
- Точно так же организуется работа с радиокнопками.
- **Application Window – Вывод диагностической информации.**
- Для отладки и диагностики ошибок можно использовать Application Window/Global Script/GSC Diagnostics, куда можно печатать из макроса: `printf("Something wrong!\n")`.

Объект Picture Window

- Данный объект может содержать в себе кадр, что полезно для уменьшения количества объектов управления на мнемосхеме при отображении их в отдельном кадре и при разбиении мнемосхемы на отдельные кадры. Свойства объекта позволяют изменять имя кадра, в нем отображаемого, а также управлять видимостью объекта в текущем кадре. Рассмотрим пример, в котором органы управления задвижкой (кнопки "Вкл.", "Выкл.", поле ввода/вывода, отображающее состояние задвижки и текстовое поле с именем задвижки, а также кнопка закрытия кадра) вынесены в отдельный кадр, отображаемый в объекте Picture Window при клике на изображении задвижки.

Объект Picture Window

- **Окно управления задвижкой с кнопкой «Отмена»**
- Скопируем (выделив при нажатой кнопке Shift) кнопки управления задвижкой. Создадим новый кадр с именем Valve. В его верхний левый угол вставим скопированные кнопки, создадим текстовое поле с именем задвижки (задано статически) и окно ввода/вывода, связанное с тегом Valve_1. Добавим кнопку с текстом "Выход", для ее событий Mouse Action с помошью Direct Connect передадим константу 0 в объект Current Window, свойство Display. Уменьшим размер кадра до минимально допустимых размеров.

Объект Picture Window

- В основном кадре с мнемосхемой вставим объект Picture Window, зададим его свойства: Display - No, Foreground - Yes, AdoptSize - Yes, PictureName - имя нового кадра с элементами управления. Можно задать Heading - текст заголовка окна.
- Для отображения этого объекта настроим событие клика по задвижке: для Mouse Action с помощью Direct Connect передадим константу 1 в объект PictureWindow1, свойство Display.
- Аналогичное окно можно создать для каждого объекта мнемосхемы, но лучше использовать одно окно для всех однотипных объектов.

Объект Picture Window

- **Одно окно для двух объектов.** Для использования одного окна при управлении несколькими объектами необходимо использовать тег типа text 8 bit, в котором будет храниться имя тега, содержащего состояние объекта. В нашем случае в теге ValveName будет записано Valve_1 или Valve_2, в зависимости от того, какой задвижкой мы собираемся управлять. Обращение к переменной, имя которой хранится в другой переменной, называют косвенной адресацией. Выделим задвижку и посмотрим ее события. Если при клике мыши выполнялось Direct Connection, то выбор C_Action приведет к его преобразованию в скрипт. В сценарии нужно выбрать функцию Internal/teg/set/SetTagChar, в которой в тег ValveName нужно записать Valve_1 или Valve_2.

Объект Picture Window

- Далее необходимо активировать кадр с элементами управления, то есть выбрать функцию Internal/ graphic/set/misc/SetVisible. Выбираем кадр, на котором находится PictureWindow, выбираем сам объект и указываем состояние TRUE. Сценарий будет выглядеть так:
- `SetTagChar("ValveName","Valve_2");`
- `SetVisible("Start1.Pdl","PictureWindow1",TRUE);`
- Аналогично поступаем со второй задвижкой.
- Далее необходимо настроить элементы управления в кадре управления Valve. Открыв его, берем текстовое поле и связываем его напрямую с тегом ValveName с обновлением по изменению и **отмечаем checkbox в столбце Indirect.** Аналогично поступаем с полем ввода/вывода.

Объект Picture Window

- Затем изменим скрипты установки состояния задвижки по кликам на кнопки "Вкл." и "Выкл.". В них надо вместо имени изменяемого тега вставить вызов функции получения значения текстового тега: Internal/teg/get/GetTagChar. Скрипт для кнопки "Вкл." будет выглядеть так:
- `SetTagBit(GetTagChar("ValveName"),1);`
- Если требуется отображать имя управляемого элемента мнемосхемы не в текстовом окне, а в заголовке PictureWindow, то в сценарий открытия окна добавим установку свойства Capture Text через функцию internal/graphics/set/property/SetPropChar, которой в качестве параметров передаем имя кадра, объекта, свойства и его значения. В качестве значения можно использовать результат, возвращаемый функцией (GetTagChar("ValveName").

Принципы построения систем визуализации

- Традиционно в мнемосхемах окно разбивается на три части: обзорная область (10%, видно всегда), в которой расположены окно сообщений, кнопки навигации по мнемосхеме, обобщенные индикаторы состояния, системное время, текущий пользователь, состояние связи с контроллерами, рабочая область (80%, может меняться), в которой расположены мнемосхемы, экраны диагностики, графики и таблицы, область дополнительной клавиатуры (10%, видимость зависит от рабочей области), в которой содержатся кнопки вызова диагностики, сообщений, графиков, выхода из среды исполнения и др.

Принципы построения систем визуализации

- Для реализации этого подхода создадим новый кадр (main.pdl) размером 1024x768, в него вставим 3 объекта Picture Window размерами 1024x100, 1024x600, 1024x68. Для подгонки отображаемых кадров под размер объекта в свойствах Picture Window установим Adopt Picture в Yes. В свойство Picture Name верхнего объекта введем "menu.pdl", среднего - имя кадра с мнемосхемой.
- **Смена изображений в главном окне**
- Создадим новый кадр menu.pdl размером 1024x100. В нем нарисуем кнопки "Мнемосхема", "Графики". На их нажатие повесим скрипты, изменяющие свойство Picture Name объекта Picture Window2 кадра main.pdl на имя кадра с мнемосхемой (1 кнопка) и кадр graphics.pdl (2 кнопка), который (размером 1024x600) еще предстоит создать.

Принципы построения систем визуализации

- В тех же сценариях будем менять цвет кнопки при ее нажатии.
- Экранная клавиатура подключается в свойствах компьютера, на закладке Runtime (Enable monitor keyboard).
- **Объекты пользователя**
- Очень часто некоторые совокупности объектов приходится отображать вместе, например, имя тега и его значение, имя объекта и параметры его настройки, такие как AL, AH, WL, WH, окно сообщения и кнопка его квитирования, и другие. Такие совокупности объектов удобно оформить как единый объект, занеся который в библиотеку, мы получим возможность создавать его одним действием. Такие составные объекты называют объектами пользователя.

Объекты пользователя

- Разместим в любом кадре Static Text (для отображения имени тега) и I/O Field (для отображения его значения). Выделим их с нажатой кнопкой Shift и по клику правой кнопки мыши выберем в меню Customize object/Create. Через свойства объекта переименуем его, скажем, в Paramon, и из меню правой кнопки мыши вызовем Configuration Dialog. В нем свойства составных частей объекта можно перетащить в свойства всего объекта. У Static Text возьмем Text и BackColor, у I/O Field - Output Value. Для каждого взятого свойства изменим Attribute Name. Зададим также Name of Property и переименуем группу свойств Userdefined в Svoistva.
- Аналогично можно добавить события объекта. Нам хватит Mouse Action на его обоих составных частях.
- Готовый объект можно перетащить в библиотеку объектов, вызываемую кнопкой тулбара Display Library. Если положить объект в Global Library, то он будет доступен во всех проектах на этом компьютере, а если в Project Library, то его, вместе с проектом, можно переместить на другой компьютер.

Структурные типы данных

- **Назначение структур**
- Для работы с объектами пользователя удобны структуры. Как и в классическом Си, это пользовательский тип данных, включающий в себя произвольное количество полей базовых типов. В структурной переменной можно описать все параметры какого-либо объекта, включая его имя.
- Для создания структуры откроем WinCC Explorer, выберем Structure Type и из меню правой кнопки мыши - New. Выберем вид переменных - внутренние, зададим имя нового типа - paramon и добавим поля - text 8 bit (Name), float (Value), DWORD (BackColor).
- Далее надо создать переменные этого нового типа: Internal Tag - New - Data type - Paramon. Получим СРАЗУ все теги, входящие в структуру!

Структурные типы данных

- **Связь структуры с объектом**
- Выделяем объект типа Paramon и в его свойствах устанавливаем динамическую связь с полями структурного тега.
- Привязка структурной переменной к экземпляру объекта
- Чтобы не делать так для каждого объекта этого типа, скопируем объект и в палитре Dynamic Wizard выберем закладку Standard, а там - Link a prototype to a structure и два клика левой кнопкой. В диалоге вместо привязки к существующей структуре создадим новую, того же типа. Заметим, что часть названия структуры, до точки, отделяющей имя поля, называется TagPrefix.

Структурные типы данных

- **Графическое окно настройки параметров. Свойство TagPrefix**
- Создадим графическое окно настройки параметров пользовательского объекта, отображаемое по клику на этом объекте. Оно похоже на окно управления задвижками, но вместо косвенной адресации установим свойство TagPrefix объекта PictureWindow с помощью функции internal/graphics/set/misc/SetTagPrefix. Лишь после этого объект можно делать видимым: internal/graphics/set/misc/SetVisible.

Структурные типы данных

- При работе со структурами есть ряд ограничений: если структурная переменная уже создана, то изменить структуру нельзя! Надо вначале удалить все переменные типа этой структуры. А чтобы не потерять структурные теги, их можно скопировать, удалить переменную, изменить структуру и вставить теги, а потом - переименовать.

Полезные примеры

- **Запуск приложений в одном экземпляре**
- Для этого используются стандартные функции Windows для поиска окна с указанным заголовком и отображения его на переднем плане. Если окно не найдено, то запускается необходимое приложение с указанием полного пути к нему и, при необходимости, с параметрами командной строки (полным именем открываемого файла). Если по каким-то причинам приложение не может быть запущено, то формируется сообщение об ошибке:

Полезные примеры

- `HWND Handle;`
- `Handle=FindWindow("XLMAIN","Microsoft Excel - test");`
- `If (Handle) { SetForegroundWindow(Handle); return;}`
- `Handle=FindWindowNULL,"WinCC-Runtime -");`
- `If (ProgramExecute("C:\\Program Files\\Microsoft Office\\Office 10\\Excel.exe d:\\test.xls") <= 31)`
- `MessageBox(Handle,"При запуске Excel возникла ошибка", "Ошибка",MB_OK|MB_ICONERROR);`

Полезные примеры

- **Сворачивание окна WinCC Runtime**
- Для этого используются стандартные функции Windows для поиска окна с указанным заголовком и отображения его в одном из указанных состояний, в нашем случае - в свернутом (минимизированном) виде:
- `HWND Handle;`
- `Handle=FindWindow("PDLRTisAliveAndWaitsForYou","WinCC-Runtime -");`
- `ShowWindow(Handle, SW_MINIMIZE);`

Полезные примеры

- **Завершение и принудительное закрытие порожденных программ.**
- Различается необходимостью сохранения измененных данных (при завершении) и ликвидацией процесса, породившего окно.
- `void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)`
- `{`
- `HWND hwnd;`
- `#pragma code ("Kernel32.dll")`
- `VOID Sleep(`
- `  DWORD dwMilliseconds  // sleep time`
- `);`

Полезные примеры

```
BOOL TerminateProcess(
  HANDLE hProcess,// handle to the process
  UINT uExitCode//exit code for the process
);
HANDLE OpenProcess(
  DWORD dwDesiredAccess,  // access flag
   BOOL bInheritHandle,//handle inheritance option
  DWORD dwProcessId  // process identifier
);
#pragma code()
```

Полезные примеры

```
#pragma code("User32.dll")
BOOL PostMessage(
  HWND hWnd,//handle to destination window
  UINT Msg,      // message
  WPARAM wParam,// first message parameter
  LPARAM lParam// second message parameter
);
DWORD GetWindowThreadProcessId(
  HWND hWnd,              // handle to window
  LPDWORD lpdwProcessId //pid
);
#pragma code()

DWORD pid; //ID процесса
HANDLE hProcess;//HANDLE процесса
```

Полезные примеры

```
#pragma code("User32.dll")
BOOL PostMessage(
  HWND hWnd,//handle to destination window
  UINT Msg,      // message
  WPARAM wParam,// first message parameter
  LPARAM lParam// second message parameter
);
DWORD GetWindowThreadProcessId(
  HWND hWnd,             // handle to window
  LPDWORD lpdwProcessId //pid
);
#pragma code()
```

Полезные примеры

- DWORD pid; //ID процесса
- HANDLE hProcess;//HANDLE процесса
- DWORD PROCESS_TERMINATE=1;//Флаг доступа ProgramExecute("C:\\winnt\\notepad.exe");
- hwnd=FindWindow(NULL,"Безымянный - Блокнот");
- ShowWindow(hwnd,SW_MAXIMIZE); //Развернули Sleep(1000);// подождать
- //PostMessage(hwnd,WM_CLOSE,0,0);
- GetWindowThreadProcessId(hwnd, &pid);
- hProcess=OpenProcess(PROCESS_TERMINATE, FALSE, pid);//Получаем HANDLE процесса
- printf("PID=%ld\r\n",hProcess);
- TerminateProcess(hProcess,0); }

Полезные примеры

- **Определение ClassName выполняемой в ОС программы.**
- В функции FindWindow при отсутствии заголовка окна или если в заголовок включено имя обрабатываемого файла, в качестве первого параметра удобно использовать ClassName приложения, породившего окно:
- HWND hwnd;
- char name[20];
- LPTSTR lpclassname=name;
- #pragma code("user32.dll")
- int GetClassName(HWND hwnd, LPTSTR lpclassname, int nmaxcount);
- #pragma code()

Полезные примеры

- ProgramExecute("C:\\Winnt\\System32\\mspaint.exe");
- hwnd=FindWindow(NULL,"Безымянный - Paint");
- GetClassName(hwnd,lpclassname,20);
- printf("Handle=%lx, ClassName=%s\r\n",hwnd,name);
- **Воспроизведение звуковых и видеофайлов.**
- Для этого порождается в свернутом виде соответствующее приложение, завершаемое по истечении заранее известного времени воспроизведения:
- HWND hwnd;
- #pragma code ("Kernel32.dll")
- VOID Sleep(DWORD dwMilliseconds);
- #pragma code()

Полезные примеры

- #pragma code("User32.dll")
- BOOL PostMessage(
- HWND hWnd, // handle to destination window
- UINT Msg, // message
- WPARAM wParam,
- LPARAM lParam);
- #pragma code()
- ProgramExecute("C:\\Program Files\\Windows Media Player\\mplayer2.exe C:\\winnt\\Media\\Выход из Windows.wav");
- hwnd=FindWindow(“Media Player 2”, NULL);
- ShowWindow(hwnd, SW_MINIMIZE);
- Sleep(4000);// подождать
- PostMessage(hwnd, WM_CLOSE, 0, 0);