

SIEMENS

WinCC

Руководство по конфигурации

Том 1

Данное руководство является частью пакета документации
с порядковым номером:

6AV6392-1CA05-0AB0

C79000-G8276-C157-01

Выпуск: сентябрь 1999

WinCC, SIMATIC, SINEC, STEP являются торговыми марками Siemens.

Другие использованные в данном руководстве названия могут быть торговыми марками; их авторские права могут быть нарушены в случае их использования третьими сторонами в личных целях.

(Воспроизведение, передача и использование данного документа или его содержания не разрешается без получения на то документально подтвержденных полномочий. Нарушение этих требований влечет за собой возмещение ущерба. Мы сохраняем за собой все права, в частности в случаях выдачи патента и регистрации товарных образцов.)

(Содержание данного руководства было проверено на соответствие программным и аппаратным средствам. Тем не менее, возможны расхождения, в связи с чем мы не можем гарантировать полное соответствие. Данные, приведенные в настоящем документе, регулярно подвергаются проверке и необходимые исправления вносятся в последующие издания. Мы будем благодарны за все предложения, направленные на улучшение руководства.)

© Siemens AG 1994 - 1999 Все права защищены

Мы сохраняем за собой право на внесение технических изменений

C79000-G8276-C157
Напечатано в ФРГ

Siemens Aktiengesellschaft

Содержание

1	РУКОВОДСТВО ПО КОНФИГУРАЦИИ	1-1
1.1	Руководство по конфигурации - замечания относительно структуры и применения	1-2
2	WinCC – Общая информация	2-1
2.1	WinCC – Основные понятия.....	2-2
2.1.1	Интерфейсы WinCC	2-3
2.2	WinCC – разъяснения терминов.....	2-5
3	Конфигурация – Общие предметы	3-1
3.1	Перед началом проекта	3-2
3.2	Детальная спецификация	3-3
3.2.1	Спецификация: Название проекта WinCC	3-4
3.2.2	Спецификация: Название тегов.....	3-5
3.2.3	Спецификация: Название кадров	3-7
3.2.4	Спецификация: Сценарии и процедуры	3-9
3.2.5	Спецификации: Пользовательский интерфейс.....	3-10
3.2.6	Спецификация: Концепция управления	3-15
3.2.7	Спецификация: Определение цвета	3-17
3.2.8	Спецификация: Циклы обновления	3-18
3.2.9	Спецификация: Права пользователя	3-19
3.2.10	Спецификация: Alarming (регистрация аварийных сообщений).....	3-20
3.2.11	Спецификация: Для реализации	3-21
3.3	Особенности конфигурации WinCC	3-22
3.3.1	Циклы обновления – Где и как устанавливаются ...	3-23
3.3.1.1	Обновление на кадре.....	3-23
3.3.1.2	Типы циклов обновления	3-25
3.3.1.3	Значение циклов обновления	3-27
3.3.1.4	Информация относительно применения циклов обновления.....	3-28
3.3.1.5	Исполнение фоновых сценариев (Global Script)	3-36
3.3.2	Добавление динамики в WinCC.....	3-40
3.3.2.1	Превращение свойств в динамические	3-40
3.3.2.2	Превращение событий в динамические	3-41
3.3.2.3	Типы динамизации для объектов	3-41
3.3.3	Системное окружение WinCC	3-45
3.3.3.1	Структура каталогов WinCC.....	3-45
3.3.4	Среда проекта WinCC	3-48
3.3.4.1	WinCC проект – Структура папок.....	3-48
3.3.5	Автоматический запуск проекта в WinCC.....	3-51
3.3.6	Координированное завершение работы WinCC	3-55
3.3.6.1	Замечания по установке UPS	3-55

3.3.7	Резервное копирование данных.....	3-57
3.3.8	Перенос резервной копии проекта WinCC на другой компьютер	3-59
3.3.9	Повторное использование – Передача частей проекта в новый или существующий проект	3-62
3.3.9.1	Передача кадров.....	3-63
3.3.9.2	Передача символов и побитовых изображений.....	3-65
3.3.9.3	Передача библиотеки проекта (с ранее skonфигурированными символами и модифицированными объектами).....	3-66
3.3.9.4	Передача процедур	3-68
3.3.9.5	Передача тегов	3-70
3.3.9.6	Передача многоязыковых текстов (из кадров, в сообщения).....	3-78
3.3.9.7	Передача сообщений.....	3-79
3.3.10	Online конфигурация (в режиме исполнения) - замечания, ограничения	3-115
4	Курс Си для WinCC.....	4-1
4.1	Среда разработки сценариев Си.....	4-3
4.1.1	Редактор процедур и графический редактор.....	4-4
4.1.2	Редактор глобальных сценариев	4-12
4.2	Переменные.....	4-20
4.2.1	Пример 1 — Типы данных Си (целые)	4-22
4.2.2	Пример 2 — Пользовательские типы данных (целые).....	4-24
4.2.3	Пример 3 — Теги WinCC (целые).....	4-26
4.2.4	Пример 4 — Типы данных Си (числа с плавающей точкой).....	4-28
4.2.5	Пример 5 — Теги WinCC (числа с плавающей точкой).....	4-29
4.2.6	Пример 6 — Статические и внешние переменные	4-30
4.3	Операторы и математические функции в Си.....	4-32
4.3.1	Пример 1 — Основные математические процедуры.....	4-34
4.3.2	Пример 2 — Операторы инкремента и декремента	4-35
4.3.3	Пример 3 — Битовые операции	4-37
4.3.4	Пример 4 — Перестановка старшего и младшего байтов	4-39
4.3.5	Пример 5 — Математические функции	4-40
4.4	Указатели	4-42
4.4.1	Пример 1 — Указатели	4-44
4.4.2	Пример 2 — Массивы	4-46
4.4.3	Пример 3 — Указатели и массивы	4-47
4.4.4	Пример 4 — Строки	4-49
4.4.5	Пример 5 — Текстовые теги WinCC.....	4-50

4.5	Циклы и условные выражения.....	4-51
4.5.1	Пример 1 — Цикл while	4-53
4.5.2	Пример 2 — Цикл do – while	4-54
4.5.3	Пример 3 — Цикл for.....	4-55
4.5.4	Пример 4 — Бесконечные циклы.....	4-56
4.5.5	Пример 5 — Выражение if–else	4-58
4.5.6	Пример 6 — Выражение switch–case.....	4-59
4.6	Функции	4-60
4.6.1	Пример 1 — Передача параметров по значению ...	4-61
4.6.2	Пример 2 — Передача параметров по адресу.....	4-63
4.6.3	Запись в переданный диапазон адресов	4-65
4.6.4	Возврат результата по указателю	4-67
4.7	Структуры.....	4-70
4.7.1	Пример 1 — Структурная переменная	4-71
4.7.2	Пример 2 — Определение пользовательского типа данных	4-72
4.7.3	Пример 3 — Структурный тип WinCC.....	4-74
4.7.4	Пример 4 — Функция для чтения структурного типа WinCC.....	4-76
4.8	Программный интерфейс WinCC (API).....	4-80
4.8.1	Пример 1 — Изменение свойств с помощью функций RT	4-82
4.8.2	Пример 2 — Создание связи с тегом с помощью функции RT	4-84
4.8.3	Пример 3 — Создание нового объекта при помощи функций CS	4-86
4.8.4	Пример 4 — Изменение свойств при помощи функции CS	4-89
4.8.5	Пример 5 — Создание связи с тегом с помощью функции CS	4-92
4.8.6	Пример 6 — Перечисление объектов с помощью функции CS	4-95
4.9	Среда проекта.....	4-98
4.9.1	Пример 1 — Определение файла проекта.....	4-99
4.9.2	Пример 2 — Определение пути проекта	4-101
4.9.3	Пример 3 — Определение пути проекта с помощью функции проекта	4-103
4.9.4	Пример 4 — Определение инсталляционного каталога.....	4-105
4.9.5	Пример 5 — Определение имени компьютера	4-107
4.9.6	Пример 6 — Определение имени пользователя	4-108
4.10	Windows API	4-109
4.10.1	Пример 1 — Установка свойств окна.....	4-110
4.10.2	Пример 2 — Считывание системного времени	4-111
4.10.3	Пример 3 — Воспроизведение звуковых файлов ...	4-112
4.10.4	Пример 4 — Запуск программы.....	4-114

4.11	Стандартные диалоги.....	4-115
4.11.1	Пример 1 — Переключение языка	4-116
4.11.2	Пример 2 — Выбор тега	4-118
4.11.3	Пример 3 — Диалоговое окно сообщения об ошибке	4-120
4.11.4	Пример 4 — Диалоговое окно вопроса	4-121
4.11.5	Пример 5 — Стандартное диалоговое окно выбора файла.....	4-123
4.12	Файлы.....	4-125
4.12.1	Пример 1 — Запись данных	4-127
4.12.2	Пример 2 — Чтение данных	4-128
4.12.3	Пример 3 — Формирование отчета.....	4-129
4.13	Динамический мастер.....	4-131
4.13.1	Создание функций динамического мастера	4-132
4.13.2	Структура функции динамического мастера.....	4-134
5	Приложение.....	5-1
5.1	Полезные советы.....	5-2
5.1.1	Форматированный ввод/вывод в поля ввода/вывода	5-3
5.1.2	Специфичные для объекта процедуры на открытом кадре	5-4
5.1.3	WinCC Score.....	5-5
5.1.4	Доступ к базе данных.....	5-6
5.1.4.1	Доступ к базе данных из MS Excel/MS Query.....	5-6
5.1.4.2	Доступ к базе данных из MS Access.....	5-10
5.1.4.3	Доступ к базе данных из ISQL	5-11
5.1.4.4	Доступ к базе данных из WinCC Scope	5-13
5.1.4.5	Экспорт данных из базы данных с помощью процедур Си.....	5-14
5.1.4.6	Выборки из базы данных	5-16
5.1.5	Последовательная связь.....	5-17
5.1.6	Цветовая таблица.....	5-18
5.2	Документация на аварийную систему S5.....	5-19
5.2.1	Перечень программных блоков.....	5-20
5.2.2	Требования к аппаратному обеспечению	5-21
5.2.3	Внедрение аварийной системы S5 в программу SIMATIC S5	5-22
5.2.3.1	Структура блока данных смещения	5-25
5.2.3.2	Основной номер сообщения.....	5-26
5.2.3.3	Смещение сообщения/состояния сигнала сообщений.....	5-27
5.2.3.4	Блок статуса сигнала.....	5-28
5.2.3.5	Адрес последнего блока статуса сигнала	5-29
5.2.3.6	Состояния сигнала.....	5-30

5.2.3.7	Состояния бездействия	5-30
5.2.3.8	Биты подтверждения	5-31
5.2.3.9	Флаги установки фронтов	5-31
5.2.3.10	Структура блока данных параметров	5-31
5.2.3.11	Структура блока сообщений	5-33
5.2.3.12	Номер сообщения	5-33
5.2.3.13	Состояние сообщения	5-34
5.2.3.14	Временная метка	5-34
5.2.3.15	Теги процесса	5-34
5.2.3.16	Номер задачи / идентификатор процесса	5-34
5.2.3.17	Зарезервировано	5-34
5.2.3.18	Создание блока сообщений	5-34
5.2.3.19	Внутренний FIFO буфер (кольцевой)	5-35
5.2.3.20	Почтовый ящик отправки - передача данных на верхний уровень системы WinCC	5-35
5.2.4	Описание интерфейса	5-37
5.2.4.1	Системный блок данных 80	5-37
5.2.4.2	Блок данных смещения	5-37
5.2.4.3	Блок данных параметров	5-37
5.2.4.4	Почтовый ящик отправки (Send Mailbox)/ почтовый ящик передачи (Transfer Mailbox)	5-37
5.2.5	Назначение параметров для аварийной системы S5 / системного DB 80	5-38
5.2.6	Пример конфигурации для аварийной системы S5	5-45
5.2.6.1	Параметризация DB 80	5-45
5.2.6.2	Установка блоков данных	5-46
5.2.6.3	Инициализация блоков данных смещений	5-46
5.2.7	Документация на командные блоки SIMATIC S5	5-51
5.2.7.1	Перечень программных блоков	5-51
5.2.7.2	Требования к аппаратному обеспечению	5-52
5.2.7.3	Параметры вызова FB 87: EXECUTE	5-52
5.2.8	Описание интерфейса	5-53
5.2.8.1	Пример конфигурации командных блоков S5	5-55
5.2.9	Назначение и функции синхронизации времени S5	5-56
5.2.9.1	Перечень программных блоков	5-56
5.2.9.2	Требования к аппаратному обеспечению	5-56
5.2.10	Параметры вызова FB 86: MESS:CLOCK	5-57
5.2.11	Форматы даты и времени	5-59
5.2.11.1	Область данных часов ЦПУ 944, ЦПУ 945	5-60
5.2.11.2	Область данных часов ЦПУ 928В, ЦПУ 948	5-61
5.2.11.3	Область данных часов ЦПУ 946, ЦПУ 947	5-62
5.2.11.4	Часовые форматы данных для блоков сообщений	5-63
5.2.12	Описание интерфейса	5-64
5.2.13	Взаимодействие с аварийной системой WinCC	5-65

5.3	Интерфейс с динамической библиотеки формата (Format DLL) для системы регистрации аварийных сообщений (Alarm Logging) и системы регистрации тегов (Tag Logging).....	5-66
5.3.1	Разделяемые интерфейсы для системы регистрации аварийных сообщений (Alarm Logging) и системы регистрации тегов (Tag Logging).....	5-67
5.3.2	Дополнения, специфичные для системы регистрации тегов (Tag Logging)	5-69
5.3.3	API функции динамической библиотеки формата (Format DLL) WinCC.....	5-70
5.3.3.1	Инициализация динамической библиотеки формата (Format DLL)	5-70
5.3.3.2	Запрос свойств динамической библиотеки формата (Format DLL)	5-71
5.3.3.3	Запрос имени динамической библиотеки формата (Format DLL)	5-73
5.3.4	Завершение работы динамической библиотеки формата (Format DLL)	5-74
5.3.4.1	Расширения конфигурации.....	5-74
5.3.4.2	Диалоговое расширение во время конфигурации сообщений S7PMC.....	5-74
5.3.4.3	Диалоговое расширение во время конфигурации архивных тегов.....	5-77
5.3.4.4	Online службы	5-78
5.3.4.5	Регистрация всех архивных тегов	5-79
5.3.4.6	Переключение языка	5-81
5.3.5	Форматирование	5-82
5.3.5.1	Вывод одиночных сообщений	5-82
5.3.5.2	Подтверждение, блокировка/разрешение сообщений.....	5-83
5.3.5.3	Обработка в случае изменения состояния	5-85
5.3.5.4	Обновление сообщений динамической библиотеки формата (Format DLL) S7PMC.....	5-85
5.3.5.5	Форматирование архивных тегов.....	5-87
5.3.5.6	Вывод отдельных значений архивных тегов.....	5-87
5.3.5.7	Блокировка/разрешение архивных тегов	5-88
5.3.5.8	Обработка в случае изменения состояния	5-88
5.4	Глобальная библиотека.....	5-90
5.4.1	Системные блоки.....	5-91
5.4.1.1	Двигатели.....	5-91
5.4.1.2	ПК/ПЛК	5-92
5.4.1.3	Насосы	5-92
5.4.1.4	Трубы	5-93
5.4.1.5	Трубы - модифицированные объекты.....	5-93
5.4.1.6	Резервуары	5-94
5.4.1.7	Клапаны - модифицированные объекты	5-94
5.4.1.8	Клапаны	5-94

5.4.2	Дисплеи	5-95
5.4.2.1	Дисплеи	5-95
5.4.2.2	Окна	5-95
5.4.2.3	Линейки	5-95
5.4.2.4	Текстовые поля	5-95
5.4.2.5	Измерительные приборы	5-96
5.4.3	Элементы управления	5-97
5.4.3.1	3D кнопки	5-97
5.4.3.2	Панели управления	5-97
5.4.3.3	Кнопки с изображениями	5-98
5.4.3.4	Навигация по кадрам	5-98
5.4.3.5	Кнопки инкремента/декремента	5-98
5.4.3.6	Контроллеры	5-99
5.4.3.7	Переключатели языка	5-99
5.4.3.8	Клавиатуры	5-99
5.4.3.9	Переключаемые кнопки	5-100
5.4.4	Символы	5-101
5.4.4.1	Устройства выключения	5-101
5.4.4.2	Клапаны выключения	5-102
5.4.4.3	DIN 30600	5-103
5.4.4.4	E символы	5-104
5.4.4.5	Конвейеры	5-105
5.4.4.6	ISA символы	5-106
5.4.4.7	Двигатели	5-110
5.4.4.8	Клапаны	5-111
5.4.4.9	Разное 1	5-112
5.4.4.10	Разное 2	5-113

Введение

Цель руководства

Данное руководство знакомит Вас с существующими опциями конфигурации WinCC.

Руководство можно получить как в печатном, так и в электронном виде.

Используя оглавление или указатель можно быстро найти необходимую информацию. В электронной версии существуют расширенные возможности поиска.

Требования для использующих данное руководство

Знание основ WinCC, например, из руководства для начинающих “Getting Started“ или из практического опыта конфигурирования WinCC.

Дополнительная поддержка

По техническим вопросам обращайтесь в представительство компании Siemens в Вашем регионе.

Также Вы можете воспользоваться нашей горячей линией по телефону:

+49 (911) 895-7000 (Fax -7001)

Информация о продуктах SIMATIC

Актуальную информацию о продуктах SIMATIC можно найти в каталоге CA01. Данный каталог расположен по следующему адресу Internet:

<http://www.ad.siemens.de/ca01online/>

Кроме того, служба SIMATIC Customer Support (служба поддержки пользователей SIMATIC) обеспечивает клиентов текущей информацией и загружаемыми программами. Подборка наиболее часто задаваемых вопросов находится по следующему адресу Internet:

http://www.ad.siemens.de/support/html_00/index.shtml

1 РУКОВОДСТВО ПО КОНФИГУРАЦИИ

Руководство по конфигурации является частью документации WinCC и, в основном, описывает практическое применение WinCC в проектах.

Введение

За последние несколько лет сильно возрос спрос на системы мониторинга и контроля производственных процессов, а также на системы архивирования и дальнейшей обработки показателей хода производственного процесса. В связи с возникшим спросом, в последнее время, стали развиваться новые HMI системы.

Одной из таких новых систем и является WinCC. В отношении выполняемых функций, открытости и современного положения дел, WinCC, без сомнений, является уникальным продуктом.

Ранние версии HMI систем обычно предлагали только один путь решения поставленной задачи. С появлением WinCC стало возможным почти всегда решать поставленные задачи несколькими путями. Данное руководство написано для того, чтобы помочь в выборе наилучшего решения с учетом соотношения полученных результатов и затраченных усилий.

Данное описание разработано для того, чтобы познакомить Вас с предлагаемыми решениями для более эффективного применения WinCC в проектах.

Мы использовали предлагаемые нами решения в конкретных примерах. Эти примеры поставляются на диске вместе с WinCC. В своих проектах Вы можете использовать наши разработки.

1.1 Руководство по конфигурации - замечания относительно структуры и применения

Требования

Перед тем, как начать работу с руководством по конфигурации, необходимо иметь некоторый практический опыт работы с WinCC. Новички могут воспользоваться руководством для начинающих Getting Started, что будет являться идеальным вариантом для ознакомления с WinCC и позволит изучить и больше узнать этот программный продукт. В нем объясняются главные цели и функции посредством небольшого демонстрационного примера. Руководство по конфигурации является приложением к справочной системе WinCC (online справке и документации). Если в данном руководстве что-то отсутствует, то, возможно, объяснение будет дано в разделах специальных возможностей объектов, свойств, а также в других разделах справочной системы.

Содержание и структура




Руководство состоит из следующих разделов:

- WinCC – Концепция
Данный раздел содержит общую информацию о системе WinCC.
- Конфигурация – Общие сведения
Этот раздел содержит общую и специфическую информацию о планировании и эффективном управлении НМІ проектами.
- Работа с примерами
Этот раздел содержит информацию, необходимую для работы с примерами, созданными в данном руководстве.
- Курс Си WinCC
Этот раздел содержит курс языка Си для WinCC. Начинающие найдут там основные правила использования языка скриптов WinCC, а для экспертов по Си будут полезны материалы про специальные возможности среды разработки WinCC.
- Конфигурация Тегов/Переменных
В этом разделе объясняется пример Project_TagHandling (Обработка тегов проекта). В этом примере описываются общие принципы обработки тегов и простые элементы ввода/вывода.
- Конфигурация кадров
В этом разделе рассмотрен пример Project_CreatePicture (Создание кадров). В примере описываются общие принципы работы с кадрами в WinCC.
- Редакторы WinCC
В этом разделе объясняется пример Project_WinCCEditors (Редакторы WinCC). В примере описываются: система регистрации тегов, система регистрация аварийных сообщений и дизайнер отчетов.
- Пользовательские архивы
В этом разделе объясняется пример Project_UserArchive (Пользовательский архив). В этом примере описывается редактор пользовательских архивов.

- **Описание новых функций**
В этом разделе описывается возможность конфигурирования распределенных систем, добавленная в WinCC V5.
- **Многопользовательские проекты**
В этом разделе показывается использование многопользовательского проекта на конкретном примере.
- **Распределенные серверы**
В этом разделе, на конкретном примере, описывается возможность создания WinCC проекта, распределенного между несколькими серверами.
- **Резервирование**
В этом разделе на конкретном примере описывается возможность конфигурирования резервного сервера.
- **Приложение**
В этом разделе содержатся некоторые дополнительные материалы. Они основаны на данных из WinCC *Solutions (Решения WinCC)* и WinCC *Tips Tricks (Полезные советы)*.

Условные обозначения

В руководстве по конфигурации используются следующие условные обозначения:

Обозначение	Описание
	Обозначает операцию с использованием левой кнопки мыши.
	Обозначает операцию с использованием правой кнопки мыши.
	Обозначает двойное нажатие на левую кнопку мыши.
<i>Курсив</i>	Обозначает термины WinCC и термины, относящиеся к элементам программного интерфейса.
<i>Курсив, зеленый</i>	Обозначает рабочую последовательность или данные, вводимые пользователем (цвет виден только в онлайн-документе).
Голубой	Обозначает перекрестные ссылки (цвет виден только в онлайн-документе).

Поиск информации

В печатной версии руководства по конфигурации информация может быть найдена следующими способами:

- В **оглавлении** информация разделена на темы.
- В **предметном указателе** информация разделена согласно ключевым словам.

В онлайн-документе информация может быть найдена следующими способами:

- На закладке “Оглавление” информация разделена на темы.
- На закладке “Предметный указатель” информация разделена согласно ключевым словам.
- На закладке “Поиск” имеется возможность поиска слов во всем документе.

Описанный в данном руководстве пример проекта можно скопировать непосредственно из online документа на жесткий диск.

2 WinCC – Общая информация

2.1 WinCC – Основные понятия

- В целом, с точки зрения конфигурации, существует три подхода к решению задач с помощью WinCC:
- Конфигурация, использующая стандартные ресурсы WinCC
- Использование существующих приложений Windows с WinCC через DDE, OLE, ODBC и ActiveX

Создание собственного встроенного в WinCC приложения (с помощью Visual C++ или Visual Basic),.

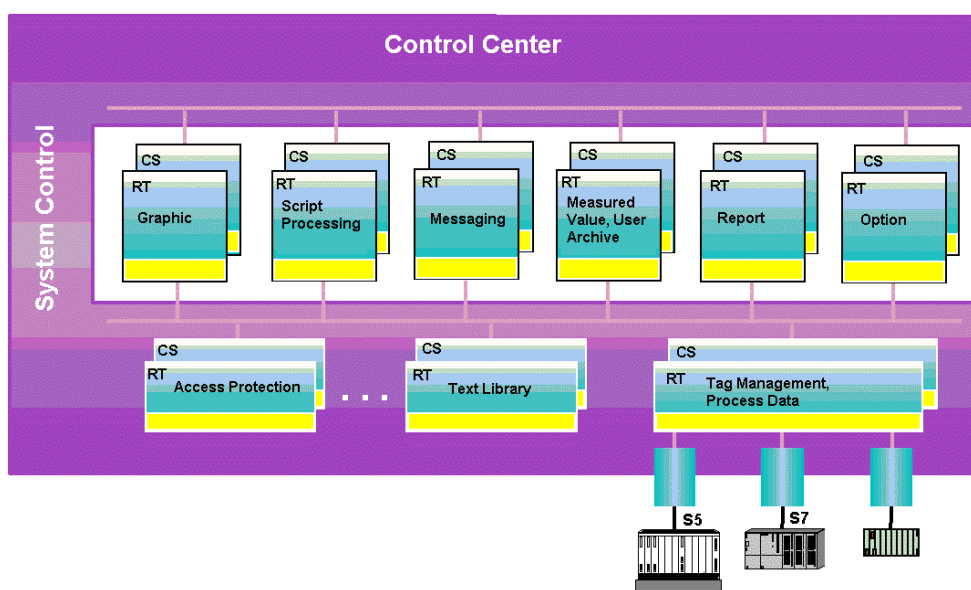
Для некоторых, WinCC является HMI системой для недорогих и быстрых конфигураций, тогда как для других она является системной платформой с неограниченными возможностями. Благодаря модульному принципу организации и гибкости WinCC, открываются абсолютно новые возможности для проектирования и выполнения задач, связанных с автоматизацией.

Операционная система: Основа WinCC

WinCC предназначена для 32-х разрядной операционной системы Microsoft (Windows NT 4.0). Эта операционная система является стандартной для платформ ПК.

Модульный принцип организации WinCC

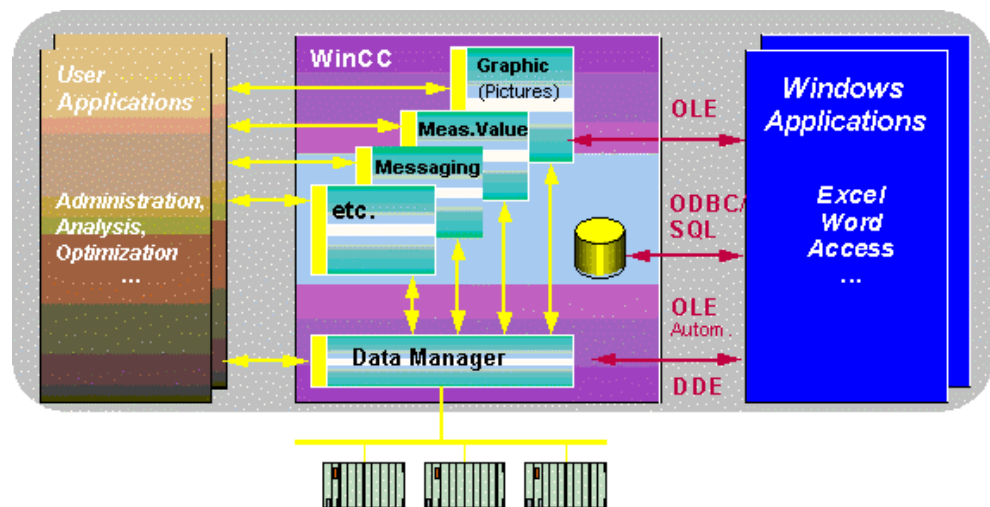
В состав WinCC входят системные модули как для визуализации, составления отчетов, получения и архивирования обрабатываемых данных, так и для согласованного объединения различных пользовательских программ. В дополнение ко всему описанному, имеется также возможность встраивания собственных модулей.



2.1.1 Интерфейсы WinCC

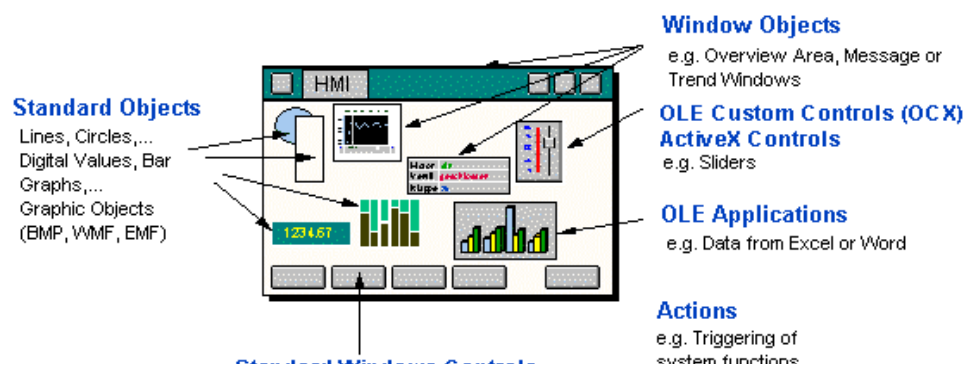
Открытость WinCC

WinCC является абсолютно открытой для любых дополнений, вносимых пользователем. Это свойство открытости достигнуто благодаря модульному принципу организации WinCC и мощному программному интерфейсу. На приведенном ниже рисунке иллюстрируются возможности совместной работы различных приложений.



Интеграция приложений в WinCC

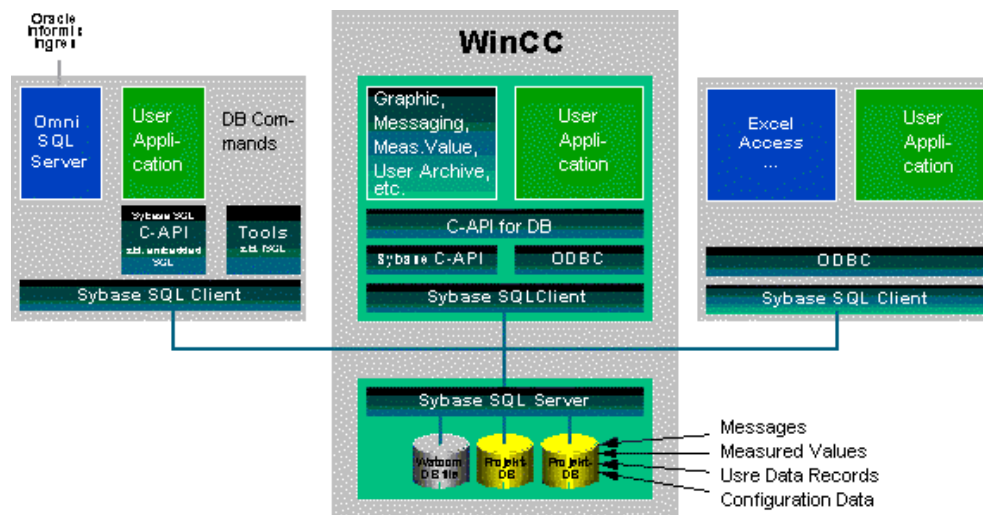
Наиболее важным является тот факт, что WinCC предлагает методы интеграции других приложений и их частей непосредственно в пользовательский интерфейс, используемый для управления технологическим процессом. Как показано ниже, OLE-приложения Windows, специальные управляющие элементы OLE (32-х битовые OCX объекты) или ActiveX объекты могут быть с легкостью интегрированы в приложения WinCC так, как если бы они были непосредственно объектами WinCC.



Управление данными в WinCC

На приведенной ниже диаграмме WinCC занимает всю центральную часть. Из диаграммы следует, что базой данных по умолчанию является Sybase SQL Anywhere. Она используется для хранения всех структурированных данных, представленных в виде списков, к которым относятся списки тегов, текстовые сообщения, а также для хранения текущих обрабатываемых данных, таких как сообщения, измеряемые значения, и пользовательские данные. Эта база данных функционирует в качестве сервера. WinCC может обратиться к данным через ODBC и с помощью открытого программного интерфейса (C-API), как клиент.

Те же права, конечно, даны и другим программам. По этой причине таблица Windows и база данных Windows имеют прямой доступ к базе данных WinCC независимо от того, запущено приложение на этом же компьютере или на сетевой рабочей станции. С помощью структурированного языка запросов SQL и соответствующих инструментов (например, драйверов ODBC) другие клиенты (например, базы данных UNIX, такие как Oracle, Informix, Ingres) могут также получить доступ к источникам данных WinCC и наоборот. Подводя итог можно сказать, что нет ничего препятствующего интегрированию WinCC в заводскую или корпоративную систему.



2.2 WinCC – разъяснения терминов

Это раздел содержит список упорядоченных в алфавитном порядке терминов, относящихся к WinCC. Возможно, что многие из них Вам уже знакомы.

НМИ Человеко-машинный интерфейс (Human Machine Interface)

ПЛК Программируемый логический контроллер (Programmable Logic Controller)

CS Система конфигурации (Configuration System)

RT Время исполнения (Runtime)

3 Конфигурация – Общие предметы

В этом разделе Вы найдете большое количество информации, инструкций и идей, способствующих правильному пониманию того, как надо управлять проектом при помощи WinCC. Некоторая информация не является характерной для WinCC. В идеале эти правила конфигурации должны стать стилистическим направлением для конфигурирования и дизайна работающих проектов.

3.1 Перед началом проекта

Перед тем, как начать конфигурацию, необходимо понять некоторые спецификации и усвоить правила структурированной работы. Это

- упрощает конфигурацию
- делает проекты более понятными
- упрощает коллективную работу
- улучшает стабильность и производительность
- снижает эксплуатационные расходы

Четкие спецификации структурных направлений являются основными предпосылками для установления и расширения корпоративных стандартов.

Эти спецификации можно поделить на две категории

Спецификации для конфигурации

- Перед тем, как начать проект, необходимо определить следующие спецификации:
- название проекта WinCC
- названия тегов
- названия экранных форм WinCC
- правила создания сценариев и процедур
- правила конфигурации (корпоративные стандарты, библиотека функций, коллективная работа)
- режим и способ документирования проекта

Спецификации для исполняемого проекта

Спецификации относительно исполняемого проекта (результата конфигурации). Эти спецификации зависят главным образом от области применения (например: автомобильная промышленность, химическая промышленность, машинное производство). Должны быть определены описанные ниже спецификации:

- пользовательский интерфейс (расположение окон, шрифт и его размер, языковые настройки, отображение объектов и т.д.)
 - общая идея (иерархия окон, способ управления, права пользователя, используемые клавиши и т.д.)
 - цвет сообщений, уставки, текст и т.д.
 - режим связи (тип соединения, тип и период обновления и т.д.)
 - количественные характеристики (количество аварийных сообщений, архивные значения, тренды, клиенты и т.д.)
- сообщения и методы архивации

3.2 Детальная спецификация

В этом разделе руководства мы изучим спецификации, используемые в дальнейших примерах. Эти спецификации предназначены для использования в качестве шаблонов при создании собственных проектов.

Замечание:

В приведенных примерах имена проектов, окон, тегов, переменных и комментарии в сценариях даны на английском языке.

Значения по умолчанию средств конфигурации

В большинстве редакторов WinCC некоторые свойства могут быть установлены по умолчанию. В этом смысле, WinCC поддерживает индивидуальный стиль конфигурации и может быть сконфигурирован оптимальным образом для выполнения специфических задач.

Замечание:

Примером этого служит опция, которая может быть установлена через *Graphics Designer (Графический дизайнер) – Tools (Инструментальные средства) – Settings (Настройка)*. Более детальное описание данного материала находится в онлайн-овой системе помощи *Graphics Designer (Графического дизайнера)*.

3.2.1 Спецификация: Название проекта WinCC

Общие сведения

Для папки, в которой будут храниться данные, касающиеся WinCC, рекомендуется оставить присвоенное по умолчанию имя, аналогичное имени проекта. Название папки можно изменить либо во время создания самого проекта, либо позднее (используя проводник Windows).

Параметры/Ограничения

Для использования разрешены все символы, за исключением некоторых специальных (например \ ? ' . ; : /). Также допустимо использование числовых значения от 0 до 9.

Спецификации

В примере, описанном во второй части руководства по конфигурации следующее относится к названию проекта:

a...a_nn

где:

- a тип обозначения (a-z, A-Z, никаких специальных символов)
- _n серийный номер для обозначения проекта среди однотипных проектов(цифры 0 - 9), диапазон 00 – 99

Пример: cours_00.mcp, или pictu_01.mcp

Замечание относительно основных применений

Название проекта WinCC, например, может быть использовано для обозначения разных технологических разделов.

Замечание:

При обновлении документации, название проекта WinCC можно добавить в распечатку. Это упрощает процесс группировки и поиска информации.

3.2.2 Спецификация: Название тегов

Общие сведения

Названия тегов теперь могут содержать более 8 символов. Несмотря на это, не рекомендуется давать тегам слишком длинные имена. Если в процессе определения названий тегов придерживаться строгих правил, то при конфигурации это будет очень полезным.

При создании проектов WinCC одной из главных задач является структура управления тегами, которая обеспечивает быструю и эффективную конфигурацию, а также качественную обработку данных во время исполнения (в сценариях).

Перед определением названий тегов, необходимо принять во внимание набор специальных характеристик, относящихся структурированному управлению тегами WinCC. Создание групп влияет только на то, как теги отображаются во время конфигурации. Названия групп теперь влияют на уникальность названий тегов. Используемые в WinCC названия тегов должны быть уникальными. Это проверяется системой.

WinCC помогает выбирать теги многими способами, например, сортировкой по столбцам (название, даты создания и т.д.) или при помощи фильтров. Тем не менее, будет очень полезным, если название тега будет содержать некоторую дополнительную информацию.

Спецификации

Описанное ниже имеет отношение к названиям тегов, приведенных в примерах данного руководства:

xxxx_z...z_a...a_nn

где:

X	Аббр.	Тип
	BIN	Двоичный тег
	U08	Без знаковое 8 битовое значение (unsigned)
	S08	Знаковое 8 битовое значение (signed)
	U16	Без знаковое 16 битовое значение
	S16	Знаковое 16 битовое значение
	U32	Без знаковое 32-х битовое значение
	S32	Знаковое 32-х битовое значение
	G32	32-х битовое число с плавающей запятой IEEE 754
	G64	64-х битовое число с плавающей запятой IEEE 754
	T08	8 битовый символьный тег
	T16	16-и битовый символьный тег
	RAW	Строковый тип данных
	TER	Текстовая ссылка
	STU	Структурные типы

У	Аббр.	Происхождение
r		Напрямую читаемый из ПЛК тег
w		Тег для чтения и записи в ПЛК (запись)
i		Внешний тег WinCC без связей с ПЛК
x		Тег с косвенной адресацией (текстовый тег, содержащий название тега)
_z		Группа (относящаяся к промышленной секции или зданию) _Paint ... напр. название промышленной секции
_a		Название тега (например, название точки измерения) _EU0815V10 ... например, название точки измерения
_n		Серийный номер примера (числа 0 - 9), диапазон 00 - 99

Параметры/Ограничения

- При назначении названий тегов необходимо принять во внимание следующие ограничения:
- Специальный символ @ должен быть зарезервирован для системных тегов WinCC (но на его использование в других местах нет никаких ограничений).
- Нельзя использовать специальные символы ' и %.
- Не разрешается использование специальных символов " и //, т.к. они имеют специальное значение в сценариях Си (начало/конец символьных строк и знаки комментариев).
- Не разрешается использование пробельных символов.
- Верхний и нижний регистры не различаются.

Замечание относительно основных применений

Назначенные нами в примерах названия тегов не обязательно использовать в своих проектах.

При использовании сценариев и Excel будет очень полезным строго придерживаться фиксированной длины различных частей названия тега (если необходимо, то можно использовать для заполнения символы 0 или x).

Можно очень просто создать и эффективно поддерживать большое количество тегов, используя, например, Excel. Если названия тегов имеют фиксированную структуру, то гораздо легче создать список тегов в Excel. Созданный в Excel список тегов может быть импортирован в рабочий проект WinCC с помощью программы \SmartTools\CC_TagImportExport\Var_exim.exe, которая находится на диске WinCC.

3.2.3 Спецификация: Название кадров

Общая информация

Если необходимо обратиться к кадрам в сценариях или внешних программах, то при определении их названий будет очень полезным использовать фиксированную структуру. Также необходимо продумать длину названий. Слишком длинные названия (названия файлов) обычно вносят неразбериху (при выборе в списках выбора, вызовах в сценариях и т.д.). Из опыта следует, что лучше всего использовать название длиной не более 40 символов.

Параметры/Ограничения

- При определении названий кадров необходимо учесть следующие ограничения:
- Максимальная дли должна составлять 255 символов
- Могут использоваться любые символы за исключением специальных
- В названиях кадров верхний и нижний регистры не различаются

Спецификации

Спецификации относительно названий кадров, используемых в проектах данного руководства:

aaaaa_k_x...x_nn

где:

- a Идентификатор кадра (a-z, A-Z, без специальных символов) для объединения кадров

Course... например, названия кадров курса Си

_k	Идентификатор типа кадра 0 – 99	Тип кадра
_0		Начальный кадр
_1		Обзорный кадр
_2		Кадр на кнопке
_3		Промышленный кадр
_4		Детализированный кадр
_5		Кадр сообщения
_6		Кадр отклонений
_7		...
_8		...
_9		Диагностический кадр (только для тестирования или ввода в эксплуатацию)

- _x Название для описания функции кадра (a-z, A-Z, без специальных символов), максимальная длина 30 символов.

_chapter ... например, название главы курса Си (C-Course)

_n Серийный номер типа (число 0 - 9), диапазон 0 - 99

Замечание относительно основных применений

Назначенные нами в примерах названия кадров не обязательно использовать в своих проектах. Однако, для поставляемых сценариев необходимо использовать предложенные нами соглашения об именовании.

3.2.4 Спецификация: Сценарии и процедуры

Общая информация

УВ проектах WinnCC имеется возможность создания своих собственных сценариев и процедур. Данное сценарию название должно отражать его суть. Это облегчит работу при дальнейшем использовании сценариев.

Использование пропорциональных шрифтов создаст неудобства при конфигурации в Global Script. В связи с этим рекомендуется использовать шрифты постоянного размера (например, Courier), что сделает текст более удобным для чтения.

Сценарии должны сопровождаться понятными комментариями. На написание комментариев тратится гораздо меньше времени, чем на понимание плохо закомментированных программ. Несмотря на то, что это общепринятый факт, он часто остается без внимания.

Спецификации

Спецификации относительно сценариев, используемых в проектах данного руководства:

Используется пропорциональный шрифт *Courier New*, размер шрифта 8

Имена всех переменных и комментарии даны на английском языке

Замечание относительно основных применений

Подробное описание использования сценариев, процедур и редакторов дано в главе Среда разработки сценариев на Си.

3.2.5 Спецификации: Пользовательский интерфейс

Основная информация

Пользовательский интерфейс необходимо настраивать с большой аккуратностью. Все объекты, созданные в *Graphics Designer (графическом дизайнере)*, отображаются на экране в рабочей области пользователя.

Созданные кадры являются лишь интерфейсом между машиной и пользователем.

Следовательно, их созданию нужно уделить особое внимание, поскольку они играют жизненно важную роль в обеспечении успеха проекта. Без сомнения функционирование завода намного важнее картинки на экране, но в будущем, небрежно нарисованные картинки могут испортить впечатление от проекта и возможно даже увеличить себестоимость обеспечения производства, поэтому лучше их хорошо продумать.

Эти экранные формы будет ежедневно видеть оператор (покупатель).

В системах с визуализацией информации на дисплее, информация о текущем состоянии производства предоставляется исключительно средствами графического изображения. Поэтому этот вид интерфейса должен быть наиболее понятным для пользователя.

Система WinCC предоставляет пользователю возможность наиболее точно сконфигурировать интерфейс проекта. Ответ на вопрос как построить пользовательский интерфейс зависит от используемого оборудования, от предоставляемых к системе требований и существующих спецификаций.

Пользователи

При конфигурировании пользовательского интерфейса, все внимание должно быть обращено на пользователей, для которых проектируется система.

Если вам удалось довести информацию до пользователя в наиболее доступном виде, результатом будет **высокое качество продукции и снижение вероятности ошибок**. Также снизится объем необходимого технического обслуживания.

Пользователям необходимо как можно больше информации. Используя эти данные как отправную точку, пользователи принимают решение о том, что необходимо для поддержания работоспособности процесса при высоком качестве продукции.

Основная задача пользователей не реагировать на аварийные ситуации (после которых нарушено нормальное течение процесса), а использовать свои знания и опыт, а также предоставляемую системой информацию для того, чтобы суметь предсказать направление развития процесса. Пользователи должны иметь возможность предотвращать аварийные ситуации до их возникновения. WinCC предоставляет возможность редактировать и наиболее эффективно отображать информацию для пользователей.

Сколько информации должно быть сосредоточено на кадре?

При обдумывании сосредоточенного на кадре объема информации, для достижения оптимального результата, необходимо учесть два основных аспекта:

- Если кадр содержит слишком много информации, он будет труден для восприятия и поиск необходимой информации займет много времени. При этом возрастает вероятность ошибки пользователя.

Если кадр содержит слишком мало информации, возрастает объем необходимой работы пользователя. Он теряет нить процесса и должен часто менять кадры, чтобы найти необходимую информацию. Это ведет к увеличению времени реакции, а также нестабильности контролируемого процесса.

Исследования показали, что опытные пользователи желают, чтобы кадры отображали как можно больше информации и чтобы не было необходимости их часто менять. И напротив, начинающие пользователи чувствуют себя неуверенно и теряются в большом объеме информации на одном кадре. Порой они даже не могут найти необходимую информацию или делают это не вовремя. Но опыт научил нас одной вещи: Новичок рано или поздно становится профессионалом, но профессионал никогда не станет новичком.

Соккрытие информации

Отображаемая информация должна быть важной и понятной. Некоторая информация может оставаться скрытой (например, идентификаторы измерительных точек) до тех пор, пока она не понадобится.

Отображение информации

При отображении аналоговых значений, старайтесь с помощью указательных инструментов комбинировать их с цифровыми. Графическое представление значений (например, указательные инструменты, гистограммы ...) делает информацию доступной и понятной для пользователя.

Чтобы избежать сложностей, возникающих при работе не различающего цветов пользователя, важные изменения объекта (статуса) должны отображаться не только использованием различных цветов, но также и различного формата.

Важная информация должна быть немедленно распознана оператором. Это означает необходимость правильного использования контрастных цветов.

Кодирование цветов

Человеческий глаз воспринимает цвета быстрее, чем, например, текст. Работа с кодированием цвета может очень помочь Вам при определении статуса различных объектов, однако при этом следует соблюдать единую схему цветового кодирования. Универсальные цветовые требования для отображения состояний в проекте (например, красный для ошибки) уже стали стандартом. Корпоративные стандарты, используемые клиентами, также должны быть приняты во внимание.

Отображение текста

Чтобы сделать текст более читаемым, необходимо придерживаться следующих простых правил.

- Размер текста должен соответствовать важности содержащейся в тексте информации, а также удаленности пользователя от экрана.
- Малые буквы более предпочтительны. Они требуют меньше места и читаются легче, чем большие, даже если букву легче прочитать на расстоянии.
- Горизонтальный текст более удобен для чтения, нежели вертикальный или диагональный.
- Используйте различные шрифты для разного типа информации

Придерживайтесь вашей концепции

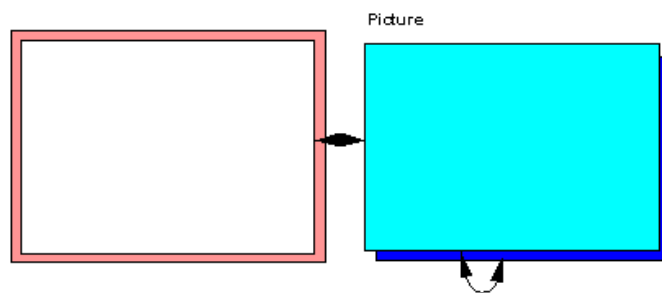
Какую бы концепцию Вы не выбрали для использования, Вы должны придерживаться ее во всем проекте. Таким образом, Вы поддерживаете интуитивный контроль над кадрами процесса. Ошибки пользователя становятся менее вероятными. То же самое относится к используемым объектам. Двигатель или насос должен всегда выглядеть одинаково, несмотря на то, на каком кадре он изображен.

Размещение на экране

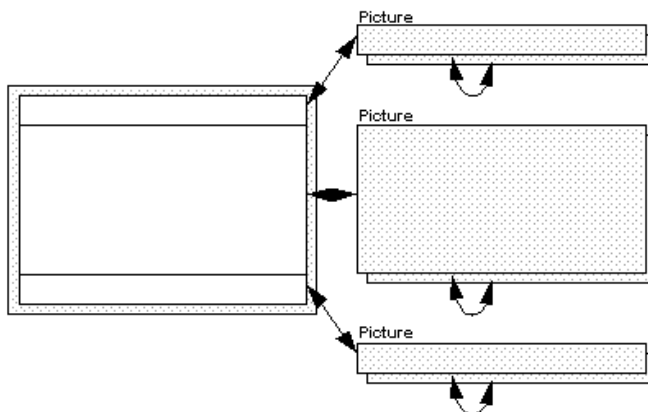
Опыт показывает, что при использовании стандартного монитора персонального компьютера, имеет смысл разделить экран на три области: обзорная область, производственную область и область кнопок.

Если же ваше приложение работает на специальном промышленном компьютере или на панели оператора с клавишами с совмещенными функциями, такой метод деления экранного пространства не всегда подходит.

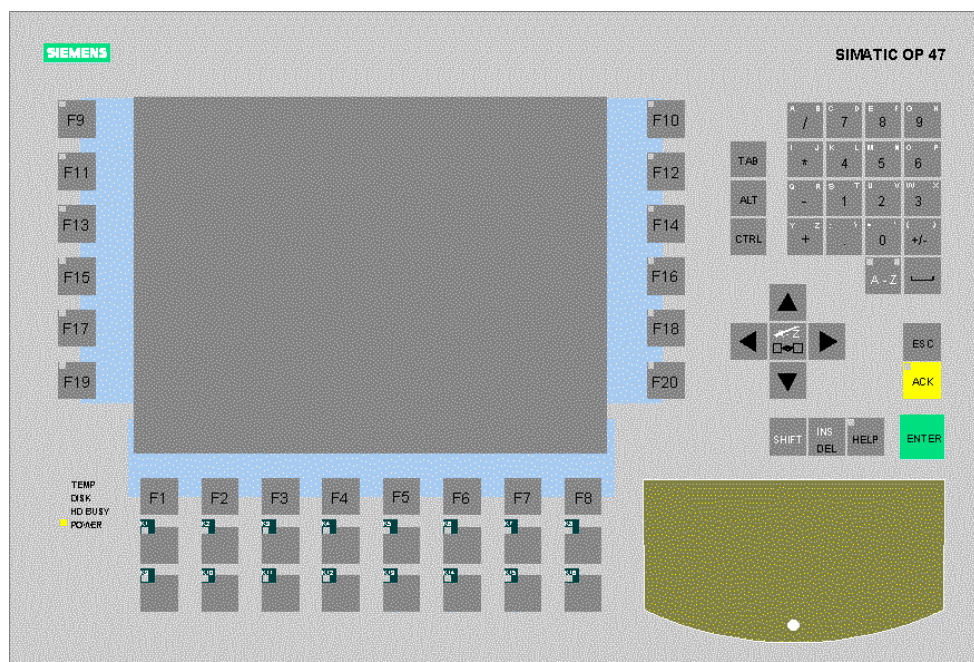
Полноэкранный кадр



Экран, разделенный на обзорную, кнопочную и производственную области



Пример панели оператора



Параметры/Ограничения

Размеры отдельных кадров могут быть установлены в пределах фиксированных границ (мин. 1 x 1, макс. 4096 x 4096 точек). В случае однопользовательской системы с 17" монитором, мы рекомендуем использовать максимальное разрешение 1024 x 768 точек. В многопользовательской системе более высокое разрешение, возможно, будет более приемлемым.

В случае операторской панели, все ограничивается доступным разрешением (TFT от 640 x 480 до 1024 x 768 точек).

Спецификация

Спецификации относительно использованных в проектах данного руководства кадров:

Разрешение

В нашем примере, используется разрешение 1024 x 768 и 800 x 600 точек (в исключительных случаях). Для корректного отображения проектов, необходимо, чтобы количество цветов Вашего монитора было установлено минимум в 65536 цветов.

Тексты

Имена точек измерения написаны шрифтом Courier, описания, и другие тексты - шрифтом Arial. Для сообщений, следуя стилю Windows, использовались шрифты MS Sans Serif и System.

Размеры шрифтов устанавливаются в соответствии с необходимыми.

Информация на кадре

Всякий раз, когда это имеет смысл, мы пытаемся скрыть некоторую информацию в кадрах. Эта информация отображается только в тех случаях, когда она требуется (управление ручное или автоматическое).

Также в наших проектах используются различные способы размещения объектов на экране. Если кадр содержит большое количество контролируемых объектов, мы предоставляем информацию о том, как их использовать, в виде подсказок.

Размещение на экране

Мы сконфигурируем основные опции для размещения на экране. В оставшихся проектах, однако, мы будем применять метод деления экрана на заголовок, рабочую область и нижнюю часть.

Замечание относительно основных применений

Вы можете использовать предложенные концепции размещения для собственных проектов.

3.2.6 Спецификация: Концепция управления

Общая информация

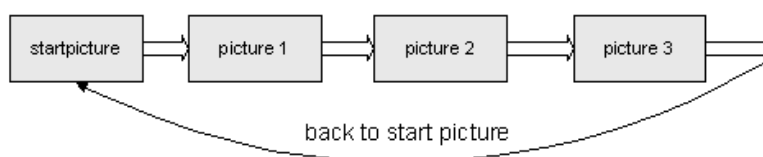
Работа Вашего приложения WinCC контролируется при помощи устройств ввода таких, как клавиатура, мышь, сенсорная панель или джойстик. Если ваш компьютер находится непосредственно на производстве с экстремальными условиями, где нет возможности использовать мышь, вы можете сконфигурировать таб курсор и буквенный курсор. Таб курсор позволяет вам перемещаться по контролируемым полям, в то время как буквенный курсор позволяет вам выполнять навигацию по полям ввода.

Каждое управляющее процедура может быть заблокировано против несанкционированного доступа.

Открытие кадров

Концепция выбора экранных форм зависит от нескольких факторов. Наиболее важно количество кадров и структура отображаемого процесса.

В небольших приложениях, рисунки могут быть упорядочены по принципу кольца или FIFO буфера.



Если вы работаете с большим количеством кадров, обязательным требованием является иерархическое упорядочивание их появления. Для того чтобы операторы могли быстро освоить управление, выбирайте простую и постоянную структуру. Без всякого сомнения, кадры должны открываться напрямую, и это имеет особое значение для небольших приложений (например, холодильный склад).

Иерархия

Иерархическая структура процесса делает его более доступным для понимания, упрощает управление и позволяет осуществить быстрый доступ к информации. Наиболее распространена трехслойная иерархическая структура.

Первый слой

Под первым слоем подразумеваются обзорные кадры.

Этот слой, в основном, содержит информацию о различных частях системы, а также о том, как эти части взаимодействуют между собой.

Первый слой также сигнализирует о каком-либо событии (сообщении), произошедшем на нижних слоях.

Второй слой

Под вторым слоем подразумеваются кадры процесса.

Этот слой содержит подробную информацию о некоторой части процесса и показывает, какой объект производства относится к этой части. Слой также показывает, к какому объекту относится аварийный сигнал.

Третий слой

Под этим слоем подразумеваются детальные кадры. Он предоставляет информацию об отдельных объектах производства, например контроллерах, моторах и др. Он отображает сообщения, состояния и значения процесса. Он также может содержать информацию, касающуюся взаимодействия объектов между собой.

Спецификация

Следующие спецификации применяются к проектам из руководства: В наших проектах мы будем использовать несколько различных концепций управления и укажем на различия между ними.

Замечание относительно основных применений

Наши проекты должны рассматриваться только как предложения по созданию собственной концепции управления. При расширении производства, Вы должны принять во внимание существующую концепцию управления. Большое число пользователей найдут, что их компания уже имеет общие соглашения и стандарты, которых необходимо придерживаться при конфигурировании системы.

Замечания:

Дополнительный пакет WinCC Basic Process Control предоставляет уже готовую концепцию управления. Этот дополнительный пакет также содержит другие полезные и мощные функции (например, хранение).

3.2.7 Спецификация: Определение цвета

Основная информация

Тема цвета очень популярна в дискуссиях относительно систем с человеко-машинным интерфейсом. WinCC предоставляет вам возможность свободно выбирать цвета для линий, границ, фона, теней и шрифтов. Вы имеете возможность выбирать из всей палитры цветов поддерживаемых системой Windows. Естественно, цвета, и другие графические элементы, могут быть изменены в течении времени исполнения WinCC.

Определение цвета особенно важно в обеспечении правильности отображения процессов.

- Цвета должны быть всегда определены для следующих областей. Определение цветов должно соответствовать стандарту DIN EN 60073 , который относится к VDE 0199, но при этом должно соответствовать запросам пользователя:
- Цвета сообщений (активировано / очищено / подтверждено)
- Цвета состояний (включено / выключено / ошибочно)
- Цвета символьных объектов (схемы / уровни заполнения)
- Цвета предупреждающих и ограничивающих значений

Спецификация

К цветам, используемым в проектах данного руководства, применяются следующие спецификации:

Для корректного отображения проектов необходимо, чтобы на Вашем компьютере был установлен режим с количеством цветов большим, чем 256.

Для удобства, мы будем использовать различные цвета фона для различных тем (теги, курс Си, конфигурация кадра). Цвет фона в обзорной области и области кнопок темнее.

Для систем с аварийными сигналами, каждый класс и тип сообщений имеет свой собственный цвет.

Замечание относительно основных применений

После определения цветов, в случае необходимости, следует настроить установки WinCC по умолчанию.

Таблица кодировки цветов в процедурах Си находится в приложении, глава Таблица цветов.

3.2.8 Спецификация: Циклы обновления

Основная информация

При определении циклов обновления, всегда рассматривайте систему в целом: Что именно обновляется и как часто выполняется обновление. Выбор неправильного цикла обновления может иметь негативный эффект, который скажется на производительности системы.

При работе с полной системой (ПЛК - связь - ЧМИ), изменения должны фиксироваться там, где они происходят, а именно в процессе (ПЛК). В большинстве случаев, это система с шиной, которая имеет определенную пропускную способность.

При определении режима обновления измеряемых значений вы должны обращать внимание на то, насколько быстро меняются реально измеряемые значения. Для контроля температуры бойлера с объемом 5,000 литров, обновление фактического значения с интервалом в 500 мс является нецелесообразным.

32-битовая система с человеко-машинным интерфейсом (ЧМИ)

WinCC является 32-битовой системой с человеко-машинным интерфейсом, работающей в системе Windows NT. Эта операционная система оптимизирована для управления по событиям. Если принять во внимание этот принцип при конфигурировании WinCC, проблемы с производительностью практически исчезнут, даже если вы оперируете с большим объемом информации.

Спецификация

В проектах, описанных в данном руководстве, имеет место следующая относящаяся к обновлению спецификация:

По определению задачи, обновление совершается по некоторому событию. Так как в основном мы работаем с внутренними тегами, то этими событиями являются обновления тегов. При использовании внешних тегов, это может привести к увеличению загрузки системы, зависящей от количества соединений драйвера процесса. Если связь поддерживает событийную передачу, она должна быть выбрана для критичных по времени данных. Не критичные данные могут быть извлечены с помощью человеко-машинного интерфейса (ЧМИ) по соответствующим циклам (процедура поллинга).

Замечание относительно основных применений

Подробное описание применения циклов обновления можно найти в главе Циклы обновления – Как и где устанавливаются?

3.2.9 Спецификация: Права пользователя

Общая информация

При управлении технологическим процессом необходимо защитить некоторые функции оператора от несанкционированного доступа. Следующее требование заключается в том, что только ограниченное количество пользователей должно иметь доступ к конфигурированию системы.

Вы можете задать пользователей или группу пользователей и определить различные уровни доступа в User Administrator (администраторе пользователей). Эти уровни доступа могут быть связаны с элементами управления на кадрах.

Группам пользователей и пользователям могут быть присвоены различные права на индивидуальном уровне.

Спецификация

В примерах проектов *Project_C-Course* и *Project_TagHandling* каждый пользователь имеет право управлять процессами проекта.

В проекте *Project_CreatePicture* пользователи могут управлять процессами проекта только после регистрации. Паролем является название проекта (*Project_CreatePicture*).

Замечание относительно основных применений

Описание того, как присваивать различные права доступа, находится во второй части руководства по конфигурации в проекте *Project_CreatePicture*, глава Авторизация пользователей.

3.2.10 Спецификация: Alarming (регистрация аварийных сообщений)

Общая информация

- В основном, WinCC поддерживает две процедуры регистрации:
- Битовая процедура сообщения является универсальной процедурой, позволяющей оповещать о сообщениях из любой автоматизированной системы. WinCC контролирует изменение фронта выбранного битового тега и получает от него события.
- Последовательный отчет требует, чтобы автоматизированные системы сами генерировали сообщения и посылали их в заранее определенном формате в систему WinCC с временной меткой и, возможно, со значениями процесса. Это именно та процедура, которая делает возможным последовательное получение сообщений от различных автоматизированных систем. Смотрите главу Документация на систему аварийных сообщений S5.

Что должно быть сообщено?

При определении событий и состояний, о которых имеет смысл сообщать, большинство пользователей идут по наиболее безопасному пути и программируют систему на сообщения обо всех возможных событиях и изменениях состояний. Это ведет к тому, что оператору нужно выбирать какое сообщение он будет осматривать сначала, а какое потом.

Опыт показывает, что если сообщается о слишком большом количестве событий, то действительно важные события порой игнорируются.

Замечание относительно основных применений

Как отображать сообщения, и какие сообщения выбирать для архивирования, Вы должны решить в соответствии с требованиями Вашей задачи.

3.2.11 Спецификация: Для реализации

Общая информация

При реализации проекта для хранения данных вполне обоснованным является использование фиксированной структуры. Спецификация начинается с определения диска, на котором будет создан WinCC проект. Следующий шаг затрагивает структуру папок, и т.д.

Опыт показывает, что наиболее целесообразно хранить все данные проекта в одной директории, которая содержит соответствующие поддиректории. Преимущества данного метода дадут о себе знать при работе с проектом и при резервном копировании данных.

Замечание:

Конфигурации персональных компьютеров сильно отличаются друг от друга. Для того чтобы избежать проблем, возникающих при определении диска для расположения проекта, мы советуем использовать **виртуальные диски**. Назначение папки на виртуальный диск может быть изменено в любое время.

Определение папок

В дополнение к папкам, созданным WinCC, в случае необходимости создавайте специальные папки для файлов Word, Excel и временных файлов.

3.3 Особенности конфигурации WinCC

Следующие главы посвящены вопросам, касающимся всех аспектов конфигурирования WinCC.

Обсуждаемые ниже вопросы являются дополнением к службе помощи WinCC.

3.3.1 Циклы обновления – Где и как устанавливаются

Одной из наиболее важных процедур при конфигурировании системы визуализации является установка циклов обновления. Установка влияет на следующие свойства:

- структура кадра
- обновление состояния объекта в текущей экранной форме (Графический дизайнер)
- обработка фоновых сценариев (*Глобальные сценарии*)
- активизация менеджера данных и соединения с процессом

Другие временные переменные устанавливаются во время обработки измеренных значений (Регистрация тегов) в соответствии с времени архивации.

Менеджер данных

Текущие значения тегов запрашиваются менеджером данных в соответствии с установленными циклами обновления. См. главу Добавление динамических свойств в WinCC.

Менеджер данных получает новые значения процесса с помощью каналов связи и предоставляет эти значения приложениям. Таким образом, запрос данных подразумевает переключение между задачами (Графический дизайнер, менеджер данных, и тд.). В зависимости от конфигурации, это может привести к разнообразным системным загрузкам.

3.3.1.1 Обновление на кадре

Обновление кадра

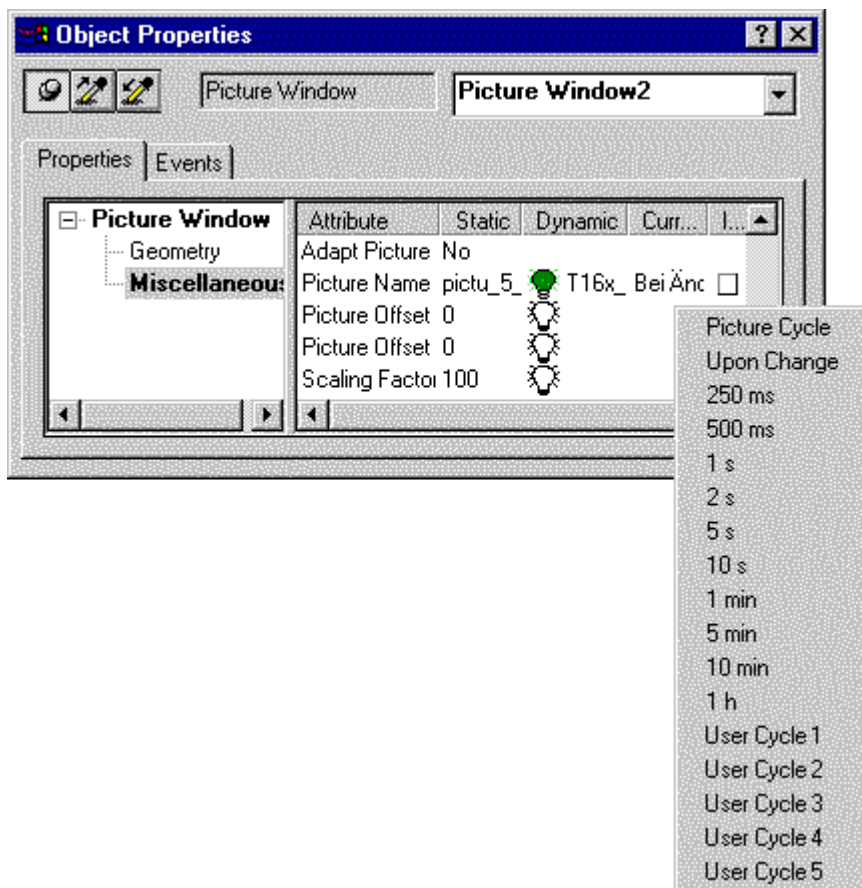
Обновление отдельных свойств объектов кадра относится к объектам, которые были динамически созданы после открытия кадра. Задача цикла обновления состоит в определении текущего статуса определенного объекта кадра. Циклы обновления объектов, созданных динамически, могут быть установлены конфигуратором или системой для следующих динамических типов:

Типы динамических свойств	Настройки по умолчанию	Индивидуальные настройки
Диалоговое окно конфигурации	Триггер тега 2 сек. Или Триггер события (например, управление)	Настройка временных циклов
Мастер динамики	<ul style="list-style-type: none"> • В зависимости от типа динамики вы можете выбрать: • Триггер события • Временной цикл • Триггер тега 	Настройка временных циклов, событий или тегов
Прямое соединение	Триггер события	

Типы динамических свойств	Настройки по умолчанию	Индивидуальные настройки
Соединение с тегом	Триггер тега 2 сек.	Настройка временных циклов
Диалоговое окно динамики	Триггер тега 2 сек.	Настройка временных циклов, триггеров тега
Процедура Си для свойств	Временной цикл 2 сек.	Настройка временных циклов, триггеров тега Прямое чтение из ПЛК
Свойство объекта	Установка, зависящая от свойства	Редактирование столбца циклов обновления

Циклы обновления заранее определены в WinCC и могут быть дополнены пользовательскими.

Выбор цикла обновления, например, для свойства объекта:



3.3.1.2 Типы циклов обновления

Существуют следующие разновидности циклов обновления:

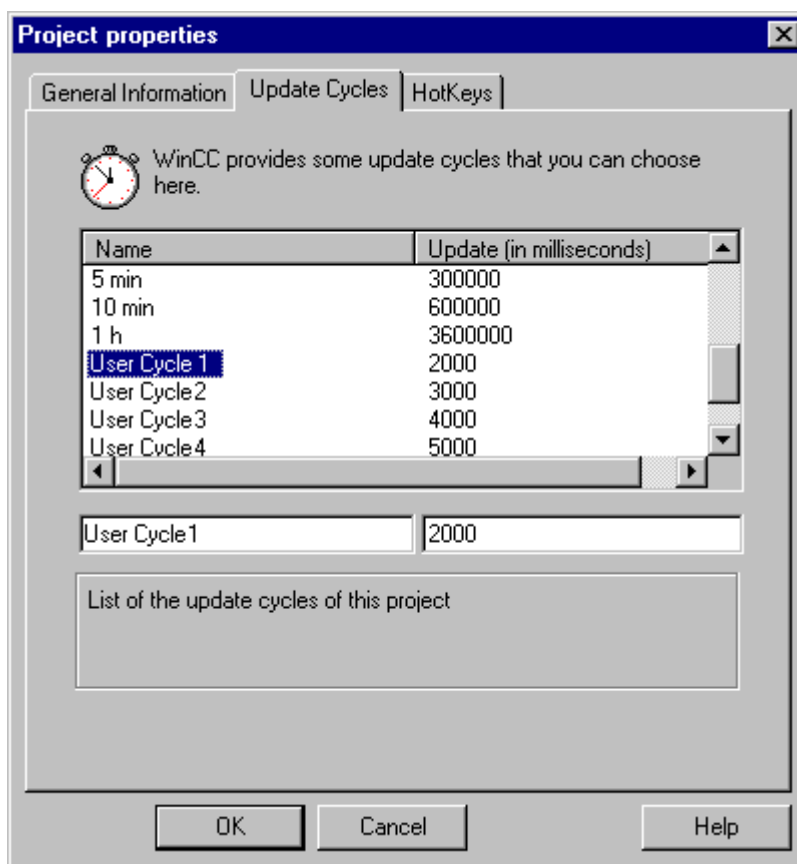
Тип	Настройки по умолчанию
Цикл по умолчанию	Временной цикл 2 секунды
Временной цикл	2 секунды
Триггер тега	2 секунды
Цикл кадра	2 секунды
Цикл окна	По изменению
Пользовательский временной цикл	Пользовательский цикл 1: 2 сек. Пользовательский цикл 2: 3 сек. Пользовательский цикл 3: 4 сек. Пользовательский цикл 4: 5 сек. Пользовательский цикл 5: 10 сек.

Пользовательский цикл

Пользователь может определить до 5 циклов относящихся к одному проекту. Если в левой части дерева *проводника WinCC* выбрано название проекта, то по нажатию на кнопку, изображенную снизу, открывается диалоговое окно *Project Properties (Свойства проекта)*.



На закладке *Update Cycles (Циклы обновления)*, пользователю предоставляется 5 пользовательских циклов, расположенных в конце списка всех циклов обновления. Впоследствии можно будет изменять только эти циклы.



Пользователь имеет возможность задать собственные временные циклы, отличные от уже имеющихся в системе (например, 200 мс).

Можно задавать цикл любой длительности от 100 мс до 10 часов. Вы можете задать циклу любое имя.

Эти временные параметры проекта могут использоваться для объектов, цикл обновления которого может быть изменен позже. Изменение временных циклов может пригодиться при оптимизации проекта. Пользовательские циклы обновления могут впоследствии пригодиться при модификации установленного временного цикла. В этом случае, отдельные объекты кадра не нужно больше настраивать. Вот почему этот метод установки временных циклов наиболее предпочтителен, если вы хотите сделать свой проект наиболее простым в эксплуатации.

3.3.1.3 Значение циклов обновления

Перед тем как вы начнете использовать циклы обновления, необходимо понять, что означают различные циклы обновления.

Для циклов обновления существуют следующие градации:

Тип	Значение
Цикл по умолчанию	Временной цикл
Временной цикл	В соответствии с установленным временем, свойство или процедура объекта будет обновляться. Это означает, что менеджер данных запрашивает каждый тег отдельно.
Триггер тега	В соответствии с установленным временным циклом для соответствующих тегов выполняется проверка на изменение значения. Если значение хотя бы одного выбранного тега меняется в течение временного интервала, то происходят изменения в свойствах или процедурах, зависящих от данного изменения. Все значения тегов запрашиваются менеджером данных одновременно.
Цикл кадра	Обновление свойств текущего объекта кадра и всех объектов, изменение которых связано с циклом обновления кадра
Цикл окна	Обновление свойств окна, а также всех объектов, изменение которых связано с циклом обновления окна.
Временные циклы, определенные пользователем	Могут быть определены специально для конкретного проекта.
Процедура Си прямого чтения из ПЛК	С помощью внутренних функций процедур Си значения могут быть считаны непосредственно из ПЛК. Дальнейшее изменение инструкций в процедуре Си могут быть продолжены только после того, как значения процесса считаны (синхронное чтение).

Замечание:

Запрос текущих значений тега менеджером данных всегда ведет к смене задачи или к переключению между задачами. К тому же, значения тега должны запрашиваться менеджером данных посредством канала связи с присоединенными программируемыми контроллерами. В зависимости от типа связи, это осуществляется с помощью телеграмм запроса, отправленных интерфейсу связи (FETCH) и телеграмм данных, отправленных контроллером для WinCC.


3.3.1.4 Информация относительно применения циклов обновления

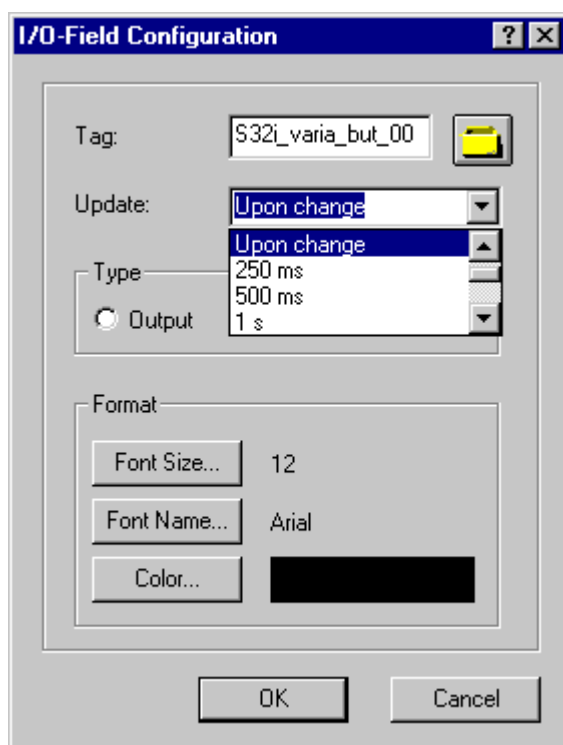
При применении циклов обновления рекомендуются следующие установки, зависящие от типа цикла:

Тип	Установки по умолчанию	Рекомендации по конфигурации
Цикл по умолчанию	Временной цикл в 2 секунды	<i>Диалоговое окно динамики или процедура Си:</i> Если теги взаимозависимы, Вы всегда должны использовать активизацию тегов. Это уменьшает количество переключений между задачами и обмен данными между задачами. Активизация тегов “по изменению“ может быть использована не во всех случаях, так как это может привести к увеличению загрузки системы! В этом случае теги постоянно проверяются на изменение. Для стандартных объектов мы рекомендуем цикл от 1 до 2 секунд.
Временной цикл	2 секунды	Нужно чтобы временной цикл зависел либо от самого объекта, либо от его свойства. Инерция компонентов процесса (заполнение резервуара или температуры в отличие от операций переключения) должны также приниматься во внимание. Для стандартных объектов мы рекомендуем цикл от 1 до 2 секунд
Триггер тега	2 секунды (для <i>диалога динамики</i>)	Если данную опцию обновления можно сконфигурировать (зависит от типа динамики), целесообразнее использовать именно ее. Если теги взаимозависимые, всегда принимайте во внимание теги, которые изменяются для свойства или запуска процедуры. Только содержащиеся в списке теги действуют как триггеры для свойства или процедуры, которые были созданы динамически. Активизацию тегов “по изменению“ следует использовать избирательно. Как только один из выбранных тегов изменился, триггер этого свойства или процедуры активизируется. Этот механизм опроса ведет к увеличению нагрузки на систему.
Цикл кадра	2 секунды	Этот цикл следует уменьшить лишь в том случае, если динамические свойства кадра меняются и следовательно требуют обновления чаще. Увеличение интервала обновления снижает нагрузку на систему.
Цикл окна	по изменению	Эти установки имеют смысл в том случае, если окно открыто, и используется, например, для настройки переменных процесса (окно переменных процесса). Если окно отображается постоянно (например, формат экрана), обновление окна и его содержимого должно быть установлено на активизацию тега или


Тип	Установки по умолчанию	Рекомендации по конфигурации
		временной цикл.
<i>Процедуру Си</i> чтения напрямую из ПЛК		Внутренние функции (например, GetTagWordWait) для синхронного чтения значений процесса (напрямую из ПЛК) следует использовать избирательно. Использование этих функций требует системного опроса (управления сценариями) и ведет к увеличению нагрузки на систему связи.

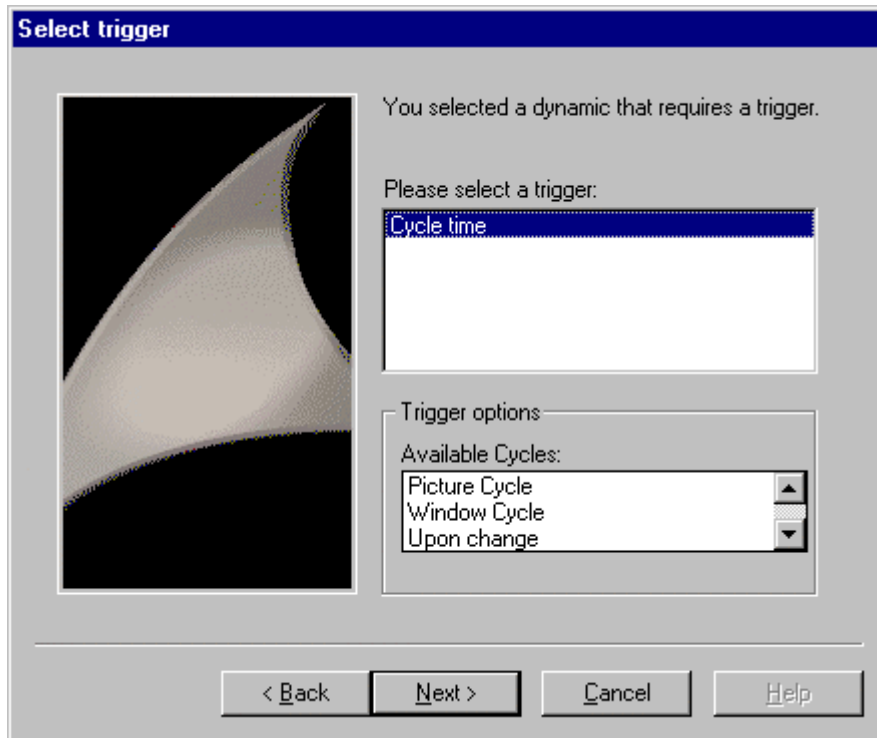
Диалоговое окно конфигурации

Отображается лишь в том случае, если сконфигурировано *Smart Object* (Интеллектуальные объекты) → *I/O Field* (поле ввода/вывода). Это окно также может быть запущено с помощью  (правой кнопки мыши) на соответствующем объекте.




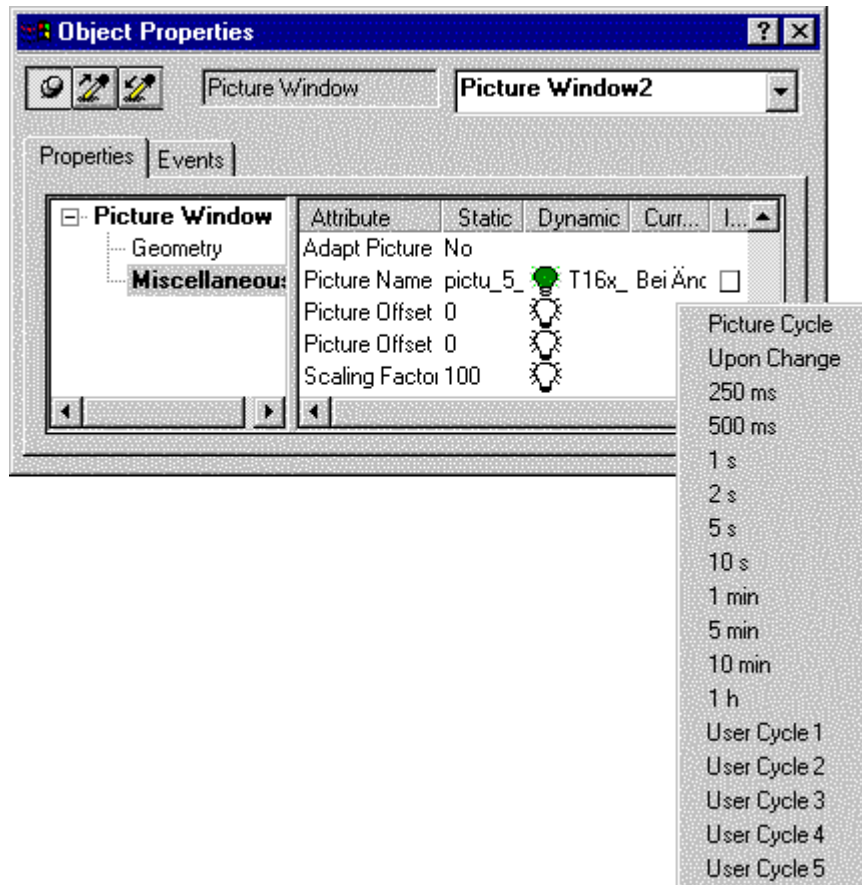
Мастер динамики

Эта страница отображается при  (двойном щелчке левой кнопки мыши) на элементе Make Property Dynamic на закладке Standard Dynamics в мастере динамики.



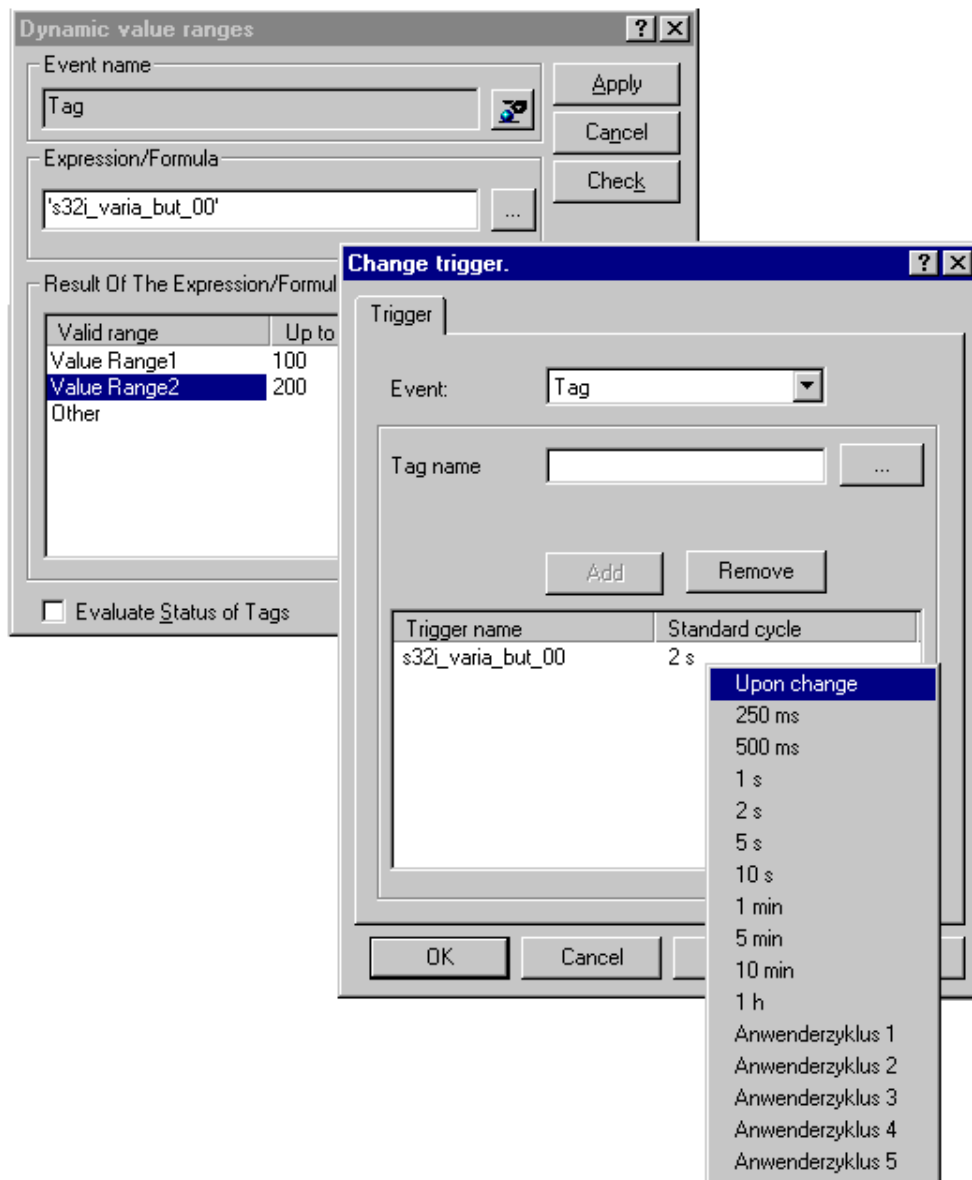
Связь с тегом для свойства объекта

Данное меню отображается выбором столбца Current (Текущий) для свойства объекта, которое было сделано динамическим с помощью тега, нажатием  (правой кнопки мыши).



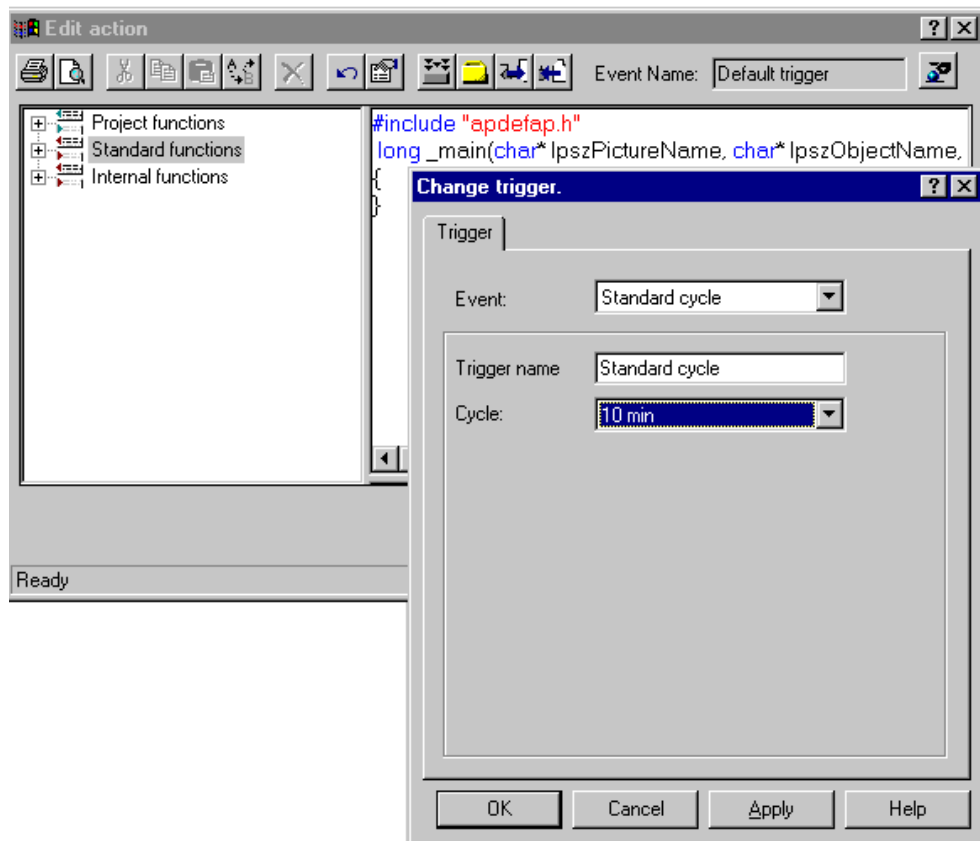
Диалоговое окно динамики

Если в диалоговом окне динамики Вы нажали на кнопку триггера, то появится диалоговое окно для изменения цикла обновления.



Процедура Си для свойства

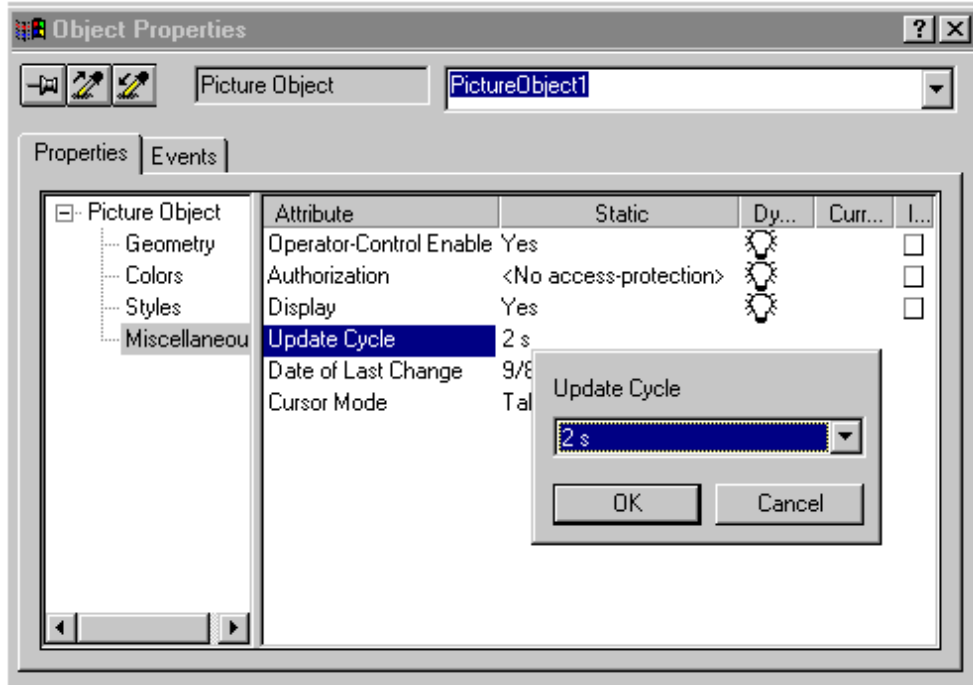
Если в редакторе конфигурации процедуры Си Вы нажали на кнопку триггера, выводится диалоговое окно цикла обновления.



Циклы обновления по умолчанию могут быть установлены следующим образом:

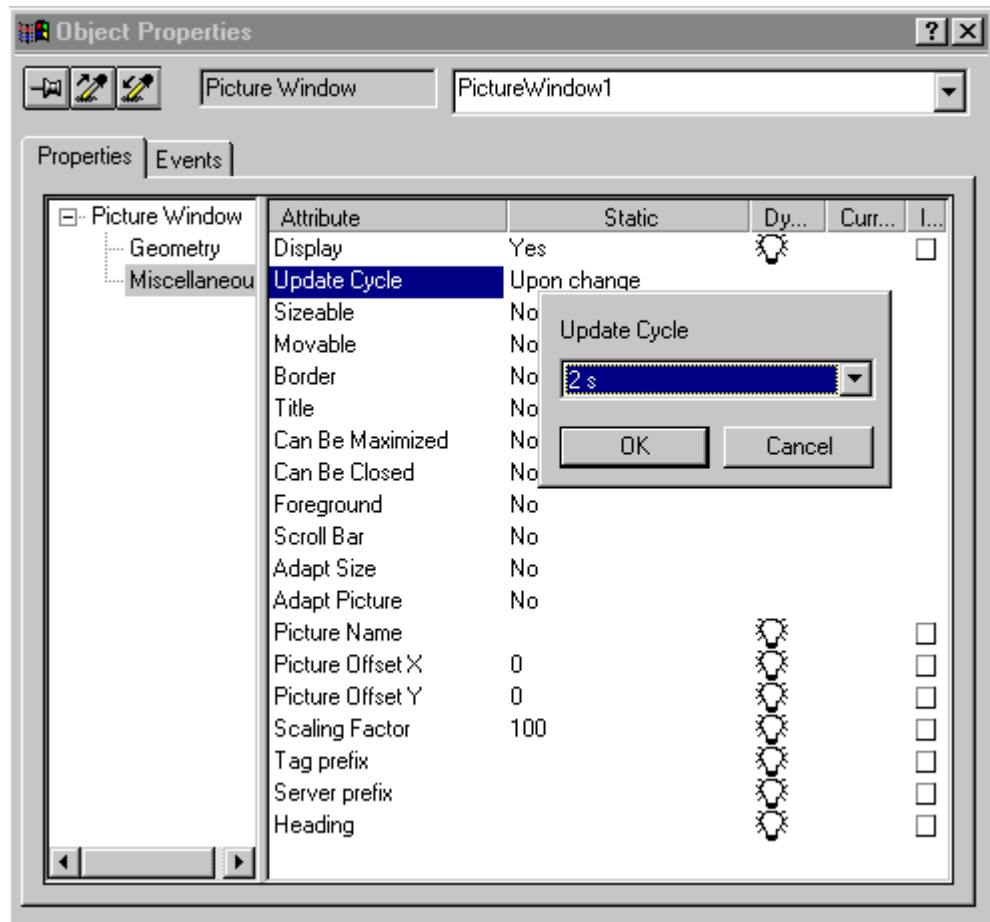
Цикл кадра

Изменение цикла кадра.



Цикл окна

Изменение цикла окна.



3.3.1.5 Исполнение фоновых сценариев (Global Script)

- Исполнение фоновых сценариев (Global Script) зависит от различных переменных в соответствии с конфигурацией:
- Временной триггер (циклическое выполнение, исключение: ациклическое = один раз)
- Временной цикл
- Время

Триггер события



или только один раз

Trigger

Event: Single

Trigger name: happy new year

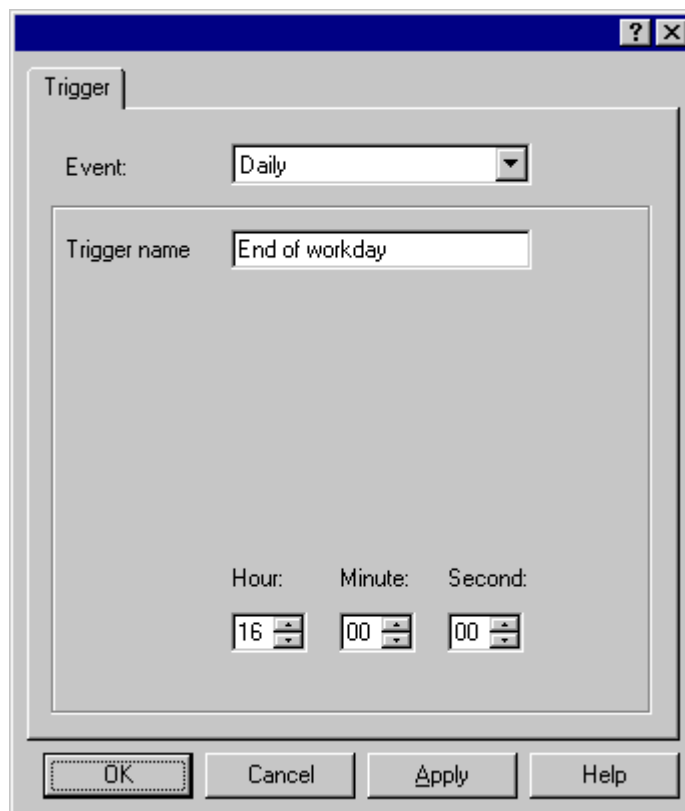
Month: 12 Day: 31 Year: 1997

Hour: 23 Minute: 59 Second: 59

OK Cancel Apply Help

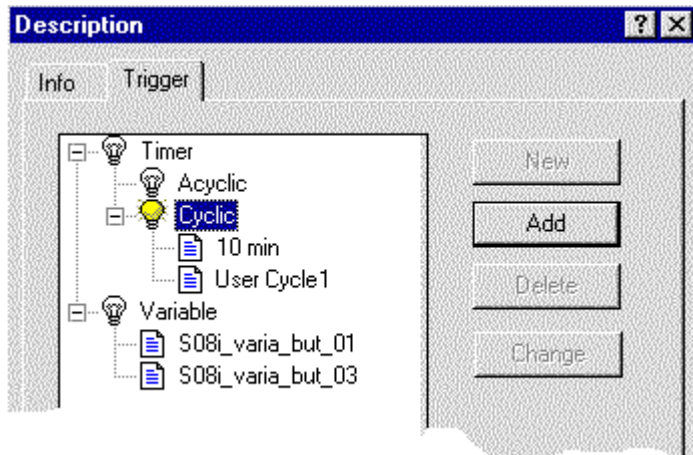
Временной цикл

- Сконфигурированная временная составляющая глобальной процедуры определяет, когда выполняется некоторая последовательность процедур. В дополнение к уже описанным циклам по умолчанию и ассоциированным временным установкам от 250 мс до 1 часа (или определенные пользователем циклы 1-5) могут быть выбраны следующие триггеры:
- Ежечасно (минуты и секунды)
- Ежедневно (часы, минуты, секунды)
- Еженедельно (дни недели, часы, минуты, секунды)
- Ежемесячно (дни, часы, минуты, секунды)
- Ежегодно (месяцы, дни, часы, минуты, секунды)

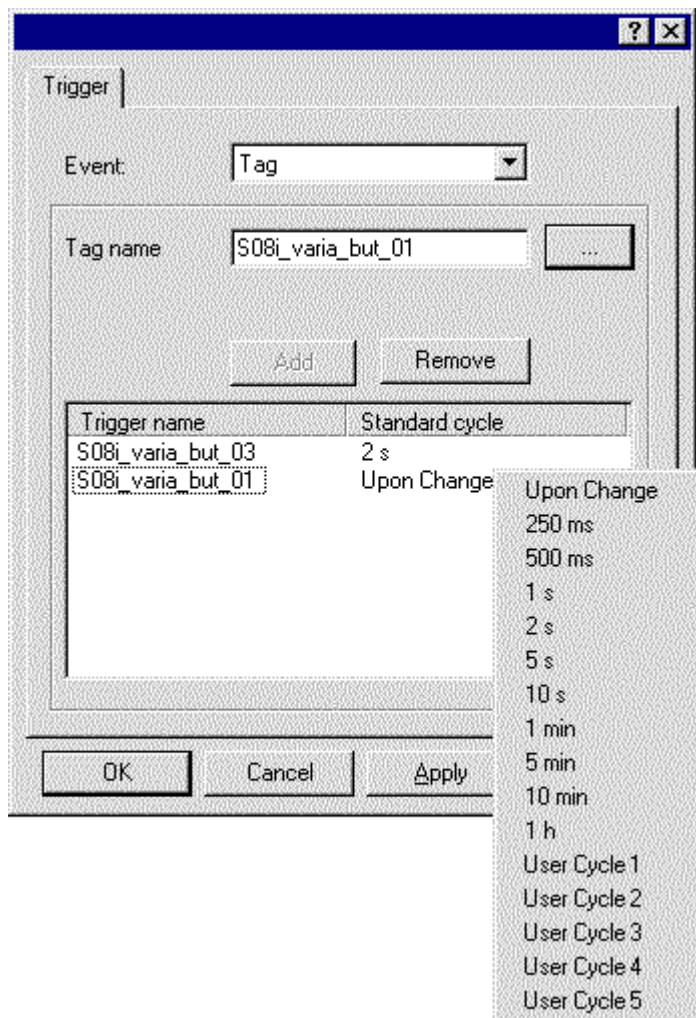


Триггер тега

Если активизация процедуры зависит от одного или более тегов, то триггер события должен быть установлен как триггер тега. Это делается аналогично триггеру тега для свойств объекта.



По умолчанию устанавливается цикл в 2 секунды. Человек, осуществляющий конфигурацию, однако, может устанавливать следующие временные характеристики:



В начале и в конце отрезка времени происходит определение значений выбранных тегов. Если значение по крайней мере одного из них изменилось, триггер глобальной процедуры активизируется.

При использовании триггеров “по изменению” необходимо не забывать об увеличении загрузки системы. Использование таких триггеров не всегда оправдано. Здесь целесообразно прислушаться к советам, относящимся к обновляемому объекту. Все глобальные процедуры не могут активизироваться по изменению состояния объекта, т.е. они могут зависеть только от временных циклов и установленных триггеров событий. По этой причине, используйте глобальные процедуры избирательно и избегайте в них ненужных операций для предотвращения перегрузок системы. Для выполнения процедур никогда не используйте большое количество коротких временных циклов.

3.3.2 Добавление динамики в WinCC

Определения

Термин добавление динамики означает изменение статических свойств (например, позиции, цвета, текста, и тд.) и реакции на события (например, нажатие кнопки мышки, клавиатуры, изменение значения, и тд.) во время исполнения.

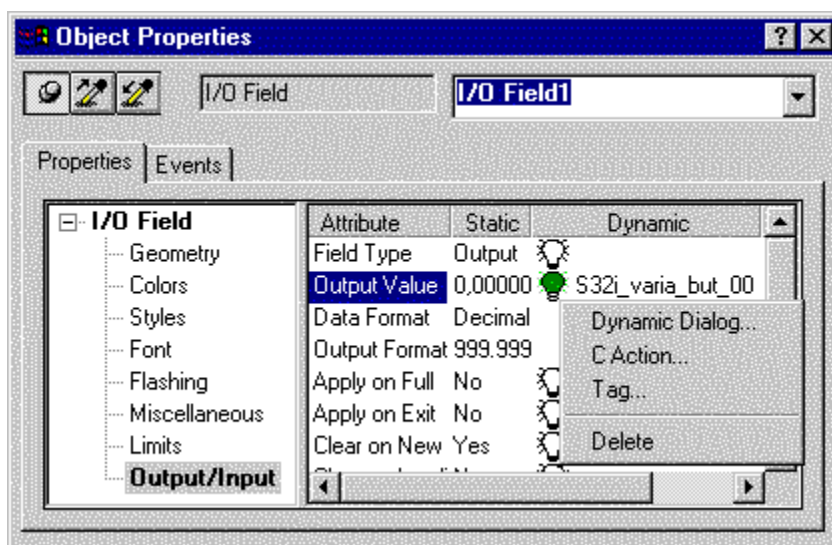
Каждый элемент окна рассматривается как независимый объект. Само окно является объектом типа *Picture Object(объект кадра)*.

Каждый объект в графической системе WinCC имеет *Свойства* и *События*. За некоторым исключением, эти свойства и события можно сделать динамическими. Небольшое число исключений касается *Свойств* и *Событий*, не использующихся в процессе исполнения. Они не имеют индикатора динамики.

3.3.2.1 Превращение свойств в динамические

Свойства объекта (позиция, цвет, текст, и тд.) можно установить как статически, так динамически во время исполнения.

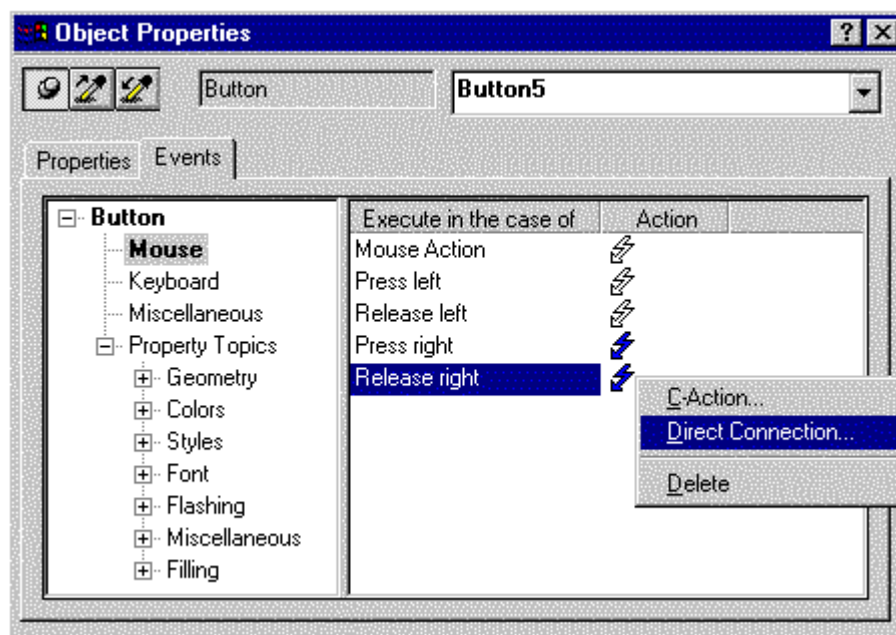
Все свойства с лампочкой в колонке Dynamic могут быть превращены в динамические. Если свойство было сделано динамическим, то вместо лампочки в колонке отображается символ, зависящий от типа динамики. Темы (например, геометрия), созданные динамически, изображаются жирным шрифтом.



3.3.2.2 Превращение событий в динамические

События объекта (например нажатие кнопки мыши, клавиатуры, изменение значений, и тд.) могут быть определены во время исполнения.

Все события с подсвеченным символом молнии в колонке Action можно сделать динамическими. Если событие было сделано динамическим, то в зависимости от типа динамики, отображаемый символ молнии меняет свой цвет. Темы (например, Разное), созданные динамически, отображаются жирным шрифтом.



3.3.2.3 Типы динамизации для объектов

Существуют различные способы динамизации объектов кадра. Различные стандартные диалоговые окна, в которых выполняется динамизация ориентированы на различные предметные области и призваны решать различные задачи.

Обзор




Тип динамики	А	В	Преимущество	Недостаток
Мастер динамики	x	x	Стандартный предлагаемый при конфигурации метод	Только для некоторых типов динамики. Всегда создает процедуру Си!
Прямое соединение		x	Самый быстрый метод, высочайшая производительность во время исполнения	Ограничен только одним соединением и может применяться только для кадра.
Соединение с тегом	x		Легко конфигурируется	Ограниченные опции для динамизации
Диалоговое окно динамики	x		Быстро и понятно; для области значений и набора альтернатив; высокая производительность	Не для всех типов динамизации.
Процедура Си	x	x	Практически безграничные возможности для динамизации благодаря мощному языку сценариев (ANSI-C)	Возможность ошибки из-за неправильных Си-инструкций Производительность ниже по сравнению с другими типами, поэтому рекомендуется проверять сможете ли вы достигнуть цели, используя различные типы динамики.

Обозначения:

А: Динамизация свойства объекта

В: Динамизация события объекта

Запуск диалоговых окон для динамизации

Диалоговое окно	Команда запуска
Окно конфигурации	<p>Не у всех объектов есть это окно. Автоматически при создании этих объектов. Выберите объект в кадре → нажмите и держите клавишу SHIFT →  D (двойной щелчок левой кнопкой мыши) на объекте. Выберите объект в кадре →  R (щелчок правой кнопкой мыши) на объекте для открытия всплывающего меню → выберите диалоговое окно конфигурации</p>
Мастер динамики	<p>Выберите объект на кадре → выберите событие или свойство → выберите мастер  D на мастере для его запуска. Мастер должен быть помечен в меню View (Вид) → Toolbars (Панели инструментов)</p>
Прямое соединение	<p>Выберите объект на кадре → отобразите его свойства → выберите закладку событий →  R на колонке процедур для отображения меню → выберите прямое соединение.</p>
Соединение с тегом	<p>Выберите объект на кадре → отобразите его свойства → выберите закладку свойств →  R на колонке динамики для вывода меню → выберите тег → в диалоговом окне, выберите и примените соответствующий тег.</p>
Диалоговое окно	<p>Выберите объект на кадре → отобразить его свойства → выберите закладку свойств →  R на колонке динамики для отображения меню → выберите диалоговое окно динамики → в диалоговом окне, сконфигурируйте и примените соответствующую динамику</p>
Процедура Си	<p>Выберите объект на кадре → отобразить его свойства → выберите закладку свойств или событий →  R на колонке динамики или процедур для отображения меню → выберите процедуру Си → сконфигурируйте и скомпилируйте соответствующую процедуру.</p>

Результаты и Визуальные изменения

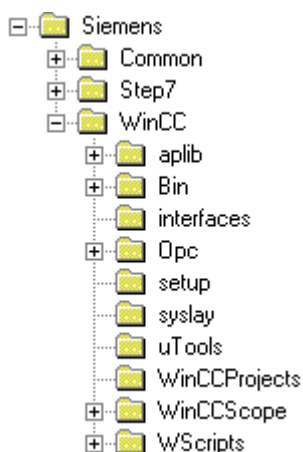
Диалоговое окно	Результат	Визуальные изменения (Символ)
Мастер динамики	Создание процедуры Си.	Зеленая молния
Прямое соединение		Синяя молния
Соединение с тегом		Зеленая молния
Диалоговое окно динамики	Автоматическое создание процедуры Си (InProc), эта процедура может быть дополнена, но в этом случае производительность снизится.	Красная молния В зависимости от изменений в процедуре меняется на зеленую молнию
Процедура Си	Сконфигурированный сценарий Си	Зеленая молния Желтая молния – процедура все еще компилируется

3.3.3 Системное окружение WinCC

По умолчанию, WinCC устанавливается в каталог `C:\Siemens\WinCC\`. При установке вы можете изменить этот путь.

3.3.3.1 Структура каталогов WinCC

Структура каталогов WinCC – без опций и примеров – выглядит следующим образом:



Файлы в каталоге WinCC по умолчанию

В каталоге WinCC имеются следующие важные для разработчика файлы и директории:

Директория	Имя файла, Расширение	Комментарии
Diagnostics	License.log	Текущие сведения относительно проверки лицензии и/или нарушений.
	License.bak	Файл протокола с информацией о лицензии созданный с момента последнего запуска системы.
	WinCC_Op_01.log	Сообщения оператора, сгенерированные WinCC во время исполнения.
	WinCC_Sstart_01.log	Системные сообщения, сгенерированные WinCC при запуске. Важный файл при поиске неисправностей. Файл содержит сообщения о пропущенных тегах и неправильно запущенных сценариях.
	WinCC_Sys_01.log	Системные сообщения, сгенерированные WinCC во время выполнения. Важный файл при поиске неисправностей. Файл содержит сообщения о пропущенных тегах и неправильно запущенных сценариях.

Директория	Имя файла, Расширение	Комментарии
	S7chn01.log	Системные сообщения об используемом канале (в данном случае S7)
aplib	Путь к библиотеке	Все заголовочные файлы, стандартные и внутренние функции содержатся в поддиректориях.

Файлы в каталоге WinCC по умолчанию

В каталоге WinCC, глобальные функции и символы содержатся в следующих поддиректориях:

Директория	Поддиректория, Имя файла	Комментарий
Aplib	library.pxl	Символы библиотеки по умолчанию WinCC.
	Report, Wincc, Windows	Директории для стандартных функций; могут быть скорректированы в любое время.
	Allocate, C_bib, Graphics, Tag	Директории для внутренних функций; не могут быть скорректированы.
syslay		Все компоновки страниц автоматически копируются WinCC в папку <i>prt</i> в директории проекта во время его создания.
wscripts	Dynwiz.cwd	<i>Динамический мастер графического дизайнера</i> . Вы можете создавать собственные сценарии. Им присваивается расширение .wnf.
	wscripts.deu	Здесь содержатся файлы сценариев для <i>немецкого языка</i> . Этот путь зависит от установленного языка.
	Wscripts.enu	Здесь содержатся файлы сценариев для <i>английского языка</i> . Так как английский язык является языком по умолчанию, то данный путь создается всегда.
	Wscripts.fra	Здесь содержатся файлы сценариев для <i>французского языка</i> . Этот путь зависит от установленного языка.

Файлы в каталоге WinCC по умолчанию

Во время установки WinCC, в ниже перечисленные папки копируются следующие приложения:

Папка\Файл	Комментарии
\sqlany\isql.exe	Интерактивная программа, используемая для просмотра информации в базе данных проекта WinCC.
\bin\Wunload.exe	Мастер, используемый для очистки online таблиц в базе данных проекта WinCC, например, удаляет сообщения и сохраненные значения. Мастер автоматически предоставляет runtime таблицы для выгрузки; в любое время пользователь может удалить или добавить дополнительные таблицы. Этот инструмент должен использоваться в то время, когда проект WinCC находится в режиме offline. Его нельзя использовать во время исполнения. Сообщения и измеренные значения могут экспортироваться во время выполнения с помощью дополнительного пакета STORAGE.
\bin\Wrebuild.exe	Мастер для восстановления базы данных; не может быть использован во время выполнения!
\SmartTools\CC_GraficTools\metaVw.exe	Программа просмотра графических файлов (например, задания для печати, экспортируемые символы) в формате EMF (extended meta file).
\SmartTools\CC_GraficTools\wmfdcode.exe	Программа просмотра графических файлов в формате WMF (windows meta file).
\SmartTools\CC_OCX_REG\ocxreg.exe	Программа предназначена для регистрации или разрегистрации дополнительных элементов управления OLE (OCX).
\SmartTools\CC_OCX_REG\Rsgsvr32.exe	Вызывается программой ocxreg.exe.

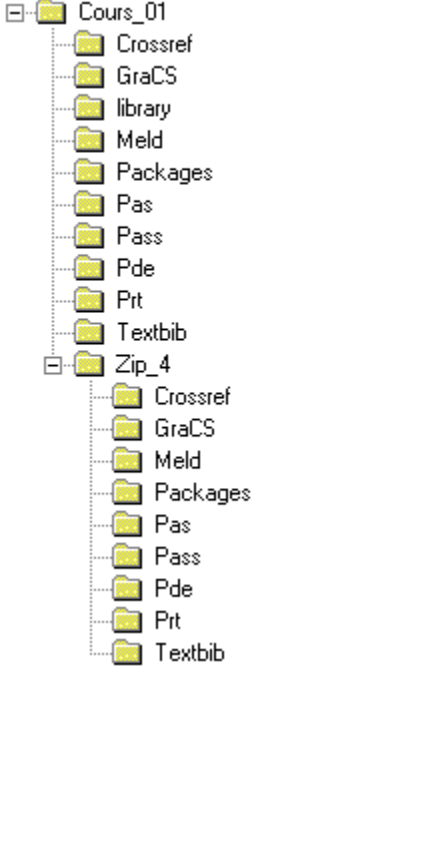
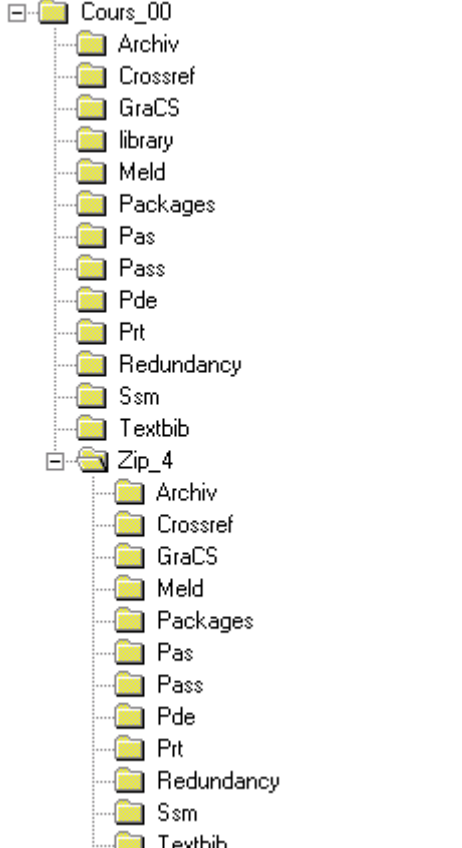
3.3.4 Среда проекта WinCC

Замечания:

Для проектов WinCC рекомендуется создавать специальную директорию, например, WinCC_Projects. В этом случае, система WinCC и сконфигурированные данные хранятся отдельно друг от друга, упрощая тем самым процесс восстановления данных. Также Вы предотвратите возможную потерю данных (из-за ошибки оператора) в случае деинсталляции WinCC.

3.3.4.1 WinCC проект – Структура папок

Проект WinCC представляет определенную структуру папок с соответствующим содержимым. После создания нового проекта в WinCC Explorer (с помощью пункта меню File (Файл) -> New (Создать)), создается новая структура папок:

Стандартная система WinCC	WinCC с опциями
 <ul style="list-style-type: none"> [-] Cours_01 <ul style="list-style-type: none"> Crossref GraCS library Meld Packages Pas Pass Pde Prt Textbib [-] Zip_4 <ul style="list-style-type: none"> Crossref GraCS Meld Packages Pas Pass Pde Prt Textbib 	 <ul style="list-style-type: none"> [-] Cours_00 <ul style="list-style-type: none"> Archiv Crossref GraCS library Meld Packages Pas Pass Pde Prt Redundancy Ssm Textbib [-] Zip_4 <ul style="list-style-type: none"> Archiv Crossref GraCS Meld Packages Pas Pass Pde Prt Redundancy Ssm Textbib

Содержимое папок проекта

Папка	Расширение	Комментарии
Корневая директория проекта	.db	База данных с конфигурационной информацией.
	rt.db	База данных с runtime информацией, измеренные значения и сообщения.
	.mcp (master control program)	Основной файл проекта WinCC. Проект открывается с этим файлом.
	.pin	Project.pin
GraCS	.pdl (picture design language)	Сконфигурированные кадры.
	.sav	Резервные файлы кадров и последней конфигурации.
	.gif (bitmap), .wmf (windows meta file), .emf (extended meta file)	Файлы рисунков
	.act (action)	Экспортированные <i>процедуры Си</i>
	.pdd	Default.pdd Параметры установки графического редактора (установки по умолчанию отдельных объектов в палитре объектов)
Library	.h (header)	<i>Ap_pbib.h</i> (Объявление функций проекта)
	.pxl	<i>Library.pxl</i> (Библиотека символов проекта)
	.fct	Функции проекта
	.dll (dynamic link library)	Библиотеки функций созданные в среде разработки Си.
Meld		
Pas	.pas (action definition)	Процедуры проекта, работающие как фоновые процедуры в зависимости от установленного триггера.
Pass		
Pde		
Prt	.rpl (report picture language) .rpl (line layout)	Компоновки страниц для заданий на печать. Все предопределенные компоновки WinCC начинающиеся с @. Все системные переменные (включая теги) определяются этим префиксом.
Название компьютера , например, Zip-ws1	\GraCS\GraCS.ini	Файл инициализации графического редактора.

Необязательное: Файлы, которые могут быть созданы во время конфигурации

Папка	Расширение	Комментарии
До некоторой степени свободное определение	.ini	Файл инициализации симулятора с информацией для вызова.
	.sim	Внутренние теги с установками для симуляции.
	.csv	Экспортируемые тексты из текстовой библиотеки.
	.txt	Экспортируемые сообщения из системы сообщений (<i>Alarm Logging</i>).
	.emf	Задания на печать, которые пишут результаты печати в файл.
	.log	Файлы протоколов
	.xls .doc .wgi	Файлы, созданные другими приложениями, но при этом используемые проектом WinCC.

3.3.5 Автоматический запуск проекта в WinCC

Требование


Система с человеко-машинным интерфейсом (WinCC) на производстве должна запускаться автоматически при старте Windows. Оператору не должен ничего знать о системе Windows (например, активация WinCC из Windows NT).

Решение

WinCC запускается автоматически во время загрузки ПК. Эта функция устанавливается в папке Startup каталога Windows.



Создание соединения

Шаг	Процедуры в системе NT 4.0:
1	В проводнике Windows перейдите в каталог <i>WinNT\Profiles\All Users\Start Menu\Programs\Startup</i> . WinNT – папка, куда установлена Windows NT.
2	В этой папке, создайте новое соединение с помощью  (правой кнопки мыши) → <i>New (Создать)</i> → <i>Connection (Соединение)</i> .
3	Установите соединение с программой <i>mcp.exe</i> (Главная управляющая программа) в каталоге <i>\bin</i> WinCC.
4	Задайте название для соединения.

В результате, WinCC будет запускаться автоматически. Сама WinCC автоматически запустит последний активный проект.

Чтобы запустить систему в режиме исполнения, нужно выйти из проекта в то время, когда он активен.

Замечания:

Если комбинация клавиш *CTRL + SHIFT* не заблокирована и нажата во время запуска WinCC, WinCC запускается в режиме конфигурации.

Оператор видит знакомое окно запуска системы. Чтобы предотвратить случайное или намеренное переключение на конфигурацию (работающую в фоновом режиме) или на приложение Windows должны быть приняты соответствующие меры. Операторы не должны иметь возможность открыть окно базы данных исполнения до тех пор, пока не закрыто соединение с базой данных WinCC.

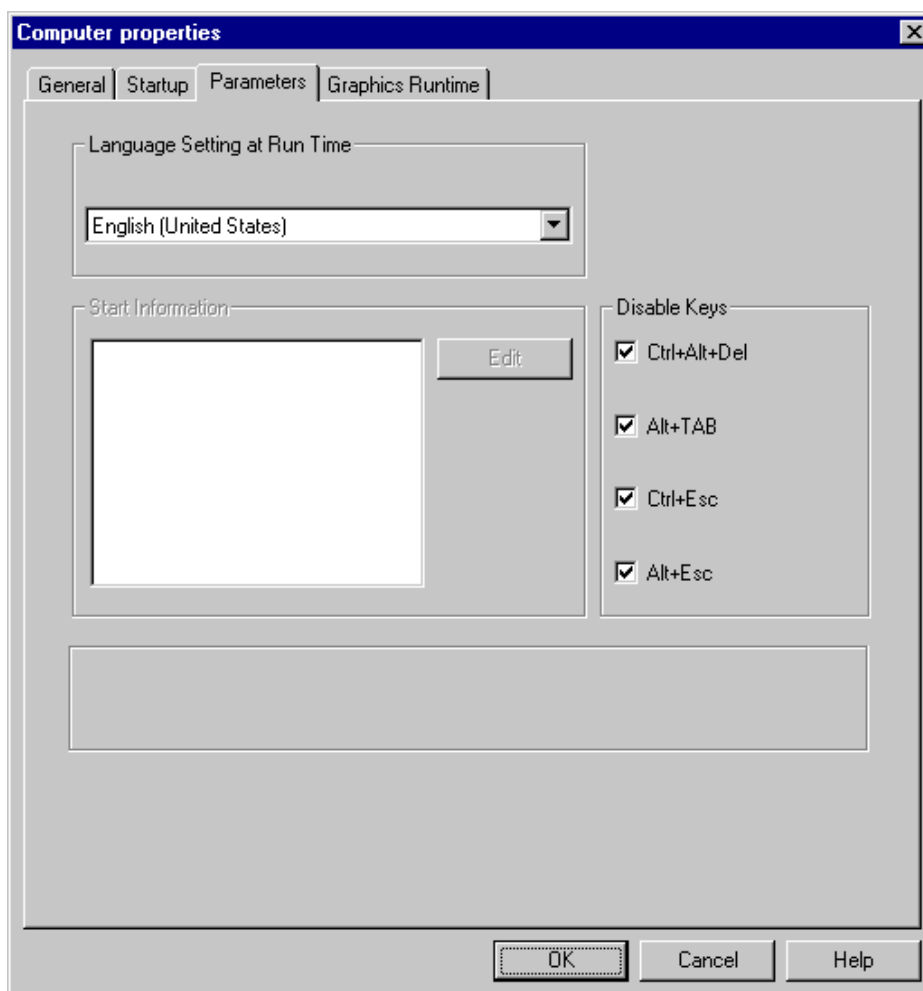
Нельзя выбирать:

Операторы не должны иметь возможность переключаться с WinCC на:

- *WinCC Explorer* системы WinCC (среда конфигурации)
- Окно исполнения базы данных WinCC (Sybase SQL Anywhere), так как таким образом они могут завершить соединения с базой данных WinCC. В этом случае WinCC не сможет более функционировать
- Панель задач Windows, так как она используется для запуска всех установленных приложений
- Диспетчер задач, так как приложение может быть закрыто оттуда

Требуемые настройки компьютера

Следующая комбинации клавиш должна быть заблокирована чтобы запретить операторам производить указанные операции.

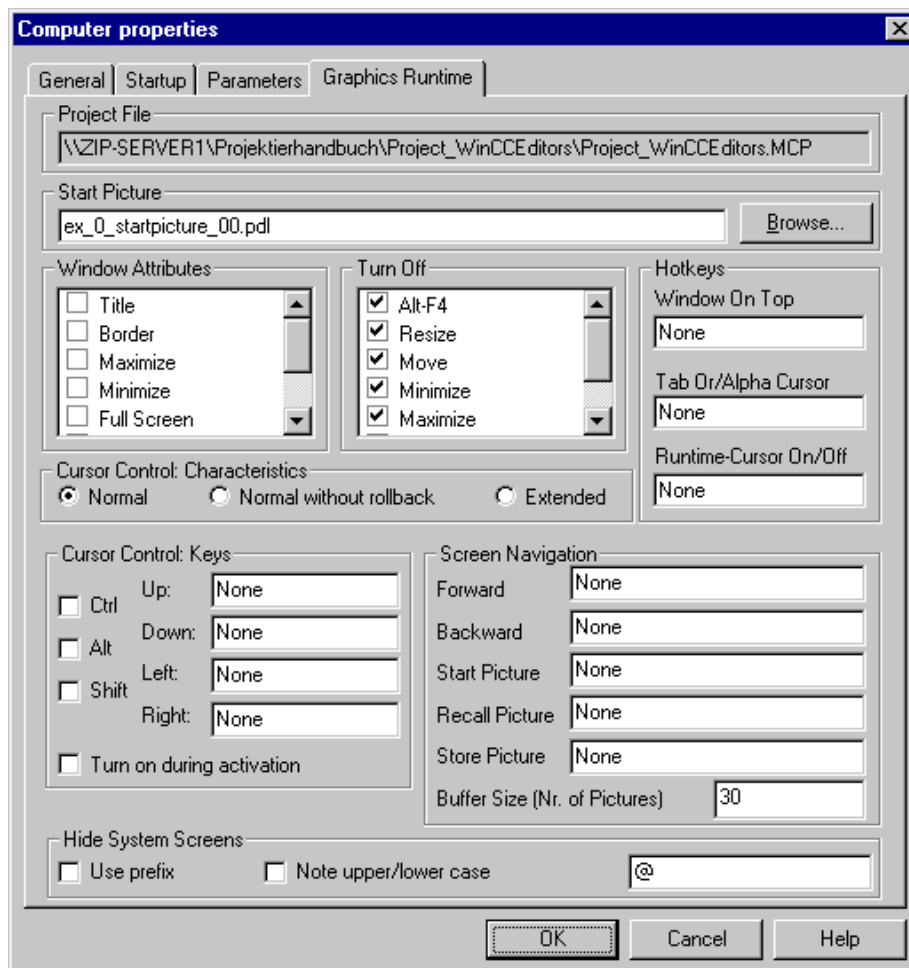


Комбинация клавиш в WinCC Explorer блокируется из диалогового окна Computer Properties (свойств компьютера).

Точное определение комбинаций клавиш можно найти в службе помощи WinCC или в справке операционной системы.

Требуемые настройки режима runtime

Технологический кадр также можно закрыть с помощью стандартных клавиш Windows; таким же образом можно выйти из WinCC. Для обеспечения такой функциональности необходимо выполнить следующую настройку свойств кадра:



Блокировки устанавливаются в *WinCC Explorer* в диалоговом окне *свойств компьютера*.

Точное определение комбинаций можно найти в службе помощи WinCC или в справке операционной системы.

- Если свойства *Resize (Изменение размеров)* и *Minimize (Минимизация)* не выключены, то в этом случае оператор может получить доступ к интерфейсу операционной системы.
- *Close (Закрытие)* (не указано на кадре выше) должна быть также деактивирована, иначе пользователи смогут выходить из системы во время исполнения и получить доступ к системе конфигурации.

Замечания:

Если упомянутые выше клавиши заблокированы частично или полностью, доступ к конфигуратору к системе конфигурации должен быть предоставлен через

специальную клавишу. Это также относится к корректному завершению работы системы.

Эти функции не должны быть легко доступными. Также необходимо предусмотреть защиту доступа с помощью установки пароля.

3.3.6 Координированное завершение работы WinCC

Обеспечение корректной работы системы

Для поддержания системы в рабочем состоянии очень важно правильно завершать работу WinCC. Выход из WinCC во время исполнения должен осуществляться только соответствующей клавишей. Только после этого необходимо закрыть проект в WinCC Explorer.

Если такой последовательности не придерживаться, то могут возникнуть потери данных, повреждения базы данных или, в худшем случае, отказ системы.

Контролируемый выход

Нельзя выключать станция WinCC без завершения работы системы. Использование КРИТИЧЕСКОГО ЗАВЕРШЕНИЕ РАБОТЫ неприемлемо для системы с человеко-машинным интерфейсом. Поэтому, должна быть сконфигурирована специальная клавиша, чтобы оператор мог выходить из системы корректно и при этом не требовалось дополнительных навыков.

Отказ питания

Во избежание потери данных из-за отказа питания, необходимо проработать вопрос о резервном копировании информации для систем с человеко-машинным интерфейсом.

В любом случае, необходима установка блока UPS (бесперебойного источника питания). Это может быть осуществлено присоединением станции к заводскому UPS или использованием отдельного UPS для сервера WinCC. Это относится как к многопользовательской системе, так и к системе с одним пользователем независимо от используемой операционной системы (Windows NT).

С используемым UPS должно поставляться специальное программное обеспечение для завершения работы Windows NT, которое в случае отказа питания после некоторого интервала автоматически завершает работу операционной системы и всех активных приложений без потери данных; например, APC UPS 600 с программой завершения работы Windows NT.

3.3.6.1 Замечания по установке UPS

Станция WinCC должна иметь последовательный интерфейс, к которому подключается UPS с соответствующей тестирующей программой. Если на станции нет свободного последовательного интерфейса (например, все занято под принтеры или ПЛК), необходимо установить дополнительную интерфейсную карту. Последовательные интерфейсы, используемые для соединения двух и более внешних устройств (например, с помощью переключателей) не поддерживаются большинством UPS и не рекомендуются.

На рабочую станцию устанавливается контролирующая служба. Эту службу необходимо настроить, задав ей параметры завершения работы системы так, чтобы гарантировать координированный и надлежащий выход из нее. Этот процесс завершения работы для приложений должен быть активизирован при любых обстоятельствах для того, чтобы WinCC завершала работу без потери данных. Вы должны выбрать время сохранения перед завершением работы, которое должно быть достаточно большим, чтобы успели завершиться все приложения.

Большинство программного обеспечения для UPS также предлагают *Завершение работы* по времени, например, на выходные или ночью. Эта функция может использоваться для **преднамеренного завершения работы WinCC без привлечения оператора.**

3.3.7 Резервное копирование данных

Где должна создаваться резервная копия?

- Проект WinCC необходимо постоянно сохранять и создавать резервную копию в следующих случаях:
- Во время конфигурации.
- Перед экспортом/импортом данных (например, при импорте тегов, текстов многоязыковых кадров, текстов сообщений).
- Перед формированием и выгрузкой базы данных WinCC.
- Перед редактированием базы данных при помощи интерактивного SQL доступа.
- Для конфигурационной информации должна быть создана резервная копия перед установкой конечному пользователю.
- Перед перекачкой данных в проект с подобной структурой.

Какие носители подходят?

Носитель	Преимущество	Недостаток
Гибкий диск	Может быть прочитан любой системой.	Недостаточная емкость (даже если данные заархивированы).
ZIP диск	Недорогой, достаточной емкости, прямой и быстрый доступ с помощью Windows; легко устанавливается; переносимый для использования на производстве.	
Стримеры (например, сетевые)	Возможность автоматического резервного копирования (ежедневно), очень большая емкость.	В основном доступен в офисе, нет прямого доступа к данным из-за специального формата хранения.
Винчестер на другом компьютере (например, Laplink)	Не нужно использовать другие носители, данные можно использовать напрямую.	Медленный способ, не подходит для больших объемов данных.
MOD (Магнито – оптический диск)	Высокий уровень интегрированности данных, для многократного использования, копирование сообщений и измеренных значений возможно во время исполнения.	Необходимо специальное устройство для чтения и записи.
CD-ROM	Большая емкость, может быть прочитан любой системой, подходит для долгосрочного архивирования.	Необходимо специальное устройство для записи, не многократного использования.

Очистка проекта перед резервным копированием данных

Чтобы **очистить** проект и сохранить его наименьшего размера перед резервным копированием для конечного потребителя или переноса данных проекта, можно уничтожить следующую информацию.

- Все файлы резервных копий в папке *\GraCS*.sav* .проекта

Если вы не создавали собственных компоновок (дизайнер отчетов) для документирования, можно уничтожить системные компоновки в папке *\Prt*. При создании нового проекта, все системные компоновки автоматически копируются из папки *\Siemens\WinCC\syslay* в папку проекта.

Для какой информации нужно создавать резервную копию?

Если нужно создавать копию только для данных проекта WinCC, то в резервную копию необходимо включить следующие файлы и папки со всеми файлами в них.

- Из папки проекта:
- файлы **.mcp, *.pin, *.db*
- папки *\GraCSand \Library*
- если были созданы собственные процедуры, папка *\Pas*
- если были созданы собственные компоновки, папка *\Prt*

Если Вы создавали глобальные компоненты (стандартные функции, объекты в библиотеке проекта), нужно создать копию для следующих файлов из

- папка WinCC по умолчанию:
- файлы *\Siemens\WinCC\aplib*.fct*
- файл *\Siemens\WinCC\aplib\library.pxl*

Эти данные **не** создаются при переустановке WinCC.

3.3.8 Перенос резервной копии проекта WinCC на другой компьютер

Установка системы

Система WinCC устанавливается на интегрированные системы как с помощью диалогового окна конфигурации, которое вызывается автоматически, так и с помощью установочного CD и лицензионных дискет из пакета WinCC.

Добавления

Если Ваш проект использует дополнительные пакеты (например, пакеты опций), специальные интерфейсы связи или интерфейсы с другими Windows приложениями (например, WORD, EXCEL, и т.д.), эти пакеты также должны быть установлены на компьютер. Соответствующая авторизация для пакетов опций, добавления или интерфейс связи (DLL каналов) также нужно установить. На новую станцию также нужно импортировать все необходимые авторизации (для всех использующихся DLL каналов) для того, чтобы можно было работать с проектом WinCC.

Программное обеспечение Windows

Если в кадрах WinCC используются OLE соединения с другими Windows приложениями – например, WORD, ClipArt или EXCEL, то соответствующий программный пакет необходимо установить на целевой компьютер, т.е. внесен в реестр Windows.

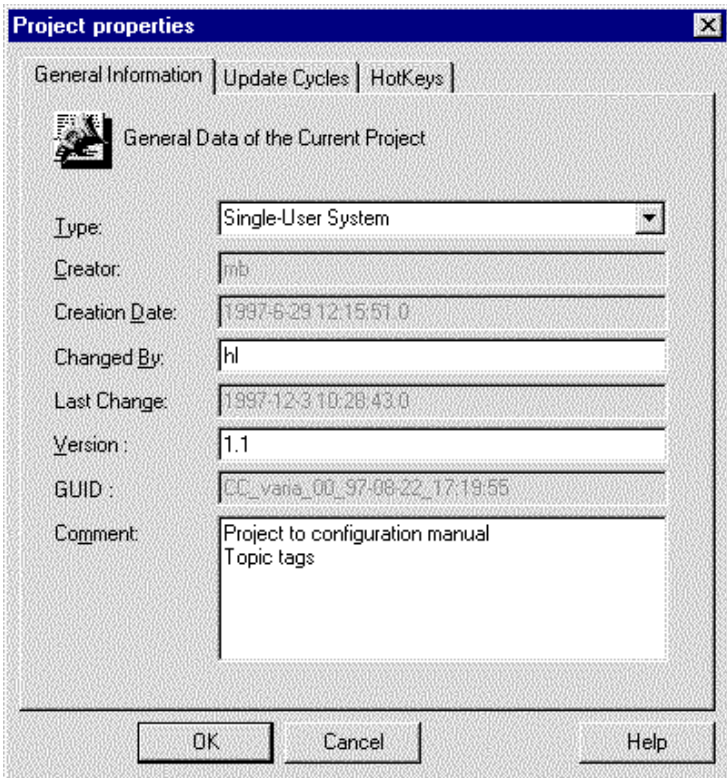
OCX, ActiveX

Если используются компоненты OCX (OLE Control, ActiveX), то они должны быть установлены в системном реестре Windows. Зарегистрируйте или проверьте наличие регистрации этих OCX компонентов, например, с помощью инструмента SmartTools\CC_OCX_REG\ocxreg.exe с WinCC CD-ROM. Если регистрация OLE или OCX объектов не найдена при запуске проекта WinCC в режиме исполнения (также как во время конфигурации используя графический дизайнер), этот объект отображается с пометкой неизвестный объект.

Сеть

Если проект сконфигурирован для **многопользовательской системы**, сеть должна быть полностью установлена на целевой компьютер перед копированием данных WinCC. Запомните имя компьютера, которое вы дали ему в сетевом окружении, оно понадобится при установке параметров копируемого проекта, оно также важно в случае однопользовательской системы. Следовательно, вы должны знать имя компьютера, на который устанавливается проект, или определить его с помощью панели управления Windows.

Копирование данных и запуск проекта

Шаг	Процедуры при переносе данных
1	Создать папку проекта (например, <i>WinCC_Projects</i>).
2	Скопировать резервные данные в поддиректорию этой папки (например, <i>\WinCC_Projects\Varia_00</i>). Имя папки проекта WinCC может быть в изменено по желанию. При переименовании файлов в папке проекта (<i>varia_00.mcp, varia_00.db, varia_00.pin, varia_00.log</i>), удостоверьтесь, что фалам даются те же имена (исключая расширение).
3	Откройте проект в <i>проводнике WinCC</i> .
4	<p>Если необходимо, измените специфические настройки проекта. Если нужно изменить тип, необходимо сделать соответствующие настройки свойств компьютера. Эти настройки описаны в справочной базе WinCC.</p> 
5	Проверьте имя компьютера, и, в случае необходимости измените его в свойствах компьютера в разделе General Information (Общая информация). Если имя компьютера в проекте WinCC не совпадает с реальным, возникнет сообщение об ошибке при активизации проекта.
6	<ul style="list-style-type: none"> • Если Вы использовали в проекте собственные стандартные функции (*.fct), резервная копия функций должна быть перенесена в папку WinCC по умолчанию <i>\Siemens\WinCC\aplib</i> и они должны появиться в дереве функций: • Открыв проект, запустите редактор <i>Global Script</i>. Обновите структуру объявлений, используя Options (Опции) → Regenerate Header (Перегенерировать заголовок). После этого новые функции станут видимыми в дереве функций.

Шаг	Процедуры при переносе данных
7	Активируйте проект и проконтролируйте его запуск.

3.3.9 Повторное использование – Передача частей проекта в новый или существующий проект

Причины повторного использования данных

Проект WinCC может быть сгенерирован разными способами. Наиболее важные аспекты касаются повторного использования частей сходных проектов или передачи данных из примеров проектов.

Команда проекта

Для всех членов команды в этом отношении определяются аналогичные задачи, поскольку проект WinCC в конце должен быть объединен в единое целое. Проект WinCC состоит из отдельных файлов (например, кадры) и конфигурационной информации в базах данных (регистрация аварийных сообщений, управление тегами).

Информация в базе данных

Данные, хранящиеся в конфигурационной базе данных, не могут быть созданы отдельно в двух проектах и после этого объединены. Поэтому основной проект, использующийся для этого типа конфигурации, должен быть создан при конфигурировании информации базы данных (например, структура регистрации аварийных сообщений). Необходимо создать резервную копию основного проекта перед любым изменением в базе данных (также для промежуточных шагов). Если при изменении что-то пошло неправильно, вы можете восстановить состояние до изменения.

Замечание:

Любое изменение, сделанное в базе данных влияет на структуру и доступ к базе. Большое количество ненужных изменений (возможно с удалением) может привести к дисбалансу и потере производительности базы данных WinCC.

Однопользовательская система

На следующем шаге конфигурации, например системы регистрации аварийных сообщений, нельзя изменять базу данных в другом месте (например, в системе регистрации тегов) в однопользовательской системе WinCC.

Многопользовательская или многоклиентская система

Напротив, если проект конфигурируется в многопользовательской или многоклиентской системе, можно проводить изменения в различных частях базы одновременно. Например, один пользователь может конфигурировать систему регистрации аварийных сообщений в то время как другой редактирует систему архивации (сбора измеренных значений).

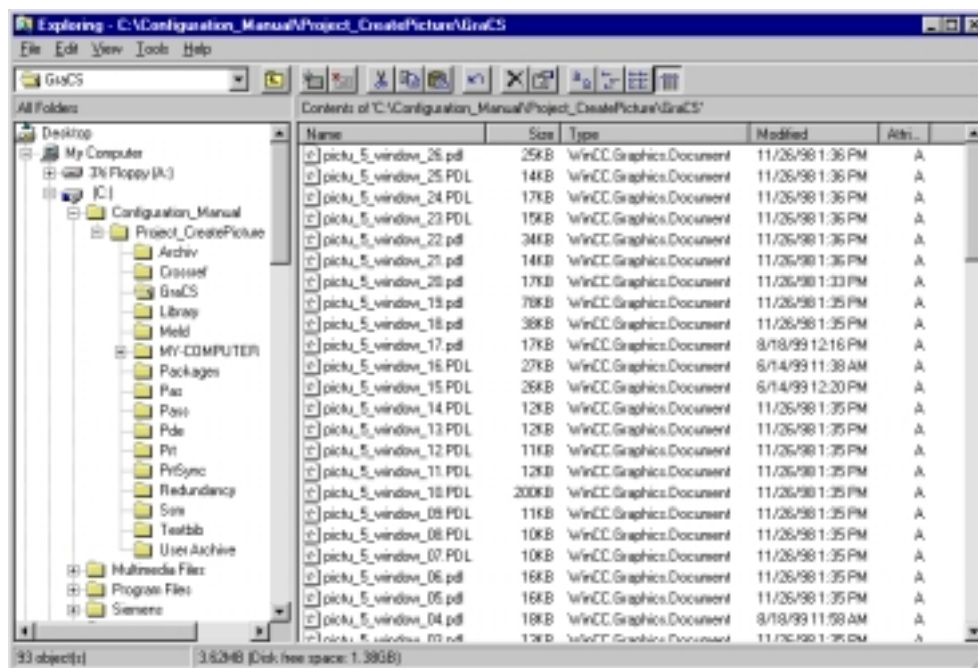
Изменение типа проекта: Преобразование однопользовательских и многопользовательских систем

Любой проект может быть переведен из одной системы конфигурации WinCC в другую. Если изменение конфигурации планируется заранее, то нельзя использовать специфические элементы WinCC, ориентированные на многопользовательскую систему. Например, нельзя использовать внутренние теги на локальной станции, если конфигурируемая система является однопользовательской.

При создании нового проекта (независимо от того на сколько пользователей ориентирована система), Вы должны прежде всего знать будет ли проект основан на существующем проекте с сконфигурированной системой регистрации аварийных сообщений и/или системой архивации. Хранящаяся в базе данных информация может быть перенесена в другой проект с помощью переконфигурации или, если возможно, с помощью экспорта - импорта.

3.3.9.1 Передача кадров

Передачу сконфигурированных кадров можно осуществить в любое время. Можно скопировать файл кадра (*.pdl) прямо в нужный каталог проекта WinCC \GraCS используя Windows Explorer (целесообразно скопировать несколько файлов кадров). Ниже представлена часть проекта по конфигурации кадра:



Можно также передать кадр, открыв файл кадра (*picture.pdl*) в *графическом дизайнера* с помощью пунктов меню *File (Файл) → Open (Открыть)*. После этого, кадр сохраняется в текущей папке кадров (\GraCS) с помощью меню *File (Файл) → Save As (Сохранить как)*. Этой процедурой можно пользоваться, если кадры используются как основа и могут быть сразу настроены.

Ссылки в кадрах

- Структуры из области типа данных
- Внутренние или внешние теги
- Системные теги
- Объекты кадров, которые хранятся как растровые или векторные файлы
- Другие кадры, используемые как графические контейнеры или окна
- Используемые функции проекта

Права доступа

Следующие ссылки также должны быть определены:

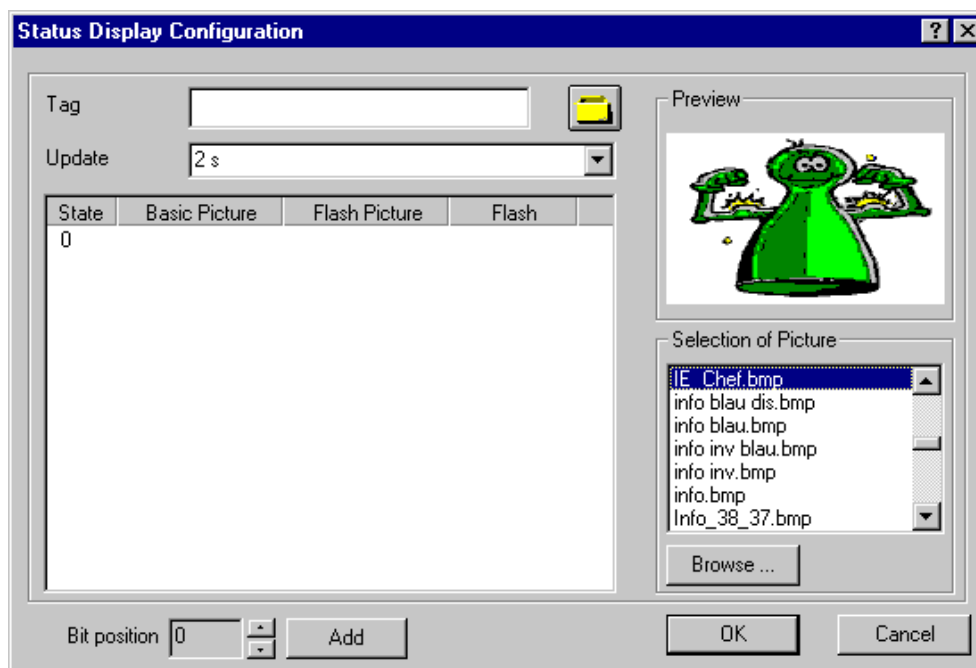
- Определение структур типов данных, например, контроллер или временные структуры для определенных объектов
- Определение каналов связи и логических соединений с определением тегов (возможно с группой тегов)
- Определение внутренних тегов или имен системных тегов (начиная с @)
- Передача элементов кадра (*.gif или *.emf) копированием из папки \GraCS
- Передача содержимого окна кадра копированием дополнительных файлов кадра (*.pdl) из папки \GraCS
- Используемые функции проекта должны быть скопированы из исходного проекта в папку \Library нового проекта. В дополнении, эти функции должны быть установлены в дерево функций редактора глобальных сценариев с помощью пункта меню Regenerate Header (Перегенерировать заголовки). Подробно эта процедура описана в главе „Среда Разработки сценариев Си”.

Определение прав доступа должно осуществляться в администраторе пользователей. Используемые права доступа (например, для кнопок управления) должны быть определены для группы спецификаций.

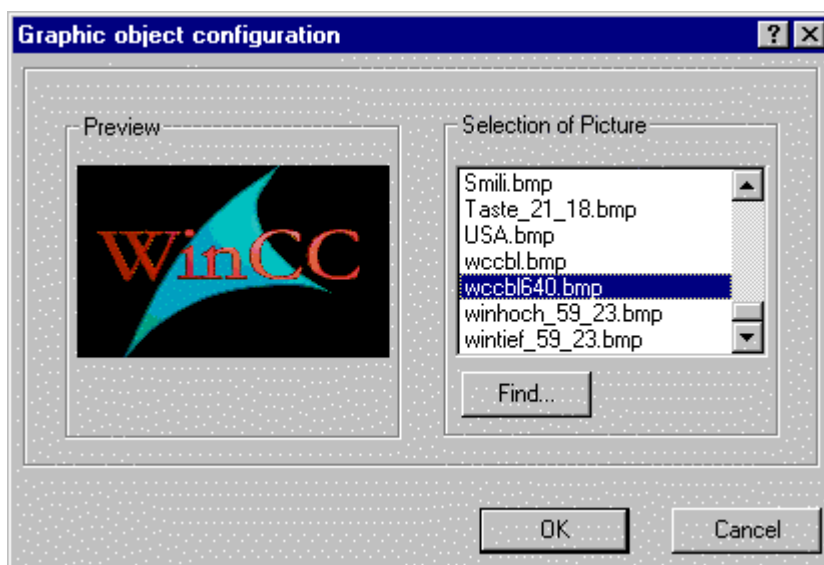
3.3.9.2 Передача символов и побитовых изображений

Копирование

Символы для окон состояний или графических объектов в файлах кадров хранятся в отдельных фалах в папке кадров проекта. Эта процедура осуществляется копированием нужных файлов символов (*.emf или *.gif) в соответствующую папку \GraCS нового проекта. Эти кадры сразу становятся доступными в списках выбора для окна состояния или графических объектов (см. “Палитра объектов в графическом дизайнере”). Ниже приводится часть диалогового окна конфигурации для окна состояния:



Окно выбора кадра для графического объекта:



Импорт

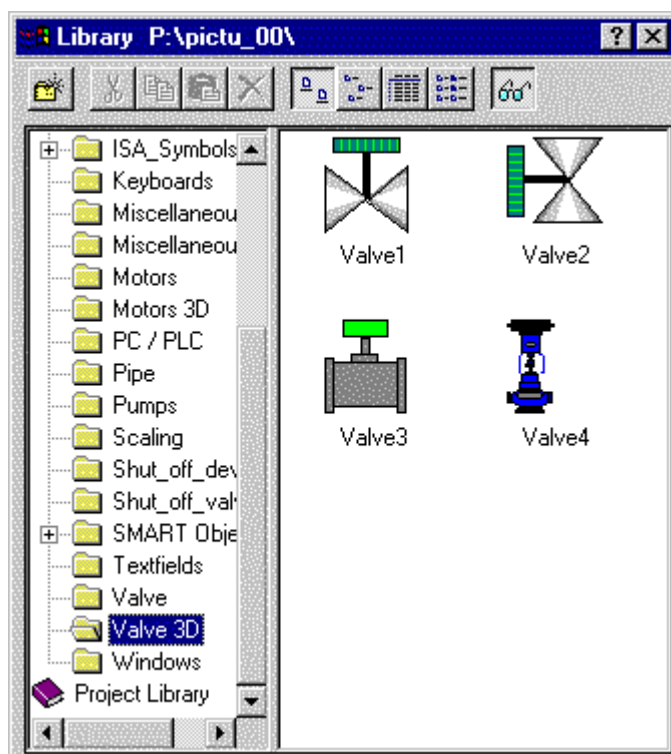
Символы могут быть интегрированы в кадр с помощью только что описанного метода, или могут быть скопированы прямо на кадр с помощью пунктов меню *Insert (Вставка)* → *Import (Импорт)*. Впоследствии, вам не нужно копировать файл, вместо этого вы импортируете нужный символ напрямую назначая путь к исходному проекту (*\GraCS*) и затем выбирая нужный файл (*.bmp, *.emf, *.wmf). После того как символ импортирован, он сразу отображается как объект на кадре (в левом верхнем углу).

Если символы хранятся в библиотеке проекта, ее можно использовать в другом проекте путем полного переноса. Как это делается - описывается в следующей главе.

3.3.9.3 Передача библиотеки проекта (с ранее сконфигурированными символами и модифицированными объектами)

Глобальная библиотека

Если символы хранятся в библиотеке проекта, она может быть использована в другом проекте путем копирования файла *library.pxl* в папку *\Library*.
Заранее сконфигурированные блоки теперь могут использоваться в новом проекте:




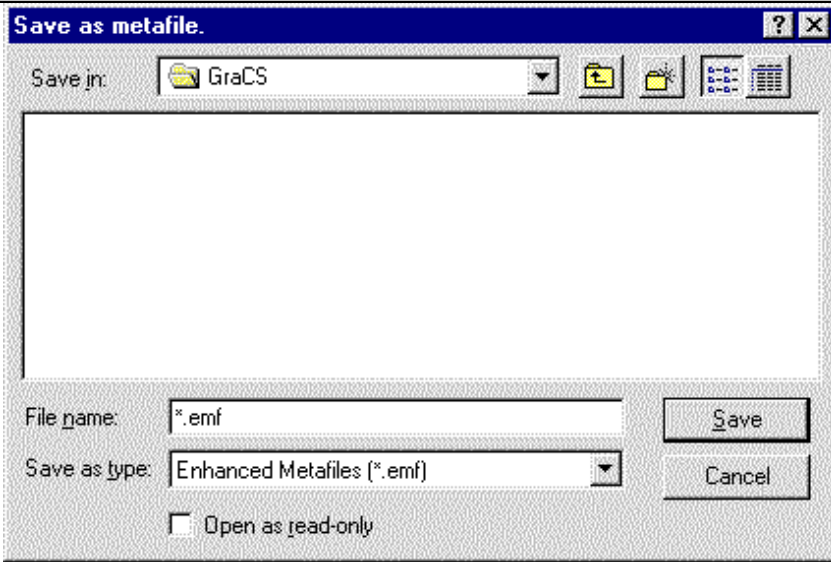
Замечания:

Примите во внимание тот факт, что связанные символы могут указывать на недоступные ссылки (или теги), которые сначала нужно определить. В зависимости от конфигурации этих символов, необходимо настроить процедуры или связи. Поэтому, после использования символов из библиотеки, проверьте какие свойства/события связи присутствуют, а какие нуждаются в настройке.

Отдельные символы

Если несколько определенных символов из библиотеки проекта используются в новом проекте, они экспортируются индивидуально (файл символов *.emf).

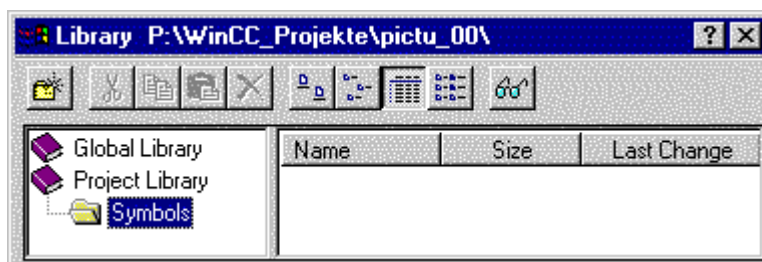
Шаг	Процедура: Передача символов
1	Открыть библиотеку.
2	Выбрать нужный символ используя  и перенести символ на кадр держа кнопку мыши нажатой (Drag and Drop).
3	С помощью пунктов меню <i>File (Файл)</i> → <i>Export... (Экспорт)</i> , откройте диалоговое окно сохранения символов.

Шаг	Процедура: Передача символов
	
4	Сохраните символ.

Библиотека нового проекта

Экспортируемые символы становятся доступными как отдельные файлы символов и могут использоваться отдельно с помощью импорта. Если эти символы часто используются в проекте, они должны быть интегрированы в библиотеку нового проекта. Это можно сделать, вызвав символьную библиотеку, в частности библиотеку проекта.

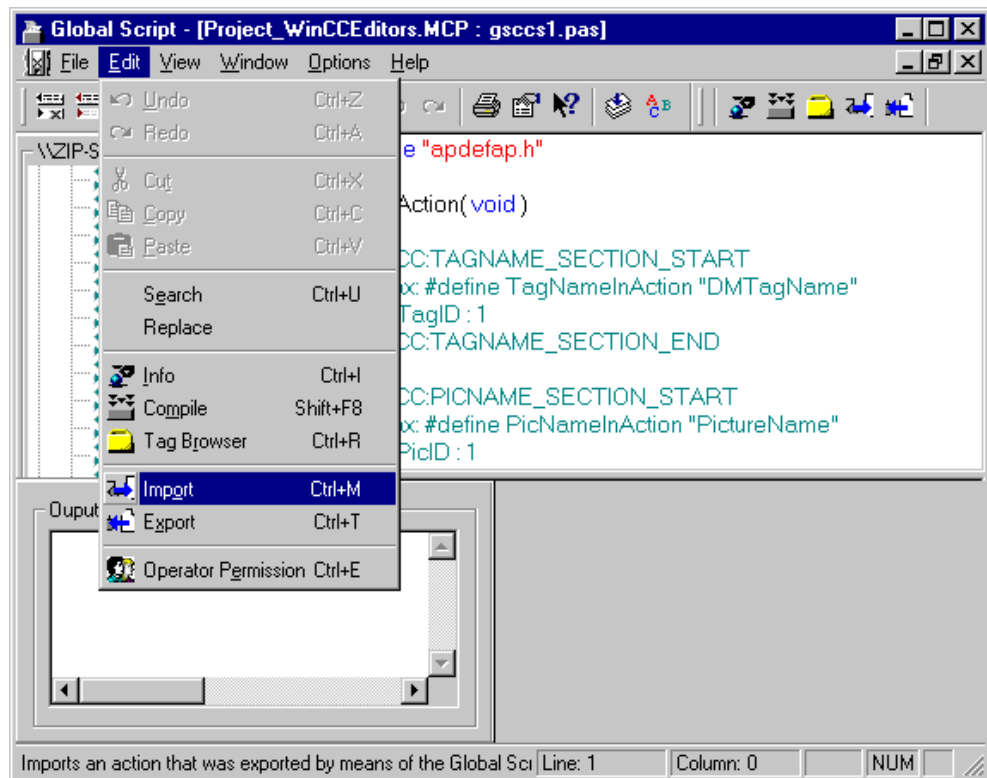
Создайте собственную папку для символов, например, с помощью иконки папки на панели инструментов окна библиотеки, и скопируйте импортируемые символы с помощью Drag and Drop. В этом случае, вы можете передать некоторые символы из проектов и добавить дополнительные символы чтобы создать специальную библиотеку проекта.



3.3.9.4 Передача процедур

Процедуры, которые часто требуются в проекте или которые копируются от одного объекта к другому хранятся как отдельные файлы. Эти файлы хранятся в папке \GraCS с расширением .act. Они могут быть переданы в любое время копированием из одной папки в другую.

Файл процедур сохраняется в файле с именем, заданным пользователем (с расширением.act для процедуры) с помощью кнопки на панели инструментов Export Action (Экспорт процедуры) в редакторе процедур Си.



Сохраненный файл процедуры передается процедуре объекта на кадре нового проекта с помощью кнопки Import Action (Импорт процедуры) на панели инструментов. Смотрите также описание в главе “Среда разработки сценариев Си”.

Замечание:

Часто используемые процедуры могут быть определены как функции проекта или стандартные функции.

3.3.9.5 Передача тегов


- Управление тегами в WinCC может осуществляться множеством различных способов:
- Чтение тегов данных S5 или S7 используя мастера (мастер динамики)
- Передача тегов S7 с помощью PCS7 Mapper
- С помощью экспорта или импорта текстовых списков через программу *Var_Exim*
- Онлайнный доступ к таблицам баз данных (таблицы тегов)
- Специально запрограммированные мастера динамики или программы, использующие WinCC API для генерации новых данных в управлении тегами

Две последние описанные опции требуют хорошего знания баз данных и программирования через интерфейс приложения. Ими должны пользоваться люди, имеющие соответствующие навыки.

Перед тем как передать данные в новый проект, необходимо выяснить где находится базис этого проекта. Если в управлении тегами WinCC уже присутствует большое количество тегов, список тегов WinCC должен быть импортирован в новый проект. Внутренние теги всегда должны передаваться из управления тегами WinCC. Это делается с помощью инструмента Var_Exim.exe .

Передача тегов данных S5/S7 с помощью мастера динамики

Определения областей данных, сгенерированных программами STEP5/STEP7 могут быть прочитаны управлением тегов WinCC с помощью мастера динамики. Должны быть выполнены следующие шаги:

Шаг	Процедура: Передача данных S5 или S7
1	Создать резервную копию проекта. Изменения делаются в базе данных.
2	Экспортировать список назначения, используя программы STEP. Создается файл с именем <i>prj_zuli.SEQ</i> .
3	Из экспортируемого файла удалить любые специальные символы (например, для вызова программ) которые не требуются для импорта в WinCC. Можно сделать это обычным текстовым редактором, как например Wordpad. Список назначения не должен содержать пустых строк.
4	Откройте проект в <i>проводнике WinCC</i> . Проект должен быть в режиме конфигурации (исполнение не активно).
5	Открыть <i>графический дизайнер</i> . В любом кадре перейдите на <i>мастер динамики</i> (с помощью <i>View (Вид) → Toolbars...(Панели инструментов)</i>) и выберите раздел <i>Import Functions (Импорт функций)</i> . Здесь выберите функцию <i>Import S7 - S5 Assignment List (Импорт списков назначений S5 – S7)</i> . Затем должны быть определены <i>файл источник (.seq)</i> и его путь (используя кнопку).  Также определите логическое соединение, в котором должны быть помещены описания тегов из списков назначений. Теперь данные введены в управление тегами WinCC. Имена тегов, использованные в проекте WinCC должны быть уникальными. Имя тега является его ключем при добавлении его в управление тегами.

Передача тегов с помощью справочной программы

Соединения (канал, DLL, логическое соединение и параметры соединения) должны быть заранее определены до импорта в новый проект.


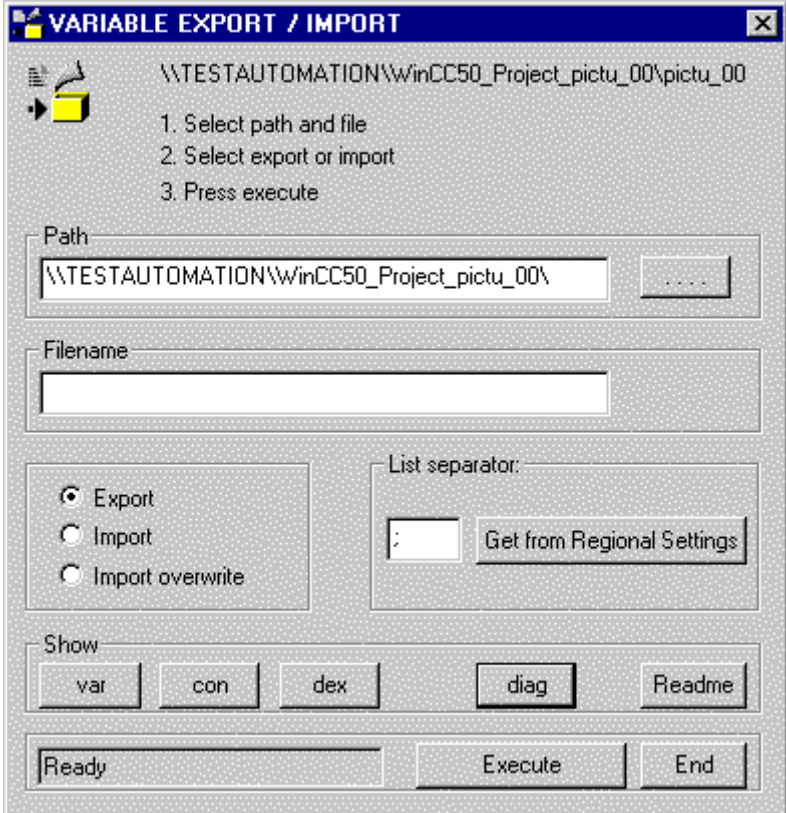
Замечания:

Для автоматической генерации соединений и данных в базе данных WinCC, можно скомпилировать специальную программу, которая будет выполнять генерацию этих определений с помощью интерфейса WinCC API. В этом случае, существующие производственные данные могут быть добавлены автоматически. Эта программа должна быть написана специалистом по WinCC API или по SQL.

Теги, определенные в управлении тегами могут быть **экспортированы** в любой момент как текстовый файл в качестве дополнения списка тегов. После этого, сгенерированные данные должны быть **импортированы** обратно в управление тегами проекта. Эти файлы будут в формате CSV (comma separated values) и могут быть прочитаны и обработаны редактирующей программой.

- Для этого существует отдельное приложение на WinCC CD-ROM в папке \SmartTools\CC_VariablenImportExport. Это Windows приложение является справочной программой для:
- Экспорта данных из управления тегами
- Импорт тегов, созданных внешними программами
- Массовая конфигурация данных

Для импорта или экспорта данных должно быть выполнено следующее:

Шаг	Процедура: Импорт/экспорт тегов
1	Откройте проект WinCC в <i>проводнике WinCC</i> .
2	Определите соединения (канал-DLL – логическое соединение- параметры соединений), которые в данный момент недоступны, но позднее понадобятся для импорта. Делать это нужно только в новом проекте. Может потребоваться вторая процедура импорта-экспорта.
3	<p>Активизируйте программу <i>Var_exim</i> с помощью . Пользовательский интерфейс этой программы изображен ниже.</p> 

Импорт - экспорт

Для импорта или экспорта, необходимо сделать следующее:

Нахождение, процедура	Импорт	Экспорт
<i>Путь</i>	Выберете папку проекта, содержащую файлы для импорта тегов. Необходимо выбрать файлы с расширением <i>.mcp</i> . Файлы, содержащие данные для импорта должны находиться в той же директории, что и файл проекта.	Выберете папку проекта для экспорта. Необходимо выбрать файлы <i>.mcp</i> .
Процедура	Выберете <i>Import (Импорт)</i> . Если необходимо перезаписать данные, выберете <i>Import Overwrite (Импорт с перезаписью)</i> .	Выберете <i>Export (Экспорт)</i> .
<i>Исполнение</i>	Щелкните на исполнении. Открывается диалоговое окно с установленными параметрами, и после нажатия ОК запускается процедура. Из-за выполняющейся проверки импорт занимает больше времени.	Щелкните на исполнении. Открывается диалоговое окно с установленными параметрами, и после нажатия ОК запускается процедура.
Статическое отображение	Завершение экспорта/импорта	Завершение экспорта/импорта
Файл тегов <i>Name_vex.csv</i>	Базис для импорта: Состоит из заголовка и записей.	Сгенерированный список тегов хранится в этом файле как текст. Этот файл может быть открыт нажатием на кнопку <i>var</i> или отредактирован с помощью текстового редактора (Notepad) или EXCEL.
Файл тегов <i>Name_cex.csv</i>	Базис для импорта: Состоит из заголовка и записей (структурные компоненты).	Файл содержит текущие сконфигурированные соединения, на которые имеются ссылки в файле тегов. Этот файл может быть открыт нажатием кнопки <i>con</i> или отредактирован текстовым редактором (Notepad) или EXCEL.
Файл структуры данных <i>Name_dex.csv</i>	Базис для импорта: Состоит из заголовка и записей данных.	Если есть теги с типом структуры данных, то также генерируется этот файл, содержащий структурную

Нахождение, процедура	Импорт	Экспорт
		информацию. Вы можете отредактировать его с помощью текстового редактора (Notepad) или EXCEL.
Файл диагностики <i>Diag.txt</i>	Файл диагностики содержит информацию о тегах, которые невозможно импортировать.	

Из этой программы можно выйти нажатием кнопки *End*.

Списки тегов

Следующая таблица описывает структуру списка тегов.

Поле	Тип	Описание
Tag name	char	Имя тега
Conn	char	Соединение
Group	char	Имя группы
Spec	char	Внутренние теги или адреса (совпадающие с типом соединений)
Flag	DWORD	
Common	DWORD	
Stype	DWORD	Тип тега 1 BIT 2 SBYTE 3 BYTE 4 SWORD 5 WORD 6 SDWORD 7 DWORD 8 FLOAT 9 DOUBLE 10 TEXT_8 11 TEXT_16 12 Raw Data Type 13 Field 14 Structure 15 BITFIELD_8 16 BITFIELD_16 17 BITFIELD_32 18 Text Reference
CLen	DWORD	Длина тега
CPro	DWORD	Внешний или внутренний тег
CFor	DWORD	Формата преобразования
Protocol		

Поле	Тип	Описание
P1	BOOL	Ошибка по верхней границе
P2	BOOL	Ошибка по нижней границе
P3	BOOL	Ошибка преобразования
P4	BOOL	Ошибка записи
P5	BOOL	
P6	BOOL	
L1	BOOL	Замена значения при ошибке по верхней границе
L2	BOOL	Замена значения при ошибке по нижней границе
L3	BOOL	Начальное значение
L4	BOOL	Замена значения при ошибке соединения
L5	BOOL	Правильная верхняя граница
L6	BOOL	Правильная нижняя граница
L7	BOOL	Правильное начальное значение
L8	BOOL	Правильная замена значения
LF1	double	Верхняя граница
LF2	double	Нижняя граница
LF3	double	Начальное значение
LF4	double	Замена значения
Scaling		
SCF	DWORD	1 - если масштабирование определено
SPU	double	Область значений, обработка от
SPO	double	Область значений, обработка до
SVU	double	Область значений, тег от
SVO	double	Область значений, тег до

Список соединений

Поле	Тип	Описание
ConName	char	Имя логического соединений
Unit	char	Канальное устройство
Common	char	Общие
Specific	char	Специальные параметры соединения
Flag	DWORD	

Список структур данных

Поле	Тип	Описание
DataStructure	short	Имя структуры данных или компонента
Type ID	short	Идентификация (используется в списке тегов в Стуре)

Поле	Тип	Описание
Creator ID	short	

Для того чтобы работать с текстовыми списками в EXCEL (Версия 7.0 или 8.0), нужно открыть экспортируемые файлы, которые имеют тип текстовых файлов [* .prn; *.txt; *.csv].

Инструкции

Преобразование информации о тегах из текстового списка осуществляется с помощью следующих инструкций:

Тип	В текстовых списках	В WinCC
Соединения	Основное описание	Если не доступно, логические соединения должны быть переопределены!
	Специфичные описания канала-DLL	Если не доступно, канал-DLL должны быть переопределены!
Группа тегов	Информация о группе отсутствует Если группы определены в проекте, не содержащем тегов, группы также не экспортируются.	Информация о группе автоматически генерируется с первым тегом группы.
Теги	Основное описание	
	Специфичное описание Существующий канал-DLL или внутренний тег	Канал-DLL или внутренние теги
	Во время экспорта, отсутствующие секции заменяются на *.	Теги, чье специфическое описание отсутствует, не импортируются!
Теги типа структуры данных	Присвоение соответствует определению структуры данных списка структур.	Присваивается типу данных.
Определение структуры данных		Должно быть определено конфигуратором системы.
Уставки	Не импортируются и не экспортируются через текстовый список.	Должны быть определены конфигуратором системы.

Перед началом импорта редактируемых или новых тегов нужно выполнить **резервное сохранение** проекта, до изменений в базе данных. Эти изменения в WinCC не обратимы.

3.3.9.6 Передача многоязыковых текстов (из кадров, в сообщения)

Многоязыковые тексты кадров

Многоязыковые тексты, хранящиеся в кадре могут быть переданы в новый проект копированием самого кадра.

Соответствующий язык должен быть добавлен в текстовую библиотеку нового проекта. Проверьте установки текстовой библиотеки. Каждый язык должен иметь собственную колонку!

Text-ID	German	English
1		
2	Benutzerverwaltung	Useradministration
3	Freigabe für Bereich	Authorization for area
4	Systemwechsel	Systemchange
5	Beobachten	Monitoring
6	Prozeßbedienungen	Processcontrolling
7	Höherwertige Prozeßbedienungen	Higher Processcontrolling
8	Reportsystem	Reportsystem
9	Archive bedienen	Archive controlling
10	Bildwechsel	Picturechange
11	Runtime beenden	Runtimeend
12	Projekt bedienen	Projectuse

Только несколько языком нужно передавать в новый проект?

Так как текстовая информация кадра хранится, переконфигурация соответствующего языка должна быть осуществлена непосредственно в кадре. Вам необходимо определить, нужно ли удалять ранее сконфигурированный текст. Переключение в режим исполнения должно осуществляться только специальными клавишами. Если вы тем не менее хотите удалить текст на уже подключенном языке, мы рекомендуем импорт и экспорт на языке справочной службы.

Передача кадров с текстовыми ссылками

- Если в передаваемом кадре используются тестовые ссылки, следующие данные также необходимо передавать:
- Ассоциированные теги (экспорт или переопределение) из управления тегами проекта WinCC
- Тексты из текстовой библиотеки
- Теги текстовых ссылок должны быть снабжены правильными идентификационными номерами (текстовые ID). Проверьте соответствие текстовых ID ассоциированному тексту.

Передача текстов из текстовой библиотеки

Если из текстовой библиотеки передаются не все тексты, то текстовые ID нужно настроить соответствующим образом. Тексты из текстовой библиотеки могут быть переданы с помощью механизма экспорта/импорта *текстовой библиотеки*.

3.3.9.7 Передача сообщений

- Информационный базис для сообщений (аварийных сообщений) требует
- модификации и
- большой конфигурационной работы из-за объема информации.

Поэтому, передача сообщений из предыдущего проекта используется очень часто. В зависимости от источника сообщений могут использоваться следующие методы передачи сообщений:

- Передача сконфигурированных сообщений из предыдущей системы (например, COROS)
- Импорт (единичных) сообщений из существующего проекта WinCC

Передача сообщений из COROS

Процедура приведенная выше выглядит следующим образом:

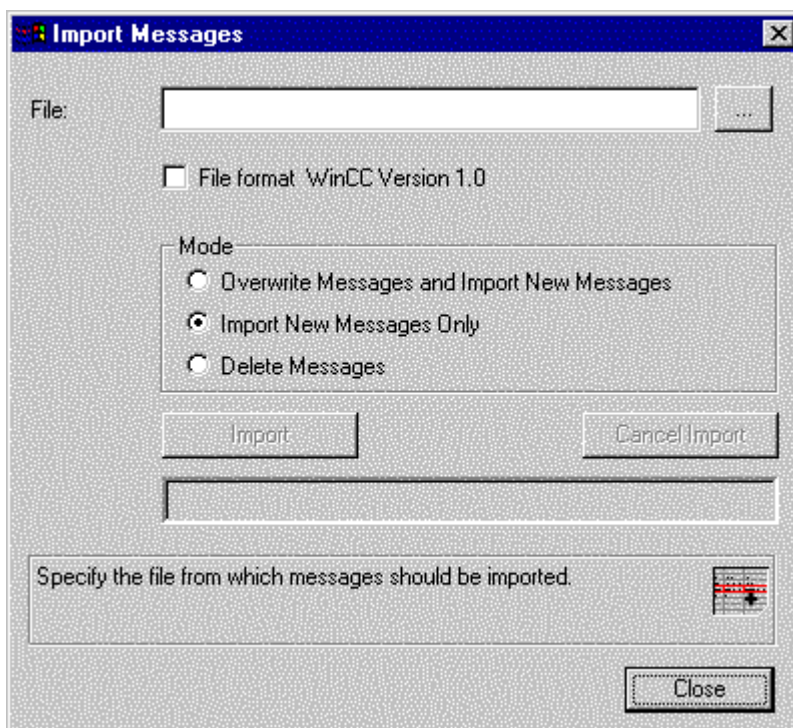
Шаг	Процедура: Передача существующего текста сообщения COROS
1	В COROS экспортировать информацию о сообщении (<i>mldtecte.txt</i>)
2	В WinCC импортировать файл сообщений с помощью мастера динамики Import Functions (Функции импорта) → Import Messages (Импорт сообщений). Данные импортируются в текущий проект WinCC.

Передача сообщений из проекта WinCC

Если сообщения передаются из существующего проекта, то сначала необходимо проверить была ли в базе установлена структура соответствующей системы сообщений (система регистрация аварийных сообщений). Различия можно увидеть, например, в блоках значений пользователя или процесса. Если возможно, немедленно организуйте блоки данных в соответствии с последовательностью (а также в соответствии с длиной текстовых элементов). Иначе, перед импортом, Вам придется делать настройки в отдельных колонках .

Шаг	Процедура: Передача текста сообщений WinCC
1	В текущем проекте, откройте <i>Alarm Logging (редактор аварийных сообщений)</i> .
2	Иницируйте экспорт сообщений выбрав пункт меню <i>Messages (Сообщения)</i> → <i>Export Single Messages (Экспорт одиночных сообщений)</i> .
3	Определите конечный текстовый файл, в который будет экспортироваться информация. Выберите экспортируемые сообщения используя критерий, например, номер, класс сообщений.
4	Запустите процедуру экспорта, нажав <i>Export</i> . Создается текстовый файл, содержащий разделенную запятыми информацию.

Шаг	Процедура: Передача текста сообщений WinCC
5	Закройте текущий проект и откройте новый. Откройте снова <i>Alarm Logging</i> и определите необходимые типы и классы сообщений. Определите по одному сообщению каждого типа, чтобы сохранить основную структуру для последующего импорта.
6	Для экспорта основных сообщений, выберите Messages (Сообщения) → Export Single Messages (Экспорт отдельных сообщений). Повторите шаги 3 и 4.
7	Теперь откройте, например в EXCEL, файл сообщений проекта источника и файл сообщений конечного проекта. Колонки разделены запятыми. Сравните структуру блоков сообщений и, если необходимо, выполните необходимые настройки. В каждом блоке с текстовым ID, введите индекс 0. Это означает, что текст будет автоматически организован в текстовые таблицы при импорте. Ни в коем случае не следует оставлять старые ID! Модифицированный файл должен быть сохранен еще раз как текстовый.
8	Иницируйте процедуру импорта с помощью <i>Messages (Сообщения) → Import Single Messages (Импорт единичных сообщений)</i> .
9	Теперь определите файл источник с экспортируемой информацией. Вы должны решить следует ли перезаписывать существующую информацию. Сообщения присваиваются в соответствии с существующими номерами, номера должны быть уникальными.
10	Сообщения импортируются и дополняют существующую систему сообщений (регистрация аварийных сообщений). Проверьте импортированную информацию.



Замечания:

Если сообщения передаются из проекта WinCC V 1.10, вы должны обратить внимание на заголовки столбцов в текстовом файле сообщений!

Импорт сообщений из фазы понятия с помощью таблиц EXCEL

Информация о сообщениях уже доступна в таблицах EXCEL. Сообщения могут быть переданы объединением столбцов в структуру сообщений проекта WinCC. Это нужно сделать путем создания файла сообщений WinCC. Этот файл создается следующим образом:

Шаг	Процедура: Создание структуры сообщений
1	В <i>проводнике WinCC</i> откройте новый проект. Откройте редактор <i>Alarm Logging</i> и определите необходимые блоки сообщений, классы сообщений и их типы. Определите по одному сообщению каждого типа, чтобы сохранить структуру для последующего импорта.
2	Для экспорта основных сообщений, выберите <i>Messages (Сообщения)</i> → <i>Export Single Messages (Импорт единичных сообщений)</i> .
3	Определите конечный файл, в который должна быть сохранена экспортируемая информация.
4	Запустите процедуру экспорта, нажав на соответствующую кнопку. Создается текстовый файл, содержащий информацию о сообщениях, разделенную запятыми.
5	В программе EXCEL, откройте файл сообщений и только что созданный файл нового проекта. Столбцы разделены запятыми. В таблице создайте копию строки соответствующего класса/типа сообщений. Передайте тексты сообщений из источника и введите их в соответствующие блоки. Например: <i>Block 1 (Блок 1)</i> → <i>Message Text (Текст сообщения)</i> . Пронумеруйте строки сообщений (например, начиная с 1). Это можно быстро сделать в EXCEL с помощью нумерации в колонке номеров сообщений.
6	В каждом блоке с текстовым ID введите индекс 0. Это означает, что текст будет автоматически организован в текстовые таблицы при импорте. Ни в коем случае не следует оставлять старые ID! Модифицированный файл должен быть сохранен еще раз как текстовый.
7	В редакторе Alarm Logging запустите процедуру импорта, выбрав <i>Messages (Сообщения)</i> → <i>Import Single Messages (Импорт единичных сообщений)</i> .
8	Теперь определите файл источник с экспортируемой информацией сообщения. Теперь определите параметры перезаписи существующих сообщений. Сообщениям присваиваются существующие идентификационные номера, которые должны быть уникальными в проекте.
9	Затем сообщения импортируются и дополняют существующую систему сообщений (регистрация аварийных сообщений). Проверьте импортированную информацию.

3.3.9.8 Передача измеренных значений

Так как спецификации точек измерений, определения архивов значений процесса и пользовательские архивы вместе со своими свойствами интегрированы напрямую в структуру базы данных, то невозможно передавать измеренные значения (без прямого доступа к базе данных, который требует превосходного знания базы данных). Это означает, что эти архивы и точки измерений должны быть либо переконфигурированы, либо данные должны передаваться автоматически в начале конфигурации путем копирования всего основного проекта.

3.3.9.9 Передача компоновок печати

Скопируйте необходимые компоновки печати, *.rpl для компоновок страниц или *.rpl для компоновок строк из исходной папки в папку \PRT нового проекта.

3.3.9.10 Передача глобальных процедур

Скопируйте необходимые глобальные или фоновые процедуры *.pas из исходной папки в папку \Pas нового проекта.

3.3.9.11 Передача функций проекта

Скопируйте необходимые функции проекта *.fct из исходной папки в папку \Library нового проекта. К этим функциям можно обратиться через пункт меню Options (Опции) -> Regenerate Header (Перегенерировать заголовок) в редакторе глобальных сценариев (Global Script). Детальное описание данной тематики содержится в главе “Среда разработки сценариев Си”.

3.3.9.12 Применение стандартных функций

В отличие от функций проекта, стандартные функции не должны копироваться. Эти функции сразу же становятся доступными в проекте, так как они видны всем WinCC проектам, находящимся на данной рабочей станции.

3.3.9.13 Передача системы User Administrator (Администратор пользователей)

Так как спецификации для групп пользователей, пользователей и прав доступа вместе со своими свойствами интегрированы напрямую в структуру базы данных, то передача системы User Administrator невозможна. Это означает, что требуется переконфигурация.

Из сложившейся ситуации есть только один выход: инициировать автоматическую передачу данных в начале конфигурации посредством копирования всего базового проекта.

3.3.10 Работа без мыши

Технологические кадры в WinCC в основном управляются с помощью мыши. Нажатие клавиши мыши является одним из наиболее часто используемых событий и используется различными способами (нажатие или отжатие левой или правой клавиши мыши) для различных типов динамизации. Тем не менее, имеются системы, управляемые клавиатурой и мышью или только с клавиатуры.

3.3.10.1 Управление с помощи клавиатуры

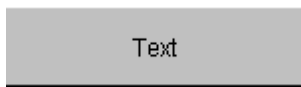
- Клавиатура предлагает следующие опции для ввода:
- Функциональных клавиши от F1 до F12
- Специальные функциональные клавиши (например функциональные клавиши панели оператора SF10)
- Стандартный ввод с клавиатуры
- Переход к полям ввода/вывода или к клавишам управления используя курсорную клавишу или специальные клавиши.

Конфигурация управляющих процедур без мыши должна быть рассмотрена отдельно для следующих областей конфигурации:

- Управляющие кнопки на технологическом кадре (например, для его изменения)
- Использование функциональных клавиш
- Использование специальных клавиш
- Использование стандартных клавиш
- Любые клавиши управления
- Перемещение посредством управляющих объектов
- Поля ввода на технологических кадрах
- Поля ввода/вывода
- Специальные объекты ввода (переключатель, ...)
- Система регистрации аварийных сообщений (окна сообщений)
- Управляющие процедуры с помощью функциональных клавиш
- Управляющие процедуры с помощью специально сконфигурированных клавиш
- Система регистрации тегов (окна таблиц или трендов)
- Управляющие процедуры с помощью функциональных клавиш
- Управляющие процедуры с помощью специально сконфигурированных клавиш
- Начало печати с помощью клавиши
- Регистрация и выход из системы с помощью клавиатуры

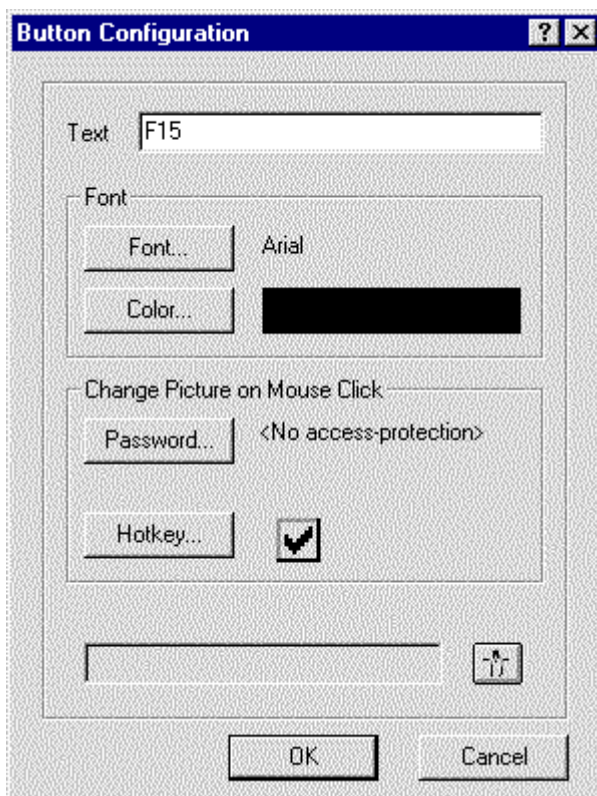
Кнопки управления сконфигурированы в обычном *стиле Windows*. По этой причине в палитре объектов (Object Palette) существует стандартная кнопка управления Windows. Также имеется возможность добавления других графических элементов на кнопку.

Управляющие кнопки



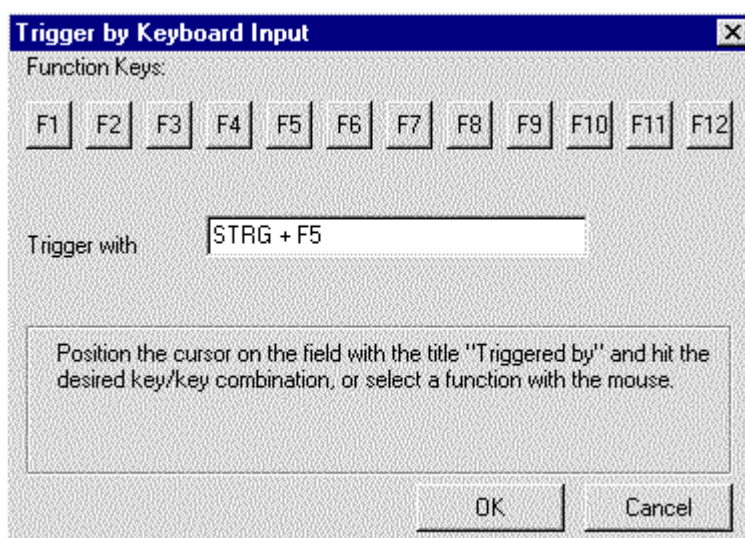
Управляющие процедуры с помощью функциональных клавиш

Функциональные клавиши от F1 до F12 часто используются в качестве (дополнительных) способов воздействия на управляющие кнопки при изменении кадров в их промышленной иерархии. Эти функциональные клавиши в любой момент можно установить в качестве “горячих” клавиш для сконфигурированных кнопок Windows. “Горячая” клавиша предоставляет самый быстрый доступ к соответствующей функции.

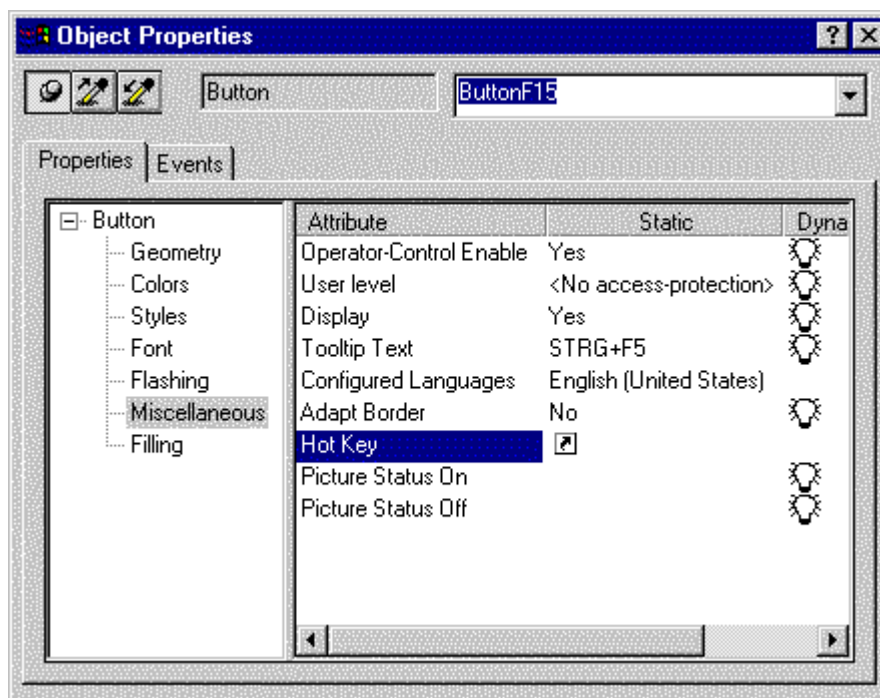


Упомянутым функциональным клавишам могут быть назначены “горячие” клавиши. Эти клавиши уже отображены в диалоговых окнах конфигурации как кнопки выбора. Если необходимо соединить клавишу, например, с клавишами SHIFT или CTRL, то необходимо просто ввести требуемую последовательность (например, SHIFT+F2) в поле ввода при помощи нажатия соответствующих клавиш. Специальные коды вводить не требуется.

Выбранная комбинация клавиш отображается в поле ввода.

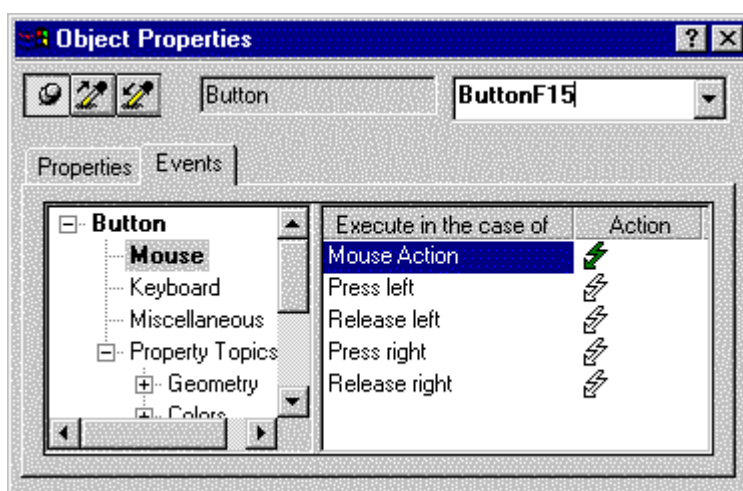


Выбранная последовательность клавиш хранится в свойствах объекта и может быть изменена как через диалоговое окно конфигурации, так и через *Miscellaneous* (Разное) → *Hotkey* (Горячая клавиша).



Процедура для “горячих” клавиш

Процедура, которая должна быть вызвана при нажатии функциональной клавиши (определение “горячей” клавиши) хранится в том же событии кнопки Windows, в котором оно хранится будучи процедурой, привязанной к нажатию на кнопку мыши. Процедура выполняется в момент отпускания кнопки мыши, но только если нажатие и отпускание кнопки мыши происходит над одним и тем же объектом. Если по событию *Mouse Action (Нажатие на кнопку мыши)* не сконфигурирована на запуск ни одна процедура, но для похожего по смыслу события *Press Left (Нажатие на левую кнопку)* такая процедура сконфигурирована, то процедура **не** будет выполняться по нажатию на функциональную клавишу! При конфигурировании стоит также учесть тот факт, что функциональная клавиша может быть использована на кадре только один раз.



Специальные функциональные клавиши

Если для управления кадрами Вы используете специальные функциональные клавиши, например, F13, S1 на операторской панели, то их можно переопределить комбинациями клавиш, например, комбинацией SHIFT+F1. Вдобавок к вышеописанному использованию выбранных комбинаций клавиш, для конкретных устройств также могут быть указаны специфические комбинации клавиш. Для таких целей Вы найдете индивидуальные настройки клавиатуры, зависящие от конкретных устройств. Например, файл F125.key настройки кодов клавиш для промышленных ПК. В таких устройстве – зависимых файлах хранятся коды функциональных клавиш. После необходимой настройки (функциональных клавиш с соответствующими шестнадцатеричными кодами) новые коды клавиш могут быть использованы для работы с кадрами.

Стандартные клавиши

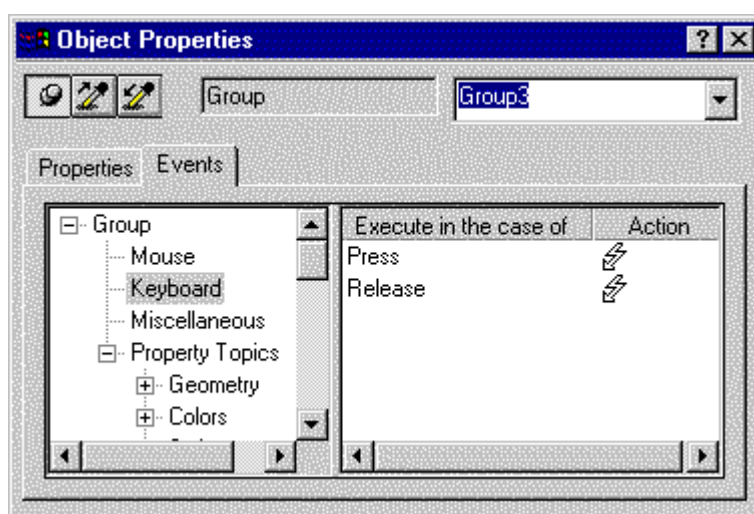
Если процедура назначена не на функциональную клавишу, а на клавишу стандартной клавиатуры, например, букву m, то эта клавиша хранится в качестве “горячей” для объекте Windows Button (Кнопка Windows).

Управление любой клавишей

Для технологических кадров можно в свою очередь создать дизайн управляющих кнопок. Вы найдете много дополнительных кнопок управления, например, в пользовательской библиотеке в разделе 3D Buttons (3D кнопки) или User Objects (Пользовательские объекты).

Для самостоятельно разработанных объектов, которые не основаны на кнопках Windows, „горячие“ клавиши не могут быть сконфигурированы. Все остальные объекты должны быть сконфигурированы с помощью свойства объекта *Button Event* (Событие кнопки). Для объектов возможны следующие события клавиатуры:

- Нажатие клавиши
- Отпускание клавиши



Для обеспечения возможности конфигурирования управляющих процедур с клавиатуры данное событие должно быть доступным. При использовании predetermined кнопок из пользовательской библиотеки необходимо в первую очередь проверить, возможно ли работать с этими кнопками только с помощью клавиатуры, т.е. без использования мыши. Например, кнопки перемещения пользовательских объектов не всегда доступны для управляющих процедур с клавиатуры.

Предварительно сконфигурированные кнопки управления (например, прокрутка иерархии кадров) находятся в дополнительных пакетах. (например, Basic Process Control (Основное управление процессом) - Picture Tree Manager (Менеджер работы с деревом кадров), и т.д.).

Если один из этих объектов используется как управляющий элемент, то запускающая процедура кнопка конфигурируется на срабатывания по событию Keyboard - Press (Клавиатура – Нажатие) или Keyboard – Release (Клавиатура – Отпускание). В качестве процедуры может быть сконфигурировано как Direct Connection (Прямое соединение), так и C-Action (Процедура Си).

Запускающим событием для кнопки может быть.

- Нажатие любой клавиши или
- Нажатие выбранной клавиши на стандартной клавиатуре

Если это нажатие любой клавиши, то можно использовать *Direct Connection* (Прямое соединение). При использовании выбранной клавиши должна использоваться *C-Action* (Процедура Си). Перед продолжением текущей последовательности процедур *Процедура Си* проверяет введенные коды клавиш символ за символом.

3.3.10.2 Перемещение через управляющие объекты (поля ввода и управляющие поля)

Мышь может быть использована для нажатия непосредственно на каждый управляемый объект. Признаком контролирования объекта служит изменение указателя мыши. Как организовать такую функциональность без мыши?

Буквенный курсор

Вы можете передвигаться между контролируемых объектов в режиме исполнения посредством “клавиш передвижения” (“movement keys”). Различают:

- объекты буквенного курсора (объекты ввода/вывода) и
- объекты перехода по клавише табуляции

Объекты ввода/вывода выбираются с помощью буквенного курсора (клавиша табуляции или комбинация клавиш SHIFT+табуляции).

Все управляющие элементы (контролируемые мышью, клавиатурой или и тем и другим) могут быть интегрированы в управление как с помощью буквенного курсора, так и с помощью последовательности табуляции.

Последовательность табуляции

Последовательность табуляции (может быть установлена с помощью Edit (Правка) → TAB Sequence (Последовательность табуляции) → Alpha Cursor (Буквенный курсор) или → Tab Order (Последовательность табуляции)) предоставляет возможность влиять на порядок, в котором в режиме исполнения происходит перемещение указателя по контролируемым объектам. Текущий выбранный объект может быть визуализирован во время исполнения. Это курсор времени исполнения, который можно выключить. (Computer Properties (Свойства компьютера) → Graphics Runtime (Режим исполнения графического редактора)). При переходе фокуса на кнопку в стиле Windows выполняется ее подсветка с помощью прямоугольника с штрих - пунктирной границей. Переход между контролируемыми объектами зависит от настроек режима исполнения (Computer Properties (Свойства компьютера) → Graphics Runtime (Режим исполнения графического редактора)).

Переход	Стандартные клавиши	Настройка клавиш
Вверх, Вниз Влево, Вправо	Клавиши управления курсором или клавиши Tab (следующий) или SHIFT+Tab (предыдущий)	Настройка других клавиш с помощью <i>Computer Properties</i> → <i>Graphics Runtime</i> → <i>Cursor Control Keys (Клавиши управляющего курсора)</i>
Буквенный курсор/Порядок табуляции	Порядок табуляции	Переключение между буквенным курсором и табуляцией с использованием “горячих” клавиш. (<i>Computer Properties</i> → <i>Graphics Runtime</i> → <i>Hotkeys (Горячие клавиши)</i>) или клавиш, определенных пользователем (используя внутреннюю функцию <i>SetCursorMode</i>)
Переход в таблицах (Группа курсора)	Нормальный режим редактирования, т.е. строчка за строчкой. Если курсор достиг конца группы курсора, то он	Этот режим можно изменить с помощью <i>Computer Properties</i> → <i>Graphics Runtime</i> → <i>Cursor Control Characteristics (Характеристики управляющего</i>

Переход	Стандартные клавиши	Настройка клавиш
	останется в этой позиции.	<i>курсора)</i>

Поля ввода/вывода

В сконфигурированные поля ввода/вывода ввод с клавиатуры можно производить сразу же после перехода на конкретное поле. Полями, предназначенными только для вывода нельзя управлять (*Properties (Свойства) → Output/Input (Ввод/вывод) → Field Type (Тип поля) → Output (Вывод)*).

Запись введенных данных зависит от соответствующих настроек (*Properties → Output/Input → Apply on Exit (Применить при выходе)*) и происходит при нажатии клавиши ENTER. Выход из поля без записи изменений, если таковые были, по клавише ESCAPE (ESC).

Другие объекты ввода

В дополнение к типовым вариантам полей ввода имеются и другие, предлагаемые Windows-приложениями. Эти отдельные поля ввода находятся в Object Palette (Палитре объектов) меню Windows Objects (Объекты окна)

- Независимый переключатель (Check-Box)
- Радио кнопка (Radio - Button)

Выбор полей независимого переключателя и радио кнопки делается с помощью пробела (spacebar), а навигация по ним осуществляется с помощью стрелок Вверх/Вниз. Эти установки являются установками по умолчанию.

Другим объектом ввода является текстовый список. Вы можете сделать выбор с помощью выпадающего списка, и он будет зависеть от введенных элементов: Объектами можно также управлять с помощью стандартной клавиатуры. Нет никакой необходимости хранить специальную конфигурацию для управляющих процедур клавиатуры.

Список открывается нажатием клавиши ENTER, двигаться по списку можно с помощью клавиш Вверх/Вниз, а подтвердить текущий выбор клавишей ENTER. Дополнительные объекты ввода могут быть использованы в WinCC посредством ОСХ-элементов. Управление и конфигурация этих объектов зависит, тем не менее, от событий и свойств, определенных для объекта. Это необходимо выяснять в каждом отдельном случае..

3.3.10.3 Функциональные клавиши для кнопок панели управления системы Alarm Logging (Регистрация аварийных сообщений)

На панели управления в окнах сообщений установлены различные управляемые с помощью мыши кнопки.

Наиболее часто встречающимися управляющими процедурами в окнах сообщений являются:

- выбор сообщения для подтверждения
- передвижение вверх/вниз по списку сообщений
- скроллинг по списку сообщений



При открытии окна сообщения или дальнейших управляющих действиях элемент управления должен находиться в окне сообщений, а не в главном окне. В зависимости от текущей управляемости, кнопки (или функциональные клавиши) воздействуют на панель кнопок главного окна или на кнопки, находящиеся на окне сообщений. Этого можно достигнуть, например, установкой фокуса в данную область окна. Фокус обычно устанавливается нажатием кнопки мыши. Посредством клавиатуры, фокус может быть установлен в окно сообщений с помощью следующих конфигурируемых механизмов:

- изменение окна с помощью “горячих” клавиш
- установка фокуса с помощью кнопок управления или установка фокуса определенному элементу в окне сообщений при открытии кадра.

Смена (переключение) окна сообщений может быть осуществлена с помощью “горячих” клавиш, которые также могут быть использованы для смены любых окон, например, окон трендов. Это определено в параметрах запуска системы Graphics Runtime. Комбинация клавиш (например, CTRL+W) вводится в Computer Properties (Свойства компьютера) → Graphics Runtime (Режим исполнения графической системы) → Hotkeys (Горячие клавиши) → Window On Top (Окно поверх остальных).

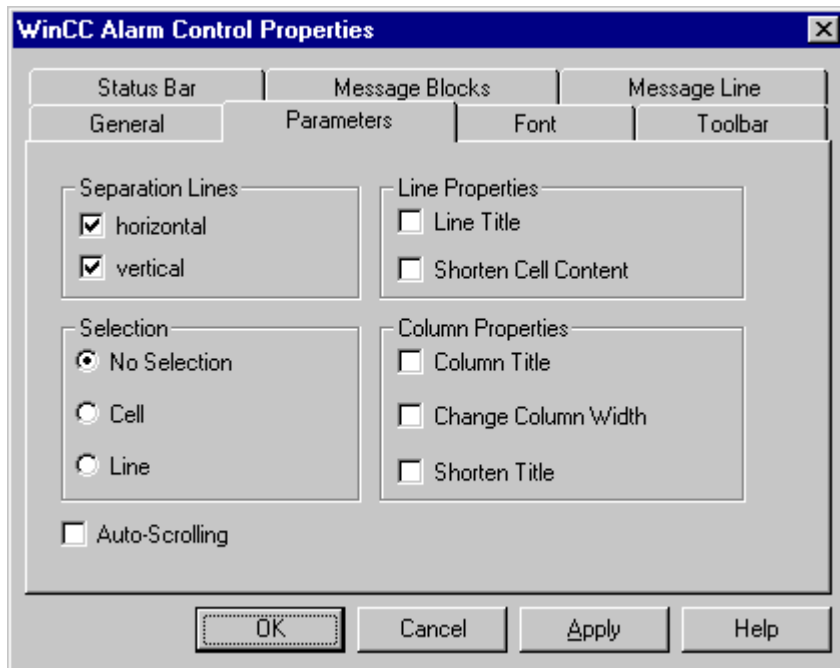
Как только окно сообщений открыто, кнопки панели инструментов могут быть активизированы с помощью данной последовательного нажатия клавиш.

Установка фокуса в окно сообщений производится с помощью внутренней функции Set_Focus. Процедура Си для кнопки управления или команда открытия окна (*Picture Object (Объект кадра)* → *Events Miscellaneous (Различные события)* → *Open Picture (Открытие кадра)*) могут повлиять на активизацию окна сообщений.

Функции, отвечающие за фокус кадра, доступны из меню *Internal Functions (graphics set focus)*. Например, для установки фокуса управления вызывается следующая функция:

```
Set_Focus(lpszPictureName, lpszObjectName);
```

В качестве параметров должны быть переданы имя главного окна (имя кадра) и Alarm Control (имя объекта). Сообщение в окне сообщений выбирается путем выбора соответствующей строки. При открытии окна сообщений курсор устанавливается на младшее сообщение (последнее сообщение в кадре). Выбор сообщения или скроллинг в окне сообщений зависит от активности стрелок скроллинга. Стрелки скроллинга можно включить/выключить или с помощью кнопки на панели управления или напрямую установить при конфигурации WinCC Alarm Control.

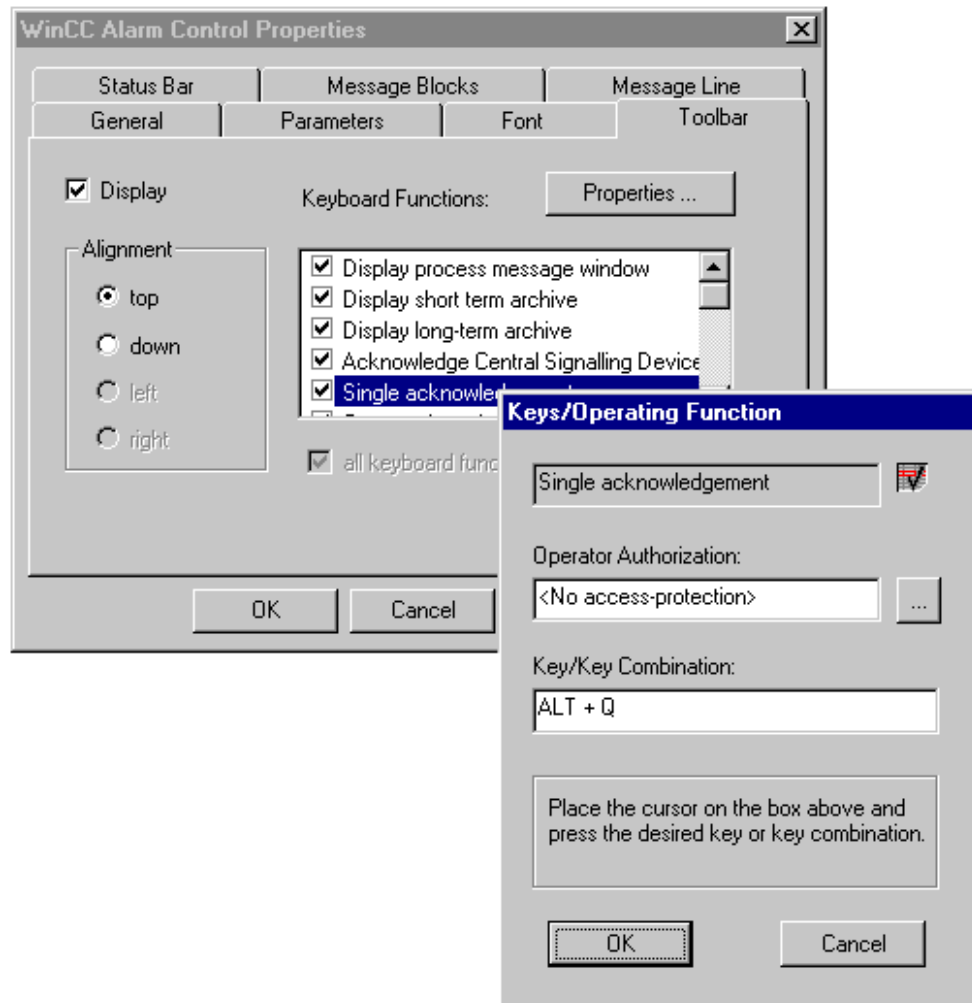


Если стрелки скроллинга активны и фокус находится в окне сообщений, то передвижение может осуществляться следующим образом:

Передвижение	Стандартны клавиши	Настройки клавиш
Вверх, Вниз в списке сообщений	Клавиши со стрелками	Отдельные строки сообщений
Начало, Конец в списке сообщений	Клавиша Pos1, Клавиша End	В начало или в конец списка сообщений
Скроллинг	Клавиши скроллинга (Page Up/Page Down)	Несколько строк сообщений

Вдобавок к возможности активизации кнопок на панели управления (таких как подтверждение выбранного сообщения) с помощью стандартных процедур мыши Вы можете определить процедуры с помощью функциональных клавиш.

В свойствах каждой отображаемой на панели управления кнопки может быть сконфигурировано соответствующее управление с клавиатуры. Например:



Эти шаги конфигурации дают Вам возможность определить комбинацию клавиш для каждой кнопки, использованной в окне сообщений.

3.3.10.4 Alarm Logging (система регистрации аварийных сообщений) - кнопки панели управления, разработанные специально для промышленности

Все кнопки на панели управления в WinCC определены и не могут быть изменены. Если необходимо переконфигурировать разработанное для промышленности расположение кнопок, то Вам необходимо деактивизировать панель управления WinCC (т.е. убрать ее) и спроектировать необходимые кнопки самостоятельно. Все эти новые кнопки могут быть спроектированы для удовлетворения желаний покупателей, например при предоставлении ими иконок.

Определенные для кнопок функциональные возможности должны, тем не менее, быть сконфигурированы как связанные процедуры. В процедурах Си для такого связанного события (например, нажатия клавиши) из дерева функций должна быть выбрана соответствующая стандартная функция.

Функции, предусмотренные для управления кнопками находятся в *Standard Functions (Стандартные функции)* → *Alarm (Аварийное сообщение)*. Для каждой кнопки на панели управления в списке содержится функция, соответствующую данной кнопке. Например, кнопка Single Acknowledgment вызывается с помощью следующей функции:

```
AXC_OnBtnSingleAckn(lpszPictureName, lpszObjectName);
```

В качестве параметра вводится имя окна Alarm Logging Control.

Несколько примеров таких кнопок можно найти в дополнительных пакетах для аварийной системы (например, Основное управление процессом – Подтверждение сигналов, и т.д.).

3.3.10.5 Функциональные клавиши для кнопок панели управления системы Tag Logging (системы регистрации тегов)

В элементах управления трендами или таблицами Tag Logging (окна трендов и таблиц) используемых для отображения измеренных величин, на панели управления могут быть сконфигурированы различные кнопки управления, которыми можно управлять при помощи мыши.

- Наиболее частыми процедурами в окнах трендов являются:
- скроллинг измеренных величин (временная ось)
- выбор временного интервала
- выбор трендов
- использование линейки чтения

После открытия окна тренда показывается текущая кривая тренда, в зависимости от конфигурации.

При открытии окна тренда, управление должно находиться не в главном окне, а в окне тренда. В зависимости от текущего управления, кнопки управления (или функциональные клавиши) воздействуют на поле функциональных клавиш главного окна или на хранящиеся кнопки управления окон трендов или таблиц. Этого можно достигнуть, например, установкой текущего фокуса в этот раздел окна. Фокус обычно устанавливается при нажатии на кнопки мыши.

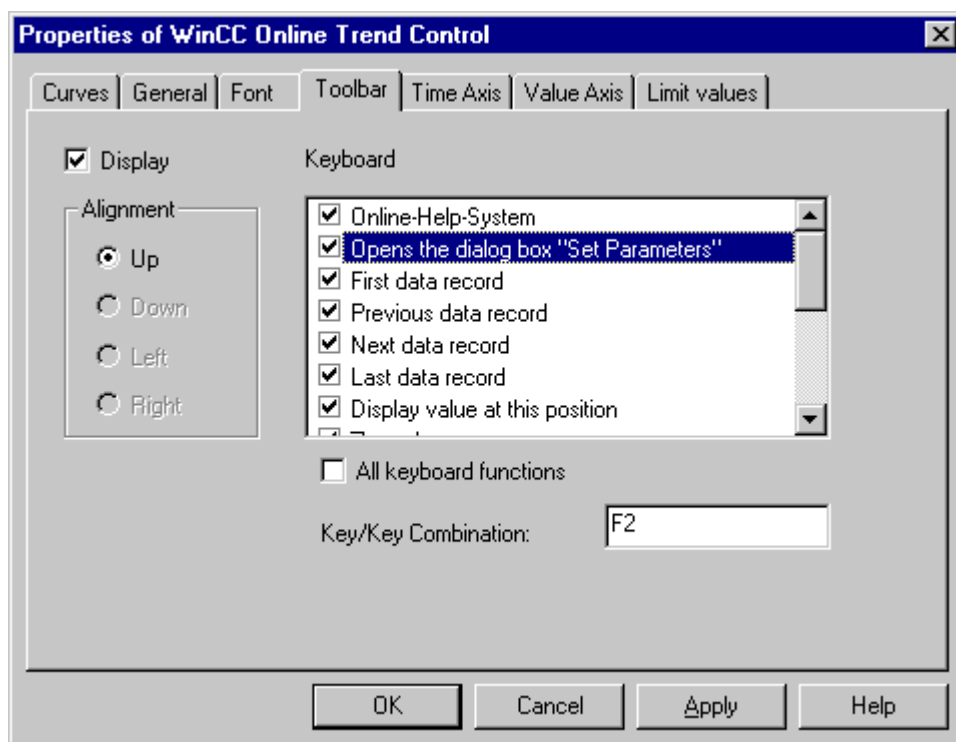
При использовании клавиатуры, фокус может быть установлен в окно тренда или таблицы при помощи следующих процедур:

- изменения окон при помощи “горячих” клавиш
- установки фокуса при помощи кнопок управления или
- напрямую определенному элементу окна тренда при открытии кадра

Реализация этих различных вариантов может быть найдена в данной главе, в описании Alarm Logging (регистрации аварийных сообщений).

Вдобавок к возможности при помощи стандартных процедур мыши активизировать кнопки на панели управления, также Вы можете определить управляющие процедуры при помощи функциональных клавиш. По умолчанию, отдельные кнопки заняты функциональными клавишами от F1 до F10.

Для каждой кнопки, изображенной на панели управления, в свойствах может быть сконфигурировано соответствующее управление с клавиатуры. Например:



Эти шаги конфигурации дают Вам возможность определить комбинацию клавиш для каждой кнопки, используемой для окон трендов или таблиц. Кнопки управления для окон трендов или таблиц должны быть определены дополнительно.

Следующие стандартные клавиши могут быть использованы в окне тренда с того момента, как соответствующая функциональная клавиша активизирована:

Переход	Стандартные клавиши	Настройки клавиш
Линейка чтения	Клавиши со стрелками	Для сдвига линейки чтения влево или вправо
Изменение масштаба	Для выбора секции, предназначенной для масштабирования	Для настройки и активизации дополнительных клавиш для управления мышью (см. системные настройки) Клавиши со стрелками и клавиша INS дают возможность определить изменяемое окно масштабирования
Диалоговые окна, например, Archive Tag Selection(Выбор архивных тегов)	Клавиша табуляции	Для перехода между полями ввода
	Клавиши со стрелками	Для перехода в пределах выбора тегов или табуляции
	Клавиша + (клавиша-)	Для раскрытия или закрытия ветки дерева архивных тегов
	Клавиша пробела	Выбор или отмена выбора
	Клавиша ENTER	Для подтверждения и выхода из диалогового окна
	Клавиша ESC	Для отмены диалогового окна

Tag Logging – кнопки панели управления, спроектированные специально для промышленности

Все кнопки панели управления определены в WinCC и их дизайн не может быть изменен. Если необходимо переконфигурировать разработанное для промышленности расположение кнопок, то Вам необходимо деактивизировать панель управления WinCC (т.е. убрать ее) и спроектировать необходимые кнопки самостоятельно. Все эти новые кнопки могут быть спроектированы для удовлетворения желаний покупателей, например, при предоставлении ими иконок. Определенные для кнопок функциональные возможности должны, тем не менее, быть сконфигурированы как связанные процедуры. В процедурах Си для такого связанного события (например, нажатия клавиши) из дерева функций должна быть выбраны соответствующая стандартная функция.

Функции, предусмотренные для управление кнопками находятся в *Standard Functions* → *Taglog* → *Toolbarbuttons*. Для каждой кнопки на панели управления в списке содержится функция, соответствующую данной кнопке. Например, линейка чтения вызывается с помощью следующей функции:

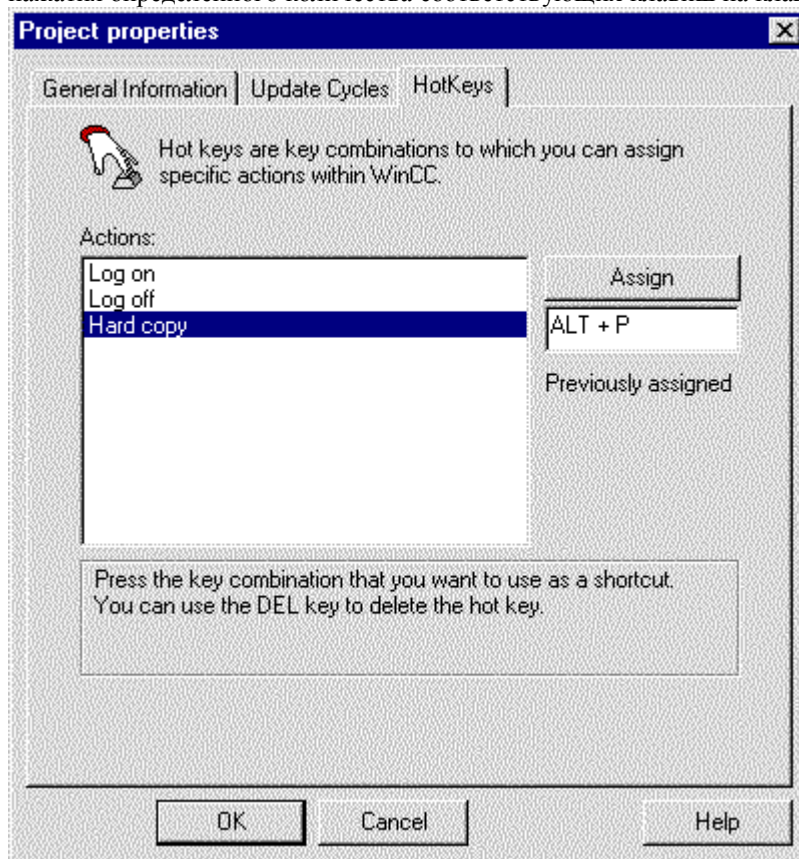
```
TlgTrendWindowPressLinealButton( lpszWindowName );
```

В качестве параметра вводится имя окна Tag Logging Control.

3.3.10.6 Запуск задания на печать

Задания на печать можно запустить несколькими различными способами. Например, в *WinCC Explorer (Проводнике WinCC)*, задание на печать активизируется напрямую при помощи выбора соответствующего пункта в списке выбора. Тем не менее, Вы имеете возможность в технологическом окне создать кнопку Print (Печать) самостоятельно, которую можно использовать для начала печати. Эта кнопка также присутствует на панели управления для списка сообщений и, как было описано на предыдущих страницах, может быть активизирована при помощи функциональной клавиши или определенной Вами клавиши.

Функция печати содержимого экрана – “жесткая копия” – может быть запущена в любой момент с помощью “горячей” клавиши. Эта “горячая” клавиша устанавливается глобально в свойствах проекта. Для ее настройки необходимо выбрать из WinCC Explorer (Проводника WinCC) - Project Properties (Свойства проекта) → Hotkey (Горячая клавиша) → Hardcopy (Жесткая копия) и, для определения “горячей” клавиши, напрямую ввести комбинацию клавиш (путем нажатия определенного количества соответствующих клавиш на клавиатуре).



Если в технологическом кадре Вы конфигурируете свою собственную кнопку для печати определенного задания на печать, то это процедура должно быть инициировано с помощью *C-Action (Процедуры Си)*. Как было описано в начале главы, кнопка активизируется с помощью функциональной клавиши или с помощью клавиши клавиатуры (например, клавиша D). *Процедура Си* должна быть сконфигурировано соответственно событию в вопросе (например, *Mouse Action (Процедура мыши)* или *Keyboard – Press (Нажатие клавиатуры)*). WinCC обеспечивает эту функциональность с помощью функций, находящихся в меню

Standard Functions (Стандартные функции) → Report (Отчет) → ReportJob (Задание для отчета).

```
ReportJob(pszJobName, pszMethodName);
```

Первым параметром, передаваемым в эту функцию является имя задания на печать. Как второй параметр передается "PRINT" (Печать) если печать необходимо выполнить сразу или "PREVIEW" (Просмотр) если необходимо выполнить предварительный просмотр.

3.3.10.7 Вход в систему и завершение сеанса работы

В дополнение к переконфигурируемым “горячим” клавишам для запуска и завершения процедуры входа и выхода из системы, можно также сконфигурировать клавишу, которая будет показывать диалоговое окно входа в систему. Сеанс работы можно также завершить с помощью комбинации клавиш. Для этих целей Вам необходимо сконфигурировать отдельную кнопку, которая, может быть активизирована как мышью, так и с помощью клавиатуры. В начале главы описаны различные варианты управления кнопками. Функция, которая используется для входа в систему и завершения сеанса работы является функцией приложения WinCC. Эта функция должна быть сконфигурирована как процедура Си. Сохраните эту процедуру Си, например, в событиях Mouse Action (Процедура мыши) или Press Button (Нажатие кнопки).

Для выхода из системы используются следующие функции:

```
#pragma code("USEADMIN.DLL")
#include "PWRT_API.H"
#pragma code()

PWRTLogout();
```

Вход в систему выполняется функцией *PWRTLogin()*.

Ниже приведен пример использования этой функции:

```
#pragma code("USEADMIN.DLL")
#include "PWRT_API.H"
#pragma code()

PWRTLogin('1');
```

Всплывающим диалоговым окном можно управлять с помощью стандартных клавиш:

Переход	Стандартные клавиши	Настройки клавиш
Отдельные поля ввода	Клавиша табуляции (вперед) или SHIFT+клавиша табуляции (назад) Клавиши со стрелками	Для перевода красной линии влево или вправо
Подтверждение (ОК)	Клавиша ENTER	Для выхода из диалогового окна и подтверждения ввода
Отмена (Abort)	Клавиша ESC	Отмена диалогового окна или ввода

3.3.11 Технология модулей кадров

Технология модулей кадров – ключевая стратегия для получения возможности быстрой и простой конфигурации, повторного использования и восстанавливаемости компонентов кадров.

Сконфигурированное окно процессов используется, например, для нескольких однотипных компонентов процесса (например, клапанов или контроллеров).

Изначально сконфигурированное окно кадров может быть использовано заново для модулей управления, которые должны работать и показываться в проекте. Это сделано согласно следующим принципам:

- копирование окна кадров и переподключение полей тегов
- использование окон кадров, чьи поля тегов определены для вызова (косвенная ссылка)
- применение модифицированных объектов с прототипами и конечными объектами
- создание прототипов кадров и их интеграция
- создание ОСХ модулей кадров и внедрение их как объектов WinCC ОСХ

Сравнение различных технологий

Эти технологии достаточно сильно различаются в способах их применения, сложности конфигурации и в их возможностях. По этой причине, мы начнем со сравнения альтернатив.

Тип	Преимущество	Недостаток
Копирование окон кадров	Простая процедура	Все ссылки объектов должны быть изменены Изменение построения кадра является причиной сложной пост – обработки
Окна кадров с косвенным соединением	<ul style="list-style-type: none"> Только единовременная конфигурация окна кадра с помощью простых процедур Си Повторное использование без копирования основного окна кадра 	Изменение построения кадра является причиной сложной пост – обработки
Модификация объектов	Только единовременная конфигурация объекта с помощью соединения, использующего динамический мастер	<ul style="list-style-type: none"> Изменение построения кадра является причиной сложной пост – обработки, например, обновление кадра Нельзя изменить централизованно
Прототипы кадров	<ul style="list-style-type: none"> Только единовременная конфигурация объекта Нельзя изменить централизованно 	Обязательно (Хорошее) знание языка Си
ОСХ	<ul style="list-style-type: none"> Простое внедрение в конфигурацию WinCC, как объекта в кадр Дальнейшая модификация объекта ОСХ не требует пост – обработки при создании объектов, кроме случаев, когда изменены свойства объекта Высокая производительность Другие графические возможности Покупка новых объектов (например, модулей PCS7) 	Должно быть создано путем написания программы (C++, VB 5); не может быть создано посредством конфигурации WinCC.

Если в проекте используется только несколько модулей кадров, то одного из первых вариантов вполне достаточно для того, чтобы удовлетворить потребности человека, конфигурирующего систему. Эти варианты могут быть применены даже без какой-либо специальной подготовки. Пользовательский объект особенно подходит для простых объектов, меньшей и средней сложности и для связи с тегом. Если Вы предполагаете, что объект будет подвергнут некоторым изменениям, то имеет смысл использовать концепцию прототипов кадров. Если графических блоков много или требуется более сложное слежение за процессом, то необходимо использовать ОСХ компоненты. В этой области доступность ОСХ объектов в будущем будет расти все больше и больше. В следующей главе будут объяснены различные типы конфигураций модулей кадров и показано как их вместе использовать в технологическом кадре. Это позволит Вам сформировать свою собственную картину в различных вариантах и их применении в Вашем проекте.

3.3.11.1 Блок процесса как модуль кадра

Для изображения текущего статуса объекта (контроллер, клапаны, двигатели и т.д.) или для определения уставленных значений, на технологических кадрах имеются специальные информационные окна. Эти окна процесса обычно содержат как текущие состояния (актуальные значения), так и установленные значения, которые могут быть введены оператором, обладающим соответствующими правами.

Создание информационных окон

Это информационное окно создается как окно кадра, чьи компоненты соединены с соответствующими тегами (процесса).

Шаг	Тип	Конфигурация
1	Структура данных	С помощью средств системы Tag Management определите структуру данных, которая будет использована в модуле кадра, например двигатель с текущим значением, установленное значение, переключатель.
2	Модуль кадра	Используя <i>Graphics Designer</i> , сконфигурировать кадр, который изображает состояния устройств, например, панели, поля ввода/вывода и кнопки управления. Размер окна кадра (свойство объекта кадра – переменные X и Y) должен соответствовать размерам окна кадра.
3	Определение тегов	В системе Tag Management определите теги (процесса), например, Motor_T01 со структурным типом Motor, которые используются для окна процесса.
4	Соединение тегов	Теперь сделайте динамическими отдельные компоненты кадра, например, поля ввода/вывода, панели и т.д. с помощью их соединения с соответствующими тегами (процесса).
5	Окно кадра	На технологическом кадре создайте объект окна кадра и соедините его с содержимым окна кадра, созданным на шагах 2 – 4 с помощью свойства имени окна кадра,.
6	Свойства – Настройки	Этот кадр не должен изображаться при первоначальном открытии окна. Следовательно, свойство изображения должно быть статически установлено в нет (no). Параметры появления окна кадра также должны быть определены в свойствах окна кадра.

Шаг	Тип	Конфигурация
7	Вызов окна кадра	Окно кадра должно появиться, например, при нажатии на кнопке мыши или при работе с клавиатурой. Спроектируйте кнопку, которая будет связана с появлением окна кадра (например, с помощью прямого соединения).

Это окно кадра, содержимое окна кадра и соответствующий вызов окна кадра (кнопка) могут быть использованы повторно. Все, что Вам надо сделать - это скопировать окно кадра, модуль кадра и кнопку. Каждый раз ссылки должны быть прописаны заново. Как окно кадра, так и кнопка могут быть скопированы простой операцией “перетащить и оставить” в графическую библиотеку (например, библиотеку проекта).

Модификация модулей кадров

Следующие отдельные шаги должны быть выполнены при использовании созданного модуля кадра:

Шаг	Тип	Конфигурация
1	Теги процессов	Определите новый тег процесса, например, Motor_T02, для определенной структуры данных.
2	Копирование модуля кадра	Скопируйте содержимое окна кадра (Motor02.PDL) и измените все постоянно хранящиеся ссылки (например, вместо Motor_T01.ActValue, будет теперь Motor_T02.ActValue).
3	Копирование окна кадра	Скопируйте объект окна кадра в нужный технологический кадр (с помощью функции “перетащить и оставить” из графической библиотеки). Назначьте ссылку содержимому окна кадра в <i>Properties (Свойства)</i> → <i>Picture Name (Название кадра)</i> (Motor02.PDL).
4	Копирование кнопки	Скопируйте кнопки в нужный технологический кадр (с помощью функции “перетащить и оставить” из графической библиотеки). Назначьте ссылку новому объекту окна кадра в Direct Connection (Прямом соединении) (Object → Picture Window2 → Display).

Таким образом, отдельные окна кадров и их содержимое могут быть созданы для каждого устройства и использованы заново при помощи копирования. Как видно, эта работа неизбежно влечет за собой установление постоянно хранящихся ссылок для содержимого окна кадра. Для этих целей, существует более легкий метод повторного использования - с помощью косвенной адресации. Количество работы по адаптации должно быть сведено к минимуму.

Альтернативным является следующее решение: сконфигурировать модуль кадра без связи с окном кадра. Это означает, что сам модуль кадра конфигурируется как не

изображаемый на технологическом кадре объект. Тем не менее, большим неудобством этого является то, что при изменении модуля кадра необходимо изменить также все кадры, в которых использован этот модуль.

3.3.11.2 Модуль кадра с косвенной адресацией

До настоящего момента, отдельные компоненты модуля кадра были постоянно соединены с соответствующими тегами (процесса). Если такое соединение не будет носить постоянный характер, а будет устанавливаться динамически во время исполнения, то созданный модуль кадра можно будет использовать с большими возможностями. Такое динамическое соединение тегов (процесса) достигается путем косвенной адресации отдельных компонентов в модуле кадра. Это означает, что прямые соединения с тегами (процесса) отсутствуют. Соединение с тегами делается только для контейнера, в котором, во время исполнения, хранятся текущие названия соответствующих тегов (процесса).

При таком подходе будет значительно легче осуществлять адаптацию и повторно использовать модуль кадра.

Конфигурация выполняется аналогично тому, как это было описано выше. Ниже перечислены основные шаги:

Шаг	Тип	Конфигурация
1	Спецификация данных	С одной стороны определение данных, используемых в модуле кадра (например, Motor001_ActValue, Motor001_SetValue, Motor001_Switch), что делается в системе Tag Management, с другой стороны, указание имени контейнера для отдельных компонентов, использующихся в модуле кадра (например, ActV_Name, SetV_Name и т.д.). Вы инициализируйте данные теги, например, именем Motor001_SetValue.
2	Модуль кадра	Используя <i>Graphics Designer</i> , сконфигурируйте кадр, изображающий состояния устройств, т.е. панели, поля ввода/вывода и кнопки управления. Размер окна кадра (свойство объекта кадра – переменные X и Y) должен соответствовать размерам окна Вашего кадра..
3	Соединение с тегом	Теперь сделайте динамическими отдельные компоненты кадра, например, поля ввода/вывода, панели и т.д. Это делается с помощью их соединения с соответствующими тегами контейнера, которые, в свою очередь, содержат имена соответствующих тегов. Тем не менее, при соединении Вы должны указать, что тег - это только имя текущего тега (процесса). Это делается путем установки переключателя <i>Indirect Addr (Косвенная адресация)</i> .
4	Окно кадра	На технологическом кадре создайте окно кадра и соедините его с содержимым окна кадра, созданным на этапах 2 и 3, по названию окна кадра.
5	Свойства – Настройки	Это окно кадра не должно отображаться при начальном открытии окна. Следовательно, свойство отображения должно быть установлено статически в no (нет). Параметры появления окна кадра должны быть также определены в свойствах окна кадра.
6	Вызов окна кадра	Данное окно кадра должно появиться при нажатии на кнопку мыши или при работе с клавиатурой. Спроектируйте кнопку, которая будет связана с появлением окна кадра (например, с помощью прямого соединения).

Шаг	Тип	Конфигурация
7	Графическая библиотека	Объект окна кадра и кнопка копируются в библиотеку (операцией “перетащить и оставить”), так что они могут быть повторно использованы.

3.3.11.3 Модифицирование объектов

Для создания повторно используемых модулей кадров можно воспользоваться модифицированными объектами и соответствующими динамическими мастерами. Модифицированный объект представляет собой графический объект, спроектированный человеком, конфигурирующим систему (например, комбинация нескольких объектов), в которой огромное количество свойств и событий уменьшено с помощью диалога конфигурации и сведено необходимому минимуму. Этот модифицированный объект становится динамическим после его определения как прототипа, что делается с помощью соответствующего мастера. Необходимо выполнить следующие шаги

Шаг	Тип	Конфигурация
1	Структура данных	В системе Tag Management определите структуру данных с, которая будет использована в модуле кадра.
2	Модуль кадра	С помощью <i>Graphics Designer</i> , сконфигурируйте модифицированный объект с определяемыми пользователем свойствами.

Модифицированный объект формируется из группы объектов WinCC. Первоначально, эти объекты не обладают динамическими свойствами. Для формирования модифицированного объекта выбираются все объекты, которые должны быть скомбинированы и для этой группы вызывается диалоговое окно конфигурации:

В этом диалоговом окне все свойства объектов уже объявлены как свойства модифицированного объекта, который в дальнейшем станет динамическим. При этом основные свойства объекта (такие как, например, позиция и размер) уже сохранены для модифицированного объекта.

Каждое отдельное свойство для группы объектов может быть выбрано в окне диалога и добавлено к новому объекту посредством операции “перетащить и оставить” в качестве *user-defined property or event (определенного пользователем свойства или события)*.

Каждому из этих свойств можно назначить имя атрибута (независимое от языка), а также, зависимое от языка имя свойства (например, для конфигурации на английском). Свойства, которые не должны быть видимыми в окне свойств, но которые, например, используются в сценариях, могут быть спрятаны при помощи символа @. Это означает, что отображается только несколько свойств и событий (являющихся динамическими). Все остальные свойства не отображаются.

Спроектированный Вами модифицированный объект теперь необходимо сделать динамическим. Для этого существует мастер:

Шаг	Тип	Конфигурация
3	Динамизация	Вызов динамического мастера <i>Add dynamics to the prototype (Добавление динамических свойств в прототип)</i> . Как для шаблона (прототипа) свяжите каждое отдельное свойство объекта с соответствующим элементом

Шаг	Тип	Конфигурация
		<p>определенной ранее структуры данных.</p> <p><i>Structure member (Элемент структуры)</i> для соединения выбирается с использованием браузера тегов.</p> <p>Тем не менее, мастер только сохраняет имя структурной компоненты в свойстве (например, Значение). Привязка каждого отдельного свойства должна производиться в отдельности.</p> <p>Этот объект теперь является динамическим, однако, он соединен лишь как прототип и не активен во время исполнения. Это означает, что во время исполнения он не может быть обновлен.</p>
4	Копирование в графическую библиотеку	Скопируйте этот прототип в графическую библиотеку.

Прототип модифицированного объекта копируется в графическую библиотеку для его повторного использования. Примером динамических объектов являются инструменты указателя в библиотеке WinCC. (пользовательская библиотека, модифицированные объекты, инструменты указателя).

Копия прототипа объекта вставляется в необходимый технологический кадр. Эта копия теперь должна быть связана с реальными тегами (процесса) из подсистемы Tag Management.

Шаг	Тип	Конфигурация
5	Тег	Определите тег (процесса) для структуры данных, определенной на шаге 1; этот тег в дальнейшем будет использоваться для модифицированного объекта.
6	Создание экземпляра	<p>Скопируйте прототип из графической библиотеки в технологический кадр с помощью операции “перетащить и оставить”</p> <p>Соедините этот объект с тегом (процесса) с помощью динамического мастера.</p> <p>Свяжите прототип со структурой:</p> <p>Мастер автоматически соединит все необходимые структурные компоненты тегов с соответствующими свойствами прототипа модуля кадра путем замены каждой связи тега прототипа на связь с реальным с тегом.</p> <p>Теперь у Вас есть объект, который обновляется текущими значениями тегов во время исполнения</p>

3.3.11.4 Динамический пример

В дополнение к Dynamic Wizard Link (Динамическому мастеру связи) прототипа и структуры, существует еще мастер, который называется Make a prototype dynamic (Создание динамического прототипа). Чем он отличается от связывания прототипа и структуры, и какие шаги необходимо предпринять?

В отличие от постоянного связывания тегов объекта, модули кадров также могут быть связаны динамически. Это означает, что во время исполнения сначала

устанавливается зависимость экземпляра от текущего содержимого тега. Например, описанный выше модуль кадра не соединен с тегом постоянно, имя тега остается динамическим. Текущее имя тега должно быть определено с помощью текстового тега. Этот текстовый тег, содержащий имя текущего тега, должен быть соединен с модулем кадра.

В отличие от постоянного варианта, т.е. соединения со структурой, в этом случае при конфигурации должны быть модифицированы следующие шаги:

Шаг	Тип	Конфигурация
1	Модуль кадра	<p>В графическом дизайнера конфигурируется модифицированный объект с определяемыми пользователем свойствами, как описано выше. Модифицированный объект должен содержать компонент Static Text (Статический текст), чье свойство “Текст” передается как определенный пользователем компонент.</p> <p>Данному свойству назначается имя атрибута <i>TagName (Имя тега)</i>. Это имя тега используется для динамического связывания тегов (процесса).</p>
2	Тег	<p>Для структуры данных, определенной на шаге 1, укажите тег (процесса), это тег должен использоваться для модифицированного объекта.</p>

Шаг	Тип	Конфигурации
3	Создание примера	<p>С помощью операции „перетащить и оставить“ скопируйте прототип объекта из графической библиотеки в технологический кадр.</p> <p>Соедините этот объект с тегом (процесса) с помощью динамического мастера</p> <p>Свяжите прототип со структурой:</p> <p>Мастер автоматически соединит все необходимые структурные компоненты тегов с соответствующими свойствами прототипа модуля кадра путем замены каждой связи тега прототипа на связь с реальным с тегом.</p> <p>Теперь у Вас есть объект, который обновляется текущими значениями тегов во время исполнения</p>

При использовании модифицированных объектов и прототипов, человек, занимающийся конфигурированием системы должен убедиться в том, что необходимые процедуры Си сохранены в объектах. Их не следует удалять, иначе функциональность модулей будет утеряна.

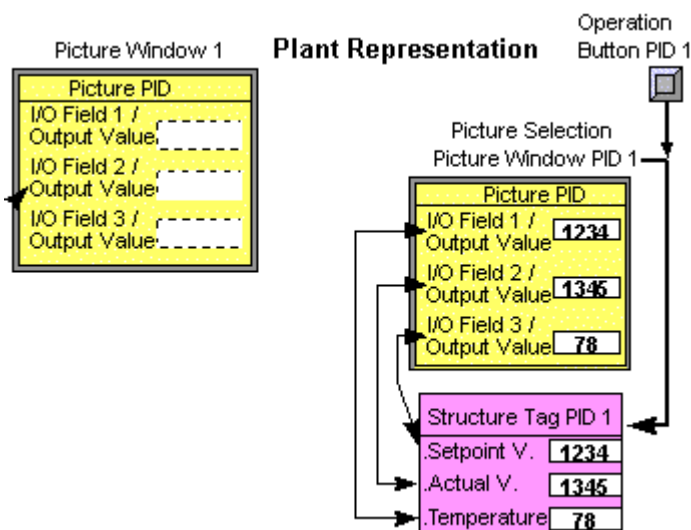
3.3.11.5 Прототип кадра

Технология прототипов кадров является довольно современной. С помощью прототипов можно создать настолько гибкую концепцию, что сделанные в прототипе изменения автоматически передаются созданному объекту. Эта гибкость требует также линковки очень гибких сценариев Си..

Технология прототипов кадров работает с использованием шаблонов кадров, которые могут быть неоднократно интегрированы в один или несколько родительских кадров. Шаблон кадра является всего лишь шаблоном, который “начинает жить” только после интеграции в реальный объект. Объект, основанный на шаблоне (=прототип кадра) появляется путем так называемой “инстанциации”. Для одного шаблона может быть создан набор экземпляров (т.е. реальных объектов).

Шаблон окна кадра

Изображает окно кадра с текущими данными из менеджера данных.



Последующие изменения делаются только в одном месте (в шаблоне) и воздействуют на все приложения (примеры). Это делает эту программу очень эффективной в том смысле, что она лишена недостатка, связанного с переносом изменений во множество различных мест.

В родительском кадре может быть отображено до 30 примеров (т.е. объектов) определенного типа шаблона. Если используются различные прототипы, то можно работать с большим числом объектов.

Прототипы кадров – это модули кадров, которые после создания копируются в библиотеку, т.е. их можно использовать повторно. Ранее использованные модули кадров в дальнейшем используются в технологических кадрах в качестве экземпляров шаблона. В таких копиях отображается текущая информация, например, с контроллеров или двигателей. Соответствующие компоненты контроллеров или двигателей отображаются автоматически.

Также можно создавать сложные модули. Модуль кадра может состоять из нескольких частично или полностью перекрывающихся друг друга компонентов. Например, вся относящаяся к двигателю информация, такая как текущее состояние, эксплуатационные характеристики и т.д. может быть объединена в одном объекте и обновлена по необходимости. В случае если имеется несколько однотипных двигателей, достаточно только один раз создать модуль кадра и потом просто копировать его. Остальное делается автоматически.

Для создания модуля кадра необходимо следовать следующим шагам:

Шаг	Тип	Конфигурация
1	Структуры данных	Определите структуры данных, которые будут использоваться в модулях кадров. Это делается в подсистеме Tag Management.
2	Построение окна кадра	В графическом дизайнера сконфигурируйте содержимое модуля кадра, т.е. панели, поля ввода/вывода и т.д.
3	Привязка структур данных к окну кадра	Свяжите компоненты кадра с отдельными компонентами структур данных с помощью специальных Си сценариев (из примера проекта).
4	Окно кадра	Свяжите модуль кадра с окном кадра.
5	Графическая библиотека	Скопируйте окно кадра в графическую библиотеку.

После этого модуль кадра можно использовать в технологических кадрах путем его вызова из графической библиотеки.


Ниже представлены этапы создания шаблона кадра.

Шаг	Конфигурация
1	Создание структурного тега в менеджере тегов; здесь указывается количество тегов, составляющих структурный тег, их названия и типы (BIT, SHORT, и т.д.). Например, PID со следующими компонентами: уставка, фактическое значение и температура.
2	Создание кадра, который будет использоваться в качестве модуля кадра. Такой кадр обычно немного меньше чем фактический размер экрана и поэтому его можно как угодно изменить. Любой правильно созданный кадр может быть использован для создания шаблона. Далее с помощью графических функций, полей ввода/вывода, панелей и т.д.

Шаг	Конфигурация
	создается кадр, но не делается никаких привязок к тегам. Внутренние связи между графическим объектами, такие как передача вводимого значения по его изменению конфигурируются на этом кадре.

Шаг	Конфигурация
3	<p>Теперь графические объекты связаны с компонентами соответствующих структурных тегов. На этом этапе происходит лишь связь тегов на уровне типов (только шаблоны), но не на уровне конкретных объектов.</p> <p>Пример для этого этапа можно найти на WinCC CD в директории \Samples. В библиотеке данного проекта (\Template) есть модифицированный объект <i>TemplateInit</i>, осуществляющий такую связь. Данный объект располагается в графической библиотеке и его можно просто перетащить на кадр.</p> <p>Объект <i>TemplateInit</i> уже имеет полную логику сценариев. Эта логика использует так называемую <i>ConnectionTable (Таблицу связи)</i>, которая заполняется в процессе конфигурации и содержит все отмеченные выше элементы. Этот метод используется для связи свойств и компонентов структурных тегов.</p> <p>Данные связи тегов могут быть установлены как внутри шаблонов, так и снаружи. Для этого специальная графическая библиотека проекта содержит модифицированные объекты, которые на первый взгляд выглядят как обычные кнопки, но на самом деле они содержат параметризованную информацию для вызова шаблонов.</p> <p>Все сценарии для реализации такой заранее подготовленной генерации прототипов должны быть скопированы в соответствующий проект. Обратитесь к заключительным шагам (8-10) описанным в конце данной главы. Без этих функций проекта эти прототипы реализовать не удастся.</p>
4	<p>Данный кадр должен использоваться как диалоговое окно процесса.</p> <p>Для этого создайте окно кадра на временном кадре (он необходим только для этого промежуточного этапа) и соедините свойство <i>Picture Name (Название кадра)</i> данного окна с кадром, в котором содержится модуль кадра</p>
5	Скопируйте данное окно кадра в графическую библиотеку.


Теперь шаблон может быть использован в технологическом кадре неоднократно. Связь с тегами процесса устанавливается автоматически при вводе имени.

Шаг	Тип	Конфигурация
6	Определение тегов	Определите тег (процесса) для соответствующего типа данных (например, PID_1 типа PID).
7	Создание примера	<p>Скопируйте модуль окна кадра из графической библиотеки.</p> <p>Назначьте окну кадра имя использованного тега (процесса) (например, PID_1): Установите <i>Picture Window Object (Объект окна кадра)</i> → <i>Picture Window (Окно кадра)</i> → <i>Object Name (Название кадра)</i> → <i>Static (Статический)</i>  в PID_1</p>

Когда модуль кадра находится в кадре, ему дается имя структурного тега (процесса), который содержит данные о состоянии объекта. Во время исполнения модуль кадра получает данные о состоянии автоматически.

Для этих прототипов кадров используется несколько сценариев. Для того чтобы иметь возможность использования подготовленных Вами сценариев, необходимо

настроить следующие сценарии из примера проекта. Следуйте изложенным ниже инструкциям:

Шаг	Тип	Конфигурация
8	Копирование функциональных файлов	Из пути
9	Определение области видимости.	Для того чтобы сделать новые функции доступными в проекте в WinCC, запустить редактор глобальных сценариев (Global Script →  R → Open) теперь Вы можете сделать новые функции доступными в дереве функций путем нажатия на кнопку Regenerate Header (Перегенерация заголовка). Теперь новые функции должны быть видны в списке функций проекта.
10	Передача модифицированных объектов	Модифицированные объекты делаются доступными с помощью библиотеки проекта. В новый проект, в библиотеку которого Вы еще не успели сохранить своих собственных символов, Вы можете просто скопировать библиотеку проекта-примера: Сору (Копировать) → Ваш путь \Library! Также для передачи модифицированных объектов можно использовать механизм экспорта. Необходимые символы проекта – примера экспортируйте в файлы .emf (File (Файл) → Export (Экспорт)) и импортируйте эти файлы в собственный проект с помощью меню Insert (Вставка) → Import (Импорт). Перенесите эти символы в библиотеку проекта путем перетаскивания. Для этого рекомендуется использовать отдельную директорию, например, Template.

Модули кадров (стандартизированные куски кадров или шаблоны) поддерживают различные варианты сохранения. Они могут моделироваться как объекты со ссылками на теги и храниться в графической библиотеке. После этого они запрашиваются из графической библиотеки и автоматически наполняются информацией. Таким образом, больше не нужно связывать отдельные теги и привязывать их графическим объектам, таким как поля ввода/вывода, панелям и т.д.


- При использовании прототипов кадров есть несколько способов для задания имен экземпляров модулей кадров. Ниже перечислены возможные способы:
- Название экземпляра определяется при выделении окна кадра: для этого есть специальная функция (EnableTemplateInstance), исполняемая по открытию окна кадра.
- Название экземпляра определяется через тег ввода/вывода, чтение которого происходит автоматически с помощью соответствующего сценария. Это делается с использованием специального модифицированного объекта *InstanceCallButton+Template*.
- Название экземпляра передается по кнопке напрямую вызванному окну кадра: это делается при помощи специального модифицированного объекта *InstanceCallButtons+Template*.

Более подробные примеры по этой тематике можно найти в примерах проектов на WinCC CD-ROM (\Samples).

3.3.11.6 ОСХ объекты

Объекты ОСХ или ActiveX представляют собой модули кадров, доступные в качестве загружаемых компонентов. WinCC предлагает довольно большое количество таких объектов, например, элемент управления цифровыми/аналоговыми часами WinCC Digital/Analog Clock Control.

Эти модули могут быть очень просто встроены в Ваши технологические кадры.

Шаг	Тип	Конфигурация
1	Вставка ОСХ объекта	Из палитры компонентов <i>графического дизайнера</i> выберите элемент <i>OLE Control</i> (ОСХ). Перетащите объект на технологический кадр, удерживая  , и из появившегося диалогового окна выберите необходимый компонент.
2	Соединение свойств	Вставленный объект, так или иначе, имеет свойства и события. Доступные свойства и события зависят от конкретного ОСХ. Связь с тегами устанавливается в свойстве соединения с драйвером процесса.

Создание

ОСХ модули кадров должны быть созданы с помощью отдельных программных продуктов. Это может быть Microsoft Visual C++ 5 или Microsoft Visual Basic 5. Этот метод используется для модификации и улучшения модулей кадров. Модули ОСХ очень мощные, и их нельзя создать в WinCC. Поэтому, для их создания и модификации всегда следует обращаться к сторонним производителям сред разработки.

В отличие от программирования ОСХ объектов, технология прототипов кадров может быть реализована ресурсами WinCC. При этом не нужно знать ничего об ОСХ программировании.

На сегодняшний день уже доступно довольно много таких модулей. Среди всего прочего готовые модули можно найти в PCS7.

Регистрация

Созданные или приобретенные ОСХ модули должны регистрироваться на соответствующей WinCC станции. Увидеть доступные на wincc станции ОСХ объекты можно в диалоговом окне выбора в графическом дизайнера (см описание выше). Все зарегистрированные на компьютере ОСХ объекты перечислены в соответствующем списке. ОСХ объекты хранятся на рабочей станции в виде файлов с расширением .ОСХ или .dll.

Если модуль еще не зарегистрирован, это можно сделать в диалоговом окне WinCC OLE Control. Это диалоговое окно содержит кнопку для регистрации и для удаления регистрационной информации выбранного элемента.

Для возможности регистрации соответствующий файл должен присутствовать на рабочей станции WinCC.

Совместимость и функциональность ОСХ компонентов должна быть протестирована человеком, производящим настройку системы. В WinCC необходимо тестировать только те модули ОСХ, которые имеют пометку о совместимости с WinCC.

3.3.10 Online конфигурация (в режиме исполнения) - замечания, ограничения

В отношении online конфигурации необходимо выделить несколько моментов. По различным причинам, некоторое количество изменений либо невозможно сделать в режиме online, либо можно сделать только при определенных условиях, либо изменения вступят в силу несколько позже.

Проводник WinCC

Следующие изменения не допустимы:

- Изменение типа компьютера в списке компьютеров

Во время исполнения, не возможно следующее:

- удаление/переименование тегов
- изменение типа данных тега

Alarm Logging

Следующие изменения не допустимы:

- изменение архивов/отчетов
- изменение групповых сообщений
- любое сообщение после 500 одиночных сообщений активно в режиме исполнения

Во время исполнения, не возможно следующее:

- без ограничений

Tag Logging

Следующие изменения не допустимы:

- без ограничений.

Во время исполнения, не возможно следующее:

- таблицы пользовательских архивов могут создаваться, но не изменяться
- удаление данных в подсистеме Tag Logging и из пользовательских архивов

Исключение по конфигурированию во время исполнения:

API времени исполнения подсистемы Tag Logging может быть использовано для редактирования и удаления таблиц пользовательских архивов

Global Script

Следующие изменения не допустимы:

- изменения, произведенные в сценарии мастера доступны только после перезапуска графического дизайнера

Во время исполнения, не возможно следующее:

- без ограничений

4 Курс Си для WinCC

В WinCC предусмотрены различные способы задания динамических свойств объектов. Среди них такие, как связи с тегом, динамические диалоги и прямые соединения. При помощи этих средств можно задавать сложные динамические свойства. Тем не менее, их функциональные возможности ограничены и не всегда полностью отвечают предъявляемым требованиям. Значительно более широкие возможности пользователю предоставляют процедуры Си, функции проекта или процедуры WinCC. Они создаются с использованием языка сценариев WinCC Си. Для большинства приложений исчерпывающего знания Си не требуется. Однако, для максимально полного использования возможностей Си как языка сценариев WinCC, необходимы базовые сведения об этом языке программирования. Эти сведения Вы сможете почерпнуть из данного курса.

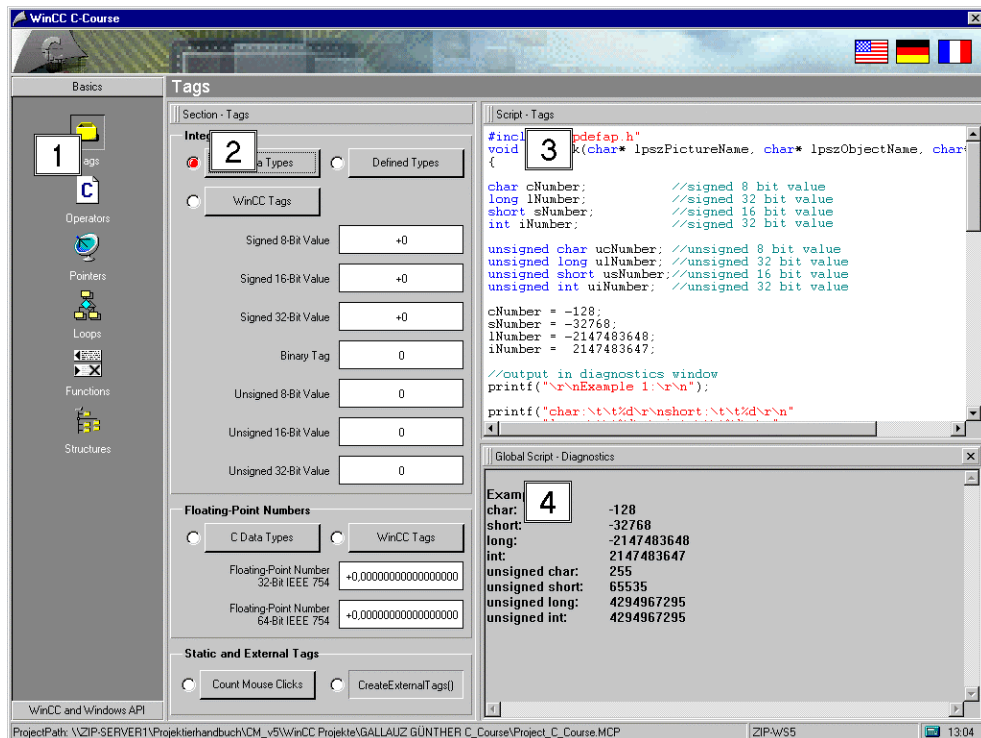
Целевая аудитория

Данный курс предназначен для первоначального ознакомления с основными принципами работы с Си для тех, кто не имеет навыков программирования на этом языке. Квалифицированные Си-программисты смогут изучить особенности применения Си в среде WinCC. Примеры проектов к данному курсу можно скопировать на Ваш жесткий диск непосредственно из online документации. По умолчанию они будут помещены в каталог C:\Configuration_Manual folder.



Project_C_Course

Примеры проектов



Окно примеров проектов разбито на несколько частей. Ниже приводится их перечень:

- Навигационная панель (1): Навигационная панель позволяет выбирать пиктограммы, соответствующие различным главам.
- Окно главы (2): Окно главы отображает пиктограммы, относящиеся к отдельным главам. Эти пиктограммы объединяют все примеры, приведенные в соответствующей главе. Большинство примеров выполнено в форме кнопок.
- Окно сценария (3): В окне сценария отображается код примера, выбранного в окне главы. Выбранный в данный момент пример в окне главы подсвечивается красным цветом.
- Окно диагностики (4): В окне диагностики приводится вся выходная информация различных примеров, генерируемая функцией printf().

4.1 Среда разработки сценариев Си


Для создания сценариев Си в WinCC используются два различных редактора. Один из них — редактор процедур — используется в графическом редакторе для создания процедур Си, связанных с объектами, другой — редактор глобальных сценариев — применяется для программирования функций проекта и глобальных процедур. Синтаксис языка сценариев соответствует стандарту ANSI C.

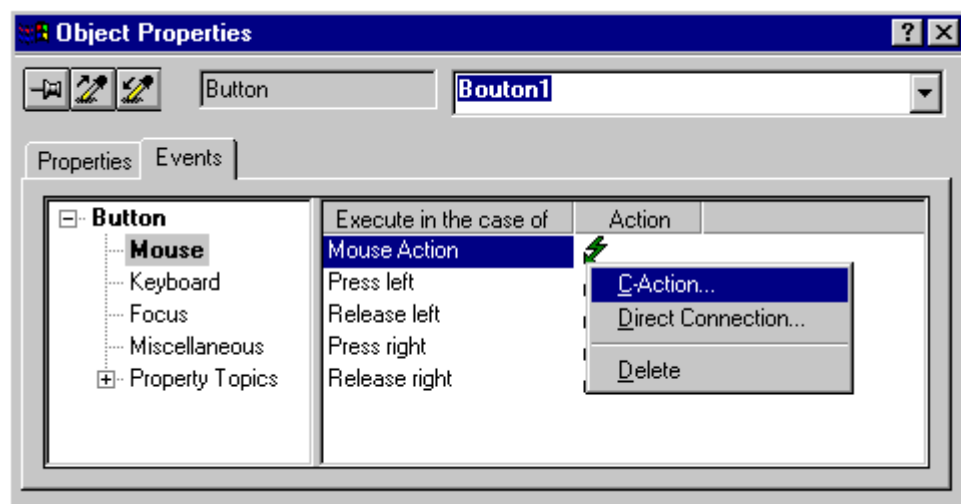
Другой областью применения языка Си в WinCC является создание динамических мастеров. Для этого предусмотрен отдельный редактор, использование которого разъясняется в отдельном примере, и в общем обзоре рассматриваться не будет.

4.1.1 Редактор процедур и графический редактор

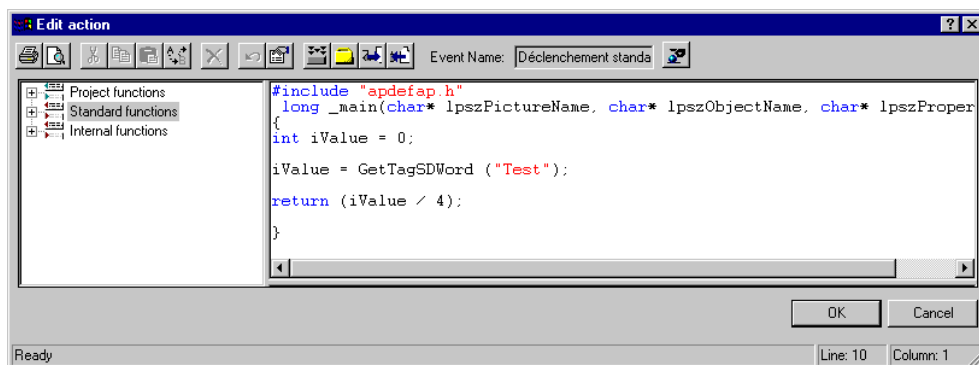
В графическом редакторе свойства объектов можно сделать динамическими при помощи процедур Си. Кроме того, процедуры Си могут использоваться для обработки событий объекта.

Редактор процедур

Для конфигурирования *процедур Си* используется специальный редактор процедур. Данный редактор запускается из диалогового окна свойств объекта (*Object Properties*) нажатием правой кнопки мыши  на нужном *свойстве* или *событии* и последующим выбором процедуры Си во всплывающем меню. Уже существующие процедуры Си обозначаются зеленой стрелкой напротив *свойства* или *события*.



В редакторе процедур можно осуществлять программирование *процедур Си*. Для *процедур Си*, связанных со свойствами объекта, следует задавать триггер. Для *процедур Си*, связанных с событиями это делать не обязательно, т.к. событие само по себе является триггером. Готовые процедуры Си необходимо компилировать. Если компилятор не обнаружит ошибок, редактор процедур можно закрыть нажатием на кнопку *OK*.



Структура процедур Си

В целом процедура Си соответствует функции в Си. Существует два различных типа *процедур Си*: процедуры, созданные для *свойств*, и процедуры, созданные для *событий*. В общем случае *процедура* для *свойства* используется для задания значения этого *свойства* в соответствии с различными внешними факторами (такими например, как значение какого-либо тега). Для *процедуры* данного типа необходимо задавать триггер, управляющий его запуском. Процедура Си для *события* используется для обработки этого *события*.

Процедура Си для свойства

```
#include "apdefap.h"
long _main(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{
    /*1*/ long lReturnValue;
    /*2*/ lReturnValue = GetTagSDword("S32i_course_test_1");
    /*3*/ return lReturnValue;
}
```

Приведенный выше фрагмент программы представляет типичную процедуру Си для свойства. Значение отдельных частей данного фрагмента разъясняется ниже.

- **Заголовок (серый):** Первые три строчки, выделенные серым цветом, образуют заголовок процедуры Си. Заголовок генерируется автоматически и не может быть изменен. За исключением типа возвращаемого значения (*long* в приведенном фрагменте), заголовки функций идентичны для всех свойств. *Процедуре Си* передается три параметра: название экранной формы (*lpszPictureName*), название объекта (*lpszObjectName*), и название свойства (*lpszPropertyName*).
- **Объявление переменных (1):** В данной части (первой из частей, доступных для редактирования) производится объявление используемых переменных. В приведенном примере декларируется одна переменная типа *long*.
- **Вычисление значения (2):** В данной части производится вычисление значения свойства. В приведенном примере считывается значение одного тега WinCC.
- **Возврат значения (3):** Вычисленное значение присваивается свойству. Это делается при помощи оператора *return*.

Процедуры Си для событий


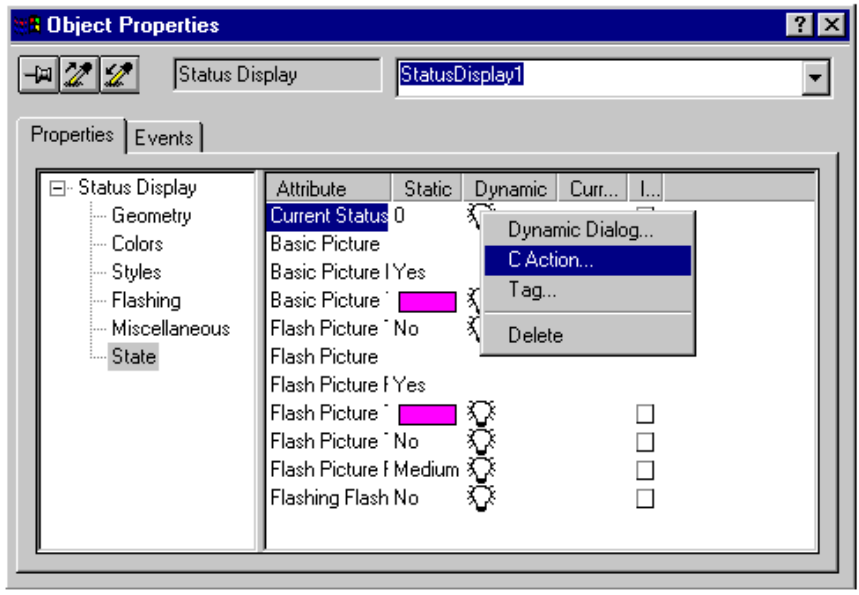
```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{
    /*1*/ long lValue;
    /*2*/ lValue = GetTagSDWord("S32i_course_test_1");
        SetLeft(lpszPictureName, lpszObjectName, lValue);
}
```


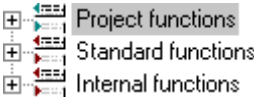

Приведенный выше фрагмент программы представляет типичную процедуру Си для события. Значение отдельных частей данного фрагмента разъясняется ниже.

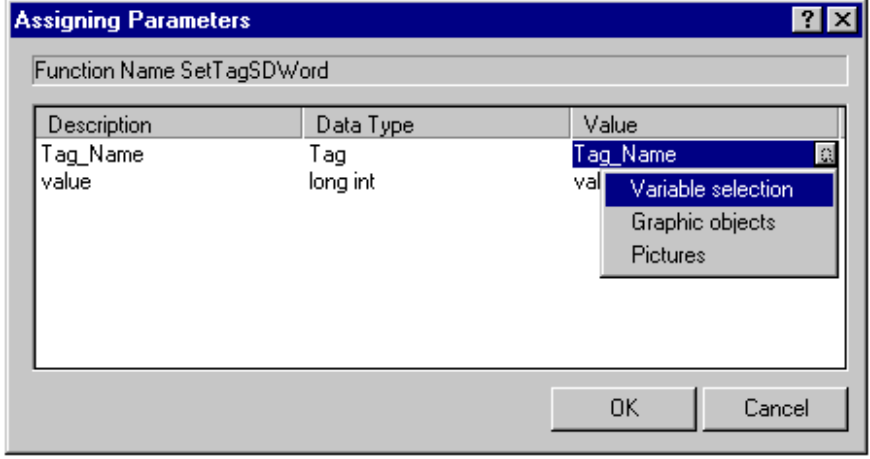



- **Заголовок (серый):** Первые три строчки, выделенные серым цветом, образуют заголовок процедуры Си. Заголовок генерируется автоматически и не может быть изменен. Заголовок функции отличается для различных типов событий. *Процедуре Си* передается три параметра: название экранной формы (*lpszPictureName*), название объекта (*lpszObjectName*), и название свойства (*lpszPropertyName*). Параметр *lpszPropertyName* содержит существенную информацию для событий, реагирующих на изменение свойств. Кроме того, имеется возможность передачи дополнительных параметров, описывающих событие.
- **Объявление переменных (1):** В данной части (первой из частей, доступных для редактирования) производится объявление используемых переменных. В приведенном примере декларируется одна переменная типа *long*.
- **Обработка события (2):** В данной части исполняются процедуры по обработке соответствующих событий. В приведенном примере считывается значение одного тега WinCC. Считанное значение присваивается координате X рассматриваемого объекта. Возвращаемое значение процедуры Си для события имеет тип *void*, т.е. оно не требуется.



Создание процедуры Си

Приведенная ниже таблица описывает последовательность шагов, необходимых для создания процедуры Си.

Шаг	Создание процедуры Си
1	<p>Запустите графический редактор (<i>Graphics Designer</i>).</p> <p>Откройте требуемую экранную форму WinCC.</p> <p>Откройте диалоговое окно свойств нужного объекта (<i>Object Properties</i>).</p>
2	<p><i>Редактор процедур</i> запускается нажатием правой кнопки мыши  на нужном свойстве или событии объекта и последующим выбором процедуры Си во всплывающем меню.</p> 
3	<p>При этом откроется окно <i>редактора процедур</i>.</p> <p>В нем будет отображена базовая структура функции.</p> <p>Помимо прочего при этом автоматически создается заголовок процедуры Си. Этот заголовок не может быть изменен.</p> <p>В первой строке заголовка процедуры Си включается файл <i>apdefap.h</i>. Данный файл используется для объявления всех функций проекта (<i>project functions</i>), стандартных функций (<i>standard functions</i>) и внутренних функций (<i>internal functions</i>) в <i>Cu-действии</i></p> <p>Вторая часть заголовка процедуры Си это заголовок функции. Заголовок функции содержит информацию о возвращаемом из процедуры Си значении, а также о передаваемых параметрах, которые в ней могут использоваться.</p> <p>Третья часть заголовка процедуры Си представлена открывающей фигурной скобкой. Ее нельзя удалить. Сам текст программы располагается между открывающей и закрывающей фигурными скобками.</p>

Шаг	Создание процедуры Си
4	<p>Другая автоматически генерируемая часть программы состоит из двух блоков комментариев. Они служат для предоставления редактору перекрестных ссылок (<i>CrossReference</i>) доступа к внутренним данным процедуры Си. Кроме того, они необходимы для переприсваивания в самой <i>процедуре Си</i>. Если в этих возможностях не будет необходимости, блоки комментариев можно удалить. Первый блок используется для описания тегов WinCC, использующихся в <i>процедуре Си</i>. В тексте программы вместо реальных имен переменных следует использовать их описанные имена. Второй блок комментариев используется для описания экранных форм WinCC, использующихся в <i>процедуре Си</i>. Здесь также следует использовать описанное имя экранной формы, а не ее реальное название. Пример текста программы, относящийся к этой теме приводится ниже, после таблицы. В нем содержится описание тега и экранной формы WinCC и демонстрируется их использование.</p>
5	<p>Необходимо также написать сам код функции, выполняющий требуемые вычисления, процедуры и т.п. При программировании пользователю доступен ряд вспомогательных инструментов. Один из них — диалоговое окно выбора тегов. Он открывается нажатием на изображенную ниже кнопку панели инструментов. В появившемся окне <i>Select Tag</i> можно найти необходимый тег WinCC и подтвердить выбор нажатием на <i>OK</i>. Имя выбранного тега будет вставлено в текст процедуры Си за текущей позицией курсора.</p> 
6	<p>Другой вспомогательный инструмент — дерево выбора функций в левом окне редактора процедур. Используя дерево функций можно автоматически вставить любую из <i>функций проекта, стандартных функций, или внутренних функций</i> в текст процедуры Си за текущей позицией курсора.</p>  <p>Для этого нужную функцию следует выбрать двойным щелчком мыши  D. После этого появится диалоговое окно присвоения параметров (<i>Assigning Parameters</i>), содержащее список всех передаваемых параметров и их типов. Передаваемые параметры можно задать при помощи столбца значения (<i>Value</i>). Помимо простого ввода текста имеется возможность использовать диалоги выбора тега (<i>Select Tag</i>), выбора графического объекта (<i>Graphic Objects</i>) и выбора экранной формы (<i>Pictures</i>). Для того чтобы вставить функцию после текущей позиции курсора в <i>процедуре Си</i>, подтвердите диалоговое окно нажатием на <i>OK</i>.</p>

Шаг	Создание процедуры Си
	
7	<p>Готовую функцию необходимо откомпилировать. Это делается нажатием на изображенную ниже кнопку панели инструментов.</p>  <p>Результаты процесса компиляции отображаются в левом нижнем углу редактора процедур. Они включают число найденных ошибок и предупреждений. Найденные ошибки не позволят исполнять процедура Си. В отличие от них, предупреждения указывают на места возможного возникновения ошибок при исполнении процедуры Си. При хорошем стиле программирования результаты компиляции процедуры Си должны выглядеть следующим образом: <i>0 Error(s), 0 Warning(s)</i>.</p> <p>0 Error(s), 0 Warning(s)</p> <p>Ошибки, обнаруженные во время компиляции, будут отображены в окне результатов. Двойным щелчком  на сообщении об ошибке можно перейти непосредственно к соответствующей ему строке программы.</p> <pre>line 2 : error (0086) : function '_main' does not return a value Größe Quellcode : 576 Zeichen - Größe P-Code 0 Bytes</pre>
8	<p>Для <i>процедур Си</i>, связанных со свойствами объекта, следует задавать триггер. Для <i>процедур Си</i>, связанных с событиями это делать не обязательно, т.к. событие само по себе является триггером.</p> <p>Триггер задается нажатием на изображенную ниже кнопку панели инструментов. Вы можете использовать временной триггер или триггер, связанный с тегом.</p> <p>Event Name: <input type="text" value="Default trigger"/> </p>
9	<p>При нажатии на кнопку <i>OK</i> редактора процедур, запрограммированная процедура Си будет связана с соответствующим свойством или событием. Свойство или событие, связанное с процедурой Си, будет помечено зеленой</p>

Шаг	Создание процедуры Си
	стрелкой.  

Описание тегов и экранных форм WinCC

```
#include "apdefap.h"
long _main(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{
// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
#define S32I_COURSE_TEST_1 "S32i_course_test_1"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
#define CC_0_STARTPICTURE_00 "cc_0_startpicture_00.Pdl"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

SetTagSDWord(S32I_COURSE_TEST_1,100);
OpenPicture(CC_0_STARTPICTURE_00);
return 0;
}
```

При создании новой процедуры Си в нее автоматически будут вставлены два блока комментариев.

Они служат для предоставления редактору перекрестных ссылок (*CrossReference*) доступа к внутренним данным процедуры Си. Кроме того, они необходимы для переписывания в самой *процедуре Си*.

- **Описание переменной:** Первый блок используется для описания тегов WinCC, использующихся в *процедуре Си*. Этот блок комментариев начинается строкой *// WINCC:TAGNAME_SECTION_START* и заканчивается строкой *// WINCC:TAGNAME_SECTION_END*. Между этими двумя строками описываются имена всех тегов WinCC, использующихся в *процедуре Си*. Определение производится при помощи директивы препроцессора *#define*, за которой следует определяемое имя (в примере — это *S32I_COURSE_TEST_1*) и имя тега WinCC (в примере — это *S32i_course_test_1*).
- **Описание экранной формы:** Второй блок комментариев используется для описания экранных форм WinCC, использующихся в *процедуре Си*. Этот блок комментариев начинается строкой *// WINCC:PICNAME_SECTION_START* и заканчивается строкой *// WINCC:PICNAME_SECTION_END*. Между этими двумя строками описываются названия всех экранных форм WinCC, использующихся в *процедуре Си*. Описание названий экранных форм производится в той же форме, что и приведенное выше описание имен тегов.
- **Применение:** В тексте программы следует использовать описания вместо реальных имен тегов и названий экранных форм. Перед компиляцией процедуры Си препроцессор производит замену всех описаний реальными именами.

4.1.2 Редактор глобальных сценариев

Редактор глобальных сценариев используется для создания функций проекта, стандартных функций и процедур.

Функции проекта

Если одни и те же элементы программ часто используются в разных процедурах Си, они могут быть оформлены в виде функций проекта. Функции проекта могут вызываться из процедур Си проекта WinCC точно так же, как любые другие функции. Ниже перечислены преимущества использования функций проекта по сравнению с написанием всего кода в процедуре Си:

- **Единая среда редактирования:** Изменения, внесенные в функцию проекта, влияют на все использующие ее процедуры Си. В противном случае все процедуры Си необходимо править вручную. Использование функций проекта упрощает не только конфигурирование, но и отладку, и последующую поддержку.
- **Возможность повторного использования:** После того, как *функция проекта* была запрограммирована и полностью оттестирована, она может использоваться где угодно не требуя при этом дополнительного конфигурирования и тестирования..
- **Уменьшение размера файла кадра:** Если не весь текст программы процедуры Си разместить непосредственно в его теле, то размер файла кадра станет меньше. При этом кадр будет загружаться быстрее и повысится общая производительность системы.
- **Защита паролем:** *Функции проекта* могут быть защищены от изменений при помощи системы паролей. Это помогает предотвратить нежелательные изменения конфигурации системы, а также позволяет защитить ваше ноу-хау.

Функции проекта доступны только внутри этого проекта. Они хранятся в каталоге WinCCProjectFolder\LIBRARY и описываются в файле ap_pbib.h, который находится в том же каталоге.

Пользователю также доступен ряд *стандартных функций*. В отличие от *функций проекта*, стандартные функции могут использоваться во всех проектах WinCC. Существующие *стандартные функции* защищены от изменений, но пользователь может создавать свои собственные *стандартные функции*..

Стандартные функции отличаются от функций проекта лишь тем, что они доступны из любого проекта, в то время как последние могут использоваться только внутри того проекта, в котором они были созданы. Стандартные функции хранятся в каталоге WinCCInstallationFolder\APLIB и описываются в файле ap_glob.h, расположенном там же.

Внутренние функции

Кроме функций проекта и стандартных функций существуют внутренние функции. Помимо прочих к ним относятся стандартные функции Си. Изменить существующую стандартную функцию или создать новую нельзя.


Процедуры

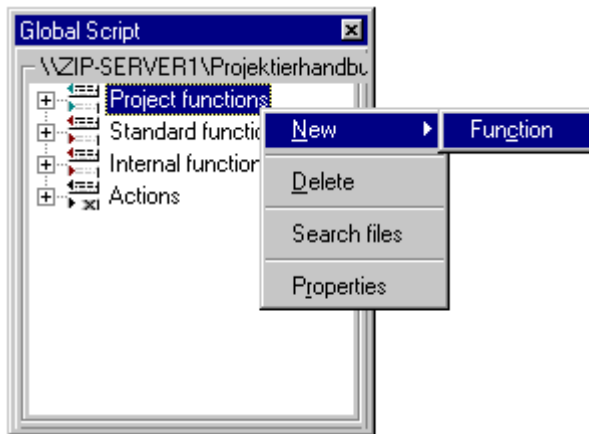
В отличие от описанных ранее функций, процедуры не вызываются из процедур Си или других функций. Для процедуры необходимо задать триггер, управляющий его запуском. Процедура выполняется независимо от того, какая экранная форма выбрана в момент исполнения проекта. Процедуры можно делать глобальными, т.е. общими для всех проектов. В этом случае они хранятся в каталоге WinCCProjectFolder\PAS folder. Вы также можете создать локальные процедуры (заданные для определенного компьютера), которые располагаются в каталоге WinCCProjectFolder\ComputerName\PAS.

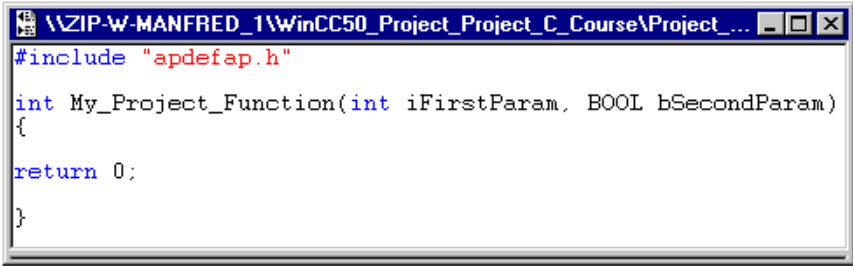




Если в списке запуска компьютера отмечена опция *Global Script Runtime*, то все глобальные и локальные процедуры, относящиеся к данному компьютеру, будут активизированы при запуске проекта.

Создание функций проекта

Процедуры, необходимые для создания функции проекта и стандартной функции идентичны. Приведенное ниже описание относится также к созданию стандартных функций.


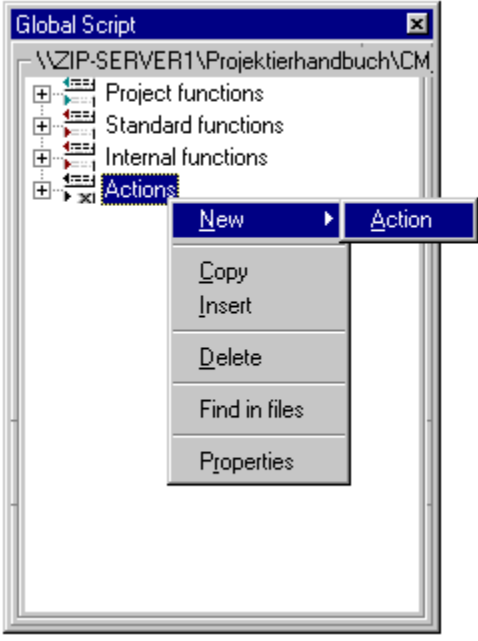
Шаг	Процедура создания функции проекта
1	Запустите редактор <i>глобальных сценариев</i> .
2	Щелкните правой кнопкой мыши  на пункт <i>Project Functions (Функции проекта)</i> и во всплывающем меню выберите <i>New (Создать) → Function (Функцию)</i> . После этого будет создан базовый шаблон новой <i>функции проекта</i> .

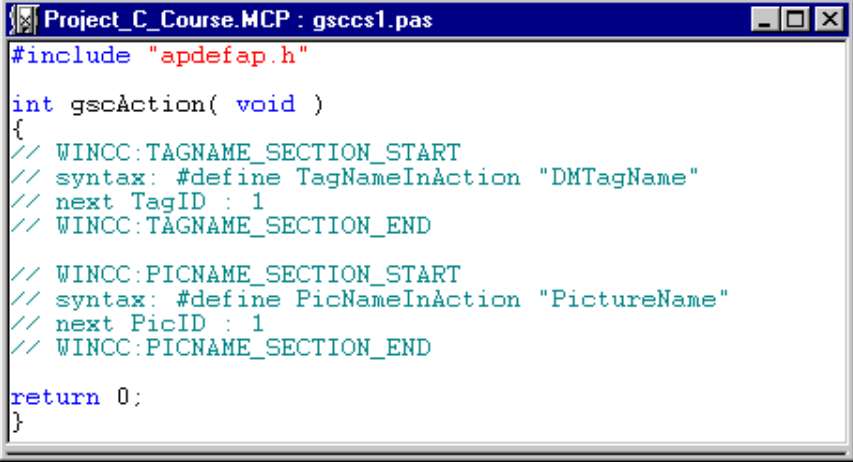



Шаг	Процедура создания функции проекта
3	<p><i>Функция проекта</i> от начала и до конца создается пользователем. В ней нет участков кода, которые недоступны для редактирования.</p> <p>Напишите заголовок функции. Функции должно быть дано имя, по которому ее можно вызывать из <i>процедур Си</i> или других функций. Кроме того, следует указать тип возвращаемого функцией значения и список передаваемых параметров.</p> <p>Если в данной функции будут использоваться другие <i>функции проекта</i> или <i>стандартные функции</i>, необходимо включить заголовочный файл <i>apdefap.h</i>. Это делается с использованием директивы препроцессора <i>#include "apdefap.h"</i>, которую следует поместить перед заголовком функции.</p>  <pre>#include "apdefap.h" int My_Project_Function(int iFirstParam, BOOL bSecondParam) { return 0; }</pre>
4	<p>Напишите код самой функции.</p> <p>При этом пользователю доступны те же вспомогательные средства, что и при программировании <i>процедур Си</i>, в частности, диалоговые окна выбора тегов и функций.</p>
5	<p>Готовую функцию необходимо откомпилировать. Это делается нажатием на изображенную ниже кнопку панели инструментов.</p>  <p>Результаты компиляции отображаются в окне вывода. Перечисляются обнаруженные ошибки и предупреждения, а также выводится их общее количество. Двойным щелчком  на сообщении об ошибке можно перейти непосредственно к соответствующей ему строке программы.</p>  <pre>Compiling... line 3 : error (0086) : function 'My_Project_Function' does not return a value 1 Error(s).0 Warning(s)</pre>
6	<p>Используя изображенную ниже кнопку панели инструментов к <i>функции проекта</i> можно добавить описание, а также пароль для защиты от несанкционированного доступа.</p> 
7	<p>Готовую функцию проекта следует записать под подходящим именем.</p>

Создание глобальной процедуры

Создание глобальной и локальной процедуры выглядит абсолютно аналогично. Приведенное ниже описание относится также и к созданию локальных процедур.

Шаг	Процедура создания функции проекта
1	Запустите редактор <i>глобальных сценариев</i> .
2	<p>Щелкните правой кнопкой мыши  на пункте <i>Global Actions (Глобальные процедуры)</i> и во всплывающем меню выберите <i>New (Создать) → Action (Процедуру)</i>. После этого будет создан базовый шаблон новой процедуры.</p> 
3	<p>Заголовок процедуры генерируется автоматически и не может быть отредактирован.</p> <p>В дополнение к заголовку будут созданы два блока комментариев для описания тегов и экранных форм WinCC.</p> <p>Назначение этих блоков комментариев уже было описано выше в разделе, относящемся к процедурам Си.</p>

Шаг	Процедура создания функции проекта
	 <pre> Project_C_Course.MCP : gscs1.pas #include "apdefap.h" int gscAction(void) { // WINCC:TAGNAME_SECTION_START // syntax: #define TagNameInAction "DMTagName" // next TagID : 1 // WINCC:TAGNAME_SECTION_END // WINCC:PICNAME_SECTION_START // syntax: #define PicNameInAction "PictureName" // next PicID : 1 // WINCC:PICNAME_SECTION_END return 0; } </pre>
4	<p>Напишите код самой функции.</p> <p>При этом пользователю доступны те же вспомогательные средства, что и при программировании <i>процедур Си</i>, в частности, диалоги выбора тегов и функций.</p> <p>Возвращаемое <i>процедурой</i> значение имеет тип <i>int</i>. Тем не менее, это значение не может быть проанализировано пользователем. По умолчанию возвращается <i>0</i>.</p>
5	<p>Используя изображенную ниже кнопку панели инструментов к <i>процедуре</i>, точно также как к <i>функции проекта</i>, можно добавить описание, а также пароль для защиты от несанкционированного доступа.</p> <p>В отличие от функции, для <i>процедуры</i> необходимо задать триггер, управляющий его запуском. При установке триггера для процедуры у пользователя есть более широкий выбор, чем в случае триггера для процедуры Си, связанного с объектом. Помимо прочего можно запрограммировать разовое исполнение процедуры.</p> 
6	<p>Готовую <i>процедуру</i> необходимо записать.</p>

Тестовый вывод

Исполнение программы можно отследить по тестовому выводу. Это облегчает отладку и диагностику ошибок во время разработки. Тестовый вывод формируется при помощи функции `printf()`. Используя эту функцию можно выводить текстовые сообщения и текущие значения тегов. Для того чтобы увидеть текстовые сообщения необходимо сконфигурировать окно диагностики глобальных сценариев (Global Script Diagnostics Window).

Функция printf()

Функция `printf()` позволяет формировать тестовый вывод. Ниже приводится пример использования этой функции:

```
printf("I am %d years old\r\n", iAge);
```

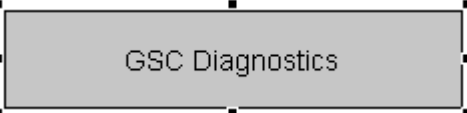
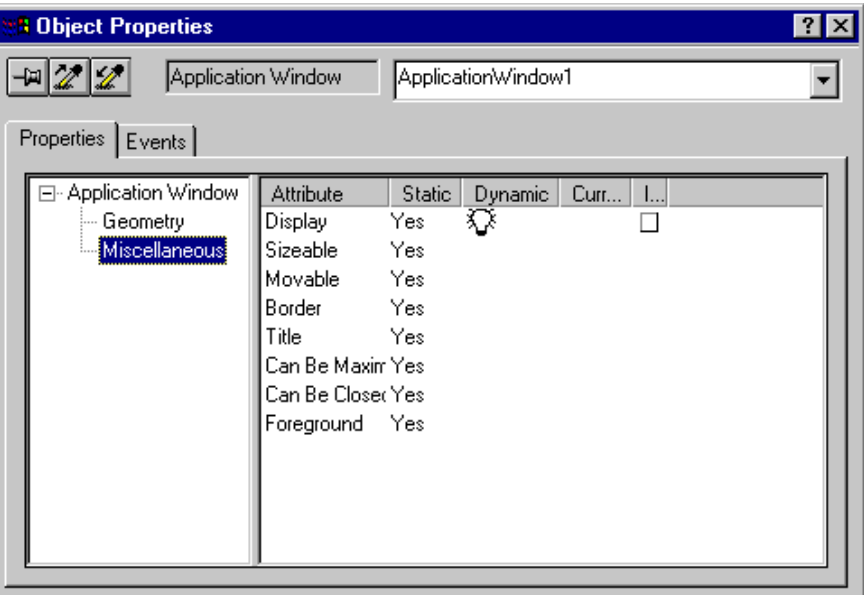
Функция `printf()` требует передачи как минимум одного параметра. Этим параметром является символьная строка. Тип и количество дополнительных передаваемых параметров зависит от этой текстовой строки.

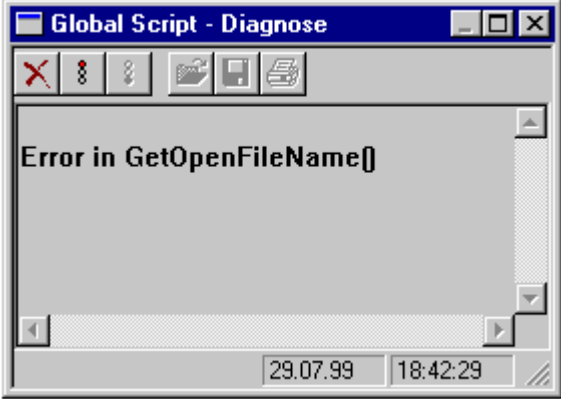
Функция `printf()` использует символ `%`, выступающий в роли идентификатора позиции для вставки значения переменной. Символ, следующий за знаком `%` определяет тип этой переменной. Используемая выше комбинация символов `%d` обозначает вывод десятичного числа. Другие возможные комбинации и их значения приводятся в следующей таблице:

Параметр	Описание
<code>%d</code>	Вывод десятичного числа (int или char)
<code>%ld</code>	Вывод переменной типа long в виде десятичного числа
<code>%c</code>	Вывод символа (char)
<code>%x</code>	Вывод числа в шестнадцатеричном формате (со строчными a...f)
<code>%X</code>	Вывод числа в шестнадцатеричном формате (с заглавными A...F)
<code>%o</code>	Вывод числа в восьмеричном формате
<code>%u</code>	Вывод десятичного числа (только для беззнаковых типов)
<code>%f</code>	Вывод числа с плавающей точкой в представлении с фиксированной точкой, т.е. 3.43234
<code>%e</code>	Вывод числа с плавающей точкой в экспоненциальном представлении, т.е. 23e+432
<code>%E</code>	То же самое, что и <code>%e</code> , только с заглавным символом E, т.е. 23E+432
<code>%s</code>	Вывод символьной строки (char*)
<code>%le</code>	Вывод переменной двойной точности
<code>%%</code>	Вывод символа %
<code>\n</code>	Вывод перевода строки (возврат каретки)
<code>\r</code>	Вывод перехода на одну строку вниз
<code>\t</code>	Вывод символа табуляции
<code>\\</code>	Вывод символа \

Окно диагностики глобальных сценариев

Текстовые сообщения, формируемые функцией `printf()`, отображаются в окне диагностики глобальных сценариев. В приведенной ниже таблице описываются процедуры, необходимые для того, чтобы сконфигурировать такое диагностическое окно:

Шаг	Процедура создания окна диагностики глобальных сценариев
1	Запустите редактор <i>Graphics Designer</i> . Откройте нужную экранную форму WinCC.
2	<p>Выберите <i>Smart Object (Интеллектуальный объект)</i> → <i>Application Window (Окно приложения)</i> и поместите его на экранную форму.</p> <p>После этого появится диалоговое окно для определения содержания окна приложения <i>Window Contents</i>. В списке выбора отметьте пункт <i>Global Script</i>. Закройте диалоговое окно нажатием на <i>OK</i>.</p> <p>После этого появится диалоговое окно шаблона <i>Template</i>. В списке выбора отметьте пункт <i>GSC Diagnostics (Диагностика GSC)</i>. Так же закройте диалоговое окно нажатием на <i>OK</i>.</p> 
3	<p>Для удобной работы с окном диагностики глобальных сценариев рекомендуется установить всем свойствам в разделе <i>Miscellaneous (Разные)</i> окна <i>Object Properties (Свойства объекта)</i> значение <i>Yes (Да)</i>.</p> 

Шаг	Процедура создания окна диагностики глобальных сценариев
4	<p>Если проект запущен, текстовый вывод, формируемый функцией <i>printf()</i>, осуществляется в окне диагностики. При помощи специальной кнопки панели инструментов можно приостановить обновление окна, после чего распечатать или записать его содержимое.</p> 

4.2 Переменные

В проекте WinCC Project_C_Course примеры, относящиеся к работе с переменными, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называющейся `ss_9_example_00.PDL`.



Tags

Переменные

Переменные — это объекты, представляющие данные, обрабатываемые программой. Переменную можно использовать только после того, как она была описана. Все используемые в программе переменные должны быть описаны до исполнения первой команды.

Переменную можно сравнить с контейнером. При помощи имени переменной мы даем контейнеру уникальное название. Тип содержимого контейнера определяется типом данных. Начальное содержимое контейнера задается инициализирующим значением. В большинстве случаев содержимое контейнера изменяется в процессе исполнения программы.

Описываемые здесь переменные не следует путать с тегами WinCC. Они используются только внутри кода программы.

Пример описания переменной иллюстрируется приведенным ниже фрагментом кода. Здесь описывается одна переменная типа `int` с именем `iNumber`. Строка программы заканчивается точкой с запятой. Префикс имени переменной указывает ее тип. Это требование не является обязательным, но позволяет в процессе написания программы по имени переменной сразу же определять ее тип.

```
int iNumber;
```

Кроме того, при описании переменной можно указать ее начальное значение.

```
int iNumber = 0;
```

Константы

Помимо переменных в программе также могут использоваться константы, формируемые непосредственным указанием численного значения. Для пояснения смысла такого численного значения при помощи директивы `#define` можно описать соответствующую символьную константу.

В приведенном ниже фрагменте программы показан пример описания символьной константы. Здесь описывается символьная константа `MAX_INT_VALUE` со значением `2147483647`. Обратите внимание, что в данном случае строка программы не должна заканчиваться точкой с запятой. Символьные константы принято записывать заглавными буквами, чтобы их можно было легко отличить от переменных.

```
#define MAX_INT_VALUE 2147483647
```

Типы данных

В Си используются перечисленные ниже типы данных.

Тип данных	Описание
char	Один байт, может содержать один символ.
int	Целое значение
float	Число с плавающей точкой одинарной точности
double	Число с плавающей точкой двойной точности

Размер переменной типа char — один байт. Ее содержимое может интерпретироваться либо как символ, либо как число.

Описатель целого типа данных может предваряться ключевыми словами signed (со знаком) или unsigned (без знака).

Описатель целого типа данных может также предваряться ключевыми словами long (длинное) или short (короткое). При использовании этих ключевых слов сам описатель int можно опустить. Размер переменной типа short (или short int) — два байта, а переменной типа long (или long int), так же как и переменной типа int — четыре байта.

Тип данных double отличается от типа с плавающей точкой только диапазоном значений. Числа могут быть представлены типом double с большей точностью.

Размер переменной типа float — четыре байта, в то время как размер переменной типа double составляет 8 байт.

Диапазоны значений типов данных

Каждый тип данных может представлять числа в определенном диапазоне значений. Различия определяются разным размером переменных и наличием или отсутствием знака.

Тип данных	Диапазон значений
int	от -2 147 483 648 до 2 147 483 647
unsigned int	от 0 до 4 294 967 295
short	от -32 768 до 32 767
unsigned short	от 0 до 65 535
long	от -2 147 483 648 до 2 147 483 647
unsigned long	от 0 до 4 294 967 295
char	от -128 до 127 (все символы ASCII)
unsigned char	от 0 до 255 (все символы ASCII)
float	от -10^{38} до 0^{38}
double	от -10^{308} до 0^{308}

4.2.1 Пример 1 — Типы данных Си (целые)

В данном примере использующиеся в Си по умолчанию типы данных применяются для отображения целых чисел. Пример сконфигурирован для изображенной ниже кнопки в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button1

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    char cNumber;           //signed 8 bit value
    long lNumber;          //signed 32 bit value
    short sNumber;         //signed 16 bit value
    int iNumber;           //signed 32 bit value

    unsigned char ucNumber; //unsigned 8 bit value
    unsigned long ulNumber; //unsigned 32 bit value
    unsigned short usNumber; //unsigned 16 bit value
    unsigned int uiNumber;  //unsigned 32 bit value

    cNumber = -128;
    sNumber = -32768;
    lNumber = -2147483648;
    iNumber = 2147483647;

    //output in diagnostics window
    printf("\r\nExample 1:\r\n");

    printf("char:\t\t%d\r\nshort:\t\t%d\r\n"
           "long:\t\t%d\r\nint:\t\t%d\r\n",
           cNumber, sNumber, lNumber, iNumber);

    ucNumber = 255;
    usNumber = 65535;
    ulNumber = 4294967295;
    uiNumber = 4294967295;

    //output in diagnostics window
    printf("unsigned char:\t\t%u\r\nunsigned short:\t\t%u\r\n"
           "unsigned long:\t\t%u\r\nunsigned int:\t\t%u\r\n",
           ucNumber, usNumber, ulNumber, uiNumber);
}
```

- Первые три строки образуют заголовок процедуры Си. Этот заголовок не может быть изменен.
- В следующей части описываются переменные. Описывается по одной переменной каждого из типов *char*, *long*, *short* и *int* и их беззнаковые аналоги. Имена переменных содержат префикс, описывающий тип переменной. Это требование не является обязательным, но позволяет в процессе написания программы по имени переменной сразу же определять ее тип. В качестве комментария в каждой строке указывается размер переменной (комментарии начинающиеся с комбинации символов // выделяются зеленым цветом).
- В следующей части переменным присваиваются значения. Это делается при помощи оператора присваивания =. Используемые в примере значения в точности совпадают с границами диапазонов значений, представимых переменными соответствующих типов.

- Численные значения выводятся в *окне диагностики* с использованием функции *printf()*. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 1:  
char:          -128  
short:         -32768  
long:          -2147483648  
int:           2147483647  
unsigned char: 255  
unsigned short: 65535  
unsigned long:  4294967295  
unsigned int:   4294967295
```

4.2.2 Пример 2 — Пользовательские типы данных (целые)

Вместо стандартных типов данных, доступных в Си, можно описывать пользовательские типы данных. Однако эти пользовательские типы данных являются лишь переименованием реальных типов данных. В данном примере для представления целых чисел используются различные пользовательские типы данных. Пример сконфигурирован для изображенной ниже кнопки *Button2* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button2

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    CHAR cNumber;           //signed 8 bit value
    SHORT sNumber;         //signed 16 bit value
    LONG lNumber;          //signed 32 bit value
    INT iNumber;           //signed 32 bit value

    BOOL bNumber;          //TRUE or FALSE
    BYTE byNumber;         //unsigned 8 bit value
    WORD wNumber;          //unsigned 16 bit value
    DWORD dwNumber;        //unsigned 32 bit value
    UINT uiNumber;         //unsigned 32 bit value

    cNumber = -128;
    sNumber = -32768;
    lNumber = -2147483648;
    iNumber = 2147483647;

    //output in diagnostics window
    printf("\r\nExample 2:\r\n");

    printf("CHAR:\t\t%d\r\nSHORT:\t\t%d\r\n"
           "LONG:\t\t%d\r\nINT:\t\t%d\r\n",
           cNumber, sNumber, lNumber, iNumber);

    bNumber = TRUE;
    byNumber = 255;
    wNumber = 65535;
    dwNumber = 4294967295;
    uiNumber = 4294967295;

    //output in diagnostics window
    printf("BOOL:\t\t%u\r\nBYTE:\t\t%u\r\nWORD:\t\t%u\r\n"
           "DWORD:\t\t%u\r\nUINT:\t\t%u\r\n",
           bNumber, byNumber, wNumber, dwNumber, uiNumber);
}
```

- В первой части производится описание переменных. Описывается по одной переменной каждого из пользовательских типов *CHAR*, *SHORT*, *LONG* и *INT* и их беззнаковые аналоги *BYTE*, *WORD*, *DWORD* и *UINT*. Кроме того, описывается переменная типа *BOOL*. Переменным типа *BOOL* могут присваиваться только предопределенные значения *TRUE* или *FALSE*. Как и в предшествующем примере, имена переменных содержат префикс, определяющий их тип.
- В следующей части переменным присваиваются значения. Используемые в примере значения опять в точности совпадают с границами диапазонов значений, представимых переменными соответствующих типов.

- Численные значения выводятся в *окне диагностики* с использованием функции `printf()`. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```

Example 2:
CHAR:          -128
SHORT:         -32768
LONG:          -2147483648
INT:           2147483647
BOOL:          1
BYTE:          255
WORD:          65535
DWORD:         4294967295
UINT:          4294967295

```

Определение типов

Используемые в данной главе типы данных были определены с использованием команды `typedef`. Приведенный ниже фрагмент кода показывает, как был определен тип `BYTE`. Тип `BYTE` — это лишь псевдоним стандартного типа данных языка Си `unsigned char`. Вы также можете использовать Ваши собственные псевдонимы.

```
typedef unsigned char BYTE;
```

В следующей таблице приводятся определенные типы данных, соответствующие стандартным типам, используемым в Си:

Определенный тип данных	Стандартный тип данных Си
BOOL	int
CHAR	char
SHORT	short
LONG	long
INT	int
BYTE	unsigned char
WORD	unsigned short
DWORD	unsigned long
UINT	unsigned int

4.2.3 Пример 3 — Теги WinCC (целые)

В большинстве случаев для динамического изменения объектов с помощью *процедур Си* или других функций необходимо использование тегов WinCC. Для этой цели предусмотрены различные функции чтения и записи значений тегов WinCC. Эти функции можно использовать с любыми тегами стандартных типов WinCC. В рассматриваемом примере значения записываются в различные теги WinCC. Содержимое тегов WinCC отображается в полях вывода. Пример сконфигурирован для изображенной ниже кнопки *Button3* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button3

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    CHAR cNumber;           //signed 8 bit value
    SHORT sNumber;         //signed 16 bit value
    LONG lNumber;          //signed 32 bit value

    BOOL bNumber;          //TRUE or FALSE
    BYTE byNumber;         //unsigned 8 bit value
    WORD wNumber;          //unsigned 16 bit value
    DWORD dwNumber;        //unsigned 32 bit value

    cNumber = -128;
    sNumber = -32768;
    lNumber = -2147483648;

    //set wincc tags
    SetTagSByte("S08i_course_tag_1", cNumber);
    SetTagSWord("S16i_course_tag_1", sNumber);
    SetTagSDWord("S32i_course_tag_1", lNumber);

    bNumber = TRUE;
    byNumber = 255;
    wNumber = 65535;
    dwNumber = 4294967295;

    //set wincc tags
    SetTagBit("BINi_course_tag_1", (SHORT)bNumber);
    SetTagByte("U08i_course_tag_1", byNumber);
    SetTagWord("U16i_course_tag_1", wNumber);
    SetTagDWord("U32i_course_tag_1", dwNumber);
}
```

- В первой части производится описание переменных. Типы переменных были выбраны в соответствии с имеющимися типами тегов WinCC.
- В следующей части переменным присваиваются значения. Используемые в примере значения опять в точности совпадают с границами диапазонов значений, представимых переменными соответствующих типов.
- Значения переменных присваиваются различным тегам WinCC с использованием соответствующих функций. Имя функции состоит из строки *SetTag* и описателя типа тега WinCC, к которому применяется данная функция. Для функций записи значений в теги WinCC *SetTag* существуют аналоги для чтения из тегов *GetTag*.

- Если переменная типа *BOOL* (псевдоним *int*) передается в качестве аргумента в функцию *SetTagBit()*, то компилятор выдаст предупреждение, т.к. в *SetTagBit()* предусмотрен аргумент типа *SHORT*. Следовательно, значение переменной *bNumber* преобразуется к типу *SHORT* перед тем, как в примере оно передается в функцию *SetTagBit()*. Этот процесс также называется приведением/преобразованием типов (*typecast*).

Преобразование типов

Перед тем как значение переменной передается в качестве параметра в функцию, или присваивается другой переменной, оно может быть преобразовано к другому типу данных. Тем не менее, тип самой переменной при этом не меняется. Следующий фрагмент кода иллюстрирует приведение переменной типа *float* к типу *int*.

```
iNumber = (int)fNumber;
```

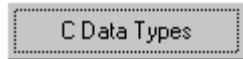
Типы данных тегов WinCC

В приведенной ниже таблице перечислены различные типы тегов WinCC и соответствующие им типы данных, существующие в Си. Переменные этих типов передаются в функции *SetTag* и возвращаются функциями *GetTag*.

Типы тегов WinCC	Типы переменных Си
Signed 8-Bit Value (знаковое 8-битное значение)	char
Signed 16-Bit Value (знаковое 16-битное значение)	short int
Signed 32-Bit Value (знаковое 32-битное значение)	long int
Binary Tag (двоичный тег)	short int
Unsigned 8-Bit Value (беззнаковое 8-битное значение)	BYTE
Unsigned 16-Bit Value (беззнаковое 16-битное значение)	WORD
Unsigned 32-Bit Value (беззнаковое 32-битное значение)	DWORD

4.2.4 Пример 4 — Типы данных Си (числа с плавающей точкой)

В рассматриваемом примере стандартные типы Си используются для представления чисел с плавающей точкой. Пример сконфигурирован для изображенной ниже кнопки *Button4* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button4

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    float fNumber;    //32 bit
    double dNumber;  //64 bit

    fNumber = 1.0000001;
    dNumber = 1.0000001;

    //output in diagnostics window
    printf("\r\nExample 4:\r\n");

    printf("float:\t%2.17f\tsizeof(float):\t%d\r\n"
           "double:\t%2.17f\tsizeof(double):\t%d\r\n",
           fNumber, sizeof(float), dNumber, sizeof(double));
}
```

- В первой части производится описание переменных. Описывается по одной переменной типов *float* и *double*.
- В следующей части переменным присваиваются значения. В данном примере одно и то же значение присваивается обоим переменным.
- Точность переменной типа *float* соответствует примерно седьмому знаку после запятой. Переменные типа *double* могут представлять числа в два раза точнее. Это можно заметить по результатам вывода значений переменных при помощи функции *printf()* в *окне диагностики*. Помимо значения переменной также выводится ее размер. Размер переменной определяется с использованием оператора *sizeof()*. Размер отображается в байтах.

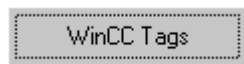
Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 4:
float:    1.00000011920928960    sizeof(float):    4
double:  1.00000010000000010    sizeof(double):   8
```

4.2.5 Пример 5 — Теги WinCC (числа с плавающей точкой)

Помимо целых чисел теги WinCC также могут содержать числа с плавающей точкой. Для этого в WinCC предусмотрены два типа тегов, соответствующие типам Си float и double. Для доступа к таким тегам WinCC с целью чтения или записи существуют соответствующие функции *SetTag* и *GetTag*. В рассматриваемом примере значения записываются в различные теги WinCC. Содержимое тегов WinCC отображается в полях вывода. Пример сконфигурирован для изображенной ниже кнопки *Button5* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button5

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    float fNumber;    //32 bit
    double dNumber;  //64 bit

    fNumber = 1.0000001;
    dNumber = 1.0000001;

    //set wincc tags
    SetTagFloat("F32i_course_tag_1", fNumber);
    SetTagDouble("F64i_course_tag_1", dNumber);
}
```

- В первой части производится описание переменных. Описывается по одной переменной типов *float* и *double*.
- В следующей части переменным присваиваются значения. В данном примере опять одно и то же значение присваивается обоим переменным.
- Значения присваиваются тегам WinCC при помощи соответствующих функций. У использующихся здесь функций записи значений в тег WinCC *SetTag* имеются аналоги для чтения из тегов *GetTag*.

4.2.6 Пример 6 — Статические и внешние переменные

Пример сконфигурирован для изображенной ниже кнопки *Button6* в окне свойств объекта в *Event* → *Mouse* → *Mouse Action* (Событие → Мышь → Процедура мыши).



Статические переменные

Переменная Си становится действительна только после ее описания. Она становится недействительной после завершения исполнения функции. При последующем вызове этой же функции переменная создается вновь. Тем не менее, ключевое слово *static*, предваряющее имя переменной, позволяет предотвратить уничтожение переменной после завершения функции. Таким образом, эта переменная сохранит свое значение до последующего вызова функции. Однако, в случае *процедур Си*, это имеет место только до тех пор, пока отображается текущая экранная форма WinCC. Статические переменные становятся недействительны, как только экранная форма перестает отображаться. При очередном открытии экранной формы статические переменные будут созданы вновь во время первого исполнения процедуры Си.

Внешние переменные

Переменная Си может использоваться только внутри той функции, в которой она была описана. Тем не менее, если переменная описывается вне функции, она становится глобальной (внешней) переменной. Такая переменная может быть объявлена в любой другой функции при помощи ключевого слова *extern*. После этого к ней можно получить доступ.

Функция проекта CreateExternalTags()

```
int ext_iNumber = 0;
void CreateExternalTags()
{
    //nothing to do
}
```

- Функция *CreateExternalTags()* служит исключительно для описания и инициализации внешней переменной типа *int*. В начале исполнения проекта функция вызывается однократно (в *Events (События)* → *Miscellaneous (Разное)* → *Open Picture (Открытие кадра)* стартовой экранной формы *cc_0_startpicture_00.PDL*). Начиная с этого момента, переменная *ext_iNumber* описана и может использоваться в любой *процедуре Си* или любой другой функции.

Процедура Си, связанная с кнопкой Button6

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //declare external tag
    extern int ext_iNumber;

    //define static tag
    static int stat_iNumber = 0;

    //output in diagnostics window
    printf("\r\nExample 6:\r\n"
        "mouseclicks since project was started: %d\r\n"
        "mouseclicks since picture was opened: %d\r\n",
        ++ext_iNumber, ++stat_iNumber);
}
```

- В первой части объявляется внешняя переменная *ext_iNumber*, для того чтобы иметь возможность использовать ее в *процедуре Си*.
- В следующей части описывается и инициализируется статическая переменная *stat_iNumber*. Это происходит во время первого исполнения процедуры Си после выбора данного кадра WinCC. В дальнейшем значение переменной между последующими запусками процедуры Си будет сохраняться. Если кадр будет закрыт и потом вновь открыт, переменная создастся повторно.
- Значения переменных увеличиваются на единицу при помощи оператора инкремента ++ и выводятся в *диагностическом окне* с использованием функции *printf()*. Таким образом, в переменной *ext_iNumber* будет содержаться число нажатий на кнопку со времени запуска проекта, а в переменной *stat_iNumber* — с момента открытия экранной формы. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 6:  
mouseclicks since project was started: 1  
mouseclicks since picture was opened: 1
```

4.3 Операторы и математические функции в Си

В проекте WinCC Project_C_Course примеры, относящиеся к работе с переменными, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называющейся `ss_9_example_01.PDL`.



Operators

Операторы

В программе операторы используются для работы с переменными и константами. В результате применения операторов к переменным и константам формируются новые значения переменных.

Операторы можно разделить на несколько различных категорий, таких, как математические операторы, битовые операторы и операторы присваивания.

Математические операторы

Оператор	Описание
+ (унарный)	Знак плюс (фактически не оказывает влияния)
- (унарный)	Знак минус
+ (бинарный)	Сложение
- (бинарный)	Вычитание
*	Умножение
/	Деление
%	Остаток от деления
++	Инкремент
--	Декремент

Битовые операторы

Данные операторы позволяют устанавливать, сбрасывать или опрашивать значения отдельных битов переменных.

Оператор	Описание
&	Побитное И (AND)
	Побитное ИЛИ (OR)
^	Побитное исключающее ИЛИ (XOR)
~	Побитное отрицание
<<	Сдвиг битов влево
>>	Сдвиг битов вправо

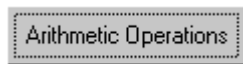
Логические операторы

Все логические операторы используют одно и то же правило: 0 соответствует FALSE (ЛОЖЬ), все остальные значения — TRUE (ИСТИНА). Результатом применения этих операторов являются значения 0 (FALSE) или 1 (TRUE).

Оператор	Описание
>	Строго больше
>=	Больше или равно
==	Равно
!=	Не равно
<=	Меньше или равно
<	Строго меньше
?&	Логическое И (AND)
	Логическое ИЛИ (OR)
!	Логическое отрицание

4.3.1 Пример 1 — Основные математические процедуры

В данном примере демонстрируется использование основных математических операций. Пример сконфигурирован для изображенной ниже кнопки *Button1* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button1

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    float fValue1 = 123.6;
    float fValue2 = 23.4;

    float fResAdd;
    float fResSub;
    float fResMul;
    float fResDiv;

    fResAdd = fValue1 + fValue2; //add
    fResSub = fValue1 - fValue2; //subtract
    fResMul = fValue1 * fValue2; //multiply
    fResDiv = fValue1 / fValue2; //divide

    //output in diagnostics window
    printf("\r\nExample 1\r\n");
    printf("%.1f + %.1f = %.1f\r\n", fValue1, fValue2, fResAdd);
    printf("%.1f - %.1f = %.1f\r\n", fValue1, fValue2, fResSub);
    printf("%.1f * %.1f = %.1f\r\n", fValue1, fValue2, fResMul);
    printf("%.1f / %.1f = %.1f\r\n", fValue1, fValue2, fResDiv);
}
```

- В первой части описываются и инициализируются две переменные типа *float*. Математические операторы будут применяться к этим двум переменным.
- В следующей части описываются еще четыре переменные типа *float*. Эти переменные используются для хранения результатов математических операций.
- В следующей части математические операторы используются для сложения, вычитания, умножения и деления.
- Результаты вычислений выводятся в *диагностическом окне* при помощи функции *printf()*. Результаты работы приведены в следующем разделе.

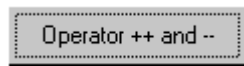
Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 1
123.6 + 23.4 = 147.0
123.6 - 23.4 = 100.2
123.6 * 23.4 = 2892.2
123.6 / 23.4 = 5.3
```

4.3.2 Пример 2 — Операторы инкремента и декремента

В данном примере демонстрируется использование операторов инкремента и декремента. Пример сконфигурирован для изображенной ниже кнопки *Button2* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Префиксный и постфиксный варианты

Операторы инкремента и декремента существуют в префиксном и постфиксном вариантах. Оба варианта осуществляют одни и те же действия: увеличивают или уменьшают значение переменной, к которой они применяются, на единицу. Различие заключается в возвращаемом значении. Префиксный вариант увеличивает или уменьшает значение переменной и возвращает полученное новое значение. Постфиксный вариант возвращает исходное значение переменной и лишь затем увеличивает или уменьшает его.

```
iValue = ++iCount; //prefix
iValue = iCount++; //postfix
```

Процедура Си, связанная с кнопкой Button2

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    static int stat_iPrefix = 0;
    static int stat_iPostfix = 0;

    printf("\r\nExample 2\r\n");

    //execute operators
    printf("Prefix operator on first tag: %d\r\n", ++stat_iPrefix);
    printf("Postfix operator on second tag: %d\r\n", stat_iPostfix++);

    //check values
    printf("Value of first tag after execution: %d\r\n", stat_iPrefix);
    printf("Value of second tag after execution: %d\r\n", stat_iPostfix);
}
}
```

- В первой части описываются и инициализируются две переменные типа *int*.
- В следующей части к этим переменным применяются префиксный и постфиксный варианты оператора инкремента. Возвращаемые операторами значения выводятся в *окне диагностики* при помощи функции *printf()*. Затем вывод значений переменных осуществляется еще раз. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод приведенного ниже текста. Несложно заметить, что префиксный вариант оператора инкремента возвращает инкрементированное значение переменной, в то время как постфиксный вариант возвращает исходное значение переменной. Тем не менее, в обоих случаях переменная в итоге инкрементируется.

Example 2

Prefix operator on first tag: 1

Postfix operator on second tag: 0

Value of first tag after execution: 1

Value of second tag after execution: 1

4.3.3 Пример 3 — Битовые операции

В данном примере демонстрируется использование битовых операторов. Пример сконфигурирован для изображенной ниже кнопки *Button3* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Описание

В данном примере битовые операторы применяются к содержимому двух беззнаковых 16-битных тегов WinCC (*unsigned 16-Bit values*). Результат преобразований записывается в другой тег WinCC того же типа. Применяемый оператор управляется и одновременно отображается кнопкой *Button6*. Предусмотрены следующие битовые операции: *AND*, *OR*, *NAND*, *NOR* и *EXOR* (И, ИЛИ, НЕ И, НЕ ИЛИ, исключающее ИЛИ). Каждому из вариантов выбора битовой операции соответствует определенное численное значение, определяющееся отдельным тегом WinCC (*unsigned 8-Bit value*).

Процедура Си, связанная с кнопкой Button3

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    BYTE byOperation;

    DWORD dwValue1;
    DWORD dwValue2;
    DWORD dwResult;

    //read tag values
    dwValue1 = GetTagWord("U16i_course_op_1");
    dwValue2 = GetTagWord("U16i_course_op_2");

    //get desired operation
    byOperation = GetTagByte("U08i_course_op_1");

    switch (byOperation)
    {
        //AND
        case 0: dwResult = dwValue1 & dwValue2;
               break;
        //OR
        case 1: dwResult = dwValue1 | dwValue2;
               break;
        //NAND
        case 2: dwResult = ~(dwValue1 & dwValue2);
               break;
        //NOR
        case 3: dwResult = ~(dwValue1 | dwValue2);
               break;
        //EXOR
        case 4: dwResult = dwValue1 ^ dwValue2;
               break;
        //???
        default: return;
    }

    //write result
    SetTagWord("U16i_course_op_3", (WORD)dwResult);
}

```

- В первой части описываются одна переменная типа *BYTE* и три переменные типа *DWORD*. Эти переменные используются для временного хранения значений тегов WinCC.
- В следующей части считываются значения двух тегов WinCC и записываются в переменные *dwValue1* и *dwValue2*. Кроме того, значение тега WinCC, определяющее тип битовой операции, помещается в переменную *byOperation*.
- В следующей части в зависимости от значения переменной *byOperation* к переменным *dwValue1* и *dwValue2* применяется одна из битовых операций. Результат вычисления записывается в переменную *dwResult*. Выбор выполняемой битовой операции осуществляется при помощи конструкции *switch-case*. Эта конструкция более подробно описывается в главе *Циклы*.
- В следующей части результат битовой операции, находящийся в переменной *dwResult*, записывается в соответствующий тег WinCC.

4.3.4 Пример 4 — Перестановка старшего и младшего байтов

В данном примере операторы побитового сдвига используются для организации перестановки младшего и старшего байтов тега WinCC (*unsigned 16-Bit value*). При этом младший и старший байты меняются местами. Пример сконфигурирован для изображенной ниже кнопки *Button4* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button4

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    DWORD dwValue;

    DWORD dwtempValue1;
    DWORD dwtempValue2;

    //read tag value
    dwValue = GetTagWord("U16i_course_op_3");

    //rotate bytes
    dwtempValue1 = dwValue<<8;
    dwtempValue2 = dwValue>>8;

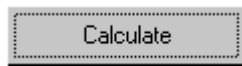
    dwValue = dwtempValue1 | dwtempValue2;

    //write result
    SetTagWord("U16i_course_op_3", (WORD)dwValue);
}
```

- В первой части описывается переменная типа *DWORD*. Эта переменная используется для временного хранения значения тега WinCC. Кроме того, описываются две вспомогательные переменные типа *DWORD*.
- В следующей части значение тега WinCC, над которым производится преобразование, записывается в переменную *dwValue*.
- В следующей части биты переменной *dwValue* сдвигаются на восемь позиций влево (один байт) и результат записывается в переменную *dwtempValue1*. После этого биты переменной *dwValue* сдвигаются на восемь позиций вправо, и результат записывается в переменную *dwtempValue2*. Затем к вычисленным значениям применяется битовый оператор ИЛИ, а результат записывается в переменную *dwValue*.
- В следующей части перевернутое значение, хранящееся в переменной *dwValue*, записывается в соответствующий тег WinCC.

4.3.5 Пример 5 — Математические функции

В данном примере демонстрируется использование стандартных математических функций, использующихся в Си. Пример сконфигурирован для изображенной ниже кнопки *Button5* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button5

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    double dValue = 123.6;
    int iValue = -24;

    double dResPow;
    double dResSqrt;
    int iResAbs;
    int iResRand;

    dResPow = pow(dValue,3); //power of 3
    dResSqrt= sqrt(dValue); //square root
    iResAbs = abs(iValue); //absolute
    iResRand= rand(); //random

    //output in diagnostics window
    printf("\r\nExample 5\r\n");
    printf("%.1f raised to the power of 3 = %.1f\r\n",dValue,dResPow);
    printf("Square root of %.1f\t = %.1f\r\n",dValue,dResSqrt);
    printf("Absolute value of %d\t = %d\r\n",iValue,iResAbs);
    printf("A pseudorandom number\t = %d\r\n",iResRand);
}
```

- В первой части производится описание переменных.
- Вначале вызывается функция *pow()*. Данной функции передаются два параметра. В рассматриваемом примере функция возвращает значение переменной *dValue*, возведенное в третью степень.
- Затем вызывается функция *sqrt()*. Эта функция возвращает квадратный корень передаваемого ей значения.
- Затем вызывается функция *abs()*. Эта функция возвращает абсолютную величину передаваемого ей значения.
- Затем вызывается функция *rand()*. У этой функции нет аргументов. Она возвращает произвольное (псевдослучайное) число.
- Результаты вычислений выводятся в *окне диагностики* при помощи функции *printf()*. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

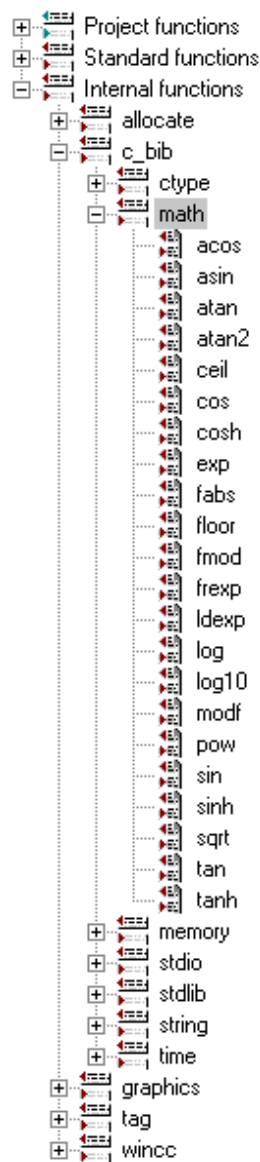
Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```

Example 5
123.6 raised to the power of 3 = 1888232.3
Square root of 123.6           = 11.1
Absolute value of -24          = 24
A pseudorandom number         = 41
  
```

Дополнительные математические функции

Перечень математических функций приводится в окне выбора функций в разделе *Internal Functions* → *c_bib* → *math*. На приведенном ниже кадре видны все доступные пользователю математические функции (выделено серым цветом).



4.4 Указатели

В проекте WinCC Project_C_Course примеры, относящиеся к работе с указателями, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называющейся `ss_9_example_02.PDL`.



Pointers

Работа с указателями

Указатели являются важной частью языка Си. Указатель — это переменная, содержащая, как правило, адрес другой переменной.

Указатели описываются точно также как обычные переменные. К названию типа переменных, на которые будет ссылаться указатель, добавляется унарный символ `*`. Этот символ не следует пугать с бинарным оператором `*`, используемым для умножения. В приведенном ниже фрагменте кода описывается указатель на переменную типа `int`.

```
int* piValue;
```

Содержимое указателя не определено. Он ссылается на некоторую случайную, недействительную переменную типа `int`. Для большей ясности при описании указателя его следует инициализировать значением `NULL`. Это позволит проверить корректность указателя перед его использованием.

```
int* piValue = NULL;
```

Чтобы указатель ссылался на переменную типа `int`, ему следует присвоить адрес этой переменной. Это делается при помощи специального унарного оператора, так называемого оператора адреса. Этот оператор возвращает адрес переменной вместо ее значения. В приведенном ниже фрагменте кода адрес переменной типа `int` присваивается указателю.

```
piValue = &iValue;
```

Доступ к значению переменной, на которую ссылается указатель, осуществляется при помощи унарного оператора `*`, который также иногда называют оператором разыменования. В приведенном ниже фрагменте кода переменной типа `int` присваивается значение переменной, на которую ссылается указатель.

```
iValue = *piValue;
```

Работа с массивами

Указатели и массивы тесно связаны. В приведенном ниже фрагменте кода описывается массив из 5 переменных типа `int`.

```
int iVector[5];
```

Доступ к отдельным элементам массива осуществляется по его индексу. В приведенном ниже фрагменте кода производится доступ к содержимому последнего элемента массива. Это делается при помощи оператора индекса [].

```
iValue = iVector[4];
```

Имя массива также может использоваться как указатель, ссылающийся на первый элемент этого массива. Доступ к определенному элементу массива можно также производить, смещая данный указатель на нужное число элементов. Как показано в приведенном ниже примере, это делается прибавлением целого числа к указателю. Содержимое переменной, на которую ссылается результирующий указатель, может быть получено с использованием оператора разыменования *. Как и в предыдущем примере, доступ осуществляется к последнему элементу массива.

```
iValue = *(iVector+4);
```

Символьные строки

В Си символьная строка может быть описана как массив символов, или как указатель, ссылающийся на символ. К концу символьной строки Си добавляет нулевой символ (null). Он служит для обозначения конца символьной строки. В приведенном ниже фрагменте кода производится описание символьных переменных с использованием обоих вариантов.

```
char* lpszString = "String1";
char szString[10] = "String2";
```

Ниже показано содержимое обеих символьных переменных. В первом случае под переменную отводится область памяти, размер которой в точности соответствует строке, инициализирующей эту переменную. Во втором случае переменной выделяется область памяти, размер которой определяется количеством элементов описываемого массива.

S	t	r	i	n	g	1	\0
---	---	---	---	---	---	---	----

S	t	r	i	n	g	2	\0	?	?
---	---	---	---	---	---	---	----	---	---

4.4.1 Пример 1 — Указатели

В данном примере демонстрируется использование основных операций с указателями. Пример сконфигурирован для изображенной ниже кнопки *Button1* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button1

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    int iValue1 = 126;
    int iValue2 = 23;

    //declare and initialize pointer
    int* piValue = NULL;

    printf("\r\nExample 1\r\n");
    printf("Address: %x\tValue: undefined\r\n", piValue);

    //point at iValue1
    piValue = &iValue1;
    printf("Address: %x\tValue: %d\r\n", piValue, *piValue);

    //point at iValue2
    piValue = &iValue2;
    printf("Address: %x\tValue: %d\r\n", piValue, *piValue);
}
```

- В первой части описываются и инициализируются две переменные типа *int*.
- Затем описывается указатель, ссылающийся на переменную типа *int*, и инициализируется значением *NULL*.
- После этого адрес, на который ссылается указатель, распечатывается при помощи функции *printf()*. Содержимое ячейки, на которую при этом ссылается указатель, не определено. Попытка использования оператора *** для доступа к содержимому привела бы к общей ошибке доступа.
- Далее указателю присваивается адрес переменной *iValue1*. Содержимый в нем адрес и содержимое, находящееся по этому адресу, опять распечатываются при помощи функции *printf()*.
- Затем указателю присваивается адрес переменной *iValue2*, и результаты распечатываются вновь. Результаты работы данной программы приводятся в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

Example 1
Address: 0 **Value: undefined**
Address: 2b4f9f8 **Value: 126**
Address: 2b4f9fc **Value: 23**

4.4.2 Пример 2 — Массивы

В данном примере демонстрируется использование основных операций с массивами. Пример сконфигурирован для изображенной ниже кнопки *Button2* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button2

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //declare and initialize int vector
    int iValue[5] = { 10, 20, 30, 40, 50 };

    int iIndex;

    printf("\r\nExample 2\r\n");

    //access vector elements
    for (iIndex = 0; iIndex<5; iIndex++)
    {
        printf("Index: %d\t Value: %d\r\n",iIndex,iValue[iIndex]);
    }
}
```

- В первой части описывается массив из 5 переменных типа *int*. Элементы массива инициализируются численными значениями непосредственно в момент его описания.
- Затем описывается переменная–счетчик *iIndex* типа *int*.
- Элементы массива распечатываются при помощи функции *printf()*. Доступ к отдельным элементам массива производится в цикле *for* с использованием оператора индекса []. Работа с массивами описывается в следующей главе *Циклы*. Результаты работы программы приведены в следующем разделе.

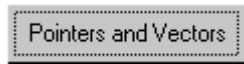
Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 2
Index: 0 Value: 10
Index: 1 Value: 20
Index: 2 Value: 30
Index: 3 Value: 40
Index: 4 Value: 50
```

4.4.3 Пример 3 — Указатели и массивы

В данном примере поясняется связь между массивами и указателями. Пример сконфигурирован для изображенной ниже кнопки *Button3* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button3

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    int iValue[] = { 10, 20, 30, 40, 50 };
    int iIndex;
    int* piElement = NULL;
    printf("\r\nExample 3\r\n");
    //////////////////////////////////////
    //access via seperat pointer
    //point to the first element
    piElement = &iValue[0];
    printf("Startaddress: %x\r\n",piElement);
    for (iIndex = 0; iIndex<5; iIndex++)
    {
        printf("Index: %d\t Value: %d\r\n",iIndex,*(piElement+iIndex));
    }
    printf("\r\n");
    //////////////////////////////////////
    //access without seperate pointer
    printf("Startaddress: %x\r\n",iValue);
    for (iIndex = 0; iIndex<5; iIndex++)
    {
        printf("Index: %d\t Value: %d\r\n",iIndex,*(iValue+iIndex));
    }
}
```

- В первой части описывается массив из 5 переменных типа *int*. Элементы массива инициализируются численными значениями непосредственно в момент его описания. В данном случае при описании массива его размер можно не указывать.
- Затем описывается переменная–счетчик *iIndex* типа *int*.
- После этого описывается указатель *piElement*, ссылающийся на переменную типа *int*, и инициализируется значением *NULL*.
- Далее указателю *piElement* присваивается адрес первого элемента массива. Этот адрес распечатывается при помощи функции *printf()*.
- Затем доступ к отдельным элементам массива производится с использованием указателя *piElement*. Доступ осуществляется в цикле *for*, с перемещением указателя на очередной элемент, и использованием оператора разыменования ***.

- После этого еще раз производится доступ к отдельным элементам массива. Только на этот раз само имя массива используется в качестве указателя. Результаты работы программы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 3
Startaddress: 2b4f9e8
Index: 0 Value: 10
Index: 1 Value: 20
Index: 2 Value: 30
Index: 3 Value: 40
Index: 4 Value: 50
```

```
Startaddress: 2b4f9e8
Index: 0 Value: 10
Index: 1 Value: 20
Index: 2 Value: 30
Index: 3 Value: 40
```


4.4.4 Пример 4 — Строки

В данном примере поясняются принципы работы со строковыми переменными. Пример сконфигурирован для изображенной ниже кнопки *Button4* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button4

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //declare and initialize string
    char szText[13] = "example text";

    int i;

    printf("\r\nExample 4\r\nCharacters:\r\n");

    //access single characters
    for (i=0; i<12; i++)
    {
        printf("[%c].",szText[i]);
    }

    printf("\r\n");

    //access hole string
    printf("String:\r\n%s\r\n",szText);
}
```

- В первой части описывается символьная строка (массив, состоящий из 13 символов). Размер символьной строки на единицу больше длины инициализирующей строки для того, чтобы зарезервировать место под завершающий null-символ.
- Затем описывается переменная *i* типа *int*.
- После этого отдельные элементы массива распечатываются при помощи функции *printf()*. Доступ к символам осуществляется в цикле *for* с использованием оператора индекса [].
- Далее вся символьная строка распечатывается функцией *printf()*. Результаты работы программы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 4
Characters:
[e].[x].[a].[m].[p].[l].[e].[ ].[t].[e].[x].[t].
String:
example text
```

4.4.5 Пример 5 — Текстовые теги WinCC

В данном примере поясняется связь между строковыми переменными Си и текстовыми тегами WinCC. Пример сконфигурирован для изображенной ниже кнопки *Button5* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button5

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //declare and initialize pointer to string
    char* pszText = NULL;

    //get wincc tag value
    pszText = GetTagChar("T08i_course_point_1");

    printf("\r\nExample 5\r\n");

    //access string
    printf("String: %s\r\nStringlength: %d\r\nStartaddress: %x\r\n",
        pszText, strlen(pszText), pszText);
}
```

- В первой части описывается символьная строка (указатель, ссылающийся на первый символ). Эта строка инициализируется значением *NULL*.
- Затем содержимое текстового тега WinCC считывается при помощи функции *GetTagChar()*. Функция выделяет область памяти, необходимую для размещения символьной строки, и возвращает ее начальный адрес.
- После этого вся символьная строка распечатывается функцией *printf()*. Кроме того, длина этой символьной строки определяется при помощи функции *strlen()* и выводится на печать вместе с начальным адресом строки. Результаты работы программы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 5
String: This is an example text!
Stringlength: 24
Startaddress: 1682828
```

4.5 Циклы и условные выражения

В проекте WinCC Project_C_Course примеры, относящиеся к работе с циклами, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называемой `ss_9_example_04.PDL`.



Loops

Циклы

Циклы используются для многократного исполнения участков программы. Цикл выполняется пока искомое условие истинно.

Всего существует два типа циклов: циклы с пред проверкой и циклы с пост проверкой. Циклы с пред проверкой определяют, следует ли исполнять тело цикла, в самом начале. Циклы с пост–проверкой после выполнения тела цикла определяют, следует ли его повторять. Таким образом, циклы с пост–проверкой выполняются как минимум один раз.

Выделяют следующие типы циклов:

while

Пример цикла *while* приведен ниже. Цикл выполняется пока истинно проверяемое условие. В данном примере цикл выполняется пока значение переменной *i* меньше 5.

```
int i = 0;
while (i < 5)
{
    //do something
    ++i;
}
```

do – while

Пример цикла *do-while* приведен ниже. Цикл выполняется как минимум один раз, после чего повторяется до тех пор, пока проверяемое условие истинно. В данном примере цикл выполняется пока значение переменной *i* меньше 5.

```
int i = 0;
do
{
    //do something
    ++i;
}
while (i < 5);
```

for

Пример цикла *for* приведен ниже. Цикл выполняется пока истинно проверяемое условие. Инициализация переменных цикла, а также их изменение можно задать в самом цикле, а не в его теле.

```
int i = 0;
for (i=0, i<5, i++)
{
    //do something
}
```

Условные выражения

У циклов их тело выполняется пока истинно проверяемое условие. В условных выражениях заданный фрагмент кода выполняется ровно один раз, и только в том случае, если проверяемое условие истинно.

Выделяют следующие типы условных выражений:

if – else

Если проверяемое условие истинно, то выполняется ветвь *if*. Если условие не выполняется, то будет исполнена альтернативная ветвь *else*. Ветвь *else* также может вовсе отсутствовать, если нет необходимости в выполнении альтернативной обработки.

```
if (i < 5)
{
    //do something
}
else
{
    //do something else
}
```

switch – case

В данном случае переменная проверяется на совпадение. *Switch* задает проверяемую переменную. Производится проверка, какая из ветвей *case* соответствует значению этой переменной. Если такая ветвь найдена, она выполняется. Ограничения на количество ветвей отсутствуют. Каждая ветвь должна заканчиваться оператором *break*. При необходимости можно также задать ветвь по умолчанию *default*. Она будет исполняться в том случае, если значение проверяемой переменной не совпадает ни с одной из ветвей *case*.

```
switch (i)
{
    case 0: //do something
        break;
    case 1: //do something
        break;
    default://do something default
        break;
}
```

4.5.1 Пример 1 — Цикл while

В данном примере поясняются принципы применения цикла *while*. Пример сконфигурирован для изображенной ниже кнопки *Button1* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button1

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //loop count
    int iCount = 0;

    printf("\r\nExample 1\r\n");

    //while loop
    while (iCount < 5)
    {
        //do something
        printf("Executed loop: iCount = %d\r\n", iCount);

        ++iCount;
    }

    printf("Exit loop: iCount = %d\r\n", iCount);
}
}
```

- В первой части описывается и инициализируется переменная–счетчик *iCount* типа *int*.
- Далее задается цикл *while*. Этот цикл будет исполняться, пока содержимое переменной–счетчика *iCount* меньше 5. При каждой итерации цикла функцией *printf()* производится вывод на печать. В конце цикла переменная–счетчик *iCount* наращивается на единицу. Результаты работы программы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 1
Executed loop: iCount = 0
Executed loop: iCount = 1
Executed loop: iCount = 2
Executed loop: iCount = 3
Executed loop: iCount = 4
Exit loop: iCount = 5
```

4.5.2 Пример 2 — Цикл do – while

В данном примере поясняются принципы применения цикла *do – while*. Пример сконфигурирован для изображенной ниже кнопки *Button2* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button2

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //loop count
    int iCount = 0;

    printf("\r\nExample 2\r\n");

    //do-while loop
    do
    {
        //do something
        printf("Executed loop: iCount = %d\r\n", iCount);

        ++iCount;
    }
    while (iCount < 5);

    printf("Exit loop: iCount = %d\r\n", iCount);
}
```

- В первой части описывается и инициализируется переменная–счетчик *iCount* типа *int*.
- Далее задается цикл *do – while*. Этот цикл будет исполняться, пока содержимое переменной–счетчика *iCount* меньше 5. Однако цикл исполняется как минимум один раз, т.к. искомое условие проверяется только в конце его выполнения. При каждом исполнении цикла функцией *printf()* производится вывод на печать. В конце цикла переменная–счетчик *iCount* наращивается на единицу. Результаты работы программы приведены в следующем разделе.

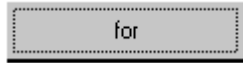
Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 2
Executed loop: iCount = 0
Executed loop: iCount = 1
Executed loop: iCount = 2
Executed loop: iCount = 3
Executed loop: iCount = 4
Exit loop: iCount = 5
```

4.5.3 Пример 3 — Цикл for

В данном примере поясняются принципы применения цикла *for*. Пример сконфигурирован для изображенной ниже кнопки *Button3* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button3

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //loop count
    int iCount = 0;

    printf("\r\nExample 3\r\n");

    //for loop
    for (iCount=0; iCount<5; iCount++)
    {
        //do something
        printf("Executed loop: iCount = %d\r\n",iCount);
    }

    printf("Exit loop: iCount = %d\r\n",iCount);
}
```

- В первой части описывается и инициализируется переменная-счетчик *iCount* типа *int*.
- Далее задается цикл *for*. Этот цикл будет исполняться пока содержимое переменной-счетчика *iCount* меньше 5. Инициализация переменной-счетчика осуществляется непосредственно в цикле, равно как и ее приращение. При каждом исполнении цикла функцией *printf()* производится вывод на печать. Результаты работы программы приведены в следующем разделе.

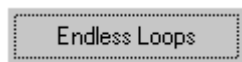
Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Example 3
Executed loop: iCount = 0
Executed loop: iCount = 1
Executed loop: iCount = 2
Executed loop: iCount = 3
Executed loop: iCount = 4
Exit loop: iCount = 5
```

4.5.4 Пример 4 — Бесконечные циклы

В данном примере объясняется, что такое бесконечные циклы. В большинстве случаев такие циклы создаются непреднамеренно, в результате программных ошибок, приводящих к тому, что условие выполнения цикла всегда истинно. Тем не менее, иногда они применяются сознательно. В этом случае завершение цикла необходимо организовать другим способом, а именно, с использованием оператора *break*. Пример сконфигурирован для изображенной ниже кнопки *Button4* в окне свойств объекта в *Event (Событие) → Mouse (Мышь) → Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button4

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //max loop executions
    #define MAX_COUNT 1000000

    //loop count
    int iCount = 0;

    int iProgressBar = 1;
    char szProgressText[5];

    //endless loop
    //another possible loop while(TRUE) {...}
    for (;;)
    {
        if (iCount > MAX_COUNT)
        {
            break;
        }

        ++iCount;

        if (iCount-(iProgressBar*MAX_COUNT/100) != 0)
        {
            continue;
        }

        //set value of progress bar
        SetWidth(lpszPictureName, "ProgressBar", (int)(iProgressBar*2.7));

        //set progress text
        sprintf(szProgressText, "%d%%", iProgressBar);
        SetText(lpszPictureName, "ProgressText", szProgressText);

        ++iProgressBar;
    }
}
```

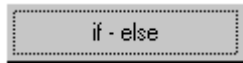
- В первой части описывается символьная константа *MAX_COUNT*. Эта постоянная задает максимальное количество итераций последующего бесконечного цикла.
- В следующей части описывается и инициализируется переменная–счетчик *iCount* типа *int*.
- Текущее количество итераций цикла должно отображаться при помощи индикатора выполнения. Данный индикатор формируется при помощи столбца,

длина которого задается переменной *iProgressBar*, и статического текста, определяемого строковой переменной *szProgressText*.

- Далее организуется бесконечный цикл. Этот цикл также можно было оформить, используя выражение *while (TRUE)*.
- В цикле производится проверка переменной-счетчика *iCount*. Если значение этой переменной превышает *MAX_COUNT*, выполнение цикла прерывается оператором *break*.
- Выполняется инкрементация переменной-счетчика *iCount*.
- Индикатор выполнения отображает процент завершенных итераций. По достижении нового процента выполнения происходит обновление индикатора. Если процент выполнения остался прежним, сразу же происходит переход к новой итерации при помощи оператора *continue*, а оставшийся фрагмент цикла пропускается.
- Параметры индикатора выполнения устанавливаются путем задания ширины объекта *ProgressBar* функцией *SetWidth()* и текста объекта *ProgressText* функцией *SetText()*. Используемый текст формируется при помощи функции *sprintf()*. Эта функция работает аналогично *printf()*. Однако, в отличие от последней, текст не выводится в *окне диагностики глобальных сценариев*, а записывается в строковую переменную. Данная строка передается функции первым аргументом.

4.5.5 Пример 5 — Выражение if-else

В данном примере поясняются принципы применения конструкции *if-else*. Пример сконфигурирован для изображенной ниже кнопки *Button5* в окне свойств объекта в *Event (Событие) → Mouse (Мышь) → Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button5

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    BYTE byValue;

    //get value to check
    byValue = GetTagByte("U08i_course_loop_1");

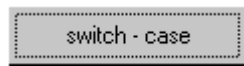
    printf("\r\nExample 5\r\n");

    if (byValue < 5)
    {
        //do something
        printf("byValue < 5\r\n");
    }
    else
    {
        //do something
        printf("byValue >= 5\r\n");
    }
}
```

- В первой части описывается переменная *byValue* типа *BYTE*. В этой переменной хранится значение тега WinCC.
- В следующей части значение тега WinCC считывается при помощи функции *GetTagByte()*, и записывается в переменную *byValue*.
- Далее организуется конструкция *if-else*. При помощи функции *printf()* осуществляется вывод сообщения, зависящего от значения переменной *byValue*.

4.5.6 Пример 6 — Выражение switch–case

В данном примере поясняются принципы применения конструкции *switch–case*. Пример сконфигурирован для изображенной ниже кнопки *Button6* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button6

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    BYTE byValue;

    //get value to check
    byValue = GetTagByte("U08i_course_loop_1");

    printf("\r\nExample 6\r\n");

    switch (byValue)
    {
        case 0:    //do something
                  printf("byValue = 0\r\n");
                  break;
        case 1:    //do something
                  printf("byValue = 1\r\n");
                  break;
        case 2:
        case 3:
        case 4:    //do something
                  printf("byValue = 2,3 or 4\r\n");
                  break;
        default:   //do something
                  printf("byValue != 0,1,2,3 and 4\r\n");
                  break;
    }
}
```

- В первой части описывается переменная *byValue* типа *BYTE*. В этой переменной хранится значение тега WinCC.
- В следующей части значение тега WinCC считывается при помощи функции *GetTagByte()* и записывается в переменную *byValue*.
- Далее организуется конструкция *switch–case*. При помощи функции *printf()* осуществляется вывод сообщения, зависящего от значения переменной *byValue*. Чтобы выполнить одинаковый фрагмент кода для нескольких разных значений проверяемой переменной, соответствующие ветви *case* следует объединить. Фрагмент кода, подлежащий исполнению, необходимо поместить в последней ветви *case*.

4.6 Функции

В проекте WinCC Project_C_Course примеры, относящиеся к функциям, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называющейся ss_9_example_05.PDL.



Functions

Функции

Функции предоставляют возможность лучше структурировать текст программы. Вместо того чтобы раз за разом программировать часто повторяющиеся фрагменты кода, их можно оформить в виде функции. В результате этого появляется возможность редактировать код программы в одном месте, что упрощает отладку и модификацию приложения.

Функция в WinCC может быть создана либо как функция проекта, либо как стандартная функция.

Передаваемые параметры

Функции могут быть переданы параметры (аргументы), определяющие ход ее исполнения. Существует несколько способов передачи параметров.

- Передача постоянного значения.
- Передача переменной по значению. При этом функции передается только значение, она не имеет доступа к самой переменной.
- Передача переменной по указателю. Этот способ дает функции возможность доступа к переменной, на которую ссылается указатель. Массивы и структуры могут передаваться функции только через указатели.

Возвращаемое значение

Функция может просто производить требуемые процедуры без возврата какого-либо значения. В этом случае возвращаемое функцией значение имеет тип *void*. Но если функция, например, производит какие-либо вычисления, их результат может быть передан в основную программу посредством значения возврата. Так можно передавать и значения и адреса переменных.

Другой способ заключается в записи результатов работы в область памяти, указанную в качестве аргумента при вызове функции. Массивы и структуры могут передаваться из функции только таким образом.

4.6.1 Пример 1 — Передача параметров по значению

В данном примере создается простая функция для вычисления среднего арифметического трех чисел. Параметры передаются в функцию по значению, результат также возвращается по значению. Пример сконфигурирован для изображенной ниже кнопки *Button1* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Функция проекта MeanValue()

```
double MeanValue(double dValue1, double dValue2, double dValue3)
{
    double dMeanValue;
    dMeanValue = (dValue1+dValue2+dValue3)/3;
    return dMeanValue;
}
```

- В заголовке в качестве имени функции указывается *MeanValue()*. В качестве аргументов ей передаются три переменные типа *double*. Тип возвращаемого значения — также *double*.
- Далее описывается переменная типа *double*, в которой будет храниться возвращаемое значение. Это значение вычисляется путем сложения значений трех переданных аргументов и последующего деления получившейся суммы на три.
- При помощи команды *return*, результат возвращается вызывающей программе.

Процедура Си, связанная с кнопкой Button1

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    double dValue1 = 126.2;
    double dValue2 = 23.9;
    double dValue3 = 45.7;

    double dMeanValue;

    //calculate mean value
    dMeanValue = MeanValue(dValue1, dValue2, dValue3);

    //output into diagnostics window
    printf("\r\nExample 1\r\n");

    printf("The mean value of %.1f, %.1f and %.1f = %.1f\r\n",
           dValue1, dValue2, dValue3, dMeanValue);
}
```

- В первой части описываются и инициализируются три переменные типа *double*. Необходимо вычислить среднее арифметическое значений этих переменных. Создается дополнительная переменная типа *double*, в которой будет содержаться результат вычислений.
- Используя ранее созданную функцию *MeanValue()*, производится вычисление среднего.
- Результат вычислений выводится на печать при помощи функции *printf()*. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

Example 1

The mean value of 126.2, 23.9 and 45.7 = 65.3

4.6.2 Пример 2 — Передача параметров по адресу

В данном примере создается простая функция для вычисления среднего арифметического элементов массива произвольной длины. Адрес массива и его длина передаются функции в качестве аргументов. Результат возвращается по значению. Пример сконфигурирован для изображенной ниже кнопки *Button2* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Функция проекта MeanValueArray()

```
double MeanValueVector(double* dValue, DWORD dwSize)
{
    double dSum = 0.0;
    int i;
    for(i=0; i<dwSize; i++)
    {
        dSum = dSum + dValue[i];
    }
    return (dSum / dwSize);
}
```

- В заголовке в качестве имени функции указывается *MeanValueVector()*. Функции передается указатель на переменную типа *double*. Этот указатель ссылается на первый элемент передаваемого массива. Кроме того, функции передается длина массива. Функция возвращает переменную типа *double*.
- Далее описывается и инициализируется переменная типа *double*. Эта переменная предназначена для хранения суммы элементов передаваемого массива. Вычисление суммы производится с применением цикла *for*.
- Посредством команды возврата *return* результат вычислений передается в вызывающую программу. Этот результат представляет собой сумму элементов массива, деленную на его длину.

Процедура Си, связанная с кнопкой Button2

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //define and initialize double vector
    double dValue[3] = { 126.2, 23.9, 45.7 };

    double dMeanValue;

    //calculate mean value of vector
    dMeanValue = MeanValueVector(dValue, 3);

    //output into diagnostics window
    printf("\r\nExample 2\r\n");

    printf("The mean value of %.1f, %.1f and %.1f = %.1f\r\n",
           dValue[0], dValue[1], dValue[2], dMeanValue);

}
```

- В первой части описывается и инициализируется массив из трех переменных типа *double*. Необходимо вычислить среднее арифметическое значений этих трех переменных. Дополнительно описывается переменная типа *double*, в которой будет храниться результат вычислений.
- Используя созданную ранее функцию *MeanValueVector()*, вычисляется среднее арифметическое элементов массива.
- Результат вычислений выводится на печать при помощи функции *printf()*. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

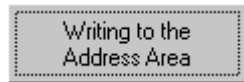
Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

Example 2

The mean value of 126.2, 23.9 and 45.7 = 65.3

4.6.3 Запись в переданный диапазон адресов

В данном примере создается простая функция для заполнения массива произвольной длины псевдослучайными числами. Функции передается адрес массива и его длина. Функция возвращает значение типа *BOOL*, указывающее, успешно ли произошло завершение работы. Пример сконфигурирован для изображенной ниже кнопки *Button3* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Функция проекта FillArray()

```

BOOL FillVector(int* piVector, DWORD dwSize)
{
    int i;

    //check received pointer
    if (piVector == NULL)
    {
        return FALSE;
    }

    //fill vector
    for (i=0; i<dwSize; i++)
    {
        piVector[i] = rand();
    }

    return TRUE;
}

```

- В заголовке в качестве имени функции указывается *FillVector()*. В качестве аргумента функции передается указатель на *int*. Данный указатель ссылается на первый элемент передаваемого массива. Дополнительно в функцию передается длина массива. Функция возвращает значение типа *BOOL*, указывающее, успешно ли произошло завершение работы.
- Далее описывается переменная–счетчик типа *int*.
- Затем проверяется переданный указатель. Вызывающая программа должна правильно указать длину массива. Передача некорректного значения может привести к общей ошибке доступа.
- В цикле *for* элементы переданного массива заполняются произвольными (псевдослучайными) значениями, генерируемыми функцией *rand()*.

Процедура Си, связанная с кнопкой Button3

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    #define VECTOR_SIZE 5

    //define int vector
    int iVector[VECTOR_SIZE];

    int i;

    printf("\r\nExample 3\r\n");

    //fill vector
    if (FillVector(iVector, VECTOR_SIZE) == FALSE)
    {
        printf("Error in FillVector\r\n");
        return;
    }

    printf("Vector Elements: ");

    for (i=0; i<VECTOR_SIZE; i++)
    {
        printf("[%d] ", iVector[i]);
    }

    printf("\r\n");
}
```

- В первой части описывается символьная константа *VECTOR_SIZE*, определяющая количество элементов массива.
- Далее описывается массив *iVector*, состоящий из *VECTOR_SIZE* элементов типа *int*.
- Затем описывается переменная–счетчик *i* типа *int*.
- Используя ранее созданную функцию *FillVector()*, массив *iVector* заполняется произвольными числами. При вызове функции *FillVector()* ее возвращаемое значение проверяется при помощи условного оператора *if*.
- Значения элементов массива *iVector* выводятся на печать при помощи функции *printf()*. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

Example 3

Vector Elements: [18467] [6334] [26500] [19169] [15724]

4.6.4 Возврат результата по указателю

В данном примере создается простая функция для формирования массива произвольной длины со случайными числами. Функции передается требуемая длина массива. Функция возвращает адрес первого элемента созданного массива. Пример сконфигурирован для изображенной ниже кнопки *Button4* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.

Функция проекта `GetFilledArray()`

```
int* GetFilledVector(DWORD dwSize)
{
    int* piVector = NULL;

    int i;

    //allocate memory for vector
    piVector = SysMalloc(sizeof(int) * dwSize);

    //check return value of SysMalloc()
    if (piVector == NULL)
    {
        return NULL;
    }

    //fill vector
    for (i=0; i<dwSize; i++)
    {
        piVector[i] = rand();
    }

    return piVector;
}
```

- В заголовке в качестве имени функции указывается *GetFilledVector()*. В качестве аргумента функции передается количество элементов в массиве, который необходимо создать. Функция возвращает указатель типа *int**, ссылающийся на первый элемент массива.

- Далее описывается указатель *piVector* на переменную типа *int*, который инициализируется значением *NULL*.
- Затем описывается переменная–счетчик *i* типа *int*.
- Для массива должна быть выделена достаточная область памяти. Это делается при помощи внутренней функции *SysMalloc()*. Ей передается размер необходимого блока памяти, вычисляемый как произведение размера переменной типа *int* на количество элементов массива. Функция возвращает адрес выделенной области памяти, или *NULL*, если памяти недостаточно.
- После этого производится проверка адреса, полученного от функции *SysMalloc()*. Если памяти для выделения требуемого блока недостаточно, функция завершается и возвращает *NULL*.
- В цикле *for* элементы массива заполняются псевдослучайными значениями, генерируемыми функцией *rand()*.
- Посредством команды возврата *return* адрес созданного массива передается в основную программу.

Процедура Си, связанная с кнопкой Button4

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
#define VECTOR_SIZE 5

//declare pointer to save address of int vector
int* piVector = NULL;

int i;

printf("\r\nExample 4\r\n");

//get address of filled vector
piVector = GetFilledVector(VECTOR_SIZE);

if (piVector == NULL)
{
    printf("Error in GetFilledVector\r\n");
    return;
}

printf("Vector Elements: ");

for (i=0; i<VECTOR_SIZE; i++)
{
    printf("[%d] ", piVector[i]);
}

printf("\r\n");

}
```

- В первой части описывается символьная константа *VECTOR_SIZE*, определяющая количество элементов массива.
- Далее описывается указатель *piVector* на переменную типа *int* и инициализируется значением *NULL*.
- Затем описывается переменная–счетчик *i* типа *int*.
- Используя ранее созданную функцию *GetFilledVector()*, создается массив из произвольных чисел, а его адрес записывается в указатель *piVector*. Значение, возвращаемое функцией *GetFilledVector()*, проверяется на корректность.

- Значения элементов созданного массива выводятся на печать при помощи функции `printf()`. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

Example 4

Vector Elements: [11478] [29358] [26962] [24464] [5705]

Замечание:

Методы передачи структур между функцией и основной программой описываются в следующей главе.

4.7 Структуры

В проекте WinCC Project_C_Course примеры, относящиеся к работе со структурами, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называющейся `ss_9_example_04.PDL`.



Structures

Определение структурного типа

Помимо применения стандартных типов данных пользователь имеет возможность создавать свои собственные структурные типы. В следующем фрагменте программы поясняются основы определения структурных типов. Созданный структурный тип данных состоит из одного элемента типа `int` и одного элемента типа `float`. Каждому структурному типу необходимо присваивать собственное имя.

```
struct ExampleStruct
{
    int iElement;
    float fElement;
};
```

Использование структурных переменных

Как показано в приведенном ниже примере, после определения нового структурного типа можно описывать переменные типа `struct ExampleStruct`. Этот пример также демонстрирует метод доступа к полям переменной структурного типа.

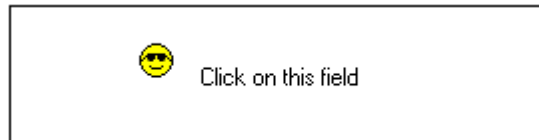
```
struct ExampleStruct esValue;

esValue.iElement = 100;
esValue.fElement = 2.5;
```

Если вместо структурной переменной имеется лишь указатель, ссылающийся на нее, то доступ к отдельным полям данной структуры можно получить так же, как это сделано в следующем примере программы. Удостоверьтесь, что указатель ссылается на реально существующую структурную переменную, или, по крайней мере, на область памяти, выделенную под нее.

4.7.1 Пример 1 — Структурная переменная

В данном примере поясняются основные принципы создания и использования переменной структурного типа. Пример сконфигурирован для изображенного ниже статического текста *StaticText1* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная со статическим текстом Static Text1

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszI
{
//define structure "CC_POINT"
struct CC_POINT
{
    int iLeft;
    int iTop;
};

//define structure tag "posObject"
struct CC_POINT posObject;

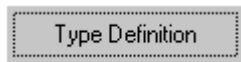
//set structure elements
posObject.iLeft = x - 8;
posObject.iTop = y - 8;

//access structure elements
SetLeft(lpszPictureName, "cool_man", posObject.iLeft);
SetTop(lpszPictureName, "cool_man", posObject.iTop);
}
```

- В первой части производится определение структурного типа *CC_POINT*, состоящего из двух элементов типа *int*. Данный структурный тип предназначен для описания координат щелчка мыши.
- Затем описывается структурная переменная *posObject* типа *struct CC_POINT*.
- Далее элементам (полям) объекта *posObject* присваиваются определенные значения. Эти значения представляют собой координаты щелчка мыши. Они передаются *Си-действию*, привязанному к событию *Event (Событие)* → *Mouse (Мышь)* → *Mouse Press Left (Нажатие левой кнопки)* в виде аргументов *x* и *y*.
- После этого координаты объекта устанавливаются равными значениям полей структурной переменной при помощи функций *SetLeft()* и *SetTop()*.

4.7.2 Пример 2 — Определение пользовательского типа данных

В данном примере поясняются основные принципы определения и применения пользовательского типа данных. В отличие от предыдущего примера этот структурный тип данных доступен в рамках всего проекта, а не только в *Си-действии*. Пример сконфигурирован для изображенной ниже кнопки *Button2* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Определение структуры в ardefar.h

```
#include "AP_PBIB.H"

//define structure tagCC_Rect
typedef struct tagCC_RECT
{
    int iLeft;
    int iTop;
    int iRight;
    int iBottom;
}
CC_RECT, //define type CC_RECT as struct tagCC_Rect
*PCC_RECT; //define type PCC_RECT as pointer to struct tagCC_Rect

//define constants for function GetFileName()
#define GFN_SAVE 0
#define GFN_OPEN 1
```

- Определяется структурный тип *tagCC_RECT*, состоящий из четырех элементов типа *int*. Данный структурный тип предназначен для описания координат и размеров прямоугольной области. Затем описывается структурная переменная типа *struct tagCC_RECT*. Чтобы избежать громоздкой нотации можно использовать псевдоним, определяемый посредством команды переименования типов *typedef*. Если Вы хотите описать переменную данного типа, достаточно указать псевдоним *CC_RECT*. Если необходимо описать указатель, ссылающийся на переменную этого типа, можно использовать псевдоним *PCC_RECT*.

Процедура Си, связанная с кнопкой Button2

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //define and initialize CC_RECT structure
    CC_RECT rect = { 10, 10, 20, 20};

    //define and initialize pointer to CC_RECT structure
    PCC_RECT prect = NULL;

    printf("\r\nExample 2\r\n");

    //access struct elements
    printf("Coordinates: %d, %d, %d, %d\r\n",
        rect.iLeft, rect.iTop, rect.iRight, rect.iBottom);

    //access struct elements via pointer
    prect = &rect;

    printf("Coordinates: %d, %d, %d, %d\r\n",
        prect->iLeft, prect->iTop, prect->iRight, prect->iBottom);
}
```

- В первой части описывается и инициализируется переменная *rect* типа *CC_RECT*.
- Далее описывается переменная *prect* типа *PCC_RECT* и инициализируется значением *NULL*. Этот тип данных является указателем на переменную типа *CC_RECT*.
- Затем, посредством оператора *.*, производится доступ к элементам структурной переменной *rect*. Ее содержимое выводится на печать при помощи функции *printf()*.
- После этого адрес переменной *rect* присваивается указателю *prect*. Затем, посредством оператора *->*, осуществляется доступ к элементам структурного типа, на который ссылается указатель *prect*. Содержимое структурной переменной вновь выводится на печать при помощи функции *printf()*. Результаты работы программы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

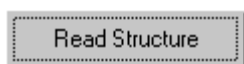
Example 2

Coordinates: 10, 10, 20, 20







Coordinates: 10, 10, 20, 20

4.7.3 Пример 3 — Структурный тип WinCC

В данном примере поясняются основные принципы определения и применения структурного типа WinCC. Его формат идентичен использованному в предыдущем примере типу *CC_RECT*. В отличие от предыдущего примера этот структурный тип данных доступен в рамках всего проекта, а не только в *Си-действии*. Пример сконфигурирован для изображенной ниже кнопки *Button3* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Создание структурного типа WinCC

Шаг	Процедура создания структурного типа WinCC
1	<p>Новый структурный тип WinCC создается в <i>WinCC Explorer</i>. Щелкните правой кнопкой мыши  на пункт <i>Structure Types</i> и выберите во всплывающем меню пункт <i>New Structure Type</i>. После этого откроется диалоговое окно для описания свойств новой структуры WinCC.</p> 
2	<p>Откроется диалоговое окно <i>Structure Properties</i>. Необходимо задать имя нового структурного типа. Это делается нажатием на правую кнопку мыши  на задаваемом по умолчанию имени <i>NewStructure</i> и последующим выбором пункта <i>Rename</i> во всплывающем меню. В данном примере используется имя <i>Rect</i>.</p> 
3	<p>Описание элементов нового структурного типа. Новый элемент можно добавить при помощи кнопки <i>New Element</i>. Имя и тип добавляемого элемента указываются нажатием на нем правой кнопкой мыши . В данном примере элемент назван <i>Left</i> и имеет тип <i>LONG</i>. Для него выбран пункт переключателя <i>Internal Tag</i>. Имена и типы остальных элементов показаны на приведенном ниже кадре.</p> <p>Теперь диалоговое окно <i>Structure Properties (Свойств структуры)</i> можно закрыть, нажав на <i>OK</i>.</p> 

Шаг	Процедура создания структурного типа WinCC										
4	<p>Теперь можно создавать теги WinCC типа <i>Rect</i>. В данном примере это означает, что для каждой переменной структурного типа будет создано четыре тега, соответствующих элементам структуры.</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>U16i_course_op_1</td> <td>Unsigned 16-bit value</td> </tr> <tr> <td>U16i_course_op_2</td> <td>Unsigned 16-bit value</td> </tr> <tr> <td>U16i_course_op_3</td> <td>Unsigned 16-bit value</td> </tr> <tr> <td>U08i_course_op_1</td> <td>Unsigned 8-bit value</td> </tr> </tbody> </table>	Name	Type	U16i_course_op_1	Unsigned 16-bit value	U16i_course_op_2	Unsigned 16-bit value	U16i_course_op_3	Unsigned 16-bit value	U08i_course_op_1	Unsigned 8-bit value
Name	Type										
U16i_course_op_1	Unsigned 16-bit value										
U16i_course_op_2	Unsigned 16-bit value										
U16i_course_op_3	Unsigned 16-bit value										
U08i_course_op_1	Unsigned 8-bit value										

Процедура Си, связанная с кнопкой Button3

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //define CC_RECT structure
    CC_RECT rect;

    //read element values of wincc structure tag
    rect.iLeft = GetTagSDWord("STRi_course_str_1.Left");
    rect.iTop = GetTagSDWord("STRi_course_str_1.Top");
    rect.iRight = GetTagSDWord("STRi_course_str_1.Right");
    rect.iBottom = GetTagSDWord("STRi_course_str_1.Bottom");

    printf("\r\nExample 3\r\n");

    //access struct elements
    printf("Coordinates: %d, %d, %d, %d\r\n",
        rect.iLeft, rect.iTop, rect.iRight, rect.iBottom);
}
}
```

- В первой части описывается переменная *rect* типа *CC_RECT*. Тип *CC_RECT* был определен в предыдущем примере.
- Затем значения элементов структурного тега WinCC записываются в поля переменной *rect*. В данном примере значения структурного тега WinCC отображаются и могут редактироваться при помощи четырех полей ввода/вывода (*I/O Fields*).
- Далее значения полей структурной переменной *rect* выводятся на печать при помощи функции *printf()*. Результаты работы программы приведены в следующем разделе.

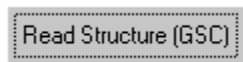
Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

Example 3
Coordinates: 0, 0, 0, 0

4.7.4 Пример 4 — Функция для чтения структурного типа WinCC

В данном примере создается функция для чтения структурного типа WinCC, описанного в предыдущем примере. Впоследствии данная функция может быть использована подобно внутренней функции *GetTag*. Пример сконфигурирован для изображенной ниже кнопки *Button4* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Функция проекта GetTagRect()

```
#include "apdefap.h"
//include file which contains definition of structure tagCC_RECT

void* GetTagRect(char* lpszTagName)
{
    PCC_RECT prect = NULL;

    //max size of struct tag name = 260
    //max size of element name = 7
    char szElementTag[267]; //260 + 7

    //allocate memory for CC_RECT structure
    prect = SysMalloc(sizeof(CC_RECT));

    //check return value of SysMalloc()
    if (prect == NULL)
    {
        return NULL;
    }

    //////////////////////////////////////
    //create tag names and get tag values
    sprintf(szElementTag, "%s.Left", lpszTagName);
    prect->iLeft = GetTagSDWord(szElementTag);

    sprintf(szElementTag, "%s.Top", lpszTagName);
    prect->iTop = GetTagSDWord(szElementTag);

    sprintf(szElementTag, "%s.Right", lpszTagName);
    prect->iRight = GetTagSDWord(szElementTag);

    sprintf(szElementTag, "%s.Bottom", lpszTagName);
    prect->iBottom = GetTagSDWord(szElementTag);
    //
    //////////////////////////////////////

    //return address of structure as void*
    return (void*) prect;
}
```

- В первой части включается заголовочный файл *apdefap.h*, содержащий определение структурного типа *tagCC_RECT*.
- В заголовке в качестве имени функции указывается *GetTagRect*. Функции передается строковая переменная, содержащая название структуры WinCC, которую следует считать. Функция возвращает указатель на область памяти неопределенного типа (*void**).

- В следующей части описывается переменная *prect* типа *PCC_RECT* и инициализируется значением *NULL*. Это указатель, ссылающийся на переменную типа *CC_RECT*.
- Затем создается строковая переменная для хранения названий элементов структурного тега WinCC.
- Далее следует выделить достаточный объем памяти для размещения переменной типа *CC_RECT*. Это делается при помощи внутренней функции *SysMalloc()*. Функции передается размер необходимого блока памяти, определяемый посредством команды *sizeof()*. Функция возвращает адрес выделенной области памяти, или *NULL*, если памяти недостаточно.
- После этого проверяется адрес, полученный от функции *SysMalloc()*. Если памяти недостаточно, функция завершается и возвращает *NULL*.
- Затем формируются названия каждого элемента структурного тега WinCC и соответствующие им значения записываются в выделенную область памяти.
- Функция возвращает адрес выделенного блока памяти, в который было записано содержимое структурного тега WinCC. Данный блок памяти останется зарезервированным и сохранит размещенные данные даже после завершения функции.

Замечание:

Если вместо приведенной здесь процедуры будет создана локальная переменная типа *CC_RECT*, поля которой содержат значения элементов структурного тега WinCC, и функция вернет ее адрес, то вызывающая программа получит недействительный указатель. Данный эффект обусловлен тем, что переменная типа *CC_RECT* становится недействительной по завершении функции, и, таким образом, будет использован адрес более не существующего объекта.

Процедура Си, связанная с кнопкой Button4

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszPropert
{
//define and initialize pointer to CC_RECT structure
PCC_RECT prect = NULL;

//read element values of wincc structure tag
prect = (PCC_RECT) GetTagRect("STRi_course_str_1");

printf("\r\nExample 4\r\n");

//check return value of GetTagRect()
if (prect == NULL)
{
printf("\r\nError in GetTagRect()\r\n");
return;
}

//access struct elements
printf("Coordinates: %d, %d, %d, %d\r\n",
prect->iLeft, prect->iTop, prect->iRight, prect->iBottom);
}
```

- В первой части описывается переменная *prect* типа *PCC_RECT* и инициализируется значением *NULL*.
- Затем структурный тег WinCC считывается при помощи ранее созданной функции *GetTagRect()*. Функция *GetTagRect()* возвращает указатель, ссылающийся на блок памяти, в котором размещены искомые данные. Этот указатель преобразуется к указателю типа *PCC_RECT*.
- Далее проверяется указатель, полученный от функции *GetTagRect()*. В случае отсутствия требуемого объема памяти функция возвращает значение *NULL*.
- Затем формируются названия каждого элемента структурного тега WinCC и соответствующие им значения записываются в выделенную область памяти.
- Функция возвращает адрес выделенного блока памяти, в который было записано содержимое структурного тега WinCC. Данный блок памяти останется зарезервированным и сохранит размещенные данные даже после завершения функции.
- После этого элементы структурного тега, на который ссылается *prect*, выводятся на печать при помощи функции *printf()*. Результаты работы программы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

Example 4

Coordinates: 0, 0, 0, 0

Функция проекта SetTagRect()

Помимо функции *GetTagRect()* также была создана соответствующая функция *SetTagRect()*. В данном примере функция связана с событием *Event (Событие)* →

Miscellaneous (Разное) → *Open Picture (Открытие экранной формы)* экранной формы *cc_9_example_04.PDL* для инициализации структурного тега WinCC.

4.8 Программный интерфейс WinCC (API)

В проекте WinCC Project_C_Course примеры, относящиеся к WinCC API, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называющейся ss_9_example_10.PDL.



WinCC API

Программный интерфейс WinCC

Будучи полностью открытой и расширяемой системой, WinCC предоставляет расширенный API (Application Program Interface — Программный интерфейс приложения). Это интерфейс, используемый приложениями для доступа к WinCC. Функции WinCC API могут также использоваться в самом проекте WinCC. Пакет WinCC ODK (Open Developers Kit — Открытый комплект разработчика) обеспечивает подробное описание WinCC API. В нем детально объясняется WinCC API, приводятся описания функций и примеры. В него также входят все заголовочные файлы с необходимыми объявлениями функций. Однако пакет WinCC ODK не является составной частью базового комплекта WinCC, его следует приобретать отдельно.

Библиотеки функций

Каждое (из числа основных) приложение WinCC (Графический дизайнер, Архиватор, Регистратор аварийных событий, и т.д.) предоставляет свой собственный API, представленный в одной или нескольких DLL. DLL (Dynamic Load Library — Библиотека динамической компоновки) — это динамически подключаемая библиотека. Объявления функций, содержащихся в DLL, приводятся в соответствующем заголовочном файле.

Подключение DLL к Си-действию или другой функции проиллюстрировано в приведенном ниже фрагменте кода. В первой строке указывается название DLL, которую следует загрузить. В приведенном примере это DLL, содержащая функции CS (Configuration System — Среда конфигурирования) графического редактора. Во второй строке подключается заголовочный файл с объявлениями функций. Если требуется использовать только одну–две функции, то их на данном этапе можно объявить явно. Завершающая строка имеет вид `#pragma code()`. В приведенном примере DLL и заголовочный файл имеют общие названия, что вполне естественно. Тем не менее, это не всегда так.

```
#pragma code("PDICSAPI.dll")
#include "pdlcsapi.h"
#pragma code()
```

Функции RT и CS

Функции API любого приложения можно грубо разбить на два типа. Это так называемые функции CS (Configuration System — Среда конфигурирования) и функции RT (RunTime — Среда исполнения).

В большинстве случаев в проекте WinCC функции RT можно вызывать без предварительной загрузки какой–либо DLL. Функции RT влияют на проект только во время исполнения. После рестарта проекта, а в большинстве случаев после смены экранной формы, изменения, внесенные функциями RT, будут утрачены.

Перед применением функций CS в проекте WinCC необходимо загрузить соответствующие DLL, в которых были запрограммированы требуемые функции.

Применение функций CS в самом проекте WinCC имеет смысл только в редких случаях. Представленный пример, тем не менее, иллюстрирует использование функций CS, т.к. на основе базовых принципов работы с этими функциями Вы сможете понять, как их применять в Ваших собственных проектах.

Пример проекта

В примере проекта не приводится детальное описание WinCC API. Основные принципы работы с WinCC API объясняются на примере API графического редактора. Примеры работают с объектами экранной формы `ss_9_example_10x.PDL`, специально созданными для этого. Эта экранная форма отображается для данной главы в окне экранных форм.

4.8.1 Пример 1 — Изменение свойств с помощью функций RT

В данном примере функции RT API *графического редактора* используются для установки свойств изображенного объекта. Координаты объекта изменяются заданием свойств *Position X* (Координата X) и *Position Y* (Координата Y). Пример сконфигурирован для изображенной ниже кнопки *Button1* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button1

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    BOOL bRet = FALSE;

    //picture name without extension ".PDL"
    char szPictureName[] = "cc_9_example_10ex";
    char szObjectName[] = "I/OField1";

    //property type
    VARTYPE vt = VT_I4;
    //new property values
    int iValueLeft = 60;
    int iValueTop = 70;

    //error structure
    CMN_ERROR Error;

    //////////////////////////////////////
    //set the propertys and check the return values
    bRet = PDLRTSetPropEx(0, szPictureName, szObjectName, "Left",
        vt, &iValueLeft, NULL, NULL, 0, NULL, &Error);
    if (bRet == FALSE)
    {
        printf("\r\nError in PDLRTSetPropEx()\r\n"
            "\t%s\r\n", Error.szErrorText);
    }

    bRet = PDLRTSetPropEx(0, szPictureName, szObjectName, "Top",
        vt, &iValueTop, NULL, NULL, 0, NULL, &Error);
    if (bRet == FALSE)
    {
        printf("\r\nError in PDLRTSetPropEx()\r\n"
            "\t%s\r\n", Error.szErrorText);
    }
    //
    //////////////////////////////////////
}
```

- В первой части описывается и инициализируется переменная *bRet* типа *BOOL*. Эта переменная используется для анализа значений, возвращаемых вызываемыми функциями API.
- Затем описываются две строковые переменные. Их содержимое — название экранной формы и название объекта — задают объект для редактирования. Проверьте, что название экранной формы не содержит расширения файла *PDL*.
- Для описания типа задаваемых свойств создается переменная. Для этого предусмотрен специальный тип данных *VARTYPE*. Для каждого существующего

типа свойств определена символьная константа. Устанавливаемые в данном примере свойства имеют тип *VT_I4* (*int* длиной в 4 байта).

- Для устанавливаемых свойств *Position X* и *Position Y* описывается по переменной типа *int*, которые задают их новые значения.
- Далее описывается переменная типа *CMN_ERROR*. При сбое во время исполнения функции в данную структуру будет записана информация о произошедшей ошибке.
- Свойства *Position X* и *Position Y* заданного объекта устанавливаются при помощи функции API *PDLRTSetPropEx()*. Первый параметр функции API описывает режим адресации объекта. Следующие три параметра задают названия экранной формы, объекта и свойства. Для указания нужного свойства следует использовать его английское название, а не немецкое. Для предыдущего примера — это *Left* и *Top*. Следующий параметр описывает тип свойства. В качестве следующего параметра указывается адрес переменной, содержащей новое значение свойства. Следующие четыре параметра не имеют отношения к рассматриваемым процедурам. В последнем параметре указывается адрес структуры для анализа ошибок.
- После вызова функций API возвращаемое ими значение проверяется в условном выражении *if*. Если при выполнении происходит ошибка, сообщение об этом выводится на печать. Часть такого сообщения формируется на основе информации, содержащейся в поле *szErrorText* структуры, предназначенной для анализа ошибок.

Замечание:

В приведенном примере возвращаемое значение функции API присваивается переменной типа *BOOL*. Затем осуществляется проверка этой переменной. Вызов функции API и проверка возвращаемого значения можно объединить в одной строке, как это будет проделано в последующих примерах данной главы.

4.8.2 Пример 2 — Создание связи с тегом с помощью функции RT

В данном примере функции RT API *графического редактора* используются для создания связи с тегом. С тегом связывается свойство *поля ввода/вывода*, что делается в меню *Property (Свойство) → Output/Input (Ввод/вывод) → Output Value (Выходное значение)*. Пример сконфигурирован для изображенной ниже кнопки *Button2* в окне свойств объекта в *Event (Событие) → Mouse (Мышь) → Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button2

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //include file with the LinkType definitions
    #include "trigger.h"

    char szPictureName[] = "cc_9_example_10ex";
    char szObjectName[] = "I/OField1";

    LINKINFO link;
    CMN_ERROR Error;

    //fill link info structure
    link.LinkType = BUBRT_LT_VARIABLE_DIRECT;
    link.dwCycle = 0;
    strcpy(link.szLinkName, "U08i_course_wincc_2");

    //set link and check the return value
    if ( PDLRTSetLink(0, szPictureName, szObjectName, "OutputValue",
        &link, NULL, NULL, &Error) == FALSE)
    {
        printf("\r\nError in PDLRTSetLink()\r\n%s\r\n",
            Error.szErrorText);
    }
}
```

- В первой части включается заголовочный файл *trigger.h*. В этом файле находятся описания используемых в данном примере символьных констант.
- Затем описываются две строковые переменные. Их содержимое — название экранной формы и название объекта задают объект для редактирования.
- Для задания свойств связи с тегом предусмотрен специальный структурный тип данных *LINKINFO*. Описывается структурная переменная *link* типа *LINKINFO*.
- Далее описывается переменная типа *CMN_ERROR*.
- Поля структурной переменной *link* заполняются данными об устанавливаемой связи с тегом. Полю *LinkType* присваивается значение символьной константы *BUBRT_LT_VARIABLE_DIRECT*. Эта константа используется для обозначения прямой связи с тегом. В поле *dwCycle* записывается *0*, соответствующий триггеру *Upon Change (По изменению)*. Поле *szLinkName* указывает используемую переменную.

- Посредством функции API *PDLRTSetLink()* у выбранного объекта создается *связь с тегом*. Первый параметр функции API описывает режим адресации объекта. Следующие три параметра задают названия экранной формы, объекта и свойства. В следующем параметре указывается адрес переменной *link*, определяющей устанавливаемую связь с тегом. Следующие два параметра не имеют отношения к рассматриваемым процедурам. В последнем параметре указывается адрес структуры для анализа ошибок. Если при выполнении происходит ошибка, сообщение об этом выводится на печать.

Замечание:

Первые два примера этой главы относятся к полю ввода/вывода *I/OField1* экранной формы *cc_9_example_10x.PDL*. Первый пример изменяет положение объекта на экранной форме, второй создает связь этого объекта с тегом. При смене экранной формы все внесенные изменения утрачиваются. Проверьте, что после выполнения всех функций CS, описываемых в последующих примерах, осуществляется смена экранной формы. При этом изменения экранной формы, внесенные при помощи двух первых кнопок, будут утрачены при нажатии на одну из других кнопок.

4.8.3 Пример 3 — Создание нового объекта при помощи функций CS

В данном примере функции CS API *графического редактора* используются для создания нового объекта *I/O Field*. Пример сконфигурирован для изображенной ниже кнопки *Button3* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button3

```

#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProper
{
//PDLCSAPI.dll contains the graphics designer cs functions
#pragma code("PDLCSAPI.dll")
#include "pdlcsapi.h"
#pragma code()
//include file with the GUID definitions
#include "pdl_guid.h";

char szProjectName[_MAX_PATH];
char szPictureName[] = "cc_9_example_10ex.PDL";//with ".PDL"
DWORD dwFlags = 1;//do not display picture
CMN_ERROR Error;
char szObjectName[] = "I/OField2";
GUID guid = GUID_IOField;//object type i/o field

//get project name
if ( DMGetRuntimeProject(szProjectName,_MAX_PATH+1,&Error) == FALSE )
{
    printf("\r\nError in DMGetRuntimeProject()\r\n"
           "\t%s\r\n",Error.szErrorText);
    return;
}
//initialize API interface of the graphics designer
if ( PDLCSGetOleAppPtr(FALSE,&Error) == FALSE )
{
    printf("\r\nError in PDLCSGetOleAppPtr()\r\n%s\r\n",
           Error.szErrorText);
    return;
}
//open picture without displaying it
if ( PDLCSOpenEx(szProjectName,szPictureName,dwFlags,&Error) == FALSE )
{
    printf("\r\nError in PDLCSOpenEx()\r\n%s\r\n",
           Error.szErrorText);
    goto OPEN_FAILED;
}
//create object
if ( PDLCSNewObjectEx(szProjectName,szPictureName,&guid,
                      szObjectName,&Error) == FALSE )
{
    printf("\r\nError in PDLCSNewObjectEx()\r\n%s\r\n",
           Error.szErrorText);
    goto ACTION_FAILED;
}
//save picture
if ( PDLCSave(szProjectName,szPictureName,&Error) == FALSE )
{
    printf("\r\nError in PDLCSave()\r\n%s\r\n",
           Error.szErrorText);
    goto ACTION_FAILED;
}
//actualize the picture which contains the created object
ActualizeObjects();
//close picture
ACTION_FAILED: PDLSClose(szProjectName,szPictureName,&Error);
//disconnect from the API interface of the graphics designer
OPEN_FAILED: PDLCSDelOleAppPtr(FALSE);
}

```

- В первой части производится загрузка DLL API *графического дизайнера*. В дополнение подключается заголовочный файл *pdl_guid.h*, содержащий описания используемых в данном примере символьных констант.
- Затем описывается строковая переменная *szProjectName* для размещения названия проекта.
- Далее описывается строковая переменная для размещения названия экранной формы. Обратите внимание, что название экранной формы должно включать расширение *PDL* (в отличие от RT).
- После этого описываются дополнительные переменные. Среди них переменная типа *GUID*, определяющая тип создаваемого объекта.

- В следующей части название проекта определяется при помощи функции API *DMGetRuntimeProject()*. Название проекта будет храниться в переменной *szProjectName*. Если при определении названия проекта произойдет ошибка, сообщение об этом будет выведено на печать. В этом случае процедура Си завершается посредством команды *return*.
- В следующей части API *графического дизайнера* инициализируется при помощи функции *PDLCSGetOleAppPtr()*. Если при инициализации API *графического дизайнера* произойдет ошибка, сообщение об этом будет выведено на печать. В этом случае процедура Си также завершается посредством команды *return*.
- В следующей части подлежащая редактированию экранная форма открывается посредством функции API *PDLCSOpenEx()*. Предпоследним параметром в функцию API передается переменная *dwFlags*, имеющая значение *1*. При этом экранная форма не отображается на экране. Если при открытии экранной формы произойдет ошибка, сообщение об этом будет выведено на печать. При помощи команды *goto* выполняется переход к фрагменту кода, где разрывается связь с API *графического дизайнера*.
- В следующей части с помощью функции API *PDLCSNewObjectEx()* создается объект *I/OField2*. Если при создании объекта произойдет ошибка, сообщение об этом будет выведено на печать. При помощи команды *goto* выполняется переход к фрагменту кода, где ранее открытая экранная форма закрывается.
- В следующей части экранная форма записывается посредством функции API *PDLCSSave()*. Если при записи экранной формы произойдет ошибка, сообщение об этом будет выведено на печать. Как и в предыдущей части, при помощи команды *goto* выполняется переход к фрагменту кода, где ранее открытая экранная форма закрывается.
- Затем редактируемая экранная форма выбирается *функцией проекта* *ActualizeObjects()*.
- Далее ранее открытая экранная форма закрывается с помощью функции API *PDLCSClose()*. Перед этой командой установлена метка для описанных ранее команд *goto*.
- После этого связь с API графического редактора вновь разрывается посредством функции API *PDLCSDelOleAppPtr()*. Перед этой командой установлена метка для описанных ранее команд *goto*.

Замечание:

Процедуры Си, создаваемые в последующих примерах очень похожи на процедуру Си, представленную здесь. В связи с этим мы будем опускать подробные описания, приведенные в данном примере. Описания исходного кода будут ограничены обзором хода выполнения программы.

4.8.4 Пример 4 — Изменение свойств при помощи функции CS

В данном примере функции CS API *графического дизайнера* используются для установки свойств изображенного объекта. Координаты объекта изменяются заданием свойств *Position X* (Координата X) и *Position Y* (Координата Y). Пример сконфигурирован для изображенной ниже кнопки *Button4* в окне свойств объекта в *Event (Событие) → Mouse (Мышь) → Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button4

```

#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty
{
#pragma code("PDLCSAPI.dll")
#include "pdlcsapi.h"
#pragma code()

char szProjectName[_MAX_PATH];
char szPictureName[] = "cc_9_example_10ex.PDL";
char szObjectName[] = "I/OField2";
char szPropertyName[2][5] = { "Left", "Top" };
VARTYPE vt = VT_I4;
int iValue[] = { 60, 130 };
int i;
CMN_ERROR Error;
//get project name
if (DMGetRuntimeProject(szProjectName,_MAX_PATH+1,&Error) == FALSE)
{
printf("\r\nError in DMGetRuntimeProject()\r\n",
"\t%s\r\n",Error.szErrorText);
return;
}
//initialize API interface of the graphics designer
if ( PDLCSGetOleAppPtr(FALSE,&Error) == FALSE)
{
printf("\r\nError in PDLCSGetOleAppPtr()\r\n%s\r\n",
Error.szErrorText);
return;
}
//open picture without displaying it
if ( PDLCSOpenEx(szProjectName,szPictureName,1,&Error) == FALSE)
{
printf("\r\nError in PDLCSOpenEx()\r\n%s\r\n",
Error.szErrorText);
goto OPEN_FAILED;
}
//set propertys
for (i=0; i<2; i++)
{
if (PDLCSsetPropertyEx(szProjectName,szPictureName,szObjectName,
szPropertyName[i],vt,iValue+i,0,NULL,&Error) == FALSE)
{
printf("\r\nError in PDLCSsetPropertyEx()\r\n%s\r\n",
Error.szErrorText);
}
}
//save picture
if ( PDLCSsave(szProjectName,szPictureName,&Error) == FALSE)
{
printf("\r\nError in PDLCSsave()\r\n%s\r\n",
Error.szErrorText);
goto ACTION_FAILED;
}
//actualize the picture which contains the created object
ActualizeObjects();
//close picture
ACTION_FAILED: PDLCSClose(szProjectName,szPictureName,&Error);
//disconnect from the API interface of the graphics designer
OPEN_FAILED: PDLCSDelOleAppPtr(FALSE);
}

```

- В первой части загружается DLL API *графического дизайнера*.
- Во второй части описываются необходимые переменные. Названия и значения задаваемых свойств хранятся в массивах, в отличие от примера 1.
- Название проекта определяется при помощи функции API *DMGetRuntimeProject()*.
- API *графического дизайнера* инициализируется функцией API *PDLCSGetOleAppPtr()*.
- Подлежащая редактированию экранная форма открывается при помощи функции API *PDLCSOpenEx()*.
- В цикле *for* с помощью функции API *PDLCSsetPropertyEx()* устанавливаются свойства объекта. Если устанавливаемые таким образом свойства объекта имеют

различные типы, то вместо переменной *vt* следует использовать массив. Этот массив будет определять тип устанавливаемого свойства.

- Экранная форма записывается посредством функции API *PDLCSSave()*.
- Редактируемая экранная форма открывается вновь с помощью *функции проекта ActualizeObjects()*.
- Экранная форма вновь закрывается при помощи функции API *PDLSCClose()*.
- Связь с API графического дизайнера разрывается посредством функции API *PDLCSDelOleAppPtr()*.

4.8.5 Пример 5 — Создание связи с тегом с помощью функции CS

В данном примере функции CS API *графического дизайнера* используются для создания связи с тегом. С тегом связывается свойство *поля ввода/вывода*, что делается в меню *Property (Свойство) → Output/Input (Ввод/вывод) → Output Value (Выходное значение)*. Пример сконфигурирован для изображенной ниже кнопки *Button5* в окне свойств объекта в *Event (Событие) → Mouse (Мышь) → Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button5

```

#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
#pragma code("PDLCSAPI.dll")
#include "pdlcsapi.h"
#pragma code()

char szProjectName[_MAX_PATH];
char szPictureName[] = "cc_9_example_10ex.PDL";
char szObjectName[] = "I/OField2";
LINK_INFO link;
CMN_ERROR Error;

//get project name
if (DMGetRuntimeProject(szProjectName, _MAX_PATH+1, &Error) == FALSE)
{
    printf("\r\nError in DMGetRuntimeProject()\r\n",
           "\t%s\r\n", Error.szErrorText);
    return;
}
//initialize API interface of the graphics designer
if ( PDLCSGetOleAppPtr(FALSE, &Error) == FALSE)
{
    printf("\r\nError in PDLCSGetOleAppPtr()\r\n%s\r\n",
           Error.szErrorText);
    return;
}
//open picture without displaying it
if ( PDLCSOpenEx(szProjectName, szPictureName, 1, &Error) == FALSE)
{
    printf("\r\nError in PDLCSOpenEx()\r\n%s\r\n",
           Error.szErrorText);
    goto OPEN_FAILED;
}
//set link info struct
link.enumLinkType = BUBRT_LT_VARIABLE_DIRECT;
link.dwCycle = 0;
strcpy(link.szLinkName, "U08i_course_wincc_1");
//set link
if ( PDLCSSetLink(szProjectName, szPictureName, szObjectName, "OutputValue",
                  &link, &Error) == FALSE)
{
    printf("\r\nError in PDLCSSetLink()\r\n%s\r\n",
           Error.szErrorText);
    goto ACTION_FAILED;
}
//save picture
if ( PDLCSave(szProjectName, szPictureName, &Error) == FALSE)
{
    printf("\r\nError in PDLCSave()\r\n%s\r\n",
           Error.szErrorText);
    goto ACTION_FAILED;
}
//actualize the picture which contains the created object
ActualizeObjects();
//close picture
ACTION_FAILED: PDLSClose(szProjectName, szPictureName, &Error);
//disconnect from the API interface of the graphics designer
OPEN_FAILED: PDLCSdelOleAppPtr(FALSE);
}

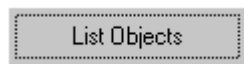
```

- В первой части загружается DLL API *графического дизайнера*.
- Во второй части описываются необходимые переменные.
- Название проекта определяется при помощи функции API *DMGetRuntimeProject()*.
- API *графического дизайнера* инициализируется функцией API *PDLCSGetOleAppPtr()*.
- Подлежащая редактированию экранная форма открывается при помощи функции API *PDLCSOpenEx()*.
- В следующей части в поля структурной переменной *link* записываются данные об устанавливаемой с тегом связи.
- Связь с тегом создается при помощи функции *PDLCSSetLink()*.

- Экранная форма записывается посредством функции API *PDLCSave()*.
- Редактируемая экранная форма открывается вновь с помощью *функции проекта ActualizeObjects()*.
- Экранная форма вновь закрывается при помощи функции API *PDLSClose()*.
- Связь с API графического дизайнера разрывается посредством функции API *PDLCSDelOleAppPtr()*.

4.8.6 Пример 6 — Перечисление объектов с помощью функции CS

В данном примере функции CS API *графического редактора* используются для перечисления объектов, размещенных на отображаемой экранной форме. Для каждого имеющегося объекта API вызовет специально созданную функцию, которой передаст в качестве параметров данные о текущем объекте. Такая функция называется *функцией обратного вызова* (Callback). Пример сконфигурирован для изображенной ниже кнопки *Button6* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Функция проекта ObjectCallback()

```
#include "pdicsapi.h"
//include file with OBJECT_INFO_STRUCT definition

BOOL ObjectCallback(void* lpData, void* item)
{
    //pointer to OBJECT_INFO_STRUCT
    LPOBJECT_INFO_STRUCT lpInfoStruct = NULL;

    //store received address of OBJECT_INFO_STRUCT
    lpInfoStruct = (LPOBJECT_INFO_STRUCT) lpData;

    //check received address
    if (lpInfoStruct == NULL)
    {
        printf("Error in ObjectCallback()\r\n");
        return FALSE;
    }

    //access structure element
    printf("%s\r\n", lpInfoStruct->szObjectName);

    return TRUE;
}
```

- В первой части включается заголовочный файл *pdicsapi.h*, содержащий определение структурного типа *OBJECT_INFO_STRUCT*.
- Тип возвращаемого значения, а также количество и типы аргументов данной функции можно узнать из WinCC ODK.
- Затем описывается указатель, ссылающийся на структурную переменную типа *OBJECT_INFO_STRUCT*. Этой переменной присваивается адрес, переданный в первом аргументе. Производится проверка этого указателя.
- Название объекта, полученного в *OBJECT_INFO_STRUCT* выводится на печать.

Процедура Си, связанная с кнопкой Button6

```

#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    #pragma code("PDLCSAPI.dll")
    #include "pdlcsapi.h"
    #pragma code()

    char szProjectName[_MAX_PATH];
    char szPictureName[] = "cc_9_example_10ex.PDL";
    CMN_ERROR Error;

    //get project name
    if ( DMGetRuntimeProject(szProjectName,_MAX_PATH+1,&Error) == FALSE)
    {
        printf("\r\nError in DMGetRuntimeProject()\r\n"
            "\t%s\r\n",Error.szErrorText);
        return;
    }
    //initialize API interface of the graphics designer
    if ( PDLCSGetOleAppPtr(FALSE,&Error) == FALSE)
    {
        printf("\r\nError in PDLCSGetOleAppPtr()\r\n%s\r\n",
            Error.szErrorText);
        return;
    }
    //open picture without displaying it
    if ( PDLCSOpenEx(szProjectName,szPictureName,1,&Error) == FALSE)
    {
        printf("\r\nError in PDLCSOpenEx()\r\n%s\r\n",
            Error.szErrorText);
        goto OPEN_FAILED;
    }
    printf("\r\nObjects in picture cc_9_example_10ex.PDL:\r\n");
    //enumerate objects
    if ( PDLCSEnumObjList(szProjectName,szPictureName,
        ObjectCallback,NULL,&Error) == FALSE)
    {
        printf("\r\nError in PDLCSEnumObjectList()\r\n%s\r\n",
            Error.szErrorText);
    }
    //close picture
    PDLCSClose(szProjectName,szPictureName,&Error);
    //disconnect from the API interface of the graphics designer
    OPEN_FAILED: PDLCSDelOleAppPtr(FALSE);
}

```

- В первой части загружается DLL API *графического дизайнера*.
- Во второй части описываются необходимые переменные.
- Название проекта определяется при помощи функции API *DMGetRuntimeProject()*.
- API *графического дизайнера* инициализируется функцией API *PDLCSGetOleAppPtr()*.
- Подлежащая редактированию экранная форма открывается при помощи функции API *PDLCSOpenEx()*.
- Функция API *PDLCSEnumObjList()* перечисляет все объекты экранной формы. Для этого ей передается адрес предварительно созданной функции проекта *ObjectCallback()*. Такая функция называется функцией обратного вызова. Она вызывается по одному разу для каждого объекта экранной формы и получает данные об этом объекте в качестве передаваемых аргументов.
- Экранная форма записывается посредством функции API *PDLCSSave()*.
- Редактируемая экранная форма открывается с помощью *функции проекта ActualizeObjects()*.

- Экранная форма закрывается при помощи функции API *PDLCSClose()*.
- Связь с API графического дизайнера разрывается посредством функции API *PDLCSDelOleAppPtr()*.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
Objects in picture cc_9_example_10ex.PDL:  
cc_9_example_10ex  
!OField1  
!OField2
```

4.9 Среда проекта

В проекте WinCC Project_C_Course примеры, относящиеся к среде проекта, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называемой `ss_9_example_11.PDL`.



Project Environment

Общие сведения

Зачастую при программировании процедур Си или других функций необходимо указывать путь к файлу, название локального компьютера и т.п. Данные параметры можно задать как абсолютные значения в соответствии с текущей конфигурацией. Это может вызвать проблемы при переносе проекта на другой компьютер. Конфигурация другого компьютера с большой степенью вероятности отличается от конфигурации того компьютера, на котором создавался проект. В связи с этим рекомендуется воздерживаться от использования каких-либо параметров, привязанных к конкретной конфигурации. Вместо этого все подобные данные следует определять во время исполнения. В данной главе приводятся примеры, иллюстрирующие возможности доступа к конфигурации локального компьютера. Для этого используются как WinCC API, так и Windows API.

4.9.1 Пример 1 — Определение файла проекта

В данном примере описывается процедура определения файла проекта WinCC. Пример сконфигурирован для изображенной ниже кнопки *Button1* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button1

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    BOOL bRet;
    char szProjectFile[_MAX_PATH+1];
    CMN_ERROR Error;

    //get the project file *.mcp
    bRet = DMGetRuntimeProject(szProjectFile, _MAX_PATH+1, &Error);

    //check return value
    if (bRet == FALSE)
    {
        printf("\r\nError in DMGetRuntimeProject()\r\n"
            "\t%s\r\n", Error.szErrorText);
        return;
    }

    //display project file
    printf("\r\nProjectFile: \r\n%s\r\n", szProjectFile);
}
```

- В первой части описывается переменная *bRet* типа *BOOL*.
- Затем описывается строковая переменная *szProjectFile* для размещения названия файла проекта. Размер этой переменной определяется максимальной возможной длиной описания пути к файлу.
- Далее описывается переменная типа *CMN_ERROR*.
- После этого название проекта определяется при помощи функции API *DMGetRuntimeProject()*. Название проекта заносится в переменную *szProjectFile*. В качестве второго параметра указывается размер области памяти, выделенной под имя проекта. Третий параметр указывает адрес структуры для анализа ошибок. Если информация о возможных ошибках не требуется, можно передать *NULL*.
- Затем проверяется значение, возвращаемое функцией API *DMGetRuntimeProject()*.
- Далее полученное имя файла проекта выводится на печать. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

ProjectFile:
\\ZIP-WS5\WinCC50_Project_Project_C_Course\Project_C_Course.MCP

4.9.2 Пример 2 — Определение пути проекта

В данном примере описывается процедура определения пути проекта WinCC. Пример сконфигурирован для изображенной ниже кнопки *Button2* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button2

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    BOOL bRet = FALSE;
    char szProjectFile[_MAX_PATH+1];
    char* psz = NULL;
    CMN_ERROR Error;

    //get the project file *.mcp
    bRet = DMGetRuntimeProject(szProjectFile, _MAX_PATH+1, &Error);

    //check return value
    if (bRet == FALSE)
    {
        printf("\r\nError in DMGetRuntimeProject()\r\n"
            "\t%s\r\n", Error.szErrorText);
        return;
    }

    //search for last backslash
    psz = strrchr(szProjectFile, '\\');

    //cut string after last backslash
    if (psz != NULL)
    {
        *(psz+1) = 0;
    }

    //display project path
    printf("\r\nProjectPath:\r\n%s\r\n", szProjectFile);
}
}
```

- В первой части описывается и инициализируется переменная *bRet* типа *BOOL*.
- Затем описывается строковая переменная *szProjectFile* для размещения названия файла проекта. Еще одна строковая переменная описывается как *char** и инициализируется значением *NULL*.
- Далее описывается переменная типа *CMN_ERROR*.
- После этого название проекта определяется при помощи функции API *DMGetRuntimeProject()*.
- Затем проверяется значение, возвращаемое функцией API *DMGetRuntimeProject()*.
- После этого функция *strrchr()* ищет последнее вхождение символа \ в полученном имени файла проекта. Сразу за найденным вхождением записывается 0. В результате остается только описание пути проекта, без имени самого файла.
- Далее полученный путь проекта выводится на печать. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
ProjectPath:  
\\ZIP-WS5\WinCC50_Project_Project_C_Course\
```

4.9.3 Пример 3 — Определение пути проекта с помощью функции проекта

В данном примере процедура определения каталога проекта WinCC переносится в функцию проекта. Пример сконфигурирован для изображенной ниже кнопки *Button3* в окне свойств объекта в *Event (Событие) → Mouse (Мышь) → Mouse Action (Процедура мыши)*.



Функция проекта GetProjectPath()

```
BOOL GetProjectPath(char* lpstrProjectPath)
{
    BOOL bRet = FALSE;
    char szProjectFile[_MAX_PATH+1];
    char* psz = NULL;
    CMN_ERROR Error;

    bRet = DMGetRuntimeProject(szProjectFile,_MAX_PATH+1,&Error);

    if (bRet == FALSE)
    {
        return FALSE;
    }

    psz = strrchr(szProjectFile, '\\');

    if (psz == NULL)
    {
        return FALSE;
    }

    *(psz+1) = 0;

    strcpy(lpstrProjectPath, szProjectFile);

    return TRUE;
}
```

- Функции проекта передается строковая переменная, в которую будет записан полученный путь проекта. При вызове этой функции необходимо удостовериться в том, что строковой переменной была выделена достаточная область памяти. Успешность завершения работы функции можно оценить по ее возвращаемому значению.
- Процедура определения пути проекта организована так же, как и в предыдущем примере.
- Полученный путь проекта копируется в переданную строковую переменную посредством функции *strcpy()*.

Процедура Си, связанная с кнопкой Button3

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    BOOL bRet = FALSE;
    char szProjectPath[_MAX_PATH+1];

    //project function to get project path
    bRet = GetProjectPath(szProjectPath);

    //check return value
    if (bRet == FALSE)
    {
        printf("\r\nError in GetProjectPath()\r\n");
        return;
    }

    //display project path
    printf("\r\nProjectPath:\r\n%s\r\n", szProjectPath);
}
```

- В первой части описывается и инициализируется переменная *bRet* типа *BOOL*.
- Затем описывается строковая переменная *szProjectFile* для размещения пути проекта.
- Используя ранее созданную функцию проекта *GetProjectPath()* определяется путь проекта. Затем проверяется значение, возвращаемое этой функцией.
- Далее полученный путь проекта выводится на печать.

4.9.4 Пример 4 — Определение инсталляционного каталога

В данном примере описывается процедура определения инсталляционного каталога WinCC. Пример сконфигурирован для изображенной ниже кнопки *Button4* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button4

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    char szProjectFile[_MAX_PATH+1];
    DM_DIRECTORY_INFO dmDirInfo;
    CMN_ERROR Error;
    char* psz = NULL;

    //get the project file *.mcp
    if ( DMGetRuntimeProject(szProjectFile,_MAX_PATH+1,&Error) == FALSE)
    {
        printf("\r\nError in DMGetRuntimeProject()\r\n"
            "\t%s\r\n",Error.szErrorText);
        return;
    }

    //get wincc directories
    if ( DMGetProjectDirectory("",szProjectFile,&dmDirInfo,&Error) == FALSE)
    {
        printf("\r\nError in DMGetProjectDirectory()\r\n"
            "\t%s\r\n",Error.szErrorText);
        return;
    }

    if ( (psz = strrchr(dmDirInfo.szGlobalLibDir,'\\')) != NULL)
    {
        *psz = 0;
    }
    if ( (psz = strrchr(dmDirInfo.szGlobalLibDir,'\\')) != NULL)
    {
        *(psz+1) = 0;
    }

    //display installation directory
    printf("\r\nInstallationDirectory:\r\n%s\r\n",dmDirInfo.szGlobalLibDir);
}
}
```

- В первой части описывается строковая переменная *szProjectFile* для размещения названия файла проекта.
- Затем описывается переменная *dmDirInfo* для хранения данных о каталоге. Эта структурная переменная имеет тип *DM_DIRECTORY_INFO*.
- Далее описывается переменная *CMN_ERROR*.
- После этого описывается строковая переменная типа *char** и инициализируется значением *NULL*.
- Название проекта определяется с помощью функции API *DMGetRuntimeProject()*.
- При помощи функции API *DMGetProjectDirectory()* в переменную *dmDirInfo* заносятся данные о каталогах. Один из путей, описываемых данной переменной, является путем к каталогу глобальной библиотеки (Global Library). Этот путь

хранится в поле *szGlobalLibDir*. Каталог глобальной библиотеки является подкаталогом инсталляционного каталога WinCC.

- Первый вызов функции `strchr()` возвращает позицию последнего символа \, в которую затем записывается 0. Второй вызов функции `strchr()` возвращает позицию последнего символа \ в оставшейся части пути. В ячейку, следующую непосредственно за этой позицией, записывается 0.
- Затем полученный инсталляционный каталог выводится на печать. Результаты работы приведены в следующем разделе.

Вывод в окне диагностики

Пример, описываемый в данной главе, осуществляет в *окне диагностики* вывод следующего текста:

```
InstallationDirectory:  
C:\Siemens\WinCC\
```

4.9.5 Пример 5 — Определение имени компьютера

В данном примере описывается процедура определения имени локального компьютера. Пример сконфигурирован для изображенной ниже кнопки *Button5* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button5

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    #pragma code ("Kernel32.DLL");
    BOOL GetComputerNameA(LPSTR ComputerName, LPDWORD pdwSize);
    #define MAX_COMPUTERNAME_LENGTH 15
    #pragma code();

    BOOL bRet = FALSE;
    char szComputerName[MAX_COMPUTERNAME_LENGTH + 1];
    DWORD dwSize = MAX_COMPUTERNAME_LENGTH + 1;

    bRet = GetComputerNameA(szComputerName, &dwSize);

    //check return value
    if (bRet == FALSE)
    {
        printf("\r\nComputerName:\r\nUnknown Computer\r\n");
        return;
    }

    //display project file
    printf("\r\nComputerName:\r\n%s\r\n", szComputerName);
}
}
```

- В первой части подключается DLL Windows Kernel32. Так как будет использована только одна функция этой DLL, она объявляется вручную. Кроме того, описывается символьная константа для задания максимальной длины имени компьютера.
- Затем описывается и инициализируется переменная *bRet* типа *BOOL*.
- Далее описывается строковая переменная *szComputerName* для размещения имени компьютера. В дополнение описывается переменная типа *DWORD*, инициализируемая длиной ранее созданной строковой переменной.
- Имя локального компьютера определяется функцией Windows *GetComputerNameA()*. Это имя записывается в переданную строковую переменную *szComputerName*.
- Далее проверяется значение, возвращаемое функцией Windows *GetComputerNameA()*.
- Затем полученное имя компьютера выводится на печать.

4.9.6 Пример 6 — Определение имени пользователя

В данном примере описывается процедура определения имени текущего зарегистрированного пользователя в *Windows NT*. Пример сконфигурирован для изображенной ниже кнопки *Button6* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button6

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    #pragma code ("advapi32.DLL");
    BOOL GetUserNameA(LPSTR UserName, LPDWORD pdwSize);
    #define UNLEN 256
    #pragma code();

    BOOL bRet = FALSE;
    char szUserName[UNLEN + 1];
    DWORD dwSize = UNLEN + 1;

    bRet = GetUserNameA(szUserName, &dwSize);

    //check return value
    if (bRet == FALSE)
    {
        printf("\r\nUserName:\r\nUnknown User\r\n");
        return;
    }

    //display project file
    printf("\r\nUserName:\r\n%s\r\n", szUserName);
}
```

- В первой части подключается DLL Windows *advapi32*. Так как будет использована только одна функция этой DLL, она объявляется вручную. Кроме того, описывается символьная константа для задания максимальной длины имени пользователя.
- Затем описывается и инициализируется переменная *bRet* типа *BOOL*.
- Далее описывается строковая переменная *szUserName* для размещения имени пользователя. В дополнение описывается переменная типа *DWORD*, инициализируемая длиной ранее созданной строковой переменной.
- С помощью функции Windows *GetUserNameA()* определятся имя текущего зарегистрированного пользователя в *Windows NT*. Это имя записывается в переданную строковую переменную *szUserName*.
- Далее проверяется значение, возвращаемое функцией Windows *GetUserNameA()*.
- Затем полученное имя пользователя выводится на печать.

4.10 Windows API

Примеры, описанные в данной главе, находятся на экранных формах `сс_0_startpicture_00.PDL` и `сс_2_keyboard_01.PDL` проекта WinCC Project_C_Course.

Программный интерфейс Windows

В проектах WinCC помимо WinCC API можно также использовать весь Windows API. Это предоставляет практически неограниченный доступ к системе.

Приведенные ниже примеры дают общее представление о таких возможностях. Они демонстрируют основные принципы использования Windows API. Тем не менее, это отнюдь не исчерпывающий курс по Windows API.

Функции Windows API располагаются в различных DLL, так же как функции WinCC API. Объявления этих функций находятся в различных заголовочных файлах.

Подключение DLL Windows выполняется таким же образом, как и подключение DLL WinCC. Следующий фрагмент кода иллюстрирует подключение DLL.

```
#pragma code ("comdlg32.dll")
#include "comdlg.h"
#pragma code()
```

4.10.1 Пример 1 — Установка свойств окна

Данный пример иллюстрирует возможность изменения свойств окна Windows. В нем производится изменение заголовка и формы окна среды исполнения. Пример связан с событием *Event (Событие)* → *Miscellaneous (Разное)* → *Open Picture (Открытие экранной формы)* стартовой экранной формы *cc_0_startpicture_00.PDL*.



Процедура Си, связанная со стартовой экранной формой

```
#include "apdefap.h"
void OnOpenPicture(char* lpszPictureName, char* lpszObjectName, char* lpszI
{
    //get handle of runtime window
    HWND hWnd = NULL;
    hWnd = FindWindow(NULL, "WinCC-Runtime - ");

    //set text of runtime window
    SetWindowText(hWnd, "WinCC C-Course");
    //set position and size of runtime window
    SetWindowPos(hWnd, HWND_TOP, 0, 0, 1024, 768, 0);

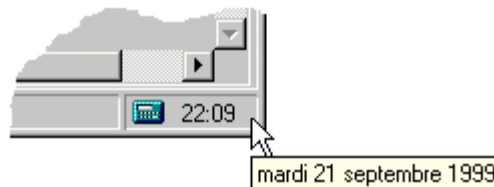
    //set active the first chapter
    SetTagByte("U08i_org_bar_1", 0);

    //
    CreateExternalTags();
}
```

- Функции Windows, используемые в данном примере, уже известны проекту WinCC. Следовательно, необходимость загрузки каких-либо DLL Windows отсутствует.
- В первой части описывается переменная типа *HWND* и инициализируется значением *NULL*. Эта переменная — так называемый описатель окна — представляет собой указатель, ссылающийся на окно Windows.
- Описатель окна может быть получен по заголовку этого окна с помощью функции Windows *FindWindow()*. Описатель окна среды исполнения WinCC можно определить в случае, если это окно имеет заданный по умолчанию заголовок
- Заголовок окна среды исполнения можно изменить при помощи функции Windows *SetWindowText()*. В данном примере в качестве заголовка указывается *WinCC C-Course*.
- Координаты и размеры отображаемого окна среды исполнения можно изменить при помощи функции Windows *SetWindowPos()*. В данном примере окно перемещается в левый верхний угол экрана (координаты 0/0) и растягивается до размера 1024 на 768.
- Остальные выражения приведенного выше фрагмента кода осуществляют инициализационные процедуры на имеющие отношения к данному примеру.

4.10.2 Пример 2 — Считывание системного времени

Данный пример показывает, как можно считывать и отображать системные дату и время. Пример привязан к экранной форме *cc_0_startpicture_00.PDL*.



Процедура Си, связанная со статическим текстом Time

```
#include "apdefap.h"
char* _main(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
#pragma code("kernel32.dll")
VOID GetLocalTime(LPSYSTEMTIME lpSystemTime);
#pragma code()

SYSTEMTIME sysTime;
char szTime[6] = "";

GetLocalTime(&sysTime);

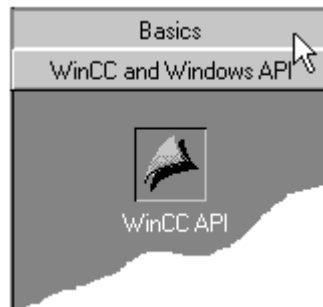
sprintf(szTime, "%02d:%02d", sysTime.wHour, sysTime.wMinute);

return szTime;
}
```

- В первой части подключается DLL Windows Kernel32. Так как будет использована только одна функция этой DLL, она объявляется вручную.
- Затем описывается переменная *sysTime* типа *SYSTEMTIME*. Это структурная переменная для хранения системного времени.
- Далее описывается строковая переменная *szTime* для размещения текущего времени в формате hh:mm.
- Текущее системное время считывается при помощи функции Windows *GetLocalTime()* и заносится в переменную *sysTime*.
- После этого посредством функции *sprintf()* текущее системное время приводится к формату hh:mm, после чего полученная строка используется как возвращаемое значение функцией. Другое процедура Си создается аналогичным образом для свойства *Property (Свойство)* → *Miscellaneous (Разное)* → *Tooltip Text (Текст подсказки)*. Это процедура Си генерирует текущую дату.
- Функция выполняется с периодичностью *1s*.

4.10.3 Пример 3 — Воспроизведение звуковых файлов

Данный пример показывает, как можно воспроизводить звуковые файлы. Воспроизведение звукового файла производится при переключении между навигационными панелями *Basics* и *WinCC and Windows API*. Пример связан с кнопкой *Button1* экранной формы *cc_2_keyboard_01.PDL*.



Функция проекта `CC_PlaySound()`

```
#include "apdefap.h"

void CC_PlaySound(char* lpszSoundFile)
{
    #pragma code("winmm.dll")
    BOOL PlaySound(LPCTSTR lpszSound, HMODULE hModule, DWORD dwSound);
    #define SND_FILENAME 0x00020000L
    #define SND_ASYNC 0x0001
    #pragma code()

    BOOL bRet = FALSE;
    char szProjectPath[_MAX_PATH];
    char szSoundPath[_MAX_PATH];

    GetProjectPath(szProjectPath);

    sprintf(szSoundPath, "%sSound\\%s", szProjectPath, lpszSoundFile);

    bRet = PlaySound(szSoundPath, NULL, SND_FILENAME|SND_ASYNC);

    if (bRet == FALSE)
    {
        MessageBeep((WORD)-1);
    }
}
```

- В первой части включается заголовочный файл *apdefap.h*. В результате этого из данной функции проекта могут вызываться другие функции проекта.
- В заголовке функции в качестве аргумента описывается строковая переменная. Она используется для передачи имени звукового файла, подлежащего воспроизведению.
- В следующей части подключается DLL Windows *winmm*. Так как будет использована только одна функция этой DLL, она объявляется вручную. Кроме того, описываются две символьные константы.

- *Функция проекта* предполагает наличие в каталоге проекта подкаталога *Sound*. В данном подкаталоге хранятся используемые в проекте звуковые файлы. Путь к искомому звуковому файлу составляется из пути проекта, имени каталога звуковых файлов и имени самого звукового файла. Этот путь будет помещен в переменную *szSoundPath*.
- Звуковой файл воспроизводится с помощью функции Windows *PlaySound()*. Если воспроизведение звукового файла невозможно, то вместо этого функцией Windows *MessageBeep()* будет сгенерирован короткий сигнал.

4.10.4 Пример 4 — Запуск программы

Данный пример иллюстрирует процедуру запуска программы. Для этой цели используется уже существующая стандартная функция, использующая Windows API. Пример приведен на экранной форме *cc_0_startpicture_00.PDL*.



Стандартная функция ProgramExecute()

```
unsigned int ProgramExecute( char* Program_Name )
{
    // This function will start any Windows Programm
    // if return value > 31 the programm started successfully

    return ( WinExec( Program_Name,
                     SW_SHOWNORMAL ) );
}
```

- *Стандартная функция ProgramExecute()* просто перенаправляет переданные ей аргументы в функцию Windows *WinExec()*. Значение, возвращаемое функцией *WinExec()* направляется функции, вызвавшей *ProgramExecute()*. Если запуск программы произведен успешно, возвращаемое значение больше 31.

Процедура Си, связанная с графическим объектом Execute

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    ProgramExecute("calc.exe");
}
```

- При помощи *стандартной функции ProgramExecute()*, производится запуск программы *calc.exe*. Это программа Windows “Калькулятор”. Путь не указывается, т.к. для программ, расположенных в каталоге Windows, в этом нет необходимости.

4.11 Стандартные диалоги

В проекте WinCC Project_C_Course примеры, относящиеся к стандартным диалогам, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называемой `ss_9_example_12.PDL`.



Standard Dialogboxes

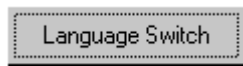
Общие сведения

Обычная процедура для создания диалога в WinCC состоит из формирования кадра WinCC и последующего его отображения в окне кадра (Picture Window). Помимо этого, существует возможность создания стандартных диалогов в процедурах Си или других функциях. В этом случае можно использовать как стандартные диалоги WinCC, так и диалоги Windows.

В данной главе демонстрируется применение некоторых стандартных диалогов. Существует ряд стандартных диалогов, которые здесь рассматриваться не будут. Сведения о них можно найти в WinCC ODK и документации по Windows API.

4.11.1 Пример 1 — Переключение языка

Приведенный ниже пример демонстрирует возможности использования стандартного диалога переключения языка в WinCC. Пример сконфигурирован для изображенной ниже кнопки *Button1* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button1

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    HWND hwndParent = NULL;
    DWORD dwFlags = 0;
    DWORD dwSetLocaleIDs[3] = { 0x0409, 0x0407, 0x040C };
    UINT  uSetIDArraySize = 3;
    DWORD dwGetLocaleID;
    BOOL  bRet;
    CMN_ERROR Error;

    hwndParent = FindWindow(NULL, "WinCC C-Course");

    //set cs language (dwGetLocaleID contains selected language ID)
    bRet = DMShowLanguageDialog(hwndParent, dwFlags, dwSetLocaleIDs,
                               uSetIDArraySize, &dwGetLocaleID, &Error);

    if (bRet == FALSE)
    {
        printf("\r\nError in DMShowLanguageDialog()\r\n"
              "\t%s\r\n", Error.szErrorText);
        return;
    }

    //set rt language
    bRet = SetLanguage(dwGetLocaleID);

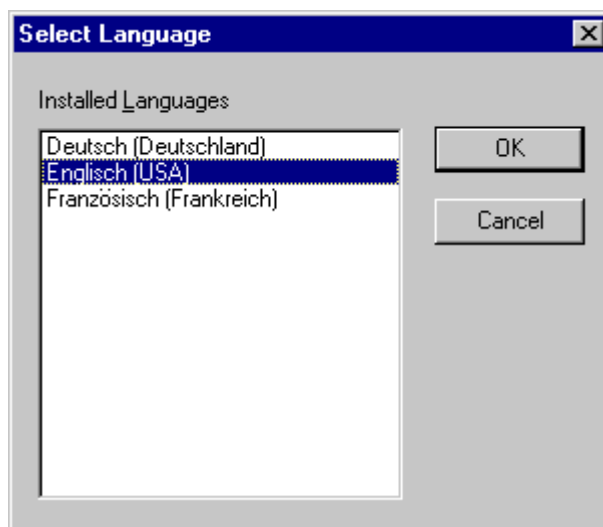
    if (bRet == FALSE)
    {
        printf("\r\nError in SetLanguage()\r\n");
        return;
    }
}
```

- В первой части описываются используемые переменные. Помимо прочих описывается массив идентификаторов (ID) трех используемых языков.
- По заголовку окна с помощью функции Windows *FindWindow()* определяется описатель окна среды исполнения. Обратите внимание, что указанный в данном примере заголовок окна не совпадает с заголовком окна среды исполнения, устанавливаемым по умолчанию.
- Стандартный диалоговое окно переключения языка отображается при помощи функции API *DMShowLanguageDialog()*. Этой функции передается массив идентификатор языков, доступных для выбора. В передаваемую по указателю переменную *dwGetLocaleID* функция записывает идентификатор выбранного языка.

- Проверяется значение, возвращаемое функцией API *DMShowLanguageDialog()*. Одним из возможных возвращаемых значений является *FALSE* означающее, что пользователь закрыл диалоговое окно нажатием на *Cancel*.
- Используемый здесь диалоговое окно переключает только язык CS. Для того чтобы переключить язык RT, необходимо использовать *внутреннюю функцию SetLanguage()*. Этой функции передается идентификатор выбранного языка.

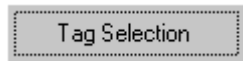
Диалоговое окно выбора языка

Исполнение описанной выше процедуры Си приводит к отображению следующего диалогового окна:



4.11.2 Пример 2 — Выбор тега

Приведенный ниже пример демонстрирует возможности использования стандартного диалога выбора тега в WinCC. Значение выбранного в диалоговом окне тега отображается в *поле ввода/вывода*. Пример сконфигурирован для изображенной ниже кнопки *Button2* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button2

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    //include file with the LinkType definitions
    #include "trigger.h"

    BOOL bRet;
    char szProjectFile[_MAX_PATH+1];
    CMN_ERROR Error;
    HWND hwndParent = NULL;
    DM_VARKEY dmVarKey;
    LINKINFO link;

    //////////////////////////////////////
    //select tag
    if ( DMGetRuntimeProject(szProjectFile,_MAX_PATH+1,&Error) == FALSE)
    {
        printf("\r\nError in DMGetRuntimeProject()\r\n"
            "\t%s\r\n",Error.szErrorText);
        return;
    }

    hwndParent = FindWindow(NULL, "WinCC C-Course");

    if ( DMShowVarDatabase(szProjectFile,hwndParent,NULL,NULL,
        &dmVarKey,&Error) == FALSE)
    {
        printf("\r\nError in DMShowVarDatabase()\r\n"
            "\t%s\r\n",Error.szErrorText);
        return;
    }

    //////////////////////////////////////
    //display tag selection
    SetText(lpszPictureName, "TagName", dmVarKey.szName);

    link.LinkType = BUBRT_LT_VARIABLE_DIRECT;
    link.dwCycle = 0;
    strcpy(link.szLinkName, dmVarKey.szName);

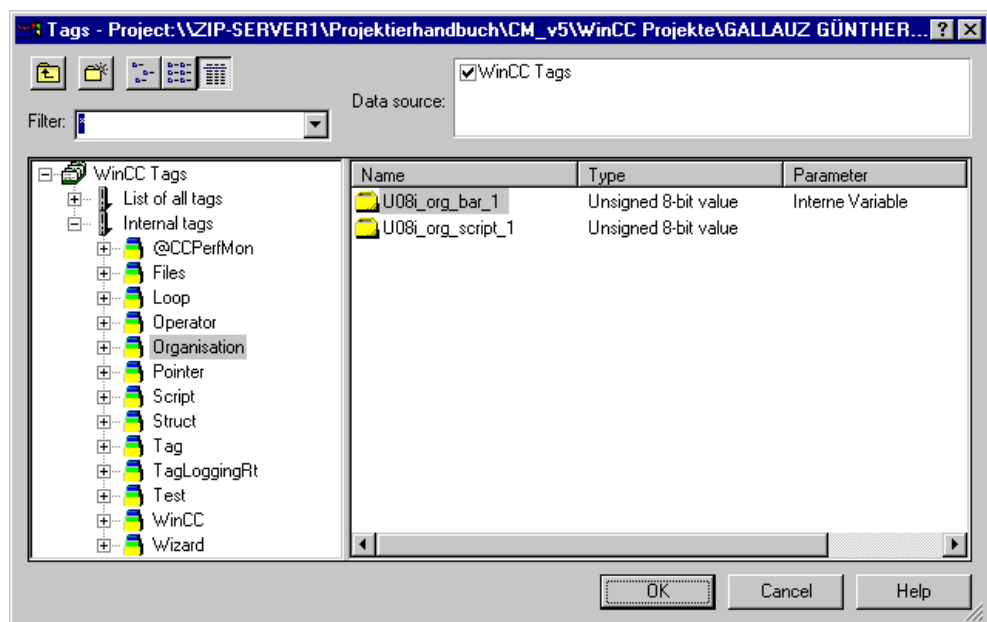
    PDLRTSetLink(0, lpszPictureName, "TagValue", "OutputValue",
        &link, NULL, NULL, &Error);
}
}
```

- В первой части включается заголовочный файл *trigger.h*. В этом файле содержатся описания используемых в данном примере символьных констант.
- В следующей части описываются используемые переменные. Помимо прочих описываются переменные *dmVarKey* для размещения данных о выбранном в диалоговом окне теге WinCC и *link* для хранения информации о связи с тегом.
- Название проекта определяется с помощью функции API *DMGetRuntimeProject()*.

- По заголовку окна с помощью функции Windows *FindWindow()* определяется описатель окна среды исполнения.
- Диалоговое окно выбора тега отображается при помощи функции API *DMShowVarDatabase()*. В передаваемую по указателю переменную *dmVarName* функция записывает информацию о выбранном в данном диалоге теге WinCC.
- Если был выбран какой-либо тег, его имя показывается в *статическом тексте*, а значение — в *поле ввода/вывода*.

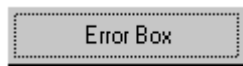
Диалоговое окно выбора тега

Исполнение описанной выше процедуры Си приводит к отображению следующего диалогового окна:



4.11.3 Пример 3 — Диалоговое окно сообщения об ошибке

Приведенный ниже пример демонстрирует возможности использования стандартного Windows-диалогового окна сообщения об ошибке. Пример сконфигурирован для изображенной ниже кнопки *Button3* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



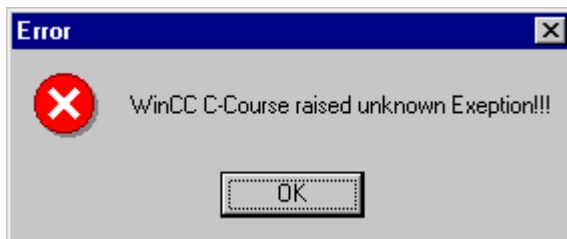
Процедура Си, связанная с кнопкой Button3

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProper
{
    HWND hWnd = NULL;
    hWnd = FindWindow(NULL, "WinCC C-Course");
    MessageBox(hWnd, "WinCC C-Course raised unknown Exception!!!",
               "Error", MB_OK | MB_ICONSTOP | MB_APPLMODAL);
}
```

- В первой части описывается переменная *hWnd* типа *HWND*. В эту переменную заносится описатель окна среды исполнения при помощи функции Windows *FindWindow()*.
- Посредством функции Windows *MessageBox()* открывается диалоговое окно сообщения об ошибке. Второй и третий аргументы задают, соответственно, текст сообщения и заголовок диалога. Четвертый аргумент определяет внешний вид и поведение диалогового окна. Диалоговое окно содержит только кнопку *OK* (*MB_OK*), символ ошибки (*MB_ICONSTOP*) и является модальным (*MB_APPLMODAL*). Таким образом, пользователь должен подтвердить сообщение об ошибке, перед тем как продолжить работу.

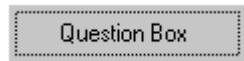
Error Box

Исполнение описанного выше процедуры Си приводит к отображению следующего диалогового окна:



4.11.4 Пример 4 — Диалоговое окно вопроса

Приведенный ниже пример демонстрирует возможности использования стандартного Windows-диалогового окна вопроса и последующих процедур, выполняемых в зависимости от того, какая кнопка была нажата в диалоге. Пример сконфигурирован для изображенной ниже кнопки *Button4* в окне свойств объекта в *Event (Событие)*
 → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button4

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    HWND hWnd = NULL;
    int iRet;

    hWnd = FindWindow(NULL, "WinCC C-Course");

    iRet = MessageBox(hWnd, "Do you want to do something?", "Question",
        MB_YESNO|MB_ICONQUESTION|MB_APPLMODAL);

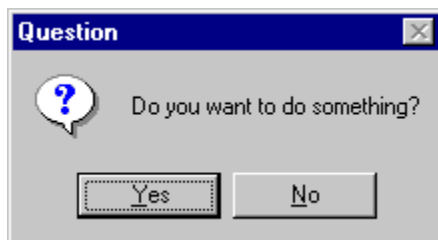
    printf("\r\nExample 3\r\n");

    if (iRet == IDYES)
    {
        printf("User selected YES button\r\n");
    }
    else // if (iRet == IDNO)
    {
        printf("User selected NO button\r\n");
    }
}
```

- В первой части описывается переменная *hWnd* типа *HWND*. Кроме того, описывается переменная *iRet* типа *int*.
- По заголовку окна с помощью функции Windows *FindWindow()* определяется описатель окна среды исполнения.
- Посредством функции Windows *MessageBox()* открывается диалоговое окно вопроса. Четвертый аргумент определяет внешний вид и поведение диалогового окна. Диалоговое окно содержит только кнопки *Yes* и *No* (*MB_YESNO*), символ вопроса (*MB_ICONQUESTION*) и является модальным (*MB_APPLMODAL*). Значение, возвращаемое функцией, заносится в переменную *iRet*.
- В последней части анализируется значение, возвращаемое функцией. Если диалоговое окно было закрыто нажатием на кнопку *Yes*, возвращаемое значение — *IDYES*, если диалоговое окно было закрыто нажатием на кнопку *No*, возвращаемое значение — *IDNO*.

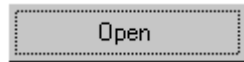
Диалоговое окно вопроса

Исполнение описанного выше процедуры Си приводит к отображению следующего диалогового окна:



4.11.5 Пример 5 — Стандартное диалоговое окно выбора файла

Приведенный ниже пример демонстрирует возможности использования стандартного Windows-диалогового окна выбора файла. Пример сконфигурирован для изображенной ниже кнопки *Button5* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button5

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    #pragma code ("comdlg32.dll")
    #include "comdlg.h"
    #pragma code()

    BOOL bRet;
    OPENFILENAME ofn;
    char szFilter[] = "Textfiles|*.txt|All Files|*.*|";
    char* psz;
    char szFile[_MAX_PATH+1];
    char szInitialDir[_MAX_PATH+1] = "C:\\";

    ofn.lStructSize = sizeof(OPENFILENAME);

    ofn.hwndOwner = FindWindow(NULL, "WinCC C-Course");

    for (psz = szFilter; *psz; psz++)
    {
        if (*psz == '|')
        {
            *psz = 0;
        }
    }
    ofn.lpstrFilter = szFilter;

    ofn.lpstrFile = szFile;
    ofn.nMaxFile = _MAX_PATH+1;

    GetProjectPath(szInitialDir); //if function fails initial
                                //directory is "C:\\";
    ofn.lpstrInitialDir = szInitialDir;

    bRet = GetOpenFileName(&ofn);

    if (bRet == FALSE)
    {
        printf("\r\nError in GetOpenFileName()\r\n");
        return;
    }

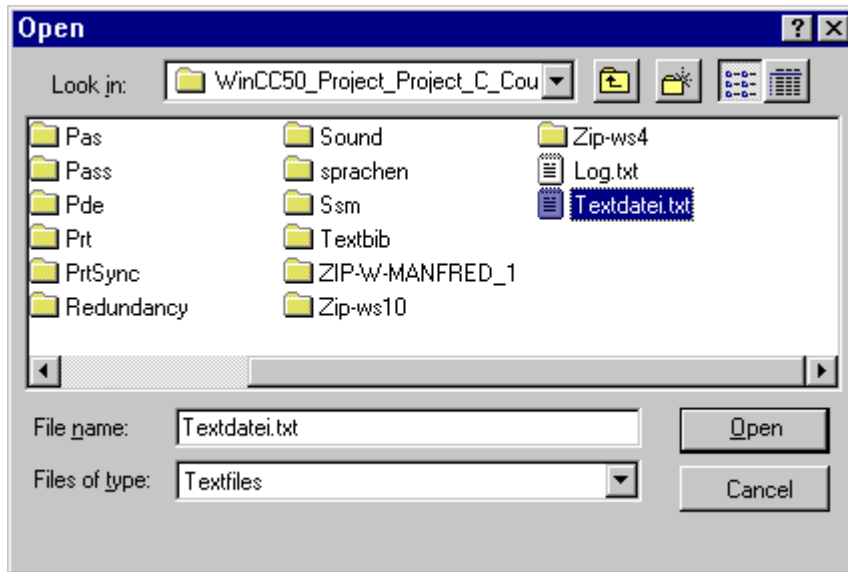
    printf("\r\nSelected File (Path+Name)\r\n%s\r\n", ofn.lpstrFile);
}
}
```

- В первой части подключается DLL Windows *comdlg32*.
- В следующей части описываются требуемые переменные. Помимо прочих описывается переменная *ofn* структурного типа *OPENFILENAME*.
- В следующей части переменная *ofn* заполняется данными.

- Переменная *ofn* передается функции Windows *GetOpenFileName()*. Эта функция открывает стандартное диалоговое окно выбора файла. Имя выбранного пользователем файла заносится в переменную *ofn*. Имя выбранного файла выводится на печать.

Стандартное диалоговое окно выбора файла

Исполнение описанного выше процедуры Си приводит к отображению следующего диалогового окна:



Дополнительные примеры

Последующие примеры этой главы так же, как и пример 5, имеют отношение к стандартным диалогам.

В примере 6 демонстрируется использование диалогового окна *Save As*.

В примере 7, создается *функция проекта GetFileName()*, облегчающая работу со стандартными файловыми диалогам. Эта функция в зависимости от передаваемой символьной константы отображает диалоговое окно *Open* или *Save As*. Для этого используются символьные константы *GFN_OPEN* и *GFN_SAVE*.

4.12 Файлы

В проекте WinCC Project_C_Course примеры, относящиеся к работе с файлами, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называющейся `ss_9_example_13.PDL`.



Files

Открытие файла

В Си файл вне зависимости от его содержания рассматривается как набор символов. Перед тем, как использовать файл в процедуре Си или другой функции, его необходимо открыть. По завершении работы с файлом его следует закрыть. Файл открывается посредством функции `fopen()`. В приведенном ниже фрагменте кода демонстрируется использование этой функции.

```
FILE* pFile = NULL;
pFile = fopen("C:\\\\Test.txt", "r");
```

Для того чтобы иметь возможность работать с файлом, необходимо определить указатель, ссылающийся на него. Для этого предусмотрен тип данных `FILE*`. Функция `fopen()` возвращает указатель, ссылающийся на открытый файл, или `NULL`, если открыть файл не удалось. В качестве первого аргумента функции `fopen()` необходимо передать имя файла с указанием пути к нему. Второй параметр задает режим доступа к файлу (например, только для чтения). Значения, определяющие режим доступа, перечислены в приведенной ниже таблице.

Режим	Описание
r	Открывает файл для чтения. Значение <i>NULL</i> возвращается в том случае, если файл не существует, или отказано в доступе на чтение.
w	Открывает файл для записи. Значение <i>NULL</i> возвращается в том случае, если файл не существует, или отказано в доступе на запись.
a	Открывает файл для продолжения записи в конец. Если файл не существует, он создается. Значение <i>NULL</i> возвращается в том случае, если файл не может быть создан, или в него нельзя осуществить запись.
r+	Открывает файл для попеременного чтения и записи. Значение <i>NULL</i> возвращается в том случае, если файл не существует, или отказано в доступе на чтение или запись.
w+	Создает файл для попеременного чтения и записи. Если файл уже существует, он затирается. Значение <i>NULL</i> возвращается в том случае, если недостаточно прав для осуществления перечисленных процедур.
a+	Открывает файл для чтения или продолжения записи в конец. Если файл не существует, он создается. Значение <i>NULL</i> возвращается в том случае, если отсутствуют права на чтение или запись.

Заккрытие файла

По завершении работы с файлом его необходимо закрыть. Файл закрывается посредством функции `fclose()`. В приведенном ниже фрагменте кода демонстрируется ее использование. Функции передается указатель, ссылающийся на файл, который необходимо закрыть.

```
fclose(pFile);
```

Запись и чтение файла

Для записи в файл существует функция, аналогичная `printf()`. Это функция `fprintf()`. Функция `fprintf()` работает аналогично функции `printf()`. При этом, однако, вывод осуществляется не в окно диагностики глобальных сценариев, а в файл. Первым аргументом функции передается указатель на этот файл. В приведенном ниже фрагменте кода демонстрируется использование функции `fprintf()`.

```
fprintf(pFile, "%d\r\n%f\r\n", iValue, dValue);
```

Для чтения из файла предусмотрена функция `fscanf()`. Структура функции `fscanf()` такая же, как у `fprintf()`. Однако, вместо перечисления переменных, значения которых записываются в файл, указываются адреса переменных, в которые заносятся считываемые из файла значения.

4.12.1 Пример 1 — Запись данных

Приведенный ниже пример показывает, как осуществлять запись в файл. Пример сконфигурирован для изображенной ниже кнопки *Button1* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button1

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    FILE* pFile = NULL;

    char szFile[_MAX_PATH+10];

    int iData;
    float fData;

    //get project path
    if (GetProjectPath(szFile) == FALSE)
    {
        printf("\r\nError in GetProjectPath()\r\n");
        return;
    }

    //create file name
    strcat(szFile, "Data.txt");

    //open or create file to write
    pFile = fopen(szFile, "w+");

    //check return value of fopen()
    if (pFile == NULL)
    {
        printf("\r\nError in fopen()\r\n");
        return;
    }

    //get data to write
    iData = GetTagSDWord("S32i_course_file_1");
    fData = GetTagFloat("F32i_course_file_1");

    //write data
    fprintf(pFile, "%d\r\n%f\r\n", iData, fData);

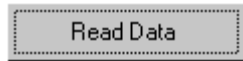
    fclose(pFile);

    //output in diagnostics window
    printf("\r\nData written in file:\r\n\t%d\r\n\t%f\r\n",
        iData, fData);
}
```

- В первой части описываются используемые переменные. Помимо прочих описывается и инициализируется переменная типа *FILE**.
- Путь проекта определяется при помощи *функции проекта GetProjectPath()*.
- Затем с помощью функции *strcat()* формируется путь к создаваемому файлу. Этот путь передается функции *fopen()*. Посредством данной функции требуемый файл открывается или создается.
- В следующей части данные, которые необходимо записать, извлекаются из тегов WinCC.
- При помощи функции *fprintf()* данные записываются в файл. Затем файл закрывается.

4.12.2 Пример 2 — Чтение данных

Приведенный ниже пример показывает, как осуществлять чтение из файла. Считываемые данные записываются в теги WinCC. Пример сконфигурирован для изображенной ниже кнопки *Button2* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Процедура Си, связанная с кнопкой Button2

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    FILE* pFile = NULL;

    char szFile[_MAX_PATH+10];

    int iData;
    float fData;

    //get project path
    if (GetProjectPath(szFile) == FALSE)
    {
        printf("\r\nError in GetProjectPath()\r\n");
        return;
    }

    //create file name
    strcat(szFile, "Data.txt");

    //open file to read
    pFile = fopen(szFile, "r+");

    //check return value of fopen()
    if (pFile == NULL)
    {
        printf("\r\nError in fopen()\r\n");
        return;
    }

    //read data
    fscanf(pFile, "%d\r\n%f\r\n", &iData, &fData);

    fclose(pFile);

    //set data
    SetTagSDWord("S32i_course_file_1", iData);
    SetTagFloat("F32i_course_file_1", fData);

    //output in diagnostics window
    printf("\r\nData read from file:\r\n\t%d\r\n\t%f\r\n",
        iData, fData);
}
}
```

- В первой части описываются используемые переменные. Помимо прочих описывается и инициализируется переменная типа *FILE**.
- Путь проекта определяется при помощи *функции проекта GetProjectPath()*.
- Затем с помощью функции *strcat()* формируется путь к создаваемому файлу. Этот путь передается функции *fopen()*. Посредством данной функции открывается требуемый файл.
- При помощи функции *fscanf()* данные считываются из файла. После этого файл закрывается.
- В следующей части считанные данные записываются в теги WinCC.

4.12.3 Пример 3 — Формирование отчета

Приведенный ниже пример показывает, как формировать файл отчета. Создается функция проекта, которой передается текст отчета. Пример сконфигурирован для изображенной ниже кнопки *Button3* в окне свойств объекта в *Event (Событие)* → *Mouse (Мышь)* → *Mouse Action (Процедура мыши)*.



Project Function LogText()

```
#include "apdefap.h"

BOOL LogText(char* lpszLogText)
{
    FILE* pFile = NULL;

    char szFile[_MAX_PATH+10];

    //get project path
    if (GetProjectPath(szFile) == FALSE)
    {
        printf("\r\nError in GetProjectPath()\r\n");
        return FALSE;
    }

    //create file name
    strcat(szFile, "Log.txt");

    //open or create file to append
    pFile = fopen(szFile, "a+");

    //check return value of fopen()
    if (pFile == NULL)
    {
        printf("\r\nError in fopen()\r\n");
        return FALSE;
    }

    //append data
    fprintf(pFile, "%s - %s\r\n", GetLocalTimeString(), lpszLogText)

    fclose(pFile);

    return TRUE;
}
```

- Функции передается строковая переменная, которая добавляется в конец файла отчета.
- В первой части описываются используемые переменные. Помимо прочих описывается и инициализируется переменная типа *FILE**.
- Путь проекта определяется при помощи *функции проекта GetProjectPath()*.
- Затем с помощью функции *strcat()* формируется путь к создаваемому файлу. Этот путь передается функции *fopen()*. Посредством данной функции открывается требуемый файл.
- При помощи функции *fprintf()* переданная часть текста отчета добавляется в файл. Перед каждой новой записью в файл записывается текущее системное время, получаемое с помощью *стандартной функции GetLocalTimeString()*. Затем файл закрывается.

Процедура Си, связанная с кнопкой Button3

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszProperty)
{
    if (LogText(GetTagChar("T08i_course_file_1")) == FALSE)
    {
        printf("\r\nError in LogText()\r\n");
    }
}
```

- В Си-действии содержимое текстового тега WinCC считывается и передается в ранее созданную функцию проекта *LogText()*. В результате, содержимое этого тега WinCC заносится в файл отчета.

4.13 Динамический мастер

В проекте WinCC Project_C_Course примеры, относящиеся к динамическому мастеру, можно открыть, щелкнув на изображенную ниже пиктограмму навигационной панели. Примеры находятся на экранной форме, называемой `ss_9_example_14.PDL`.




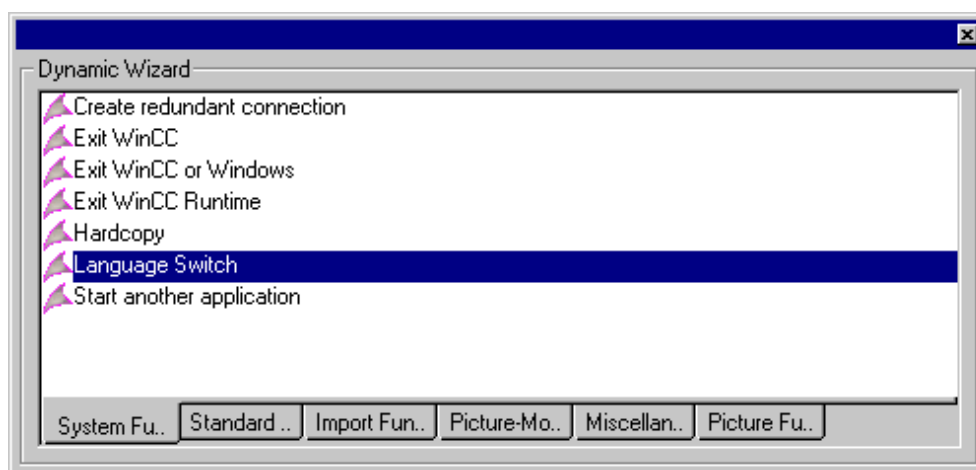
Dynamic Wizard

Общие сведения

Динамический мастер является дополнительной опцией графического дизайнера. Он упрощает для пользователя часто повторяющиеся процедуры по конфигурированию. Это ускоряет процесс создания проекта и уменьшает количество возможных ошибок. Динамический мастер представляет собой набор разнообразных динамических функций, поставляемых с WinCC. Их число может быть расширено пользовательскими функциями.

Работа с динамическим мастером

Динамический мастер отображается в графическом редакторе выбором пункта меню *View (Вид) → Toolbars (Панели инструментов) → Dynamic Wizard (Динамический мастер)*. Ниже показана структура динамического мастера. Уже существующие функции динамического мастера сгруппированы по нескольким закладкам. Для того чтобы запустить нужную функцию динамического мастера на нее нужно дважды щелкнуть мышью .



Функции динамического мастера состоят из нескольких следующих друг за другом диалоговых окон, заполняемых пользователем. К ним относятся начальное диалоговое окно, диалоговое окно триггера, диалоговое окно различных настроек и завершающее диалоговое окно, содержащее всю информацию о произведенных пользователем в предыдущих окнах настройках.

4.13.1 Создание функций динамического мастера

Для создания пользовательских функций *динамического мастера* предусмотрен отдельный редактор. Этот редактор располагается в каталоге *\bin*: программа *dynwizedit.exe*.

Каждая функция *динамического мастера* хранится в отдельном файле сценария. Для немецкого, английского и французского языков существуют отдельные файлы сценариев. Эти файлы сценариев хранятся в соответствии с используемым языком в следующих каталогах:

WinCC InstallationFolder\Wscripts\Wscripts.deu

WinCC InstallationFolder\Wscripts\Wscripts.enu

WinCC InstallationFolder\Wscripts\Wscripts.fra

После запуска редактора динамических мастеров (*Dynamic Wizard Editor*), в панели инструментов выберите язык, для которого создается динамический мастер.

Функция *динамического мастера* имеет фиксированную структуру. В рамках данного руководства приводятся два примера создания функций динамических мастеров. Файлы сценариев, в которых находятся эти примеры, располагаются в специально созданном подкаталоге *DynWiz* проекта *WinCC Project_C_Course*. Эти файлы следует скопировать в перечисленные выше папки, в которых располагаются стандартные файлы сценариев. После этого примеры можно открыть из редактора динамических мастеров.

Демонстрационный мастер Demo

В скрипт-файле *Demo.wnf* представлен *динамический мастер*, называющийся *Demo Wizard*. Этот мастер демонстрирует основные функции, позволяющие пользователю осуществлять удобный ввод данных. Однако этот *динамический мастер* не производит никаких реальных процедур.

Создание динамического мастера Motor

В скрипт-файле *Motor.wnf* приводится *динамический мастер*, называющийся *Making a Motor Dynamic (Придание двигателю динамических свойств)*. Этот мастер был специально создан для добавления динамических свойств пользовательскому объекту под названием *Motor* и не может использоваться для объектов любых других типов. Этот *пользовательский объект* хранится в библиотеке проекта *WinCC Project_C_Course* и может быть перенесен оттуда непосредственно на экранную форму. *Пользовательский объект Motor* связывается со структурным тегом WinCC типа *Motor* при помощи *динамического мастера Making a Motor Dynamic*. Точнее, для этого объекта создаются различные процедуры Си и связи с тегом. Предполагается, что существует внутренний текстовый тег WinCC *T08i_course_wiz_selected*. С помощью этого тега можно указать выбранный в текущий момент объект.

Компиляция скрипт-файлов

Полностью готовую функцию динамического мастера необходимо откомпилировать (пункт меню *Dynamic Wizard (Динамический мастер) → Compile Script (Компилировать скрипт)*) и затем сохранить. Для того чтобы использовать функцию *динамического мастера* в *графическом редакторе*, ее следует интегрировать в базу данных *динамических мастеров*. Это делается выбором пункта меню *Dynamic Wizard*

(Динамический мастер) → Reading Wizard Script (Считывание скрипта мастера).
Скрипт-файл, который необходимо считать, следует выбрать в появляющемся диалоговом окне.

4.13.2 Структура функции динамического мастера

Ниже объясняется назначение различных частей функции динамического мастера.

Подключение заголовочных файлов и DLL

Первая часть функции *динамического мастера* используется для подключения необходимых заголовочных файлов. Наиболее важным из подключаемых файлов является *dynamic.h*, в котором объявляются функции, относящиеся к пользовательскому интерфейсу динамического мастера. Здесь также подключаются все требуемые DLL Windows или WinCC API.

```
#include "dynamic.h"

#pragma code ("pdllcsapi.dll")
#include "pdllcsapi.h"
#pragma code ()
```

Описания, зависящие от языка

Если функция *динамического мастера* предусматривает использование нескольких языков, то для каждого из них необходимо создать отдельный скрипт-файл. По этой причине текстовые строки, зависящие от языка, следует описывать до текста самой программы. При этом скрипт-файл, созданный для какого-либо определенного языка, можно легко скопировать. В этом случае нужно будет модифицировать только часть функции, содержащую зависящие от языка описания.

```
////////////////////////////////////
//change only this strings to generate wizard scripts for other languages

#include "defenu.h"

char* DynWizGroupName = "WinCC C-Course";
char* DynWizDynamicName = "Make a Motor Dynamic";
char* DynWizToDoOption1 = "Select the desired Structure Tag:";
char* DynWizGenerateInfo = "The motor is made dynamic using the\r\n"
                           "structure tag\r\n%s\r\n.";

//
////////////////////////////////////
```

Список свойств

Существует возможность указать, что функция *динамического мастера* может использоваться только для объектов определенного типа. Это осуществляется указанием списка свойств объекта. Если объект имеет одно или более из перечисленных свойств, к нему может быть применена функция динамического мастера. Данная возможность была использована в *динамическом мастере Making a Motor Dynamic* для того, чтобы ограничить область его применения *пользовательскими объектами* типа *Motor*. Этот тип объекта имеет свойства *Manual (Вручную)* и *Selection (Выбор)*. Если указан пустой список свойств, функция динамического мастера может применяться к объектам любого типа. В любом случае список свойств должен присутствовать, даже если он пустой.

Функция обработки

Функция обработки — это функция, которая реализует реальные процедуры функции *динамического мастера* после нажатия кнопки *Finish*. Название этой функции следует указать в системном интерфейсе. Расширенная функция обработки представлена здесь в *динамическом мастере Making a Motor Dynamic*.

Функция конспекта Info

Функция конспекта сводит воедино все сделанные пользователем настройки и отображает их в кратком изложении в последнем диалоговом окне функции *динамического мастера*. Название этой функции следует указать в системном интерфейсе.

```
//this wizard can only be executed on the customized object "motor"
BEGIN_PROPERTY_SCHEME
{ "Hand", VT_BOOL },
{ "Selection", VT_BOOL },
END_PROPERTY_SCHEME
```

Системный интерфейс

В системном интерфейсе указываются различные свойства функции *динамического мастера*. Значения отдельных параметров разъясняются в следующем фрагменте кода.

```
BEGIN_DYNAMICS
{
    DynWizGroupName,           //group name
    DynWizDynamicName,        //dynamic name
    NULL,
    "logo16.bmp",             //use the default icon
    NULL,                       //no help string is used
    {
        "OnOption1",
        "OnOption2",
        NULL,
    },
    "OnGenerate",
    "OnShowGenerateInfo",
    {
        JCR_TRIGGERS,
        { NULL, NULL },
    },
},
END_DYNAMICS
```

- Первый параметр указывает, в какой закладке динамического мастера отображается функция.
- Второй параметр задает отображаемое название функции динамического мастера.
- В качестве третьего параметра всегда передается *NULL*.
- Четвертый параметр задает имя иконки, используемой в функции динамического мастера.

- Пятым параметром можно передать текст подсказки, описывающий функциональное предназначение и подробное описание динамического мастера.
- Шестой параметр задает список отдельных страниц настроек создаваемой функции. Список должен заканчиваться пунктом *NULL*. Можно создать не более пяти страниц настройки.
- Седьмой параметр указывает имя функции обработки, которая вызывается после нажатия на кнопку *Finish*.
- В качестве восьмого параметра задается имя функции конспекта, перечисляющей в последнем диалоговом окне все выбранные пользователем настройки.
- Восьмым параметром указывается список триггеров, которые следует перечислять в диалоге выбора. Для наиболее типичных случаев предусмотрены макросы, заполняющие этот список триггеров.

Глобальные переменные

Для каждого параметра, устанавливаемого в диалоговых окнах, необходимо описать глобальную переменную. Это гарантирует, что все задаваемые параметры известны всем создаваемым функциям и могут быть в них использованы.

```
//global tags
char g_MotorStructName[255] = "MotorStruct";
char* g_SelectedMotor      = "T08i_course_wiz_selected";
DWORD dwTypes[1];
```

Диалоговые окна настроек

Для каждого требуемого диалогового настроек должна быть определена отдельная функция. Названия этих функций следует указать в системном интерфейсе.

5 Приложение

Данное приложение содержит ряд тем, которые не были включены в *Руководство по конфигурации*.

5.1 Полезные советы

Дополнительные примеры конфигурации с помощью WinCC.

5.1.1 Форматированный ввод/вывод в поля ввода/вывода

Для форматированного отображения информации или для передачи форматированного введенного значения в ПЛК, должны быть сконфигурированы следующие процедуры:

Процедура в свойстве Output Value (Выводимое значение) поля ввода/вывода (важно: float, если требуются десятичные разряды):

```
Float a;  
a=GetTagFloat("DB21_DW1");  
return(a/100);
```

Процедура по событию Input Value (Вводимое значение) поля ввода/вывода (тег Var1 – беззнаковая 16-битная величина):

```
float a;  
a=GetInputValueDouble(IpszPictureName,IpszObjectName);  
SetTagFloat("Var1",a*100);
```

5.1.2 Специфичные для объекта процедуры на открытом кадре

Существуют приложения, где процедуры в свойствах одного или нескольких объектов кадра должны быть выполнены один раз при его открытии. Это решается путем назначения специфичной для объекта кадра процедуры в *Events (События)* → *Miscellaneous (Разное)* → *Open Picture (Открытие кадра)*.

Этот метод, однако, имеет один недостаток: действие должно производиться над объектами кадра, и, таким образом, имена объектов в процедуре должны быть определены заранее. Теперь этими объектами нельзя свободно манипулировать. Получается, что данное решение не является объектно-ориентированным. Существует возможность обойти эту проблему:

- Задайте внутренний тег (например, *dummy (пустой)*), который никогда не обновляется и не устанавливается. Укажите, что процедура должна запускаться по изменению этого тега. При открытии кадра во время исполнения, процедура отработает только один раз, так как тег *dummy* никогда не меняется.

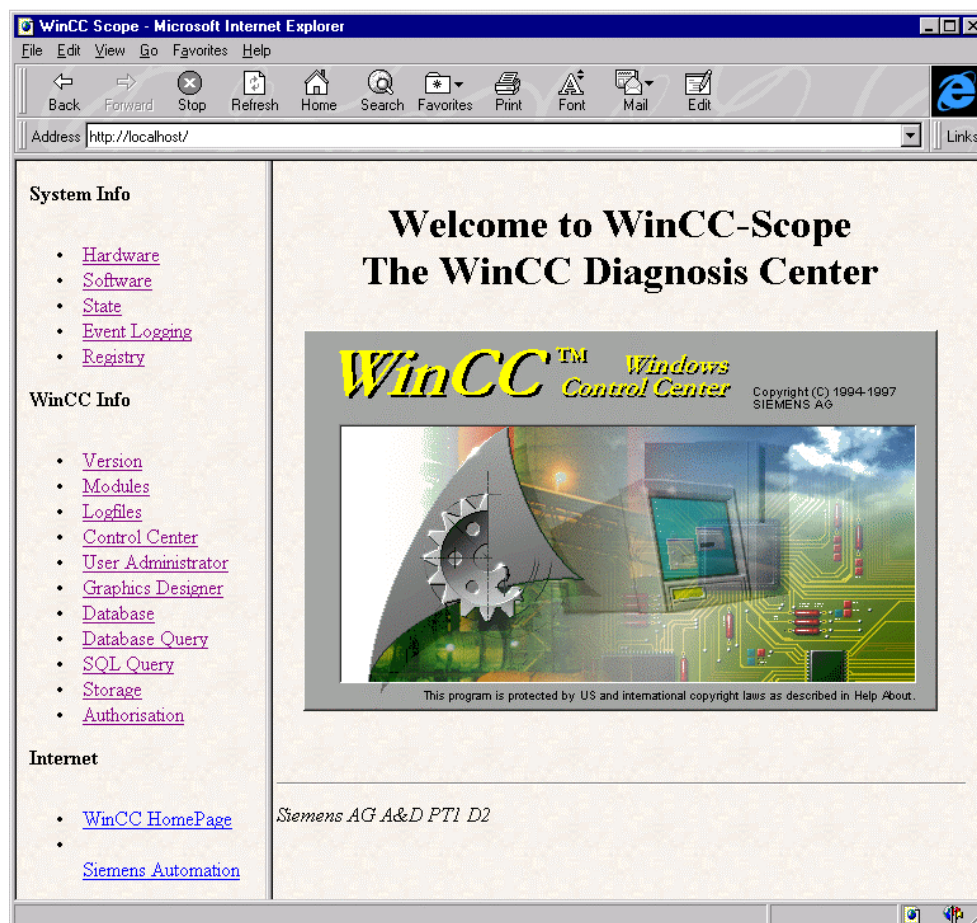
5.1.3 WinCC Scope

Общая информация

WinCC Scope - это средство для облегчения диагностики проектов WinCC. Оно предоставляет информацию о запущенном проекте и о конкретной компьютерной системе. Для работы с *Scope* требуется веб-браузер, например, Internet Explorer. Также должен быть установлен сетевой протокол TCP/IP.

Запуск и работа

По умолчанию *Scope* устанавливается вместе с WinCC. Перед использованием *Scope* должна быть запущена программа *WinCCDiagAgent.exe*, которая находится в каталоге *Siemens\WinCC\WinCCScope\bin*. Данная программа представляет собой простой сервер HTTP. После этого *Scope* может быть запущен из меню Start (Пуск). Общее описание работы *WinCC Scope* можно получить, следуя ссылке *How to use the new Diagnostics Interface (Как использовать новый интерфейс диагностики)*. Нажмите на ссылке <http://localhost> для запуска *Scope*. С помощью списка слева можно получить различную информацию. В разделе *System Info (Информация о системе)* находится общая информация о запрашиваемом компьютере, а в разделе *WinCC Info (Информация о WinCC)* можно найти информацию об активном в данный момент проекте WinCC.

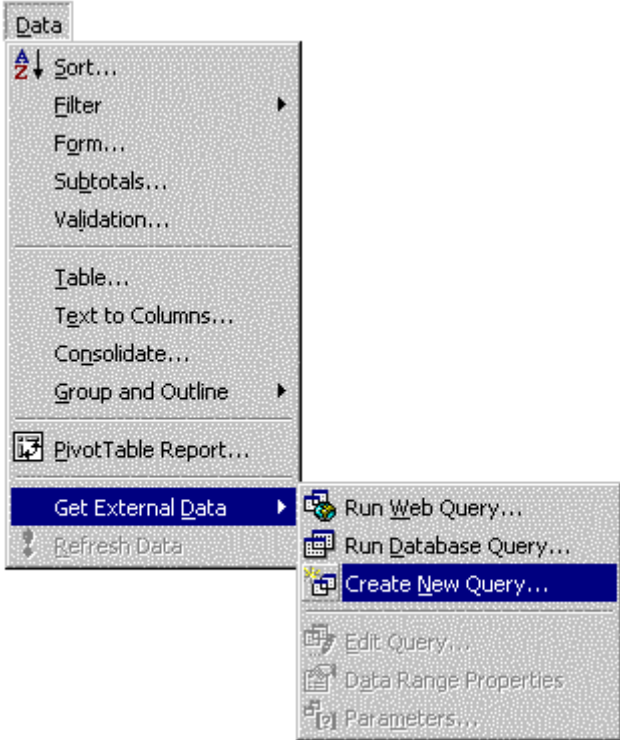


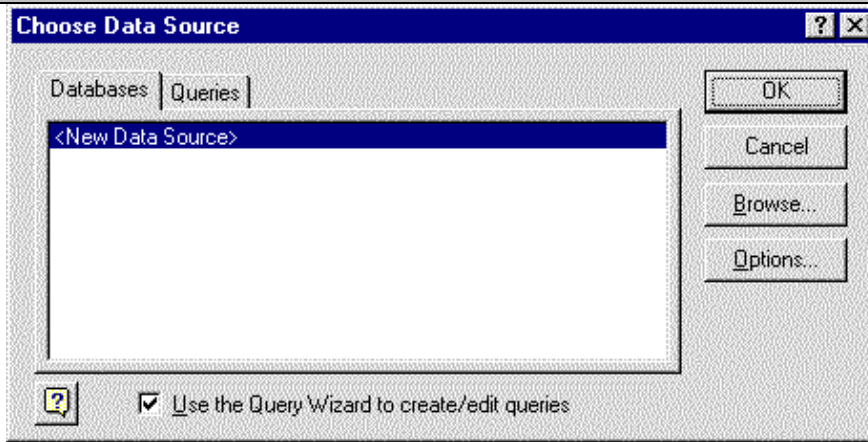
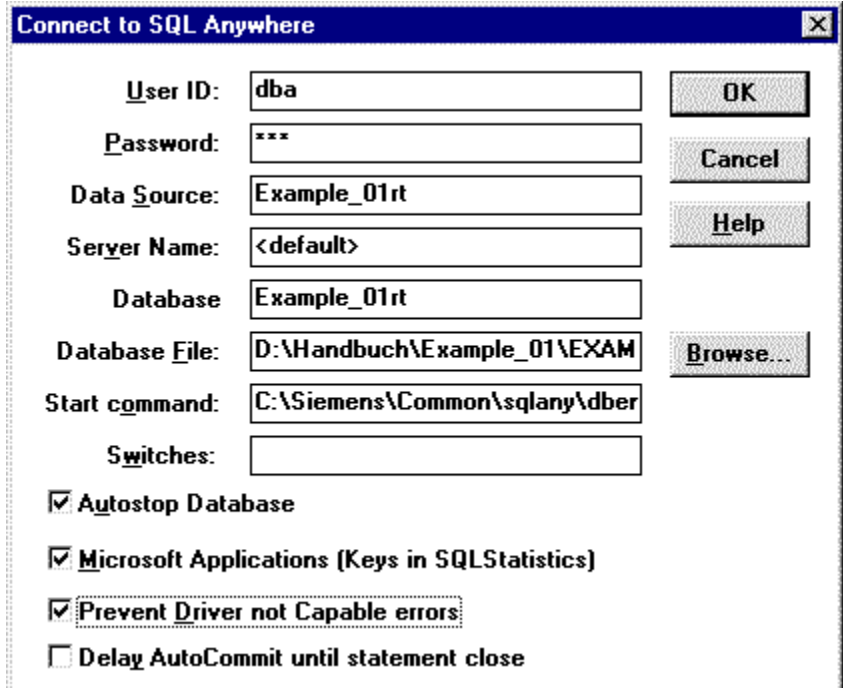
5.1.4 Доступ к базе данных

5.1.4.1 Доступ к базе данных из MS Excel/MS Query

Дальнейшее описание доступа к базе данных WinCC относится к применению Microsoft® Excel 97 с установленным SR-1.


Доступ из MS Excel/MS Query

Шаг	Процедура: Доступ из MS Excel/MS Query
1	<p>Откройте MS Excel. Через меню <i>Data (Данные)</i> → <i>Get External Data (Внешние данные)</i> → <i>Create New Query (Создать запрос)</i>, откройте диалоговое окно <i>Choose Data Source (Выбор источника данных)</i> MS Query.</p>  <p>На закладке <i>Databases (Базы данных)</i> выберите поле <i>New Data Source (Новый источник данных)</i>. После нажатия <i>OK</i> будет создан новый источник данных.</p>

Шаг	Процедура: Доступ из MS Excel/MS Query
	
2	<p>В диалоговом окне <i>Create New Data Source (Создать новый источник данных)</i> укажите имя нового источника данных. Это имя не должно совпадать с именем базы данных WinCC. В качестве драйвера укажите <i>Sybase SQL Anywhere 5.0</i>.</p> <p>По нажатию на кнопку <i>Connect.. (Подключить...)</i> открывается диалоговое окно <i>Connect to SQL Anywhere</i>, где нужно ввести информацию о драйвере. В поле <i>User ID (Пользователь)</i> вводится <i>dba</i>, в поле <i>Password (Пароль)</i> <i>sql</i>. С помощью кнопки <i>Browse (Обзор)</i> выберите базу данных для редактирования.</p> <p>Нажмите на <i>OK</i>.</p> 

Шаг	Процедура: Доступ из MS Excel/MS Query
3	<p>Если для выбранной базы данных нет сконфигурированного источника данных, появится сообщение <i>Data source name not found and no default driver specified</i> (Имя источника данных не найдено, драйвер по умолчанию не было указан).</p> <p>Нажмите кнопку <i>Connect...(Подключить...)</i> еще раз. В диалоговом окне <i>Select Data Source (Выбор источника данных)</i> выберите закладку <i>Machine Data Source</i>. Система конфигурации (CS) и база данных системы исполнения запущенного в данный момент проекта WinCC уже доступны из списка источников данных. Имена этих источников данных начинаются с <i>CC_</i> и затем следует имя проекта. Имя источника данных, представляющего собой базу данных системы исполнения заканчивается символом <i>R</i>.</p> <p>Если же требуется редактировать произвольную базу данных WinCC, для этого должен быть создан соответствующий источник данных. Это делается с помощью кнопки <i>New (Создать)</i>. На первой странице <i>Create New Data Source</i> вызывается Мастер, и Вы должны выбрать поле <i>User Data Source (Пользовательский источник данных)</i>. Затем нажмите на кнопку <i>Next (Дальше)</i>. На следующей странице выберите драйвер <i>Sybase SQL Anywhere 5.0</i>. Нажмите на <i>Next</i>. На последней странице Мастера нажмите на кнопку <i>Finish (Готово)</i>.</p> <p>Откроется диалоговое окно <i>SQL Anywhere ODBC Configuration</i>, в котором Вы должны ввести требуемую драйвером информацию. В качестве <i>User ID</i> опять введите <i>dba</i>, и <i>Password - sql</i>. Нажав на кнопке <i>Browse (Обзор)</i> выберите нужную базу данных.</p> <p>Диалоговое окно закрывается кнопкой <i>OK</i>.</p>

Шаг	Процедура: Доступ из MS Excel/MS Query
	<div data-bbox="517 353 1385 1227"><p>SQL Anywhere ODBC Configuration [X]</p><p>Data Source Name: <input type="text" value="Example_01rt"/> <input type="button" value="OK"/></p><p>Description: <input type="text"/></p><p>Connection Information</p><p>User ID: <input type="text" value="dba"/> <input type="button" value="Cancel"/></p><p>Password: <input type="text" value="***"/> <input type="button" value="Help"/></p><p>Server Name: <input type="text" value="<default>"/></p><p>Database Name: <input type="text" value="Example_01rt"/></p><p>Database Startup</p><p>Database File: <input type="text" value="D:\Handbuch\Example_01\EXAMF"/> <input type="button" value="Browse..."/></p><p><input type="radio"/> Local <input type="radio"/> Network <input checked="" type="radio"/> Custom <input type="button" value="Options..."/></p><p>Additional Connection Options</p><p>Translator Name: <input type="text" value="<No Translator>"/> <input type="button" value="Select"/></p><p><input checked="" type="checkbox"/> Microsoft Applications (Keys in SQLStatistics)</p><p><input checked="" type="checkbox"/> Prevent Driver not Capable errors</p><p><input type="checkbox"/> Delay AutoCommit until statement close</p></div>

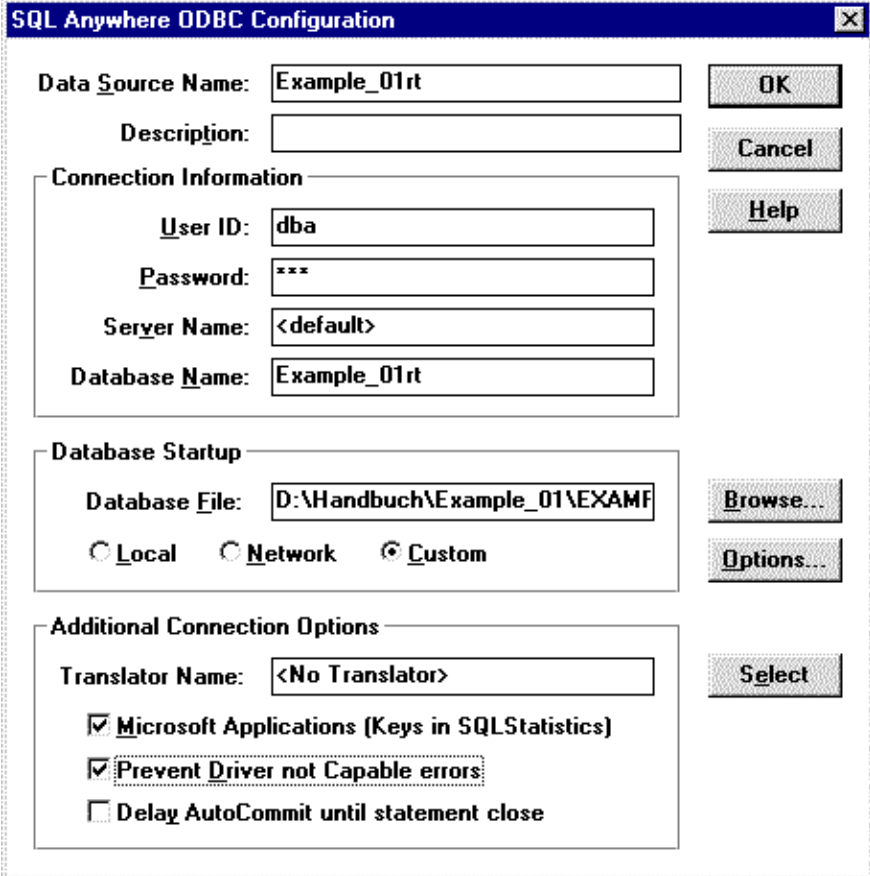
Шаг	Процедура: Доступ из MS Excel/MS Query
	<p>Выберите созданный источник данных в диалоговом окне <i>Select Data Source (Выбор источника данных)</i> и нажмите на <i>OK</i>.</p> <p>Подтвердите диалоговое окно <i>Connect to SQL Anywhere (Соединиться с SQL Anywhere)</i>.</p> <p>Конфигурация источника данных может быть также произведена через Панель Управления Windows (Control Panel). В Панели Управления откройте <i>ODBC Data Source Administrator (Администратор источников данных ODBC)</i>. Нажатие кнопки <i>Add (Добавить)</i> вызовет Мастер <i>New Data Source (Новый источник данных)</i>.</p>  <p>ODBC</p>
4	<p>Закройте диалоговое окно <i>Create New Data Source</i> кнопкой <i>OK</i>.</p> <p>В диалоговом окне <i>Select Data Source</i> выберите созданный источник данных и нажмите на <i>OK</i>.</p> <p>На первой странице <i>Query Wizard (Мастера Запросов)</i> перечислены все доступные таблицы и колонки. Выберите нужные таблицы и колонки и нажмите на <i>Next (Далее)</i>. На последующих страницах задаются фильтры для данных и порядок сортировки последних. На последней странице указывается, будут ли данные обрабатываться в MS Excel или в MS Query. Закройте диалоговое окно, нажав на кнопке <i>Finish (Готово)</i>.</p>
5	<p>В открывшемся окне <i>Returning External Data to Microsoft Excel (Возврат внешних данных Microsoft Excel)</i> укажите местонахождение баз данных, которые нужно вставить. Дополнительно могут быть заданы свойства внешнего диапазона данных. Закройте окно, нажав на <i>OK</i>.</p>

5.1.4.2 Доступ к базе данных из MS Access

Дальнейшее описание доступа к базе данных WinCC относится к применению Microsoft® Access 97 с установленным SR-1.

Доступ из MS Access

Шаг	Процедура: Доступ из MS Access
1	<p>Откройте базу данных Access или создайте новую. Через <i>File (Файл)</i> → <i>Get External Data (Внешние данные)</i> → <i>Import.. (Импорт...)</i> откройте диалоговое окно <i>Import</i>. В поле <i>File Type (Тип файла)</i> выберите <i>ODBC Databases (Базы данных ODBC)</i>.</p> <p>Откроется окно <i>Select Data Source (Выбор источника данных)</i>. В закладке <i>Machine Data Source</i> выберите источник данных. В данный момент в списке источников данных доступны и база данных режима конфигурации и база данных системы исполнения запущенного проекта WinCC. Имена этих источников данных начинаются с <i>CC_</i> и затем следует имя проекта. Имя источника данных, представляющего собой базу данных системы исполнения заканчивается символом <i>R</i>.</p>
2	<p>Если нужная база данных WinCC не приведена в списке, ее сначала нужно</p>

Шаг	Процедура: Доступ из MS Access
	<p>создать как новый источник данных; для этого нажмите кнопку <i>New (Создать)</i>.</p> <p>На первой странице <i>Create New Data Source</i> вызывается Мастер, и Вы должны выбрать поле <i>User Data Source (Источник данных пользователя)</i>. Затем нажмите на кнопку <i>Next (Дальше)</i>. На следующей странице выберите драйвер <i>Sybase SQL Anywhere 5.0</i>. Нажмите на <i>Next</i>. На последней странице Мастера нажмите на кнопке <i>Finish (Готово)</i>.</p> <p>Откроется диалоговое окно <i>SQL Anywhere ODBC Configuration</i>, в котором Вы должны ввести требуемую драйвером информацию. В качестве <i>User ID</i> опять введите <i>dba</i>, и <i>Password - sql</i>. Нажав на кнопке <i>Browse (Обзор)</i> выберите нужную базу данных.</p> <p>Диалоговое окно закрывается кнопкой <i>OK</i>.</p>  <p>Выберите созданный источник данных в окне <i>Select Data Source (Выбор источника данных)</i> <i>OK</i>.</p>
3	<p>В открывшемся окне <i>Import Objects (Импортируемые объекты)</i> выберите нужные базы данных. Нажатием на <i>OK</i> они будут вставлены в базу данных Access.</p>

5.1.4.3 Доступ к базе данных из ISQL

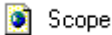
К базам данных WinCC возможен прямой доступ с использованием ISQL. Однако, это производится на свой страх и риск, так как в результате редактирования или удаления таблиц конфигурационные данные могут быть запрещены.

Доступ из ISQL

Шаг	Процедура: Доступ из ISQL
1	<p>Запустите ISQL.EXE из каталога Siemens\Common\sqlany.</p> <p>Появится окно <i>Interactive SQL Logon (Интерактивный вход в систему SQL)</i>. В качестве <i>User ID</i> укажите <i>dba</i>, в качестве <i>Password (Пароль)</i> - <i>sql</i>. Если Вы нажмете на ОК, программа соединится с открытой в данный момент базой WinCC – а именно, с базой данных системы конфигурации. Если же нужно получить доступ к другой базе данных, например, к базе данных системы исполнения, это делается через пункты меню <i>Command (Команды)</i> → <i>Connect (Присоединить)</i>. В следующем диалоге точно так же заполняются поля <i>User ID</i> и <i>Password</i>. В качестве <i>Database File (Файла базы данных)</i> указывается нужная база данных, с полным путем к ней.</p>
2	<p>В окне <i>Command (Команды)</i> теперь нужно ввести выражения SQL, которые будут выполнены по нажатию на кнопку <i>Execute (Выполнить)</i>.</p> <p>Ниже приведены некоторые примеры выражений SQL:</p> <ul style="list-style-type: none"> • <code>select* from systable</code>: отображает имена всех таблиц • <code>select * from ></code>: отображает содержимое таблицы с именем > • <code>unload table > to ></code>: экспортирует таблицу с именем > в файл с именем > • <code>drop table ></code>: удаляет файл с именем >

5.1.4.4 Доступ к базе данных из WinCC Score

Доступ из WinCC Score

Шаг	Процедура: Доступ из WinCC Score
1	<p>Перед запуском WinCC Score из меню Start (Пуск) должно быть запущено приложение WinCCDiagAgent.exe из каталога <i>Siemens\WinCC\WinCCScope\bin</i>.</p> 
2	<p>На первой странице ссылка <i>How to use the new Diagnostics Interface (Использование диагностического интерфейса)</i> предоставляет общую информацию о работе с WinCC.</p> <p>Нажмите на ссылке http://localhost для запуска Score.</p>
3	<p>Из списка слева можно вызвать различные функции.</p> <ul style="list-style-type: none"> • В <i>Database (База данных)</i> предоставляется информация о базе данных WinCC. • В <i>Database Query (Запрос к базе данных)</i> могут отображаться отдельные таблицы базы данных. База данных системы конфигурации (CS) открытого в данный момент проекта WinCC выступает в качестве <i>Источника данных (Data Source)</i>. Имя источника данных начинается с <i>CC_</i>, затем следует имя проекта. Имя источника данных, представляющего базу данных системы исполнения начинается с символа <i>R</i>. Также могут отображаться другие источники данных. • В <i>SQL Query (Запросы SQL)</i> вводятся выражения SQL, которые будут

Шаг	Процедура: Доступ из WinCC Score
	применены к источнику данных. Рекомендуется, однако, редактировать базу данных WinCC с помощью выражений SQL только если Вы обладаете обширными знаниями о системе. Примеры выражений SQL можно найти в предыдущем разделе, <i>Доступ к базе данных из ISQL (Access to the Database from ISQL)</i> .

5.1.4.5 Экспорт данных из базы данных с помощью процедур Си

Экспорт данных также может быть запущен из кадра системы исполнения WinCC. Для этого можно запустить интерактивный SQL в режиме командной строки через ProgramExecute (Запуск программы). Процедура для выполнения хранится в командном файле (в данном примере: archive.sql).

Процедура Си, пример для кнопки (Button)

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{
    char* path = "C:\\SIEMENS\\Common\\SQLANY\\ISQL-q-b-c";
    char* parameters =
        "UID=DBA;PWD=SQL;DBN=CC_Project_97-10-21_09:53:27R";
    char* action = "read D:\\WinCC\\Project\\archive.sql";

    char ExportArchive[200];

    sprintf(ExportArchive, "%s %s %s", path, parameters, action);

    ProgramExecute(ExportArchive);
}
```

- Переменная *path* (*путь*) содержит путь к программе ISQL.exe и параметры ее вызова.
- Переменная *parameters* (*параметры*) содержит поля для соединения с базой данных, которое производится в окне *Interactive SQL Logon (Интерактивный вход в систему SQL)*. Это:
 - UID (User ID, идентификатор пользователя): DBA
 - PWD (Password, пароль): SQL
- DBN (Database Name, имя базы данных): Имя источника данных ODBC. Имя этого источника данных начинается с *CC_*, затем следует имя проекта и дата/время создания проекта. Имя источника данных, представляющего базу данных системы исполнения заканчивается символом *R*. Это имя можно определить в *Windows Control Panel (Панель управления)* → *ODBC* → *Закладка User DSN*.
- Переменная *action* определяет, что должны исполняться выражения SQL, перечисленные в файле *archive.sql*.
- Выражения суммируются в *ExportArchives* и выполняются функцией *ProgramExecute()*.

Замечание:

Если экспорт должен быть произведен не из двух баз данных проекта, вместо параметра DBN должен быть указан параметр DBF (имя базы данных с указанием полного пути). Этот метод, однако, не работает с базой данных активного в данный момент проекта.

Содержимое файла: archive.sql

```
select * from PDE#HD#ProcessValueArchive#Analog;  
output to D:\WinCC\Projekt\archiv.txt format ascii
```

Из открытой базы данных выбирается архив измеренных значений *pde#hd#ProcessValueArchive#Analog* и экспортируется в ASCII файл *archive.txt* с использованием команды Output (Вывод).

5.1.4.6 Выборки из базы данных

Описанная выше команда *select* в файле команд выбирает таблицы. Подмножество этих таблиц может быть указано с помощью дополнительных параметров и затем экспортировано с помощью команды *output*. Ниже приведены некоторые примеры на данную тему.

Выборка за указанный период времени

```
select * from PDE#HD#ProcessValueArchive#Analog where T between  
'1996-5-1 10:10:0.00' and '1996-6-1 10:10:0.00'
```

Выборка, начиная с конкретного времени (Time Stamp)

```
select * from PDE#HD#ProcessValueArchive#Analog where T >  
'1996-5-1 10:10:0.00'
```

Выборка значений процесса с сортировкой и без

```
select * from PDE#HD#ProcessValueArchive#Analog where V > 100  
select * from PDE#HD#ProcessValueArchive#Analog where V > 100  
order by T
```

Выборка значений процесса с использованием столбцов T (время) и V (значение)

```
select T,V from PDE#HD#ProcessValueArchive#Analog where V > 100  
order by T
```


5.1.5 Последовательная связь

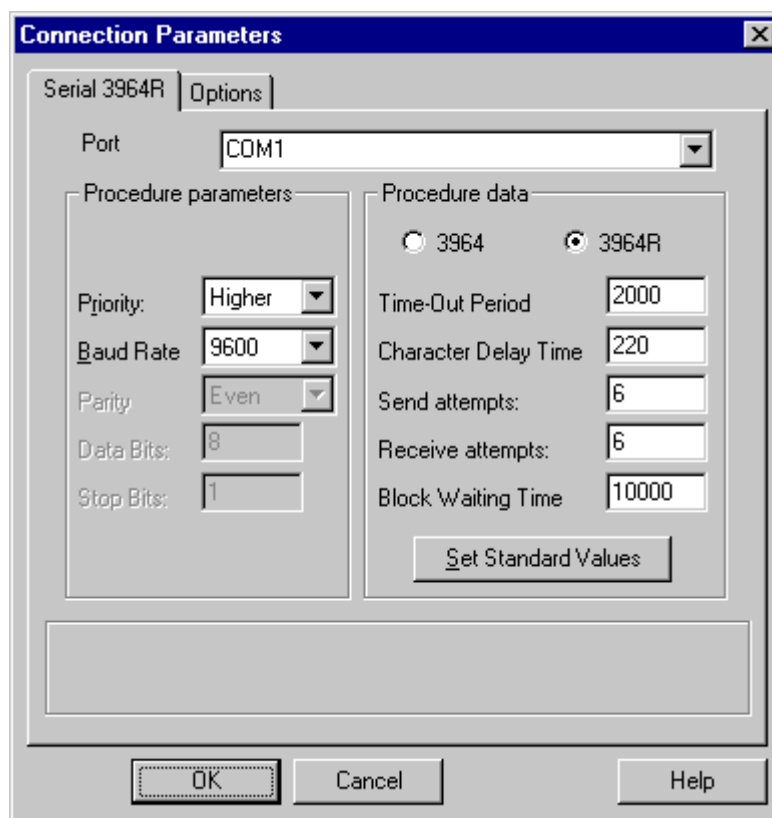
Для последовательного соединения должны быть проведены следующие настройки:

Установки CP525:

Сообщение:	Имя параметра CP525:	P3964R
Procedure (процедура):	Component: RK	Version: 01
Baud Rate (скорость передачи):	9600	Character Length (размер символа): 8
Number of Stop Bits (число стоповых битов): 1	Priority: Low (приоритет: низкий)	
Parity (четность):	Even (четный)	

В ПЛК нужно использовать SYNCHRONOUS в стартовой процедуре (startup circuit) для CP525 и SEND/RECEIVE ALL в основном цикле (cyclic program).

Установки WinCC:



В целях оптимизации один из двух партнеров должен иметь приоритет *high* (*высокий*), предпочтительно, чтобы WinCC.

5.1.6 Цветовая таблица

Значения цветов составляются из большой палитры.
Шестнадцать базовых цветов представлены ниже:

Цвет	Значение цвета (Hex)	Имя константы
Красный	0x000000FF	CO_RED
Темно-красный	0x00000080	CO_DKRED
Зеленый	0x0000FF00	CO_GREEN
Темно-зеленый	0x00008000	CO_DKGREEN
Синий	0x00FF0000	CO_BLUE
Темно-синий	0x00800000	CO_DKBLUE
Голубой (Cyan)	0x00FFFF00	CO_CYAN
Темно-голубой	0x00808000	CO_DKCYAN
Желтый	0x0000FFFF	CO_YELLOW
Темно-желтый	0x00008080	CO_DKYELLOW
Розовый (Magenta)	0x00FF00FF	CO_MAGENTA
Темно-розовый	0x00800080	CO_DKMAGENTA
Светло-серый	0x00C0C0C0	CO_LTGRAY
Серый	0x00808080	CO_DKGRAY
Черный	0x00000000	CO_BLACK
Белый	0x00FFFFFF	CO_WHITE

Символьные имена заданы заранее директивой `#define`.

Смешанные цвета получаются как промежуточные значения из палитры.

Если изменения цветов производятся с помощью динамического диалогового окна и сконфигурированные данные, затем обрабатываются в процедурах Си, значения цветов все равно могут считываться, хотя они будут получены в десятичном формате.

5.2 Документация на аварийную систему S5

Назначение и функции аварийной системы S5

Данный документ описывает функции и свойства следующего программного обеспечения SIMATIC S5:
S5 Alarm System (аварийной системы S5).

Данное ПО используется для гарантированного приема двоичных сообщений и их обработки и буферизации. Пакет программ предоставляет требуемые функции в SIMATIC S5 для реализации последовательного приема сообщений (sequenced message acquisition) системой WinCC.

Принцип работы данного ПО можно описать следующим образом: ПО следит за статусом двоичного сигнала сообщений, которые пользователь делает доступными в интерфейсе сообщений аварийной системы S5. Если состояние сигнала меняется, сообщение идентифицируется по его номеру и проставляется соответствующая временная метка (дата/время). К этим данным прибавляются 32-битовый тег и буквенно-цифровой идентификатор задачи/пакета (job/batch identifier), если пользователем сделаны соответствующие настройки. Блок сообщений, сконфигурированный таким образом, буферизируется в буфер FIFO, если это требуется. Буферизация данных необходима в том случае, если в единицу времени приходит больше сообщений, чем может быть передано в систему WinCC при данной пропускной способности шины. Благодаря этому происходит разделение по времени между приемом сообщений SIMATIC S5 и аварийной системой WinCC, что делает возможным обработку сигналов в реальном времени.

Блоки сообщений, созданные аварийной системой S5 делаются доступными приложению S5 через интерфейс блоков данных. С помощью коммуникационного ПО S5, которое должно быть реализовано пользователем, данные передаются через шину (например, SINEC H1) в высокоуровневую аварийную систему WinCC. В WinCC доступны функции всесторонней обработки сообщений, такие, как визуализация, архивация, создание отчетов и т.д. Конфигурация аварийной системы S5 выполняется пользователем через интерфейс блоков данных (системный DB 80). На этом этапе пользователь должен определить системные требования для среды работы аварийной системы. Здесь указываются области памяти, используемые аварийной системой S5, тип и область обрабатываемых сообщений, и выделение назначенных адресов памяти.

Данный раздел описывает применение и работу с аварийной системой S5 в среде SIMATIC S5. Дается обзор функций и блоков данных, используемых системой, и требования к памяти. За этим следует детальное описание всех интерфейсов данных между аварийной системой S5 и приложением S5. Для облегчения начала работы с аварийной системой S5 приводится пример.

5.2.1 Перечень программных блоков

ПО SIMATIC S5 находится на CD-ROM вместе с примерами, относящимися к данному руководству под именем *WINCC1ST.S5D*.
 Данный файл содержит следующие функции и блоки данных для аварийной системы S5:

FB	Имя	Размер	Функции
FB 80	SYSTEMFB	1114	Последовательный отчет
FB 81	STARTUPFB	135	Запуск и инициализация последовательного отчета
FB82	PCHECK	574	Используется FB 81
FB 83	MBLOCK	699	Используется FB 80
FB 84	WRITE	94	Используется FB 80
FB 87	FULL	87	Используется FB 80
DB 80	System DB	512	Назначение параметров Alarm Logging (системе регистрации аварийных сообщений)
Всего		2703	

Таблица 1

Минимальные требования к памяти зависят от конфигурации аварийной системы S5. Необходимо также наличие следующих блоков.

Буфер FIFO (min.) 2 DB = 1024 байт
Почтовый ящик пересылки в 1 DB = 512 байт
WinCC (Transfer Mailbox to
WinCC)

Для каждого блока данных смещения или параметров должны предоставляться дополнительные 512 байт.

Точный расчет размера блоков данных смещения или параметров показан в главах Структура блока данных смещения и Структура блока данных параметров.

5.2.2 Требования к аппаратному обеспечению

Для функциональных блоков аварийной системы S5, приведенных в таблице 1 требуется следующее аппаратное обеспечение:

ПЛК	ЦПУ
ПЛК 115U	ЦПУ 944 * , ЦПУ 945
ПЛК 135U	ЦПУ 928В
ПЛК 155U	ЦПУ 946/ 947, ЦПУ 948

Таблица 2

* Только ЦПУ 944 с двумя интерфейсами PG имеет системные часы.

Эти ЦПУ имеют внутренние часы, которые позволяют им предоставлять текущую дату/время для создания блоков сообщений.

Для каждого настроенного канала WinCC текущая дата/время циклически записывается в ЦПУ SIMATIC S5. Внутренние часы SIMATIC S5 синхронизируются с системными часами WinCC с помощью функционального блока **FB 86: MESS:CLOCK**.

5.2.3 Внедрение аварийной системы S5 в программу SIMATIC S5

Для внедрения ПО SIMATIC S5 для аварийной системы в приложение SIMATIC S5 должны быть предприняты следующие шаги:

Все блоки, указанные в Таблице 1, должны быть пересланы из файла WinCCST.S5D в соответствующий ПЛК.

Если блоки обработки данных не внедрены по умолчанию или они не доступны в ПЛК, загрузите их в соответствующий ПЛК.

Шаг	Процедура: Внедрение аварийной системы
1	Все блоки, указанные в Таблице 1 должны быть загружены из файла WinCCST.S5D в соответствующий ПЛК.
2	Если блоки обработки данных не внедрены по умолчанию или они не доступны в ПЛК, загрузите их в соответствующий ПЛК
3	Назначьте параметры блоку данных DB 80 в соответствии с главой Параметризация DB80 (Assigning Parameters to the DB80).
4	Настройте блоки данных для почтового ящика отправки, буфера FIFO, смещения сообщений и, если понадобится, параметров сообщений в соответствии с главой Установка блоков данных (Setup of the Data Blocks).
5	Проинициализируйте блоки данных смещения для различных категорий сообщений в соответствии с главой Блоки данных смещения (Offset Data Blocks).
6	Укажите теги процесса и указатель задачи и пакета (job and batch identifier) для индивидуальных сообщений в приложении.
7	Вызовите следующие блоки в стартовых OB (OB 20, OB 21, OB 22): <ul style="list-style-type: none"> • SPA НТВ: SYNCHRON (блок обработки данных соответствующего ЦПУ) • SPA FB 81: STARTUPFB
8	Вызовите следующие блоки в OB 1: <ul style="list-style-type: none"> • для циклической обработки сообщений SPA FB 80: SYSTEMFB • созданный пользователем функциональный блок для передачи блоков сообщений в высокоуровневую систему WinCC
9	В соответствии с последующими главами добавляются следующие функции: <ul style="list-style-type: none"> • синхронизация даты и времени с помощью FB 86: MESS:CLOCK.

Таблица 3

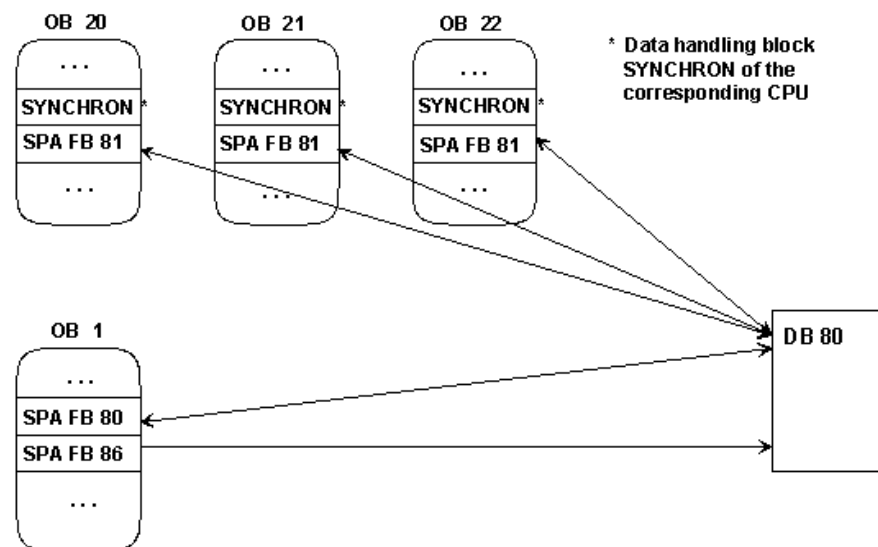


Рисунок 1

Общее описание аварийной системы S5

Дальше следует описание следующих компонентов аварийной системы S5:

- блоки данных смещения
- блоки данных параметров
- блоки сообщений
- буфер FIFO
- почтовый ящик отправки
- системный блок данных

Ниже показано взаимодействие различных компонентов:

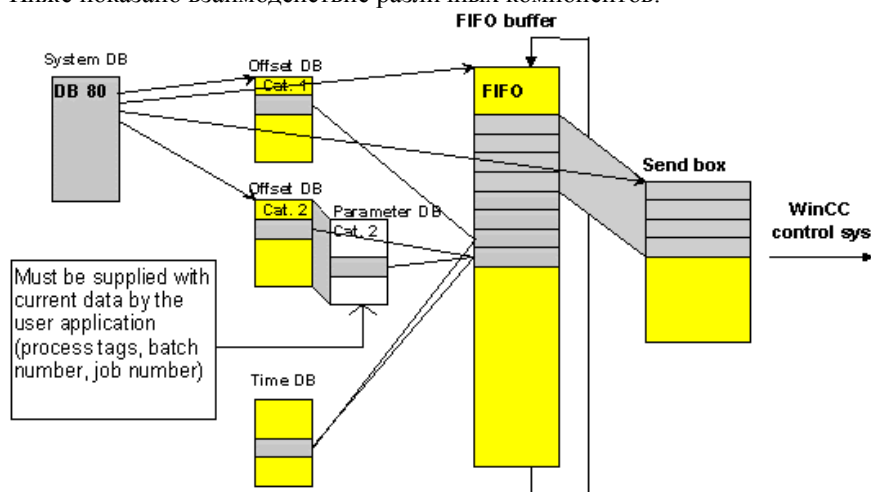


Рисунок 2

Перед тем, как система приема сообщений сможет отслеживать и принимать сообщения, последние должны быть сконфигурированы в соответствующих блоках данных. Есть четыре различные категории сообщений:

Категория	Определение: категория сообщения
1	Сообщение без параметра
2	Сообщение с тегами процесса (2 DW)
3	Сообщение с тегами процесса (2 DW) и идентификатором задачи/пакета (3 DW)
4	Сообщение с тегами процесса (2 DW), идентификатором задачи/пакета (3 DW) и зарезервированным пространством (3 DW)

Таблица 4

Для аварийной системы дата/время могут указываться глобально при создании блока сообщений. Если дата/время отсутствует, соответствующая информация добавляется в блоки сообщений системой WinCC.

5.2.3.1 Структура блока данных смещения

Блок данных смещения имеет одинаковую структуру для всех четырех категорий сообщений. Адрес соответствующего блока данных указывается для каждой требуемой категории сообщений в системном блоке данных DW 80.

Блоки данных смещений для соответствующей категории сообщений:

DW	Содержимое	Назначение
DW 0	Не назначено	Заголовок
DW 1	Основной номер сообщения	
DW 2	Адрес последнего блока статуса сигнала	
DW 3	Не назначено	
DW 4	Состояния сигнала сообщений – номера битов: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Блок статуса сигнала 1
DW 5	Статусные биты бездействия	
DW 6	Биты подтверждения	
DW 7	Флаги установки фронтов	
DW 8	Состояния сигнала сообщений – номера битов: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Блок статуса сигнала 1
DW 9	Статусные биты бездействия	
DW 10	Биты подтверждения	
DW 11	Флаги установки фронтов	
DW 12	Состояния сигнала сообщений – номера битов: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Блок статуса сигнала 3
DW 13	Статусные биты бездействия	

Таблица 5

Далее приведено описание следующих элементов:

- Основной номер сообщения
- Смещение сообщения
- Блок статуса сигнала
- Адрес последнего блока статуса сигнала
- Состояния сигнала
- Статусные биты бездействия
- Биты подтверждения
- Флаги установки фронтов

5.2.3.2 Основной номер сообщения

Каждому сообщению назначается определенный номер, по которому оно опознается из множества пришедших сообщений. Номер сообщения состоит из основного номера сообщения (basic message number) и номера смещения сообщения (offset message number).

Для каждой используемой категории сообщений необходимо указывать различные номера сообщения. Внутри категории сообщения сообщений различаются смещением. Основной номер сообщения для соответствующей категории сообщения указан в DW 1 ассоциированного с данной категорией блока данных смещения.

Особый случай

При использовании категории сообщений 1 есть возможность использовать два блока данных смещения. Для того чтобы нумерация сообщений была непрерывной, в качестве основного номер сообщения второго блока данных смещения должен вводиться базовый номер сообщения первого блока смещения плюс его объем (1008 сообщений).

Вычисление номера сообщений:

Номер сообщения = Основной номер сообщения + смещение сообщения

Пример:

Вычисление:	Описание:
Дано:	Категория сообщений 1, непрерывная нумерация сообщений, начиная с номера сообщений: 10000
Требуется:	Основной номер сообщения двух блоков данных смещения
10000	Основной номер сообщения первого блока данных смещения
$10000 + 1008 = 11008$	Основной номер сообщения второго блока данных смещения

5.2.3.3 Смещение сообщения/состояния сигнала сообщений

Состояние сигнала сообщений содержится в блоке данных смещения соответствующей категории сообщений в бите, чья позиция соответствует смещению сообщения.

Смещение соответствующего сообщения начинается с 16 битов (биты 0-15) DW 4. Затем продолжается непрерывная нумерация - DW8, DW12, т.д.

Блок статуса сигнала	Слово данных, с которого начинается блок статуса сигнала	Номера битов 0 - 15 отвечающих за смещение
1	4	0 – 15
2	8	16 – 31
3	12	32 – 47
4	16	48 – 63
...		...
62	248	976 – 991
63	252	992 – 1007

Таблица 6

Вычисление смещения сообщения:

$$\begin{aligned}
 \text{Смещение} &= \text{Номер сообщения} - \text{Основной номер сообщения} \\
 \text{Смещение} &= (\text{Слово данных} / 4 - 1) * 16 + \text{Номер бита (0-15)} \\
 \text{Смещение} &= (\text{Блок статуса сигнала} - 1) * 16 + \text{Номер бита (0-15)}
 \end{aligned}$$

Вычисление DB, DW и номера бита по смещению:

$$\begin{aligned}
 \text{Блок данных} &= \text{Блок данных смещения} \\
 \text{Слово данных} &= (\text{Смещение сообщения} / 16 + 1) * 4 \\
 \text{(Data Word)} & \\
 \text{Номер бита} &= \text{Смещение сообщения} \% 16
 \end{aligned}$$

В случае сообщений категории 1, длина слова данных может быть больше, чем 252. Тогда применяются следующие правила:

$$\begin{aligned}
 \text{Блок данных} &= \text{Блок данных смещения} + 1 \\
 \text{Слово Данных} &= \text{Слово данных} - 252 \\
 \text{Номер бита} &= \text{Номер бита}
 \end{aligned}$$

Пример 1:

Дано: DW 248, бит 7, основной номер сообщения = 10000

Требуется: номер сообщения

$$\begin{aligned}
 \text{Блок статуса сигнала} &= 248 / 4 \\
 &= 62 \\
 \text{Смещение сообщения} &= (\text{Блок статуса сигнала} - 1) * 16 + \text{номер бита} \\
 &= (62 - 1) * 16 + 7 = 983 \\
 \text{Номер сообщения} &= \text{Основной номер сообщения} + \text{смещение} \\
 &= 10000 + 983 = 10983
 \end{aligned}$$

Требуемый номер сообщения равен 10983.

Пример 2:

Дано: Категория сообщений 1 с двумя блоками данных смещения, номер сообщения = 12000, основной номер сообщения = 10000

Требуется: DB, DW, номер бита

$$\begin{aligned}
 \text{Смещение} &= \text{номер сообщения} - \text{основной номер сообщения} \\
 &= 12000 - 10000 = 2000 \\
 \text{Номер бита} &= \text{Смещения} \% 16 = 0 \\
 \text{Слово данных} &= (\text{Смещение} / 16 + 1) * 4 \\
 &= (2000 / 16 + 1) * 4 = 504
 \end{aligned}$$

Слово данных больше 252, поэтому.

$$\begin{aligned}
 \text{Блок данных} &= \text{Блок данных смещения} + 1 \\
 \text{Слово данных} &= 504 - 252 = 252 \\
 \text{Номер бита} &= 0
 \end{aligned}$$

Сообщение с номером 12000 находится во втором блоке данных смещения категории сообщений 1 в слове данных 252, бит 0.

5.2.3.4 Блок статуса сигнала

Первый блок статуса сигнала начинается со слова данных с адресом 4, следующие следуют с интервалом в 4 слова данных (DW 8, DW 12, т.д.).

См. также Таблицу 5 или Таблицу 6.

Для каждого блока данных смещения возможны 63 блока статуса сигнала (с 1 по 63). Блок статуса сигнала содержит 16 состояний сигнала. Это дает $63 * 16 = 1008$ возможных сообщений в блоке данных смещения.

Структура блока статуса сигнала:

DW	Номер бита	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	Состояния сигнала	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Состояния бездействия	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Биты подтверждения	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Флаги установки фронтов	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Таблица 7

Дополнительную информацию об этих четырех битах состояния можно найти в этой главе.

Вычисление соответствующего блока статуса сигнала:

$$\text{Блок статуса сигнала} = (\text{Смещение сообщения} / 16) + 1$$

$$\text{Блок статуса сигнала} = \text{Слово данных} / 4$$

Вычисление слова данных, с которого начинается соответствующий блок статуса сигнала:

$$\text{Первое слово данных блока статуса сигнала} = \text{Номер блока статуса сигнала} * 4$$

5.2.3.5 Адрес последнего блока статуса сигнала

Число возможных сообщений в соответствующей категории сообщений – это адрес DW последнего занятого сообщениями блока статуса сигнала.

Вычисление последнего блока статуса сигнала:

Последний блок статуса сигнала = требуемое количество сообщений в данной категории / 16

```
// Incompletely filled (16 messages) signal condition block
if ((required messages of this message category % 16) != 0)
{
    ++ last signal message block;
}
```

В категории сообщений 1 может образоваться группа из более чем 1008 сообщений, в данном случае применяется следующее:

Первый DW смещения:

$$\text{Последний блок статуса сигнала} = 63$$

$$\text{Адрес последнего блока статуса сигнала} = 63 * 4 = 252$$

Второй DW смещения:

Последний блок статуса сигнала = $(\text{Требуемое количество сообщений в категории} - 1008) / 16$

```
// Incompletely filled (16 messages) signal condition block
if ((required messages in this message category - 1008) % 16) != 0)
{
    ++ last signal message block;
}
```

Вычисление адреса DW последнего блока статуса сигнала:

*Адрес DW последнего блока статуса сигнала = Последний блок статуса сигнала * 4*

Пример:

Дано 1030 сообщений в категории сообщений 1

Первый DW смещения:

Адрес последнего блока статуса сигнала

Второй DW смещения:

Требуемое количество сообщений – 1008 = 1030 - 1008 = 22

(Требуемое количество сообщений – 1008) / 16 = 22 / 16 = 1

(Требуемое количество сообщений – 1008) % 16 = 22 % 16 = 6

Последний блок статуса сигнала = 2

Адрес последнего блока статуса сигнала

5.2.3.6 Состояния сигнала

Позиция: 1-е слово данных блока статуса сигнала (см. Таблицу 5).

Пользователь должен удостовериться, что состояния сигнала соответствующих сообщений вводятся в слова данных блоков данных смещения соответствующей категории сообщений. Это может быть произведено программой управления путем непрерывных обновлений сигналов процесса.

5.2.3.7 Состояния бездействия

Позиция: 2-е слово данных блока статуса сигнала (см. Таблицу 5).

Под состоянием/статусом бездействия сигнала (idle status of a signal) мы подразумеваем уровень сигнала при пассивном режиме работы. Состояние определяет, сигнал (сообщение) низок или высок. Эта информация нужна для определения, приходит сообщение или уходит.

Если статус события является отрицанием статуса бездействия, значит сообщений приходит. Если сообщение уходит, статус изменения события равен соответствующему статусу бездействия.

Состояния бездействия сообщений должны указываться пользователем в соответствующих позициях.

5.2.3.8 Биты подтверждения

Позиция: 3-е слово данных блока статуса сигнала (см. Таблицу 5).

Биты подтверждения не устанавливаются, а используются запущенной программой. Сообщения подтверждаются непосредственно ПК согласно принятой идеологии подтверждения. Эти подтверждения сообщений посылаются ПК соответствующему ПЛК вместе со сконфигурированными сообщениями внедренной системы подтверждения.

Соответствующий бит подтверждения устанавливается на протяжении одного цикла аварийной системой S5.

Приложение должно соответствующим образом использовать данную информацию.

5.2.3.9 Флаги установки фронтов

Позиция: 3-е слово данных блока статуса сигнала (см. Таблицу 5).

Флаги установки фронтов используются для определения изменения событий, которые могут произойти (изменение сообщения). Они не устанавливаются, а используются запущенной программой.

5.2.3.10 Структура блока данных параметров

Для сообщений категорий со 2-й по 4-ю необходимо сконфигурировать блоки данных параметров для добавочных данных сообщений, в дополнение к блоку данных смещения. Статус сигнала сообщения хранится в блоке данных смещения. Адреса блоков данных параметров хранятся в непрерывных блоках данных и непосредственно прикрепляются к соответствующим блокам данных смещения. Взаимоотношения между блоками данных параметров и смещения показано ниже:

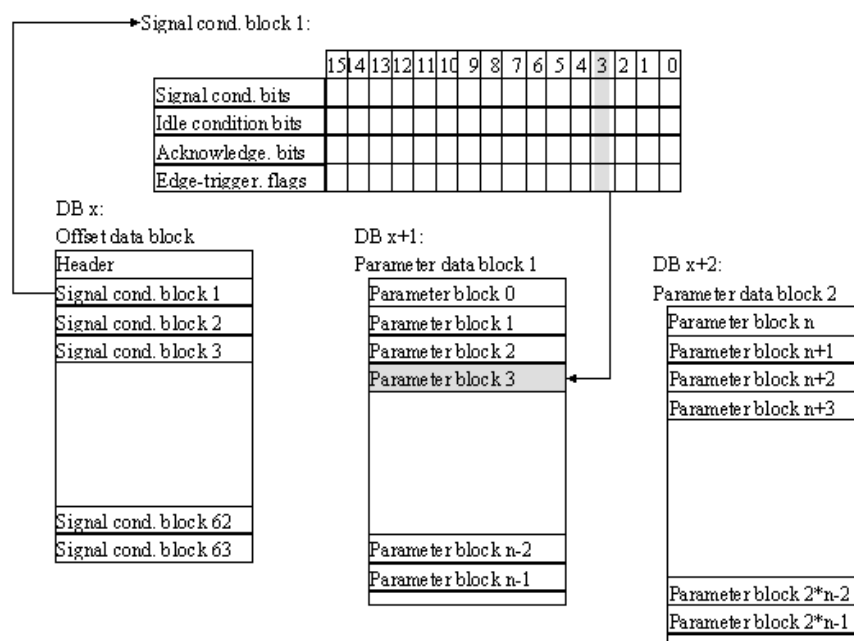


Рисунок 3

Категория	Макс. число	Размер блока параметров	Число блоков на блок данных параметров	Макс. число блоков данных параметров
1	1008 / 2016	-	-	-
2	1008	2 DW	128	8
3	1008	5 DW	51	20
4	1008	7 DW	36	28

Таблица 8

Вычисление числа блоков данных параметров:

$$\text{Number of Parameters Data Block} = \frac{\text{used Messages}}{\text{Number of parameter blocks per parameter data block}}$$

При конфигурировании нужно удостовериться, что адреса блоков данных различных категорий сообщений не пересекаются, и что число блоков данных параметров сможет выдержать последующие изменения.

Блок данных параметров содержит блоки параметров, которые назначаются различным сообщениям. Блоки параметров хранятся один за другим в блоке данных параметров, начиная с блока параметров для первого сообщения в данной категории. Блоки параметров продолжаются за пределы блока данных параметров. В случае достижения конца блока данных, блоки параметров продолжаются со следующего блока данных параметров с DW0 и далее. В блоке данных параметры хранятся только целыми блоками.

Вычисление начального адреса блока параметров:

Смещение сообщения	=	Номер сообщения – Основной номер сообщения
DW параметров	=	DW смещения + 1 + (Смещение сообщения / Число блоков параметров на DW параметров)
Начальный адрес DW параметров	=	(Смещение сообщения % Число блоков параметров на DW параметров) * размер блока параметров

Пользователь должен удостовериться, что соответствующие данные (теги процесса, номер задачи, идентификатор пакета) доступны по соответствующим адресам.

5.2.3.11 Структура блока сообщений

Блок сообщений, посылаемый системе WinCC высшего уровня, состоит из нескольких следующих одно за другим слов данных. Слова данных содержат относящуюся к сообщениям информацию. Совокупность этих слов составляет блок сообщений. Размер блоков сообщений зависит от конкретной категории сообщения. Независимо от категории сообщений, блок сообщений всегда состоит, по крайней мере, из двух слов. Это номер сообщения и его статус. В зависимости от того, проставлены ли дата/время (3 слова), и присутствуют ли дополнительные параметры, блок сообщений может быть длиной до 12 слов.

DW	Описание:
1	Номер сообщения
2	Статус сообщения
3	Время
4	Время
5	Дата
6	Тег процесса
7	Тег процесса
8	Номер задачи
9	Номер задачи
10	Идентификатор пакета
11	Зарезервировано
12	Зарезервировано

Таблица 9

Если в сообщениях не проставлены дата/время, то три слова, предназначенные для них, опускаются. Слова параметров сдвигаются к слову статуса, заполняя промежуток. Конкретный размер блока сообщений (число DW) различается, в зависимости от категории сообщений и наличия даты/времени.

Определение длины блока сообщений в зависимости от категории сообщений

Категория	Длина блока сообщений в DW без даты/времени	Длина блока сообщений в DW с датой/временем
1	2	5
2	4	7
3	7	10
4	9	12

Таблица 10

5.2.3.12 Номер сообщения

Каждому сообщению назначается определенный номер, по которому оно однозначно определяется.

5.2.3.13 Состояние сообщения

Состояние сообщения имеет следующую структуру:

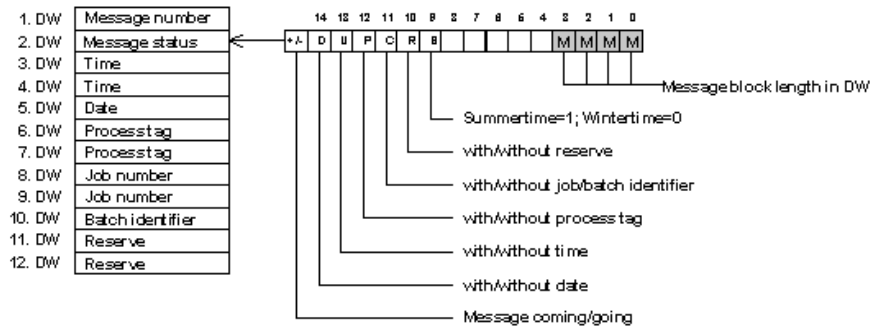


Таблица 11

5.2.3.14 Временная метка

Дата и время (date/time stamp) получаются с помощью функционального блока FB 86: MESS:CLOCK в двоичном виде.

5.2.3.15 Теги процесса

Это два слова, по которым теги процесса могут быть записаны и переданы системе обработки, если сообщение приходит.

5.2.3.16 Номер задачи / идентификатор процесса

В зависимости от конфигурации, первые два слова могут быть интерпретированы как 32-битовое двоичное число или четыре символа ASCII. Третье слово должно интерпретироваться как два символа ASCII.

В случае прихода сообщения через эти три слова системе WinCC могут быть переданы номер текущей задачи или идентификатор процесса.

5.2.3.17 Зарезервировано

Два зарезервированных слова в категории сообщений 4 предназначены для использования в будущем, и в данный момент не используются WinCC.

5.2.3.18 Создание блока сообщений

После определения факта возникновения сообщения, проверяемый бит используется для определения номера соответствующего сообщения, которое хранится в первом слове буфера FIFO. В зависимости от того, приходит сообщение или уходит, от категории и требований наличия временной метки, выбирается соответствующая маска статуса и сохраняется во втором слове данных блока сообщений в буфере FIFO. Если были указаны параметры для временной метки в соответствующем бите в системном блоке данных, в запрошенном ПК формате будут следовать три слова из системного блока данных 80 – начиная с адреса DW 190. В зависимости от категории сообщения, данный параметр считывается, если требуется, из соответствующего архива данных (блока данных параметров) и добавляется к последнему элементу в буфере FIFO для завершения блока сообщений.

Затем проверяется следующий бит статуса последующего сообщения. Это повторяется, пока не будут обработаны все сообщения, которым были назначены параметры.

5.2.3.19 Внутренний FIFO буфер (кольцевой)

Буфер FIFO представляет собой структуру для хранения, напоминающую кольцо. С одной стороны, это ограничивает возможный объем, с другой стороны, обеспечивает его бесконечность, так как только буфер заполняется, его заполнение начинается с начала.

В системе приема сообщений это означает, что самые старые данные будут перезаписаны самыми новыми, как только будет достигнут конец буфера (то есть он заполнится) – если старые данные не были удалены – и таким образом информация будет потеряна.

Буфер FIFO в памяти RAM действует, как можно предположить по его названию, в качестве буфера для принятых сообщений, пока они не переданы в ПК. В памяти буфер FIFO представляет собой область, состоящую, по крайней мере, из двух блоков данных, и может, в зависимости от параметров, быть произвольного размера в пределах максимального разрешенного числа блоков данных в ПЛК или оставшегося от размещения прикладной программы. Пользователь сообщает аварийной системе число блоков данных, которые она может использовать для архивации.

Если необходимо более одного блока данных, нужно использовать блоки данных с непрерывной нумерацией. Пользователь указывает начальный и конечный номер DB буфера в качестве параметров системного DB. Все блоки данных, чьи значения лежат между начальным и конечным блоками данных (включительно), принадлежат буферу и используются в качестве места для хранения.

5.2.3.20 Почтовый ящик отправки - передача данных на верхний уровень системы WinCC

Обычно все сообщения текущего цикла сначала записываются во внутренний буфер аварийной системы S5.

Сообщения (с максимальным размером до одного блока данных) передаются интерфейсу сообщений (почтовому ящику отправки) по завершении приема, обеспечивая готовность почтового ящика отправки. Интерфейс сообщений, в форме блока данных, выступает в качестве источника данных для функциональных блоков передачи (STEP 5 – блоки обработки данных). Блоки обработки данных формируют интерфейс к соответствующему коммуникационному процессору для работы с используемой шиной (например, шиной SINEC-H1).

Структура почтового ящика отправки:

DW	Содержимое
DW 0	Длина блока данных
DW 1	KY = [Номер ПЛК], [Номер ЦПУ]
DW 2	KY = [0], [Число сообщений]
DW 3	Начало пользовательских данных (блоки сообщений)

Таблица 12

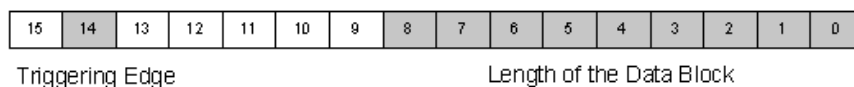
DW 0:

Таблица 13

DW 0 почтового ящика отправки определяется битом 14 – триггером фронта активации запрошенной задачи – и битами 0-8 – длиной исходных данных. Так как передаваемый блок данных может иметь максимальную длину в 256 слов, а один байт может представлять число до 255, отдельный запрос байтов с использованием команд DL или DR невозможен. В связи с этим рекомендуется передавать DW 0 в качестве вспомогательного флага. Это также дает то преимущество, что бит разрешения может быть обработан отдельно и непосредственно. Данная операция не может быть использована для слов. Если эти условия соблюдены, этот бит используется в качестве фронта для однократного сброса задачи пересылки. Оставшиеся установленные биты отвечают за размер переданных данных и могут быть записаны в область косвенного назначения параметров в качестве QLAE.

После успешного выполнения запроса WRITE (запись, SINEC-H1) в систему WinCC (free of faults, без отказов (FOF)), DW 0 почтового ящика отправки должен быть перезаписан значением 0. Это снова делает доступным почтовый ящик отправки для функционирования.

Запрос WRITE (SINEC-H1) должен быть реализован с помощью прямой функции SEND. Информацию о данной функции можно найти в соответствующем руководстве к ПЛК.

5.2.4 Описание интерфейса

Ниже будут описаны следующие интерфейсы и блоки:

- Системный блок данных DB 80: для назначения параметров аварийной системе S5.
- Блок данных смещения для соответствующей категории сообщений: двоичный интерфейс сигналов сообщений к аварийной системе S5 со спецификацией свойств сообщений.
- Блок данных параметров для соответствующей категории сообщений: для указания дополнительных данных сообщений категорий со 2-й по 4-ю.
- Почтовый ящик отправки: интерфейс передачи в системе WinCC.

5.2.4.1 Системный блок данных 80

С использованием системного блока данных становится возможным конфигурировать независимые области данных для четырех категорий сообщений, буфер FIFO и почтовый ящик отправки. Для конфигурации предоставляются слова с 0-го по 20-й в DB 80.

5.2.4.2 Блок данных смещения

Аварийная система S5 проверяет состояния сигнала соответствующих сообщений и формирует для них соответствующие блоки сообщений, если это требуется. Пользователь должен удостовериться, что...

- при конфигурировании состояний бездействия указаны отдельные различные сообщения.
- состояния сообщений записываются в соответствующие сигнальные биты состояний в процессе выполнения приложения S5.
- если это требуется, считываются и проверяются биты подтверждения.

5.2.4.3 Блок данных параметров

Для категорий сообщений со 2-й по 4-ю блоком сообщений может передаваться дополнительная информация о текущем статусе системы. Пользователь должен удостовериться, что...

- по прибытии сообщения нужные теги процесса (значение процесса, номера задачи и пакета) находятся в соответствующих блоках параметров.

5.2.4.4 Почтовый ящик отправки (Send Mailbox)/ почтовый ящик передачи (Transfer Mailbox)

Как только почтовый ящик отправки содержит блоки сообщений, он передается системе WinCC с помощью задачи WRITE (SINEC-H1). Пользователь должен удостовериться, что...

- доступны соответствующие блоки обработки данных используемого ЦПУ.
- при конфигурации системы WinCC были указаны подходящие коммуникационные каналы для соединения шины процесса.
- активизирована задача WRITE.

5.2.5 Назначение параметров для аварийной системы S5 / системного DB 80

Ниже приведено описание конфигурируемых слов системного блока данных DB 80:

DW	Описание
0	Адрес DB: начало внутреннего FIFO
1	Адрес DB: конец внутреннего FIFO
2	0: без даты и времени, 1: с датой и временем
3	DB смещения для категории сообщений 1
4	1: один DB смещения категории 1, 2: два DB смещения категории 1
5	DB смещения для категории сообщений 2
6	DB смещения для категории сообщений 3
7	DB смещения для категории сообщений 4
8	Зарезервировано
9	Зарезервировано
10	Адрес DB: почтовый ящик отправки ЦПУ -> ПК
11	1: опрос оптимизирован (АСОР)
12	АСОР начинается с n сообщений
13	Тип ПЛК (115/135/155)
14	Зарезервировано (должно быть 1)
15	Номер ПЛК: с 1 по 255; Номер ЦПУ: с 1 по 4
16	Зарезервировано
17	Зарезервировано
18	Зарезервировано
19	Зарезервировано
20	Ошибка четности

Таблица 14

DW 0, DW 1: область памяти внутреннего буфера FIFO

Область внутреннего буфера FIFO для сообщений определяется этими двумя словами.

Область хранения должна быть размером, по меньшей мере, в два блока данных; следует также удостовериться, что значение конечного DB буфера FIFO больше значения его начального DB.

Область памяти буфера состоит их блоков данных между конечным и начальным (включительно).

Выбор размера буфера FIFO:

При достижении предела буфера FIFO старые сообщения начинают перезаписываться. Число блоков данных должно быть выбрано достаточно большим для того, чтобы в случае прихода большого количества сообщений они не были перезаписаны перед экспортом. Для того чтобы быть в этом уверенными, используйте следующий практический метод:

Определение числа DB буфера FIFO:

Число сообщений на DB = $(255 \text{ DW} / \text{DB}) / \text{Длина блока сообщений}$

См. Таблицу 10

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Triggering Edge								Length of the Data Block							

Для режима *АСОР* системы приема сообщений рекомендуется выделить дополнительно один или более блоков.

DW 2: Идентификатор даты и времени

Возможность проставления временной метки относится ко всем параметризуемым сообщениям. Либо временная метка проставляется во всех принятых сообщениях, (DW2 = 1) либо ни в одном (DW2 = 0). Если установлен DW2 = 0, система WinCC добавляет временную метку к приходящим блокам сообщений.

DW 3, DW 4: DB смещения категории сообщений 1

Если должны конфигурироваться сообщения категории 1 (сообщения без параметров и идентификатора пакета), то в слове 3 должен быть задан блок данных смещения. Состояния сигналов этих сообщений должны непрерывно записываться программой управления в указанные блоки данных.

Если планируется использовать больше, чем 1008 сообщений (но не больше, чем 2016) категории 1, данной категории должен быть предоставлен дополнительный блок данных, для этого в слове 4 следует ввести 2. Второй блок данных автоматически получает следующий, более высокий номер, чем адрес в DW 3. Если максимума в 1008 сообщений категории 1 хватает, то в DW 4 вводится 1.

DW 5, DW 6, DW 7: DB смещения сообщений категорий 2, 3, 4

Так же, как и в DW 3, слова 5 - 7 содержат адреса блока данных, где хранятся сигналы сообщений.

DW 5 содержит адрес блока данных для сообщений категории 2, а слова 6 и 7 содержат адреса для сообщений категорий 3 и 4 соответственно.

Если категория сообщений не используется, в соответствующий DW вводится 0.

Адреса, введенные в DW 5 - 7 представляют так называемые DB смещения. В зависимости от категории сообщений и числа сообщений в категории, им назначается соответствующий номер второго блока данных. Второй блока данных содержит параметры сообщений. По этой причине, нужно удостовериться, что при назначении адресов DB смещения выделено достаточно места (блоков данных) для блока данных параметров между предыдущим DB смещения и тем, который указывается.

Для категорий сообщений со 2-й по 4-ю может быть сконфигурировано до 1008 сообщений. При полном использовании для различных категорий назначаются разные номера вторых блоков данных (DB параметров) DB смещения. (См. Таблицу 8).

DW 10: интерфейс сообщений с высшим уровнем системы WinCC

Параметры должны всегда назначаться слову 10. Адрес DB почтового ящика передачи назначается в DW 10. Почтовый ящик передачи выступает в качестве интерфейса между SIMATIC S5 и системой WinCC.

DW 11, DW 12: Выбор режима и числа сообщений для их оптимального приема
Возможны два режима работы:

- 0 в DW 11 -> нормальный режим работы системы приема сообщений
- 1 в DW 11 -> оптимальный режим приема системы приема сообщений

Нормальный режим:

Все сообщения, которые были приняты и сохранены во внутреннем буфере, посылаются станции WinCC по интерфейсу сообщений (который имеет ограниченный объем) за один цикл, только в случае готовности интерфейса на прием сообщений.

Данный метод отличается сравнительно большим временем цикла в том случае, если в течение цикла или нескольких смежных циклов приходит большое количество сообщений. Время цикла также увеличивается вместе с размером блока сообщений используемых категорий. В этом случае прием сообщений требует больше затрат и занимает больше времени.

Оптимизированный прием:

Хронологический прием сообщений имеет приоритет при пересылке на станцию WinCC. Внимание концентрируется на относительном времени между появлением сообщений. Тот факт, что сообщения могут отображаться на станции WinCC на несколько миллисекунд позже, рассматривается как второстепенный. Решающую роль играют медлительность человеческого зрительного восприятия и реакция оператора в зале управления.

Для уменьшения времени цикла приема сообщений в таких критичных по времени случаях была введена система оптимизированного приема сообщений. Минимальное количество появляющихся сообщений в цикле OB1 указывается в DW 12. Если число сообщений в течение текущего цикла OB1 превышает это минимальное число, сообщения только принимаются и архивируются. Они не выбрасываются и не посылаются коммуникационному партнеру в данном цикле OB1.

DW 15: Номер ПЛК/ЦПУ

Это слово требуется для создания заголовка сообщения и здесь нужно указать номер ПЛК текущего проекта и номер его ЦПУ. Номер ЦПУ особенно важен, если в одном ПЛК работают несколько ЦПУ. Только при наличии слова и идентификатора (ID) сообщения, система высшего уровня WinCC сможет интерпретировать посланные данные как сообщения, назначать текст сообщений и рассматривать их соответствующим образом.

DW 15 – это единственное слово данных, получающее формат KY S5 во время конфигурации, т.е. два байта могут рассматриваться отдельно (разделенными запятой). Левый байт содержит номер ПЛК, который может быть между 1 и 255. В правом байте указывается номер ЦПУ, который может лежать между 1 и 4..

Пример:

KY = 10,2
Номер ПЛК = 10
Номер ЦПУ = 2

DW 20: Ошибки назначения параметров

Все слова данных, которые представляют собой параметры в системном DB, проверяются на корректность при запуске аварийной системы S5. В данном случае делаются проверки на превышение предельных значений, пересечение или повторное задание блоков данных параметров и отсутствие спецификаций.

В качестве выходного параметра в формате слова эта функция использует так называемое слово PAFE (слово ошибки параметра); оно похоже на специфичные для системы блоки обработки данных. Статус слова PAFE может быть взят из DW 20 в системном DB 80. Слово PAFE может быть проанализировано на предмет ошибок, которые могли произойти после возврата программы из FB 81. Затем должны быть предприняты соответствующие действия.

Рекомендуется переводить ПЛК в состояние останова в том случае, если слово PAFE отлично от 0. Если слово PAFE игнорируется, нет никакой гарантии, что выполнение программы будет безошибочным.

Анализ слова PAFE

Если программа или ПЛК были переведены в состояние останова – что рекомендуется – в результате возникновения ошибки (неравенства слова RAFF нулю), ошибка может быть проанализирована и затем устранена по ее номеру. Следующая таблица предоставляет информацию о типе ошибки, произошедшей при задании параметров.

Формат слова PAFE:

KY = *Номер ошибки, ID группы ошибок (Group Error ID)*

Пример:

KY = 9,1

Ошибка задания параметров с номером 9 соответствует:
Адрес DB смещения категории 1 больше, чем максимально допустимый адрес DB.

№ ошибки.	Описание
1	Не определен начальный DB внутреннего буфера
2	Начальный DB внутреннего буфера имеет тот же адрес, что и системный DB (80)
3	Адрес начального DB внутреннего буфера больше, чем максимально допустимый адрес DB
4	Конечный DB внутреннего буфера имеет тот же адрес, что и системный DB (80)
5	Адрес конечного DB внутреннего буфера меньше, чем его начальный адрес
6	Адрес конечного DB внутреннего буфера больше, чем максимально допустимый адрес DB
7	DB смещения категории 1 имеет тот же адрес, что и системный DB (80)
8	DB смещения категории 1 находится в пределах области внутреннего буфера
9	Адрес DB смещения категории 1 больше, чем максимально допустимый адрес DB
10	DB смещения категории 2 имеет тот же адрес, что и системный DB (80)
11	DB смещения категории 2 имеет тот же адрес, что и DB смещения категории 1
12	DB смещения категории 2 имеет тот же адрес, что и второй DB смещения категории 1
13	Адрес DB смещения категории 2 находится в пределах области внутреннего буфера
14	Адрес DB смещения категории 2 больше, чем максимально допустимый адрес DB
15	DB смещения категории 3 имеет тот же адрес, что и системный DB (80)
16	DB смещения категории 3 имеет тот же адрес, что и DB смещения категории 1
17	DB смещения категории 3 имеет тот же адрес, что и второй DB смещения категории 1
18	DB смещения категории 3 имеет тот же адрес, что и DB смещения категории 2
19	Адрес DB смещения категории 3 находится в пределах области

№ ошибки.	Описание
	внутреннего буфера
20	Адрес DB смещения категории 3 больше, чем максимально допустимый адрес DB
21	DB смещения категории 4 имеет тот же адрес, что и системный DB (80)
22	DB смещения категории 4 имеет тот же адрес, что и DB смещения категории 1
23	DB смещения категории 4 имеет тот же адрес, что и второй DB смещения категории 1
24	Адрес DB смещения категории 4 находится в пределах области внутреннего буфера
25	Адрес DB смещения категории 3 больше, чем максимально допустимый адрес DB
26	DB смещения категории 4 имеет тот же адрес, что и DB смещения категории 2
27	DB смещения категории 4 имеет тот же адрес, что и DB смещения категории 3
28	Почтовый ящик отправки ПК имеет тот же адрес, что и системный DB (80)
29	Почтовый ящик отправки ПК не определен (0)
30	Адрес почтового ящика отправки ПК находится в пределах области внутреннего буфера
31	Адрес почтового ящика отправки ПК больше, чем максимально допустимый адрес DB
32	Почтовый ящик отправки ПК имеет тот же адрес, что и DB смещения категории 1
33	Почтовый ящик отправки ПК имеет тот же адрес, что и DB смещения категории 2
34	Почтовый ящик отправки ПК имеет тот же адрес, что и DB смещения категории 3
35	Почтовый ящик отправки ПК имеет тот же адрес, что и DB смещения категории 4
36	Почтовый ящик отправки ПК имеет тот же адрес, что и второй DB смещения категории 1
37	Зарезервированное слово DW 9 или DW 10 не равно 0
38	Зарезервированное слово DW 9 или DW 10 не равно 0
39	Зарезервированное слово DW 9 или DW 10 не равно 0
40	Зарезервированное слово DW 9 или DW 10 не равно 0
41	Зарезервированное слово DW 9 или DW 10 не равно 0
42	Зарезервированное слово DW 9 или DW 10 не равно 0
43	Зарезервированное слово DW 9 или DW 10 не равно 0
44	Зарезервированное слово DW 9 или DW 10 не равно 0
45	Зарезервированное слово DW 9 или DW 10 не равно 0
46	Зарезервированное слово DW 9 или DW 10 не равно 0

№ ошибки.	Описание
47	Отсутствует минимальная граница числа сообщений при выбранном режиме оптимизированного приема.
48	Не определен тип ПЛК
49	Зарезервированное слово DW 14 не равно 1
50	Номер ПЛК для заголовка сообщений не определен
51	Номер ЦПУ для заголовка сообщений не определен
52	Номер ЦПУ больше, чем максимально допустимый (значение должно быть между 1 и 4)

Таблица 15

5.2.6 Пример конфигурации для аварийной системы S5

Описание

Аварийная система S5 должна быть сконфигурирована для следующих категорий сообщений:

Категория	Определение: Категория сообщений
1	1200 сообщений (с номерами от 10000 до 11199) Сообщения с 11000 по 11199 – низкой активности.
2	Сообщений не ожидается.
3	11 сообщений (с номерами от 30000 до 30010)
4	Сообщений не ожидается.

Во всех сообщениях должна быть проставлена временная метка.
Планируется использовать 135U с номером ПЛК 1 и номером ЦПУ 1.

5.2.6.1 Параметризация DB 80

Категория	Макс. число	Размер блока параметров	Число блоков в DB параметров	Макс. число DB параметров
1	1008 / 2016	-	-	-
2	1008	2 DW	128	8
3	1008	5 DW	51	20
4	1008	7 DW	36	28

DB 81 используется в качестве почтового ящика отправки.

При идентичной структуре существующих сообщений, средняя длина блока (с датой и временем):

$$(1200 * 5 + 11 * 10) / (1200 + 11) = 5.05$$

Предположение:

Аварийная система S5 должна обеспечить прием до 100 сообщений за один цикл ПЛК и работать в режиме оптимизированного приема начиная с 30 сообщений..

$$5 \text{ DW/сообщ.} * 100 \text{ сообщ.} = 500 \text{ DW}$$

$$(500 \text{ DW}) / (256 \text{ DW/DB}) = 1.95 \text{ DB}$$

В результате получаем четыре блока данных для буфера FIFO, так как один или два блока данных следует добавить для режима оптимизированного приема сообщений. Буфер FIFO начинается с блока данных с адресом 82, и заканчивается на DB 85.

Для того чтобы обеспечить резерв для будущего расширения буфера FIFO, блок данных категории 1 размещается в DB 88 и DB 89 (для более чем 1008 сообщений категории 1).

DB 90 становится блоком данных смещения для сообщений категории 3. Блок данных параметров сообщений категории 3 может вместить до 51 блока параметров, и если вычесть 11 используемых блоков, остается 40 блоков для последующего расширения при использовании только одного блока данных (DB 91).

DW	Описание	Значение
0	Адрес DB начала внутреннего FIFO	82
1	Адрес DB конца внутреннего FIFO	85
2	0: без временной метки 1: с временной меткой	1
3	DB смещения для сообщений категории 1	88
4	1: один DB смещения категории 1 2: два DB смещения категории 1	2
5	DB смещения для сообщений категории 1	0
6	DB смещения для сообщений категории 1	90
7	DB смещения для сообщений категории 1	0
8	Зарезервировано	0
9	Зарезервировано	0
10	Адрес DB: почтовый ящик отправки ЦПУ -> ПК	81
11	1: оптимизированный прием (АСОР)	1
12	АСОР начинается с n сообщений	30
13	Тип ПЛК (115/135/155)	135
14	Зарезервировано	1
15	Номер ПЛК: с 1 по 255; Номер ЦПУ: с 1 по 4	1, 1
16	Зарезервировано	0
17	Зарезервировано	0
18	Зарезервировано	0
19	Зарезервировано	0
20	Ошибка четности при проверке на правдоподобность	0

В блоке данных 100 используется область с DW 10 по DW 20 для синхронизации времени.

В блоке данных 101 используется область с DW 0 по DW 255 для приема команд.

5.2.6.2 Установка блоков данных

Создание блоков данных DB 81 - DB 85, DB 88 - DB 91 и DB 101 с DW 0 - DW 255.

Создание блока данных DB 100 с DW 0 - DW 20.

5.2.6.3 Инициализация блоков данных смещений

Категория сообщений 1

DB 88 и DB 89 предоставляются для категории 1. DB 88 содержит сообщения с номерами с 10000 по 11007 и DB 89 с 11008 по 11199.

Всего для конфигурации - 1200 сообщений категории 1.

См. главу Адрес последнего блока статуса сигнала

Смещение сообщения = Номер сообщения – Основной номер сообщения = от 0 до 1199

1-й DW смещения:

Адрес последнего блока статуса сигнала: DW 252

2-й DW смещения:

Адрес последнего блока статуса сигнала: DW 252

$$1200 - 1008 = 192$$

$$192 / 16 = 12$$

$$192 \% 16 = 0$$

$$\begin{aligned} \text{Адрес последнего блока} \\ \text{статуса сигнала в блоке} \\ \text{данных смещения 2} \end{aligned} = 12 * 4 = 48$$

DB 88:

DW	Описание	Значение
DW 0	Не назначено	
DW 1	Основной номер сообщения	10000
DW 2	Адрес последнего DW	252
DW 3	Не назначено	

DB 89:

DW	Описание	Значение
DW 0	Не назначено	
DW 1	Основной номер сообщения	11018
DW 2	Адрес последнего DW	48
DW 3	Не назначено	

См. главу Смещение сообщения / состояния сигнала в сообщениях
Сообщения с 11000 по 11199 – сообщения с низкой активностью.
Позиция бита бездействия в сообщении с номером 11000:

$$\text{Смещение:} \quad 11000 - 10000 = 1000$$

$$\begin{aligned} \text{Начало блока статуса сигнала:} & \quad (\text{Смещение сообщения} / 16 + 1) * 4 = (62 + 1) * 4 * DW 252 \\ \text{Слово данных бита} & \quad DW 253 \\ \text{бездействия:} & \end{aligned}$$

$$\text{Бит:} \quad \text{Смещение \% 16} = 8$$

$$\text{Блок данных:} \quad \text{Блок данных смещения} = DB 88$$

Позиция бита бездействия в сообщении с номером 11199:

Смещение:: $11199 - 10000 = 1199$
Начало блока статуса сигнала: $(\text{Смещение} / 16 + 1) * 4 = (74 + 1) * 4 = 300$
 $300 - 252 =$
 48
Слово данных бита бездействия: $DW\ 49$
Бит: $\text{Смещение} \% 16 = 15$
Блок данных: $\text{Блок данных смещения} + 1 = DB\ 89$

Должны быть изменены следующие биты бездействия:

DB 88:

DW 253: установите биты с 8-го по 15-й в 1

DB 89:

DW 5, DW 9, с DW 13 по DW 49: установите биты с 0-го по 15-й в 1

Категория сообщений 3

Для сообщений категории 3, DB 90 выступает в роли блока данных смещения для сообщений с 30000 по 30010 и DB 91 в качестве блока данных параметров.

Всего для конфигурации - 11 сообщений категории 3.

См. главу Структура блока данных параметров

Смещение = Номер сообщения – Основной номер сообщения = от 0 до 10

DB смещения:

Адрес последнего блока статуса сигнала $11 / 16 = 0$
 $11 \% 16 = 11$
Адрес последнего блока статуса сигнала $= (0+1) * 4 = 4$

DB 89:

DW	Описание	Значение
DW 0	Не назначено	
DW 1	Основной номер сообщения	30000
DW 2	Адрес последнего DW	4
DW 3	Не назначено	

Биты бездействия – в позиции 0.
См. главу Структура блока данных параметров

DB параметров

Номер сообщения 30000:
 $DW \text{ параметров} = 90 + 1 + 0 / 51 = 91$
Номер сообщения 30010:
 $DW \text{ параметров} = 90 + 1 + 10 / 51 = 91$

Начальный адрес соответствующего блока параметров = *(Смещение % Число блоков параметров на DB параметров) * Размер блока параметров*

Номер сообщения 30000:

Начальный адрес соответствующего блока параметров = $(0 \% 51) * 5 = DW$

Номер сообщения 30010:

Начальный адрес соответствующего блока параметров = $(10 \% 51) * 5 = DW 50$

DB 91: блок данных параметров для 91 для блока данных смещения 90

Номер сообщения	Значения процесса	Номер задачи	Идентификатор пакета
30000	DW 0, 1	DW 2, 3	DW 4
30001	DW 5, 6	DW 7, 8	DW 9
30002	DW 10, 11	DW 12, 13	DW 14
30003	DW 15, 16	DW 17, 18	DW 19
30004	DW 20, 21	DW 22, 23	DW 24
30005	DW 25, 26	DW 27, 28	DW 29
30006	DW 30, 31	DW 32, 33	DW 34
30007	DW 35, 36	DW 37, 38	DW 39
30008	DW 40, 41	DW 42, 43	DW 44
30009	DW 45, 46	DW 47, 48	DW 49

Номер сообщения	Значения процесса	Номер задачи	Идентификатор пакета
30010	DW 50, 51	DW 52, 53	DW 54

5.2.7 Документация на командные блоки SIMATIC S5

Назначение и функции командных блоков S5 Command Blocks

Данное ПО используется для обработки битов, байтов, слов и двойных слов в SIMATIC S5 по шине процесса (например, промышленному Ethernet). По шине процесса в SIMATIC S5 можно адресовать только байты или слова.

По умолчанию могут быть выполнены следующие операции:

- Блоки данных (DB и DX), таймеры и счетчики должны изменяться только по словам.
- Флаги, пространства входов, выходов и периферии (P и Q) должны изменяться только по байтам.

Пакет программ предоставляет требуемые функции SIMATIC S5 для реализации следующих функций работы системы WinCC по шине процесса:

- Установка инициализирующего импульса цикла OB1
- Установка, сбрасывание и инвертирование битов в DB/DX
- Установка, сбрасывание и инвертирование битов флагов
- Запись правых/левых байтов в DB/DX
- Запись слов/двойных слов в DB/DX
- Запись байтов/слов во флаги
- Запись байтов/слов в пространство периферии
- Запись байтов/слов в пространство расширенной периферии

Требуемые изменения в SIMATIC S5 производятся WinCC Control Center через нетипизированные теги по интерфейсу данных. Команды должны быть посланы S5 через этот тег. Данные команды проверяются и выполняются прямо в S5 командным интерпретатором FB 87: EXECUTE.

Данное руководство описывает применение и управление командными блоками S5 в среде SIMATIC S5. Пользователь ознакомится с функциями и блоками данных, используемых ПО, и требованиями к памяти. Затем следует детальное описание существующего интерфейса данных. Также для облегчения понимания приводится пример конфигурации.

5.2.7.1 Перечень программных блоков

Командные блоки S5 ПО SIMATIC S5 находятся на WinCC CD-ROM в файле **WINCC1ST.S5D**.

Этот файл содержит следующие функциональные блоки для командных блоков S5:

FB	Имя	Размер в байтах	Функция
FB 87	EXECUTE	152	Реализует манипуляции с битами, байтами, словами и двойными словами по шине процесса
FB 88	OPCODE	399	Используется FB 87
Всего		551	

Таблица 16

Дополнительно требуется командный блок данных размером 512 байт.

5.2.7.2 Требования к аппаратному обеспечению

Для корректной работы указанных в таблице 16 функциональных блоков требуется следующее аппаратное обеспечение:

ПЛК	ЦПУ
ПЛК 115U	ЦПУ 943, ЦПУ 944, ЦПУ 945
ПЛК 135U	ЦПУ 928А, ЦПУ 928В
ПЛК 155U	ЦПУ 946/947, ЦПУ 948

5.2.7.3 Параметры вызова FB 87: EXECUTE

Ниже приведены параметры вызова функционального блока FB 87: EXECUTE.

Имя	Исполнение	Параметр
Bez:	DBNR	E/A/D/B/T/Z: D KM/KN/KY/KC/KF/KT/KZ/KG: KF
Bez:	DBDX	E/A/D/B/T/Z: D KM/KN/KY/KC/KF/KT/KZ/KG: KF
Bez:	RIMP	E/A/D/B/T/Z: D KM/KN/KY/KC/KF/KT/KZ/KG: KY

BNR: Номер блока данных интерфейса передачи команд

DBDX: Тип исходных данных для интерфейса передачи команд

DB: Исходные данные представляют собой блок данных (DB)

DX: Исходные данные представляют собой расширенный блок данных (DX)

RIMP: Позиция бита инициализирующего импульса

RIMP: Номер флага, номер бита

5.2.8 Описание интерфейса

Ниже приведено описание следующих интерфейсов и блоков:

- Командный функциональный блок FB 87
- Командный блок данных: интерфейс передачи команд в SIMATIC S5

В SIMATIC S5 командный интерпретатор (FB 87: EXECUTE) циклически вызывается OB 1. Тип и адрес командного DB передаются как параметры. Когда приходит команда, соответствующий код операции и четыре параметра направляются FB 88: OPCODE и исполняются непосредственно. После того, как команда была выполнена, счетчик команд (DW 1) декрементируется на единицу. Процесс передачи команды и декрементирование счетчика команд продолжается до тех пор, пока не будет обработаны все поступившие команды.

Типы и адреса блока данных должны совпадать в WinCC Control Center и программе S5, и соответствующие блоки данных должны присутствовать в S5. Блок данных DB или DX и его адрес предоставляется на выбор (например, DX 234). Этот блок данных должен быть открыт пользователем до слова 255.

Для команд, хранящихся в командном блоке данных, был определен следующий синтаксис:

DW	Описание
0	Не используется
1	Число команд для выполнения
2	Код операции первой команды
3	Параметр 1 (Код операции 1)
4	Параметр 2 (Код операции 1)
5	Параметр 3 (Код операции 1)
6	Параметр 4 (Код операции 1)
7	Код операции второй команды
8	Параметр 1 (Код операции 2)
9	Параметр 1 (Код операции 2)
10	Параметр 2 (Код операции 2)
11	Параметр 3 (Код операции 2)
12	Параметр 4 (Код операции 2)
13	Код операции третьей команды
14	Параметр 1 (Код операции 3)
.....	

Описание синтаксиса реализованных команд:

Передача кода операции и параметров в командный DB

Команда	Код операции	Параметр 1	Параметр 2	Параметр 3	Параметр 4
Установка бита в DB	10	DB	DW	Бит	-
Сброс бита	11	DB	DW	Бит	-

Команда	Код операции	Параметр 1	Параметр 2	Параметр 3	Параметр 4
в DB					
Инверсия бита в DB	12	DB	DW	Бит	-
Установка правого байта в DB	15	DB	DW	Значение	-
Установка левого байта в DB	16	DB	DW	Значение	-
Запись слова в DB	17	DB	DW	Значение	-
Запись двойного слова в DB	18	DB	DW	Значение	Значение
Установка бита в DX	20	DX	DW	Бит	-
Сброс бита в DX	21	DX	DW	Бит	-
Инверсия бита в DX	22	DX	DW	Бит	-
Установка правого бита в DX	25	DX	DW	Значение	-
Установка левого бита в DX	26	DX	DW	Значение	-
Запись слова в DX	27	DX	DW	Значение	-
Запись двойного слова в DX	28	DX	DW	Значение	Значение
Установка бита флага	30	MB	Бит	-	-
Сброс бита флага	31	MB	Бит	-	-
Инверсия бита флага	32	MB	Бит	-	-
Запись байта флага	35	MB	Значение	-	-
Запись слова памяти	36	MW	Значение		-
Запись байта в периферию	45	PB	Значение	-	-

Команда	Код операции	Параметр 1	Параметр 2	Параметр 3	Параметр 4
Запись слова периферию	46	PW	Значение	-	-
Запись байта в расширенную периферию	55	QB	Значение	-	-
Запись слова в расширенную периферию	56	QW	Значение	-	-
Установка инициализирующего пульса	60	-	-	-	-

5.2.8.1 Пример конфигурации командных блоков S5

Командные блоки S5 должны быть соответствующим образом настроены. Инициализирующий импульс обеспечивает слово флага 56, бит 4. DX 237 служит в качестве командного блока данных. Удостоверьтесь, что блок данных DX 237 открыт с DW 0 по 255 в ПЛК.
 В WinCC Control Center требуемые блоки данных вводятся при указании параметров канала (например, промышленного Ethernet).

Отрывок из ОВ 1:

```
: SPA FB 87
NAME : EXECUTE
DBNR : KF +237
DBDX : KC DX
RIMP : KY 56, 4
```

5.2.9 Назначение и функции синхронизации времени S5

Данный документ описывает функции и свойства ПО SIMATIC S5:

Синхронизация времени S5

Данное ПО используется для синхронизации системных часов SIMATIC S5. Также, оно предоставляет дату/время в нужном формате для создания блоков сообщений при хронологическом приеме сообщений в аварийной системе S5.

Функциональный блок FB 86: MESS:CLOCK предоставляет текущее время S5 в формате, используемом системой хронологического приема сообщений. Эти данные находятся в системном блоке данных 80, начиная с DW 190.

Если происходит изменение статуса сигнала сообщения, сообщение идентифицируется функциональным блоком FB 80: SYSTEMFB по номеру сообщения и в нем проставляется временная метка из системного блока данных 80. Данное руководство подробно описывает применение синхронизации времени и управление ею в среде SIMATIC S5. Пользователю предлагается обзор функциональных блоков и блоков данных, используемых ПО, и информация о затратах памяти. Для облегчения понимания предоставляется пример конфигурации.

5.2.9.1 Перечень программных блоков

Данное ПО SIMATIC S5 (синхронизация времени S5) находится на WinCC CD-ROM в файле WINCC1ST.S5D.

Файл содержит следующие функциональные блоки и блоки данных:

FB	Имя	Размер в байтах	Функция
FB 86	MESS:CLOCK	1135	Синхронизация времени
Всего		1135	

Таблица 17

Область данных часов 115U: 27 DW = 54 Байта

Область данных часов 135U/155U: 12 DW = 24 Байта

Область данных для аварийной системы S5: 3 DW = 6 Байтов

5.2.9.2 Требования к аппаратному обеспечению

Для корректной работы функциональных блоков, указанных аварийной системой S5, требуется следующее аппаратное обеспечение:

ПЛК	ЦПУ
ПЛК 115U	ЦПУ 944 *, ЦПУ 945
ПЛК 135U	ЦПУ 928B
ПЛК 155U	ЦПУ 946/947, ЦПУ 948

* Только ЦПУ 944 с двумя интерфейсами PG имеет системные часы.

UDAT:

Адрес области памяти часов
UDAT = Номер DB, номер DW

ZINT:

Интервал времени в минутах для отправки сообщения синхронизации (DCF7 = 1)

ZCLOCK:

Конечная область памяти для данных времени в формате аварийной системы
ZCLOCK = Номер DB, номер DW

ZSYN:

Конечная область памяти для сообщения синхронизации времени (DCF7 = 1)
Если с аварийной системой S5 используется хронологическая система приема сообщений, специальные данные времени в соответствующем формате должны располагаться с DB 80 по DW 190 и далее.
Этот формат данных времени основывается на системном времени S5 и записывается в соответствующую область данных ZCLOCK (DB 80, DW 190 - 192).
Взаимоотношения между хронологическим отчетом и FB 86: MESS:CLOCK:

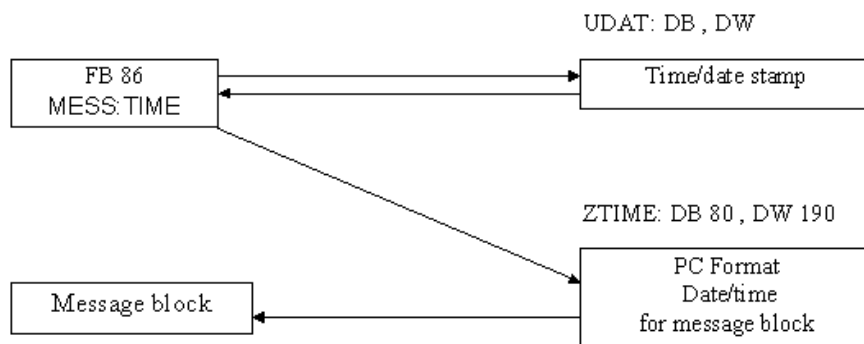


Рисунок 4

5.2.11 Форматы даты и времени

Сообщение синхронизации времени от системы (WinCC в данный момент не поддерживает сообщений времени).

Первое слово сообщения синхронизации времени содержит идентификатор (ID) источника, который посылается системой вместе со временем и датой.

Функциональный блок FB 86: MESS:CLOCK выбирает пришедшее сообщение только после того, как по этому адресу будет записан ID FFFF. В качестве уведомления о прибытии сообщения в этом слове проставляется 0. Только при прибытии нового сообщения (DW 1 = FFFF), оно снова считывается и анализируется.

Описание	Слово	Содержимое	Возможный диапазон	Комментарий
ID источника/ сообщения времени	1	FFFF		
ID сообщения	2	FFFF		Не используется
Секунды	3	00xx	xx: с 0 по 59	
Минуты	4	00xx	xx: с 0 по 59	
Часы	5	00xx	xx: с 0 по 23	
Дни	6	00xx	xx: с 1 по 31	
Месяцы	7	00xx	xx: с 1 по 12	
Года	8	00xx	xx: с 0 по 127 (1990- 2117)	Год + 1990
День недели	9	00xx	xx: с 0 по 6	Воскресенье= 0
День года	10	00xx	xx: с 1 по 365	
Летнее время, зимнее время, високосный год	11	уухх	xx: Зимнее время = 00 Летнее время = 01 уу: Високосный год Текущий год = 00 Прошлый год = 01 Два года назад = 02 Три года назад = 03	

Таблица 18

1.1.1.1 Область данных часов ЦПУ 944, ЦПУ 945

Номера слов являются относительными. Действительное расположение области определяется параметрами вызова: UDAT = Номер DB., Номер DW FB 86: MESS:CLOCK.

DW	Location
0	internal tags
1	
2	
3	
4	current time
5	
6	
7	
8	time operating range
9	
10	
11	
12	reserve - alarm time
13	
14	
15	
16	reserve - operating hours
17	
18	
19	
20	
21	
22	time / date after RUN / STOP
23	
24	
25	
26	

Таблица 19
Текущее время в области данных часов:

DW	Слово/левое	Слово/правое
4	---	День недели
6	День	Месяц
7	Год	АМ/РМ (Бит 7), Час
8	Минута	Секунда

Рисунок 5

Область установки в области данных часов:

DW	Слово/левое	Слово/правое
9	Високосный год	День недели
10	День	Месяц
11	Год	АМ/РМ (Бит 7), Час
12	Минута	Секунда

Рисунок 6

5.2.11.2 Область данных часов ЦПУ 928В, ЦПУ 948

Номера слов данных являются относительными. Действительное расположение области определяется параметрами вызова: UDAT = Номер DB., Номер DW FB 86: MESS:CLOCK.

DW	Location
0	internal tags
1	
2	
3	
4	current time
5	
6	
7	
8	time operating range
9	
10	
11	

Рисунок 7

Текущее время в области данных часов:

DW	Слово/левое								Слово/правое							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	Секунды								0							
5	Форма т		Часы						Минуты							
6	Число месяца								День недели						0	
7	Год								Секунда							

Рисунок 8

Текущее время в области установки

DW	Слово/левое								Слово/правое							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8	Секунды								0							
9	Форма		Часы						Минуты							

DW	Слово/левое		Слово/правое	
	т			
10	Число месяца		День недели	0

Рисунок 9

5.2.11.3 Область данных часов ЦПУ 946, ЦПУ 947

Номера слов данных являются относительными. Действительное расположение области определяется параметрами вызова: UDAT = Номер DB., Номер DW FB 86: MESS:CLOCK.

DW	Location
0	internal tags
1	
2	
3	
4	current time
5	
6	
7	
8	time operating range
9	
10	
11	

Рисунок 10

Текущее время в области данных часов:

DW	Слово/левое		Слово/правое	
4	10 секунд.	1 секунда.	1/10 секунды.	1/100 с.
6	10 часов	1 час	10 минут	1 минута
7	10 дней	1 день	День недели	0
8	10 лет	1 год	10 месяцев	1 месяц

Рисунок 11

Текущее время в области установки:

DW	Слово/левое		Слово/правое	
9	10 секунд.	1 секунда.	1/10 секунды.	1/100 с.
10	10 часов	1 час	10 минут	1 минута
11	10 дней	1 день	День недели	0
12	10 лет	1 год	10 месяцев	1 месяц

Рисунок 12

5.2.11.4 Часовые форматы данных для блоков сообщений

Номера слов являются относительными. Действительное расположение области определяется параметрами вызова: ZCLOCK = Номер DB., Номер DW FB 86: MESS:CLOCK.

Если в аварийной системе S5 используется хронологический прием сообщений, данные из DB 80, DW 190 должны быть введены в параметр ZCLOCK.

Дата и время предоставляются в двоичном формате функциональным блоком FB 86: MESS:CLOCK для обработки сообщений:

Текущее время в области установки:

Описание	Слово	Биты	Возможный диапазон	Комментарий
1/100 с.	1	0 - 6	от 0 до 99 (0 - 990 мс.)	С дискретностью 10 мс.
Секунды	1	7 - 12	от 0 до 59	
Минуты	0	0 - 5	от 0 до 59	
Часы	0	6 - 10	от 0 до 23	
Дни	2	0 - 4	от 1 до 31	
Месяц	2	5 - 8	от 1 до 12	
Год	2	9 - 15	от 0 до 127 (1990-2117)	Год + 1990

Рисунок 13

DW3: Время



DW4: Время



DW4: Дата

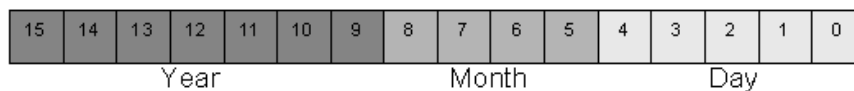


Рисунок 14

5.2.12 Описание интерфейса

Для использования ПО синхронизации времени S5 пользователь должен:

- определить параметры вызова FB 86: MESS:CLOCK согласно описанию в главе Параметры вызова FB 86: MESS:CLOCK
- открыть соответствующие области данных в ПЛК

Пример конфигурации

Предположим, что используется ЦПУ 944 с двумя интерфейсами PG. В этом ЦПУ должна быть настроена синхронизация времени S5 для аварийной системы S5 без настройки часов DCF77.

Области данных:

Сообщение синхронизации времени:	DB 100, DW 20 - DW 30
Область данных часов для системных часов S5:	DB 100, DW 31 - DW 47
Данные блока сообщений *:	DB 80, DW 190 - DW 192
*	Работа аварийной системы SIMATIC S5 основывается на использовании этой области данных.

При конфигурации параметров канала (например, промышленного Ethernet) системы, требуемый блок данных (DB 100, DW20 - DW 30) должен быть введен в поле синхронизации времени.

Удостоверьтесь, что DB 80 с DW 0 по DW 255 и DB 100 с DW 0 по DW 47 открыты.

Отрывок OB 1:

```
: SPA FB 86
NAME : MELD: UHR
CPUT : KF +1
DCF7 : KF +0
QTYP : KF +0
QSYN : KY 100, 20
UDAT : KY 100, 31
ZINT : KF +0
ZUHR : KY 80, 190
ZSYN : KF +0
```


5.2.13 Взаимодействие с аварийной системой WinCC

При взаимодействии аварийной системы WinCC с блоком сообщений S5 должно приниматься во внимание следующее:

В блоке отправки S5, в качестве числа передаваемых слов должно быть введено 256. В Control Center для устройства канальной передачи S5 должен быть сконфигурирован новый драйвер соединения. В закладке Connection (Связь) установите параметр read function fetch (выборка функции чтения) в passive (пассивно).

Для обмена данными с аварийной системой для каждого ПЛК должны быть созданы два нетипизированных тега.

Первый отвечает за прием сообщений.

Установите его адресацию следующим образом. Область данных: DB, Номер DB: xx, Адрес: слово, DW: 0, тип данных Raw: событие

Следующий блок служит для отправки информации подтверждения.

Установите его адресацию следующим образом. Область данных: DB, Номер DB: 80, Адрес: слово, DW: 90, тип данных Raw: событие

В аварийной системе тег события соединен с нетипизированным тегом приема (в этом случае битовая информация не имеет значения).

Тег подтверждения соединен с тегом отправки (в этом случае битовая информация не имеет значения).

В качестве DLL формата вводится файл S5STD.NLL.

Совет: с помощью Мастера соединения все необходимые сообщения можно соединить за один раз.

Только неотрицательные числа с фиксированной точкой могут использоваться в качестве значений процесса. Числа с плавающей точкой не поддерживаются.

S5		WinCC
prozess value	⇒	prozess value 1
job description	⇒	prozess value 2
batch description	⇒	prozess value 3
reserve	⇒	prozess value 4

5.3 Интерфейс с динамической библиотеки формата (Format DLL) для системы регистрации аварийных сообщений (Alarm Logging) и системы регистрации тегов (Tag Logging)

Назначение

Приложения, использующие систему регистрации тегов/аварийных сообщений получают данные процесса из менеджера данных WinCC. В зависимости от типа связи с процессом,

- при передаче данных используются различные DLL канала
- данные процесса хранятся в сообщениях (нетипизированных тегах) с разной структурой

Тем не менее, системы регистрации тегов/аварийных сообщений обрабатывают данные процесса одинаковым способом, вне зависимости от типа используемой связи. Для каждого типа связи используется отдельная DLL формата, которая в точности знает структуру конкретных сообщений и, таким образом, достигается универсальность обработки данных для системы регистрации аварийных сообщений и системы регистрации тегов.

По существу, DLL формата связана с DLL канала – и, как и DLL канала, DLL формата можно легко добавить в систему или удалить из нее. Тем не менее, она не имеет прямого интерфейса с соответствующей DLL канала.

Данный документ описывает интеграцию и интерфейс каждой DLL формата с приложениями системы регистрации аварийных сообщений и системы регистрации тегов WinCC. Этот документ создавался при черновом описании DLL формата S7PMC, поэтому термин DLL формата S7PMC, по большей части, - синоним термина DLL формата.

Основной процесс

DLL формата S7PMC представляет собой группу программ, которая реализует интерфейсы к приложениям систем регистрации тегов/аварийных сообщений. DLL формата S7PMC обрабатывает специфичные для S7PMC функции аварийной системы и системы регистрации тегов.

Система регистрации аварийных сообщений и система регистрации тегов регистрируются в DLL формата, используя начальный вызов (start call). При этом в стартовую структуру DLL формата передаются определенные параметры, и их свойства принимаются через идентификаторы (ID).

Для обработки функций S7PMC в режиме реального времени требуется передача данных в двух направлениях:

- от OS к ПЛК: (отправка задач logon/logoff, подтверждения)
- от ПЛК к OS: (прием сообщений и архивированных данных)

Через инициализирующий вызов система регистрации тегов/аварийных сообщений сообщает сконфигурированные имена архивных тегов и номера сообщений S7PMC DLL. Для этого DLL формата (WinCC) должна зарегистрироваться в ПЛК.

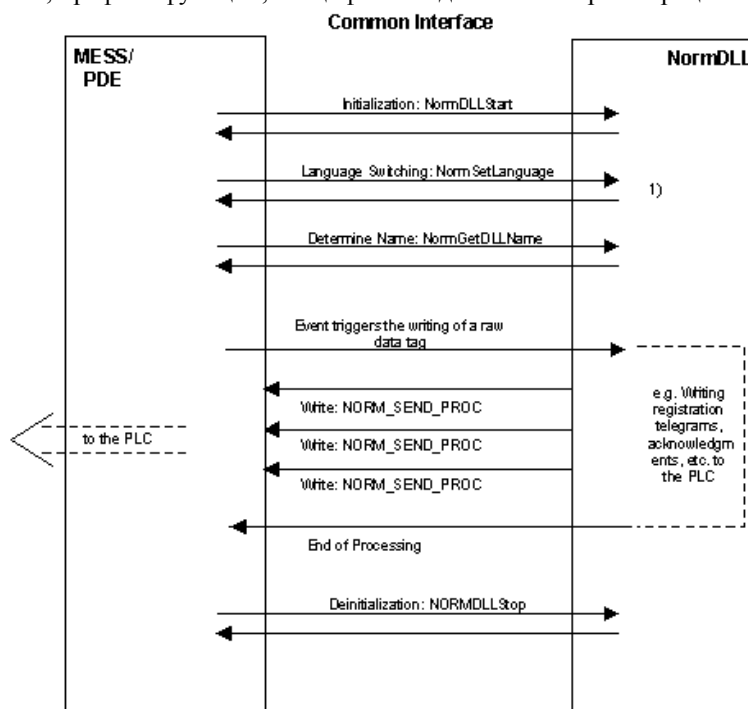
Инициализирующий вызов может быть обработан в любое время.

DLL формата вызывается системой регистрации тегов/аварийных сообщений для деинициализации, возврата ресурсов и т.п.

5.3.1 Разделяемые интерфейсы для системы регистрации аварийных сообщений (Alarm Logging) и системы регистрации тегов (Tag Logging)

Общие функции DLL формата, одинаковые для системы регистрации аварийных сообщений и системы регистрации тегов сгруппированы в разделяемый (shared) интерфейс. Все имена функций начинаются с NORM...

(Префикс функций, специфичных для системы регистрации аварийных сообщений: Mld..., префикс функций, специфичных для системы регистрации тегов: Pde...)

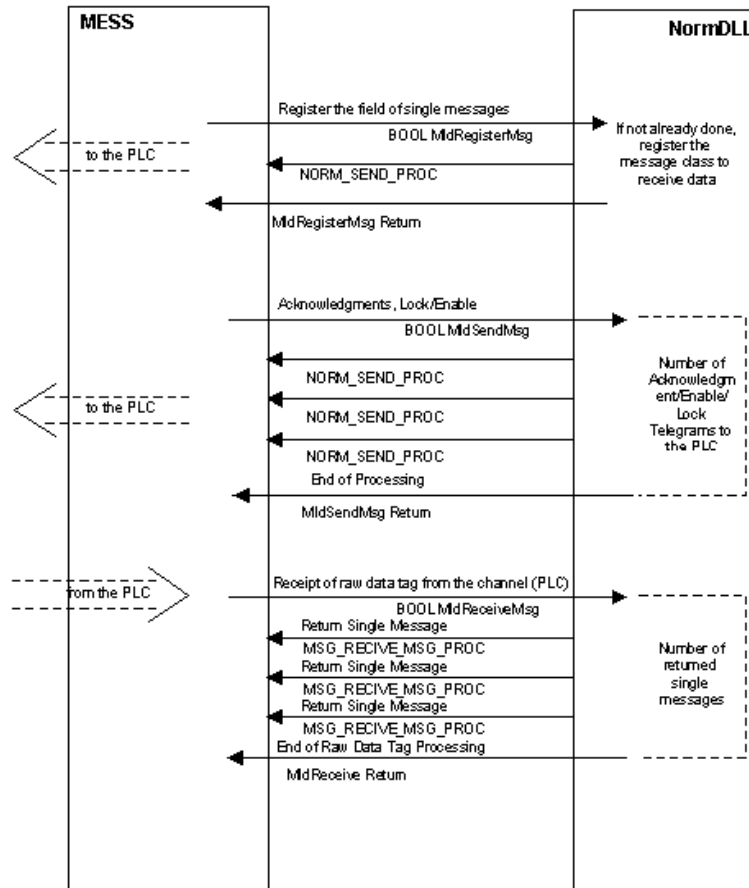


1) The language switch is only required for format DLLs that contain a dialog box

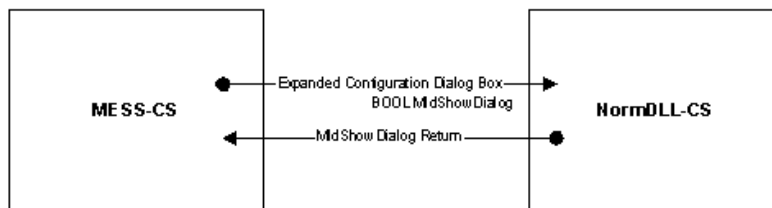
MELD = система регистрации аварийных сообщений

PDE = система регистрации тегов

Специфичные для регистрации аварийных сообщений дополнения
Система исполнения

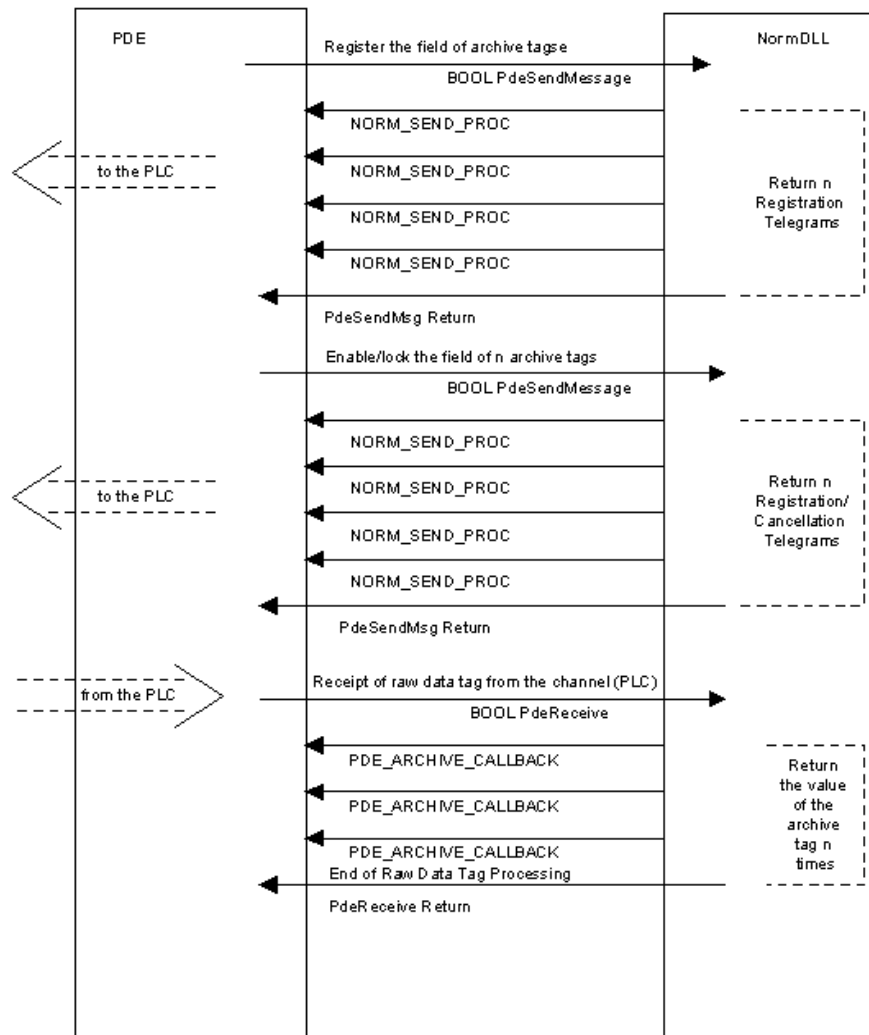


Расширенный диалог конфигурации

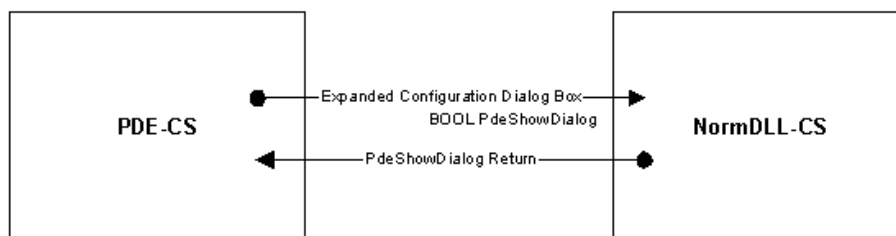


5.3.2 Дополнения, специфичные для системы регистрации тегов (Tag Logging)

Система исполнения



Расширенный диалог конфигурации



5.3.3 API функции динамической библиотеки формата (Format DLL) WinCC

DLL формата подразделяется на следующие секции:

- Инициализация DLL формата
- Инициализация операционной системой во время загрузки DLL формата (LibMain)
- Запрос свойств DLL формата
- Запрос имени DLL формата
- Прекращение работы DLL формата
- Прекращение работы системой регистрации тегов/аварийных сообщений
- Выгрузка операционной системой
- Расширения конфигурации
- Диалоговые расширения во время конфигурации сообщений
- Диалоговые расширения во время конфигурации архивных тегов
- Online службы
- Регистрация всех специфичных для DLL формата объектов (сообщений, архивных тегов)
- Переключение языка
- Форматирование
- Форматирование сообщений
- Форматирование архивных тегов

5.3.3.1 Инициализация динамической библиотеки формата (Format DLL)

Инициализация во время операции загрузки

Приложения систем регистрации тегов/аварийных сообщений загружают DLL формата WinCC с помощью системного вызова LoadLibrary. DLL формата загружается операционной системой и инициализируется через стандартные механизмы. При этом определяются все адреса входов DLL формата.

5.3.3.2 Запрос свойств динамической библиотеки формата (Format DLL)

Система регистрации аварийных сообщений и система регистрации тегов регистрируются в соответствующей DLL формата с помощью вызова NormDLLStart. Он предназначен для обмена информацией между DLL формата и приложением.

NormDLLStart

```
#include <winccnrm.h>

BOOL NormDLLStart(
    LPVOID lpUser,
    BOOL bModeRuntime,
    PNORM_STARTSTRUCT pcis,
    PCMN_ERROR lpError);
```

Параметр	Описание
lpUser	Указатель на данные приложения, перенаправляется возвратной функции без изменений
bModeRuntime	TRUE если DLL формата запущена в режиме исполнения, FALSE если в режиме конфигурации, в данный момент не проверяется DLL формата
pcis	Указатель на стартовую структуру
lpError	Указатель на структуру ошибки в WinCC по умолчанию

Возвращаемое значение	Описание
TRUE	Без ошибки
FALSE	Ошибка в функции API, описание причины ошибки можно получить через указатель lpError

NORM_STARTSTRUCT

Компонент	Описание	I/O
dwSize	Размер структуры в байтах	O
lpstrProjectPath	Путь к выбранному в данный момент проекту	I
NORM_SEND_PRO pfnWriteRwData	Указатель на возвратную функцию приложения, с которой DLL формата посылает нетипизированный тег через Data Manager в ПЛК.	I
dwAppID	Идентификатор (ID) приложения ID: 1 = Система регистрации аварийных сообщений 2 = Система регистрации тегов 3 = USER (пользователь, зарезервировано для будущих применений, в данный момент не используется)	I
dwLocalID	Установка языка во время вызова	I
dwNormCap	Свойства DLL формата в соответствии с таблицей, приведенной ниже	O

Возвратная функция для передачи нетипизированных тегов в WinCC Data Manager имеет следующий вид:

```
typedef BOOL(*NORM_SEND_PROC)(
    LPDM_VAR_UPDATE_STRUCT    lpDmVarUpdate,
    DWORD                    dwWait,
    LPVOID                    lpUser,
    LPCMN_ERROR               lpError );
```

Параметр	Описание
lpDmVarUpdate	Указатель на нетипизированный тег
dwWait	Идентификатор, определяющий, должно приложение ожидать завершения вызова write (запись) или нет: WAIT_ID_NO с SET_VALUE WAIT_ID_YES с SET_VALUE_WAIT
lpUser	Указатель на данные приложения, см. описание вызова NormDLLStart
lpError	Указатель на структуру ошибки в WinCC по умолчанию

Возвращаемое значение	Описание
TRUE	Без ошибки
FALSE	Ошибка в функции API, описание причины ошибки можно получить через указатель lpError

Каждому свойству назначается бит в соответствии со следующей таблицей:

DEFINE	Битовая маска		Описание
NORMCAP_DIALOG	0x00000001	Установлен	DLL формата предоставляет специальный диалог
		Сброшен	DLL формата не предоставляет специальный диалог
NORMCAP_REENTRANT	0x00000002	Установлен	DLL формата реентерабельна
		Сброшен	DLL формата не реентерабельна
NORMCAP_MSG_FREE_LOCK	0x00000004	Установлен	Logon/logoff возможен для сообщений
		Сброшен	Logon/logoff невозможен для сообщений
NORMCAP_ARC_FREE_LOCK	0x00000008	Установлен	Logon/logoff возможен для архивных тегов
		Сброшен	Logon/logoff невозможен для архивных тегов
NORMCAP_MSG_GENERIC	0x00000010	Установлен	Сообщения могут быть сконфигурированы обычным образом
		Сброшен	Сообщения могут быть сконфигурированы обычным образом
NORMCAP_ARC_GENERIC	0x00000020	Установлен	Сообщения могут быть сконфигурированы обычным образом
		Сброшен	Сообщения могут быть сконфигурированы обычным образом

5.3.3.3 Запрос имени динамической библиотеки формата (Format DLL)

NormGetDLLName

```
#include <winccnorm.h>
```

```
LPTSTR NormGetDLLName( void );
```

Возвращаемое значение	Описание
LPTSTR	Указатель на строку, содержащую имя DLL формата в виде простого текста, имя зависит от текущего языка

5.3.4 Завершение работы динамической библиотеки формата (Format DLL)

Завершение работы системой регистрации тегов/аварийных сообщений

Система регистрации тегов и система регистрации аварийных сообщений посылают DLL формата уведомление о том, что приложение закрывается. Ресурсы затем корректно возвращаются в DLL формата.

NormDLLStop

```
#include <winccnrm.h>
BOOL NormDLLStop (void);
```

Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API

Выгрузка операционной системой

Не требуется никаких специальных действий.

5.3.4.1 Расширения конфигурации

Для объектов S7PMС требуются особые спецификации. Эти спецификации вначале запрашиваются в диалоге с использованием стандартных способов (без MFC) и отправляются непосредственно в номер сообщения WinCC или архивный тег. Это означает, что DLL формата не должна хранить эти спецификации или оперировать с ними сама. Для гарантии уникальности номера сообщения или архивного тега по всему проекту, номеру сообщения или архивному тегу должен быть назначен нетипизированный тег. Информация о данном назначении является неотъемлемой частью номера сообщения или имени архивного тега.

5.3.4.2 Диалоговое расширение во время конфигурации сообщений S7PMС

DLL формата содержит функцию API для назначения специфичного для S7PMС номера сообщения. Данная функция вызывается аварийной системой системы конфигурации (CS) при назначении параметров отдельным сообщениям, принадлежащим DLL формата S7PMС. Номер сообщения, назначаемый DLL формата S7PMС, состоит из двух частей.

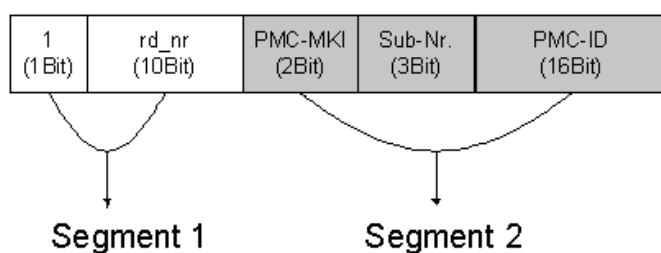
Часть 1:

Номер, который однозначно идентифицирует ЦПУ ПЛК во всем проекте (число в нетипизированном теге).

Часть 2:

Номер, принадлежащий сообщению в ПЛК, которое однозначно идентифицирует сообщение в ЦПУ ПЛК (специфично для DLL формата). В конфигурационном диалоге нужно сделать следующие настройки для установки номера сообщения:

Структура номера сообщений S7PMC (32 бита)

**Для части 1**

Каждое сообщение связано с нетипизированным тегом, который идентифицирует ЦПУ ПЛК. Для того чтобы назначить номеру сообщения нетипизированный тег, должно быть выполнено следующее.

Имя нетипизированного тега для S7PMC – и всех типов соединений с DLL формата – имеет следующую структуру:

@rd_alarm#rd_nr

@rd_alarm# Фиксированная неотъемлемая часть имени нетипизированного тега для DLL библиотек формата

rd_nr Десятичное число от 0 до 1023 для идентификации нетипизированного тега (без предваряющих нулей)

Самый значимый бит номера сообщения устанавливается для тех номеров сообщений, которые назначаются DLL формата (внешне). Эти сообщения могут быть обработаны только соответствующей DLL формата, т.е. номер сообщения не может быть изменен через конфигурационный диалог системы регистрации аварийных сообщений.

Для части 2

Данная часть номера сообщения может быть назначена только соответствующей DLL формата. Для S7PMC она имеет следующий смысл:

MKI	Класс сообщения; один из следующих: SCAN (1) ALARM/NOTIFY (2) ALARM_8P/ALARM_8 (2) LTM (3)
Sub-No.	Номер подсообщения, применим только к ALARM_8 и ALARM_8P: 1...8
PMC-ID	Номер сообщения PMC (параметр ввода блока EV-ID): 1...16386 для сообщений класса SCAN и ALARM/NOTIFY или ALARM_8P/ALARM_8 1...7 для сообщений класса LTM

MldShowDialog

```
#include <winccnrm.h>

BOOL WINAPI MldShowDialog(
    HWND          hwnd,
    LPMSG_CSDATA_GENERIC lpMsgCS,
    LPDM_PROJECT_INFO lpDMProjectInfo,
    LPCMN_ERROR    lpError );
```

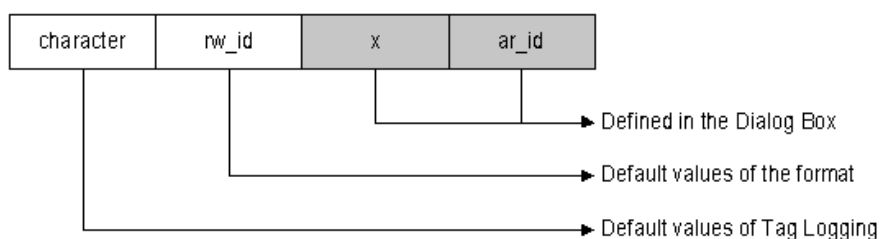
Параметр	Описания
hwnd	handle окна
lpMsgCS	Указатель на данные отдельного сообщения
lpDMProjectInfo	Указатель на структуру с информацией о проекте
lpError	Указатель на структуру ошибки WinCC по умолчанию

Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API, описание причины ошибки можно получить через указатель lpError

5.3.4.3 Диалоговое расширение во время конфигурации архивных тегов

DLL формата содержит функцию API для определения специфичного для S7PMС имени архивного тега. Эта функция вызывается системой регистрации тегов системы конфигурирования (CS) при назначении параметров архивным тегам, принадлежащим соединению S7PMС. Имя архивного тега, назначаемое DLL формата S7PMС, состоит из нескольких компонентов, которые содержат, помимо остальных вещей, номер, связанный с архивом в ПЛК. По данному алгоритму, номера тегов S7PMС содержатся уникальным образом в описании архивных тегов WinCC: при этом достигается максимально быстрое назначение в процессе исполнения. Система регистрации тегов гарантирует, что имена архивных тегов полностью уникальны.

Структура имени архивного тега S7PMС (не длиннее 18 байтов)



Имя	Длина в байтах	Назначается	Описание
character	9	Системой регистрации тегов	Строка фиксированного размера, назначаемая системой регистрации тегов, состоит из имени DLL формата и неотображаемого символа # в качестве разделителя, например, для S7PMС: NRMS7PMС
rw_id	8	Системой регистрации тегов / DLL формата	Двоичный идентификатор в шестнадцатеричном представлении (с предваряющими нулями) для уникальности назначения нетипизированного тега (соединения), с которым связан номер архива. Часть имени формируется DLL формата с использованием входных параметров системы регистрации тегов.
x	1	DLL формата из CS	Специфичный для S7PMС идентификатор для различения BSEND и AR_SEND: A = AR_SEND B = BSEND
ar_id	4	DLL формата из CS	Идентификатор в шестнадцатеричном представлении (с предваряющими нулями) В зависимости от x ID: Специфичный для S7PMС номер архива AR_ID или Специфичный для S7 R_ID при BSEND

Пример имени архивного тега, сгенерированного для S7PMC: #00000001#A#0014
PdeShowDialog

```
#include <winccnrm.h>

BOOL WINAPI PdeShowDialog(
    LPVOID          hwnd,
    LPTSTR          lpzArcVarName,
    DWORD           dwArcVarNameLength,
    LPDM_VARKEY     lpVarKey,
    LPCMN_ERROR     lpError
);
```

Параметр	Описание
hwnd	handle окна
lpzArcVarName	Указатель на строку, содержащую специфичную для DLL формата часть имени архивного тега
dwArcVarNameLength	Максимальная длина специфичной для DLL формата части имени
lpVarKey	Указатель на поле Varkey нетипизированных тегов
lpError	Указатель на структуру ошибки WinCC
Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API, описание причины можно получить через указатель lpError

5.3.4.4 Online службы

Регистрация всех сообщений

Данная функция требуется по той причине, что DLL формата не имеет конфигурационной информацией об нужных сообщениях. Сообщения не посылаются ПЛК до тех пор, пока приложение (WinCC) не зарегистрируется для получения сообщений. В этот момент система регистрации аварийных сообщений вызывает функцию MldRegisterMsg для каждого нужного сообщения и таким образом передает конфигурационную информацию для отдельного сообщения в DLL формата. Кроме описания сообщения, DLL формата также получает указатель на нетипизированный тег (соединение), который назначается этому сообщению. Это означает, что DLL формата может создать таблицу в основной памяти во время исполнения, согласно которой будет определяться структура специфичных для S7PMC сообщений регистрации.
MldRegisterMsg

```
#include <winccnrm.h>

BOOL WINAPI MldRegisterMsg(
    LPDM_VARKEY     lpDMVarKey,
    LPDWORD         lpMsgNumber,
    DWORD           DwNumMsgNumber,
    LPCMN_ERROR     lpError );
```

Параметр	Описание
lpDMVarKey	Указатель на поле Varkey нетипизированных тегов
lpMsgNumber	Указатель на поле с номерами отдельных сообщений
dwNumMsgNumber	Количество номеров отдельных сообщений
lpError	Указатель на структуру ошибки WinCC
Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API, описание можно получить через указатель lpError

5.3.4.5 Регистрация всех архивных тегов

Данная функция нужна по той причине, что DLL формата не располагает конфигурационной информацией о нужных архивных тегах. Функция PdeSendMsg вызывается для определенного количества архивных тегов, и таким образом становится известна конфигурационная и иная дополнительная информация об архивных тегах.

За один вызов могут быть зарегистрированы несколько архивных тегов соединения. Система регистрации тегов пересылает одно двойное слово на архивный тег в качестве дополнительной информации для DLL формата, которое сохраняется в памяти DLL формата. Данная дополнительная информация будет использована системой регистрации тегов, как только будет нужно обработать архивные теги (в возвратной функции TagLogging_ARCHIVE_CALLBACK).

Это означает, что DLL формата может создать таблицу в основной памяти во время выполнения, по которой можно получать структуру специфичных для S7PMС сообщений регистрации для соответствующих архивов. Сообщения регистрации нужны для уведомления ПЛК о готовности приема статуса для соответствующего номера архива. После успешной процедуры регистрации ПЛК посылает архивные данные в приложение (WinCC).

PdeSendMsg

```
#include <winccnrm.h>
```

```
BOOL WINAPI PdeSendMsg(
    NORM_SEND_PROC  IpfnCallBack,
    DWORD           dwFunctionId,
    LPSZ_ARC_VAR_NAME  lpszArcVarName,
    LPDWORD         lpdwData,
    DWORD           dwNumArchVarName,
    LPDM_VARKEY     lpVarKey,
    LPVOID          lpUser,
    LPCMN_ERROR     lpError
);
```

Параметр	Описание
lpfnCallBack	Указатель на возвратную функцию, с помощью которой нетипизированный тег, сгенерированный DLL формата, передается в менеджер данных (DM). Если 0, возвратная функция вызывается из структуры INI. Адрес функции из структуры INI не совпадает с данным параметром.
dwFunctionId	Идентификатор функции FUNC_ID_REGISTER (см. таблицу ниже), эта же функция применяется ко всем перечисленным тегам
lpzArcVarName	Указатель на поле указателя, чьи элементы относятся к именам архивных тегов
lpdwData	Указатель на поле, чьи элементы содержат дополнительную информацию для архивных тегов, может быть нулем. Дополнительное значение, принадлежащее архивному тегу, передается без изменений во внутренние списки DLL формата функцией FUNC_ID_REGISTER (зарегистрировать архивный тег) и перенаправляется в Tag Logging_ARCHIVE_CALLBACK в подходящее время. Не имеет значения для других идентификаторов функций.
dwNumArchVarName	Количество имен архивных тегов, которые должны быть обработаны
lpVarKey	Указатель на поле Varkey нетипизированного тега
lpUser	Указатель на пользовательские данные, передается без изменений в возвратную функцию
lpError	Указатель на структуру ошибки WinCC
Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API, описание можно получить через указатель lpError

Возможные функции процедуры PdeSendMsg (значения из dwFunctionId):

DEFINE	Битовая маска	Описание
FUNC_ID_LOCK	0x00000001	Заблокировать архивный тег
FUNC_ID_FREE	0x00000002	Разблокировать архивный тег
FUNC_ID_REGISTER	0x00000004	Зарегистрировать архивный тег
FUNC_ID_UNREGISTER	0x00000008	Отменить регистрацию архивного тега (в данный момент не требуется)

5.3.4.6 Переключение языка

Конфигурационные диалоговые окна должны быть зависимыми от языка, иными словами, DLL формата должна распознавать установленный в данный момент язык. При инициализации настройки языка передаются в стартовую структуру.

Динамическое переключение языка также должно быть передано в DLL формата системой регистрации тегов или системой регистрации аварийных сообщений. Для этого используется следующий вызов:

NormSetLanguage

```
#include <winccnrm.h>

BOOL NormSetLanguage(
    DWORD          dwLocaleID,
    LPCMN_ERROR    lpError
);
```

Параметр	Описание
DwLocalID	Настройки языка на момент вызова
lpError	Указатель на структуру ошибки WinCC по умолчанию
Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API, описание можно получить через указатель lpError

5.3.5 Форматирование

Если приложение зарегистрировалось на ПЛК для приема сообщений или архивных данных, эти данные будут предоставлены через соответствующий нетипизированный тег. С этого момента ПЛК может посылать сообщения. Сообщения упаковываются в нетипизированные теги и посылаются в DLL формата через DLL канала, менеджер данных и соответствующее приложение (в данном случае, система регистрации тегов или аварийных сообщений); DLL берет на себя работу с нетипизированными тегами. DLL формата интерпретирует входящие данные и затем создает сообщения или архивирует данные из них.

5.3.5.1 Вывод одиночных сообщений

Содержимое нетипизированного тега может хранить n отдельных сообщений. DLL формата должно проинтерпретировать это специфичное для S7PMC сообщение и перенаправить получившиеся отдельные сообщения в систему регистрации аварийных сообщений.

Номер сообщения (EV_ID) S7PMC является частью номера сообщения WinCC. До десяти значений процесса может быть доставлено S7PMC. В данном случае, в качестве значения процесса допустим строковый тип. Это значение процесса не поддерживается системой регистрации аварийных сообщений – дополнительные значения подобного вида должны отвергаться DLL формата.

Функция MldReceiveMsg вызывается системой регистрации аварийных сообщений каждый раз, когда изменяется состояние нетипизированного тега, т.е. менеджер данных определяет, что состояние ошибки изменяется на ОК или наоборот. Изменение состояния нетипизированного тега имеет значение только для DLL формата S7PMC. Дополнительную информацию можно найти в главе Обработка в случае изменения состояния.

MldReceiveMsg

```
#include <winccnrm.h>

BOOL WINAPI MldReceiveMsg(
    MSG_RECEIVE_MSG_PROC    lpfnMsgReceive,
    LPDM_VAR_UPDATE_STRUCT  lpDMVar,
    LPVOID                   lpUser,
    LPCMN_ERROR              lpError );
```

Параметр	Описание
lpfnMsgReceive	Указатель на возвратную подпрограмму, с помощью которой отдельные сообщения, структурированные DLL формата, передаются системе регистрации аварийных сообщений
lpDMVar	Указатель на нетипизированный тег
lpUser	Указатель на пользовательские данные, передается без изменений в возвратную функцию
lpError	Указатель на структуру ошибки WinCC
Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API, описание можно получить через указатель lpError

Возвратная функция для отправки отдельных сообщений в систему регистрации аварийных сообщений имеет следующий вид:

```
typedef BOOL (*MSG_RECEIVE_MSG_PROC)(
    LPMSG_RTCREATE_STRUCT lpMsgCreate,
    DWORD dwNumMsg,
    LPVOID lpUser,
    LPCMN_ERROR lpError);
```

Параметр	Описание
lpMsgCreate	Указатель на сообщение WinCC
dwNumMsg	Количество отдельных сообщений
lpUser	Указатель на данные приложения
lpError	Указатель на структуру ошибки WinCC по умолчанию
Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API, описание можно получить через указатель lpError

5.3.5.2 Подтверждение, блокировка/разрешение сообщений

Концепция сообщений и аварий системы регистрации аварийных сообщений WinCC и S7PMC предполагает, что сообщения подтверждаются в зависимости от их конфигурации. Информация о подтверждении известна системе регистрации аварийных сообщений, но она также должна обрабатываться в области для хранения подтверждений сообщений в ПЛК. Для этого система регистрации аварийных сообщений посылает подтверждения сообщений в ПЛК с помощью DLL формата для соответствующего соединения.

На основе полученных данных, DLL формата S7PMC создает соответствующие сообщения S7PMC, которые направляются в менеджер данных возвратной функцией системы регистрации аварийных сообщений NORM_SEND_PROC.

Эта же процедура применяется, если отдельное сообщение должно быть заблокировано/разрешено системой регистрации аварийных сообщений, т.е. их создание запрещается/разрешается на стороне ПЛК.

MldSendMsg

```
#include <winccnrm.h>

BOOL WINAPI MldSendMsg(
    NORM_SEND_PROC lpfnMsgSend,
    LPMSG_SEND_DATA_STRUCT lpSendData,
    DWORD dwNumData,
    LPVOID lpUser,
    LPCMN_ERROR lpError );
```

Параметр	Описание
lpfnMsgSend	Указатель на возвратную подпрограмму системы регистрации аварийных сообщений, с помощью которой нетипизированный тег – созданный DLL формата – передается для записи в ПЛК. Параметры описаны в главе Запрос свойств динамической библиотеки формата.
lpSendData	Указатель на отправляемые данные, их структура подробнее описана ниже

Параметр	Описание
dwNumData	Количество индивидуальных задач, которые должны быть обработаны
lpUser	Указатель на пользовательские данные, передается без изменений в возвратную функцию
lpError	Указатель на структуру ошибки WinCC
Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API, описание можно получить через указатель lpError

Структура отправляемых данных системы регистрации аварийных сообщений (индивидуальная задача)

Тег	Описание
DWORD dwVarID	ID нетипизированного тега DM
DWORD dwNotify	Уведомление: возможные значения MSG_STATE_QUIT: подтвердить сообщение MSG_STATE_LOCK: заблокировать сообщение MSG_STATE_UNLOCK: разрешить сообщение MSG_STATE_QUIT_EMERGENCY: подтвердить все сообщения
DWORD dwData	Для QUIT, LOCK, UNLOCK --> номер сообщения Для EMERGENCY ACK --> не используется

5.3.5.3 Обработка в случае изменения состояния

Об изменении состояния соединения (нетипизированных тегов) должна быть уведомлена DLL формата. Это выполняется функцией MldReceiveMsg.

Изменение состояния с – на	Обработка в DLL формата S7PMC
Ошибка – ОК	Сообщения регистрации для всех классов сообщений S7PMC, в которых было сконфигурировано, по меньшей мере, одно сообщение – передаются в ПЛК. Регистрация, производимая для сообщений S7PMC, зависит от их класса. DLL формата уже знает обо всех сконфигурированных сообщениях благодаря их регистрации.
ОК – Ошибка	DLL формата должна отвергнуть активные задачи, которые уже были переданы в ПЛК, но не смогли быть полностью обработаны из-за смены состояния (отсутствуют подтверждения).

5.3.5.4 Обновление сообщений динамической библиотеки формата (Format DLL) S7PMC

В случае обновления сообщения, DLL формата S7PMC считывает состояния всех сообщений, о которых было получено уведомление (через регистрацию) и отправляет их как отдельное сообщение в систему регистрации аварийных сообщений. Следовательно, можно составить цельную картину сообщений при инициализации системы.

Обновление сообщения необходимо при:

- изменении состояния с ошибочного на ОК (это также неявно производится при инициализации)
- отправке ПЛК сообщения об обновлении сообщения. Это сообщение отправляется всем зарегистрированным участникам, если, например, было обнаружено переполнение сообщений, сообщения от других участников подтверждаются или разрешаются.

В течение обновления сообщений ПЛК считывает состояния обновления сообщений и их идентификаторы блокировки (lock IDs). Дополнительные значения и время не посылаются. В данном случае, DLL формата предоставляет текущее системное время в качестве времени для отдельного сообщения или записывает идентификатор MSG_STATE_UPDATE в состояние сообщения.

5.3.5.5 Форматирование архивных тегов

DLL формата предоставляет две функции для системы регистрации тегов:

- Вывод отдельных значений архивных тегов из содержимого нетипизированного тега
- Блокировка/разрешение архивных тегов

5.3.5.6 Вывод отдельных значений архивных тегов

Содержимое нетипизированного тега (сообщения) может состоять из *n* значений архивных тегов. DLL формата должна проинтерпретировать эти специфичные для S7PMС сообщения и направить получившиеся значения архивных тегов в систему регистрации тегов.

Для архивного тега также должны быть переданы конвертеры значений процесса. DLL формата S7PMС затем произведет требуемые преобразования значения процесса в значение архивного тега. Этот процесс использует существующие функции масштабирования WinCC. Точная процедура, однако, все равно должна быть указана.

Функция PdeReceive вызывается системой регистрации тегов каждый раз, когда изменяется состояние нетипизированного тега, т.е. менеджер данных определяет изменение состояния ошибки ОК или наоборот. Изменение состояния нетипизированного тега имеет значение только для DLL формата S7PMС.

Дополнительную информацию можно найти в главе Обработка в случае изменения состояния.

PdeReceive

```
#include <winccnrm.h>
```

```
BOOL PdeReceive (
    LPDM_VAR_UPDATE_STRUCT      lpDmVarUpdate,
    TagLogging_ARCHIVE_CALLBACK lpfnCallBack,
    LPVOID                      lpUser,
    LPCMN_ERROR                 lpError
);
```

Параметр	Описание
LpDmVarUpdate	Указатель на нетипизированный тег
LpfnCallBack	Указатель на возвратную подпрограмму, с помощью которой DLL формата передает отдельные значения архивных тегов в систему регистрации тегов.
LpUser	Указатель на пользовательские данные, передается в возвратную функцию без изменений
LpError	Указатель на структуру ошибки WinCC
Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API, описание можно получить через указатель lpError

Возвратная функция для отправки отдельных значений архивных тегов в систему регистрации тегов имеет следующий вид:

```

BOOL (*PDE_ARCHIVE_CALLBACK) (
    LPTSTR      lpzArcVarName,
    double      doValue,
    SYSTEMTIME* lpstTime
    DWORD       dwFlags
    DWORD       dwData,
    LPVOID      lpUser,
    LPCMN_ERROR lpError
);

```

Параметр	Описание
lpzArcVarName	Имя архивного тега из идентификатора нетипизированного тега
doValue	Значение архивного тега
lpstTime	Указатель на временную метку из пользовательских данных нетипизированного тега
dwFlags	Идентификаторы, точное значение которых будет определено в будущем.
dwData	Дополнительная дата, которая была предоставлена во время регистрации, передается без изменений
lpUser	Указатель на пользовательские данные, передается без изменений в возвратную функцию
lpError	Указатель на структуру ошибки WinCC
Возвращаемое значение	Описание
TRUE	Успешное выполнение функции
FALSE	Ошибка в функции API, описание можно получить через указатель lpError

5.3.5.7 Блокировка/разрешение архивных тегов

С помощью этой функции система регистрации тегов имеет возможность контролировать прием значений архивных тегов в S7PMC. Для этой цели DLL формата S7PMC формирует вызов процедуры logoff или logon соответствующего архива, и направляет этот вызов в DM через NORM_SEND_PROC.

С точки зрения DLL формата S7PMC функция блокировки/разрешения архивных тегов почти идентичны функции для регистрации архивных тегов. В обеих функциях вызывается одна и та же функция DLL формата (PdeSendMessage).

Через идентификатор функции dwFunctionId различают функции для регистрации и блокировки/разрешения: для блокировки/разрешения дополнительные данные lpdwData архивного тега не имеют значения. См. также главу Регистрация всех архивных тегов.

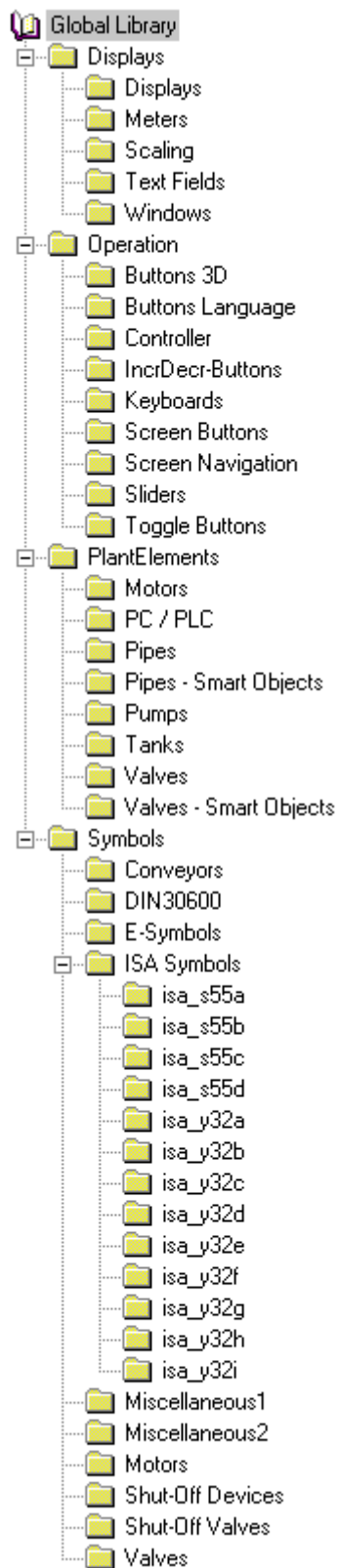
5.3.5.8 Обработка в случае изменения состояния

В случае изменения состояния соединения (нетипизированных тегов) DLL формата должна быть об этом уведомлена. Это выполняется функцией PdeReceive.

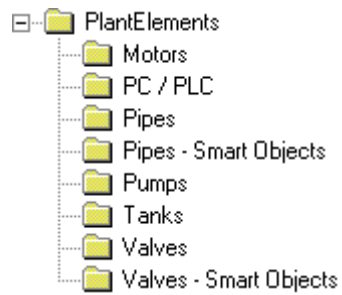
Изменение состояния с – на	Обработка в DLL формата S7PMC
Ошибка- ОК	Сообщения регистрации (logon) для всех архивных тегов

Изменение состояния с – на	Обработка в DLL формата S7PMC
	всех соединений. DLL формата уже знает обо всех сконфигурированных сообщениях благодаря их регистрации.
ОК – Ошибка	DLL формата должна отвергнуть активные задачи, которые уже были переданы в ПЛК, но не смогли быть полностью обработаны из-за смены состояния (отсутствуют подтверждения).

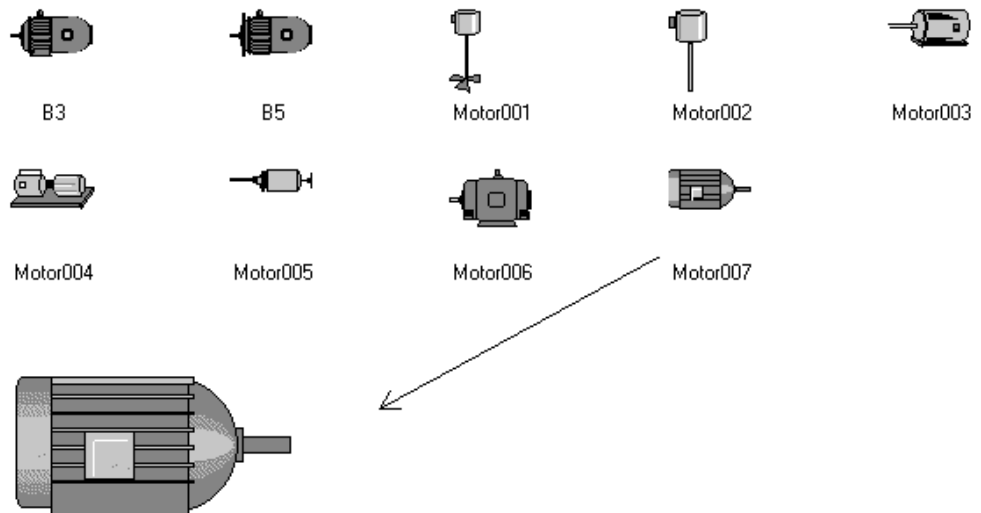
5.4 Глобальная библиотека



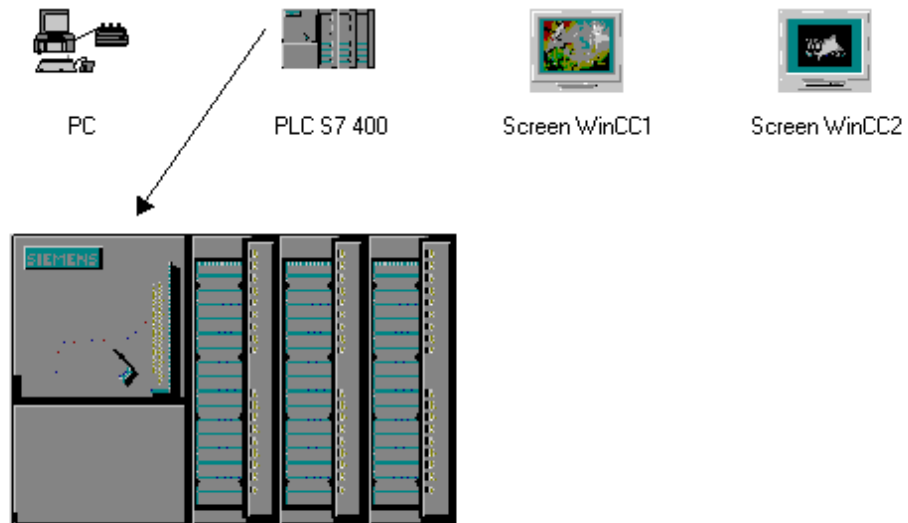
5.4.1 Системные блоки



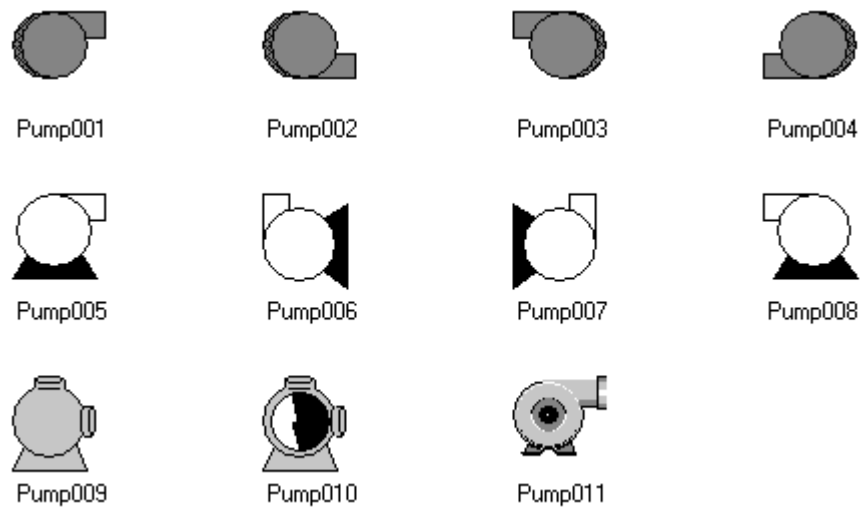
5.4.1.1 Двигатели



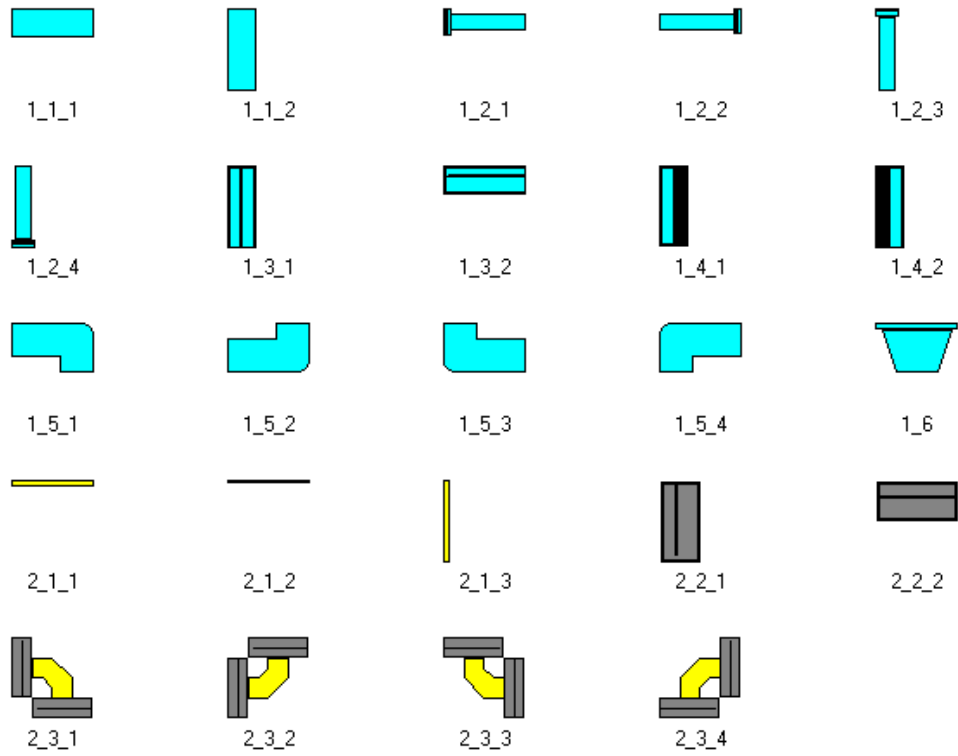
5.4.1.2 ПК/ПЛК



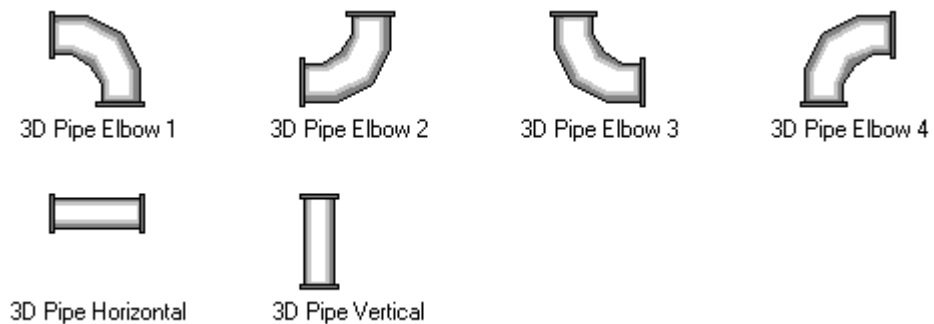
5.4.1.3 Насосы



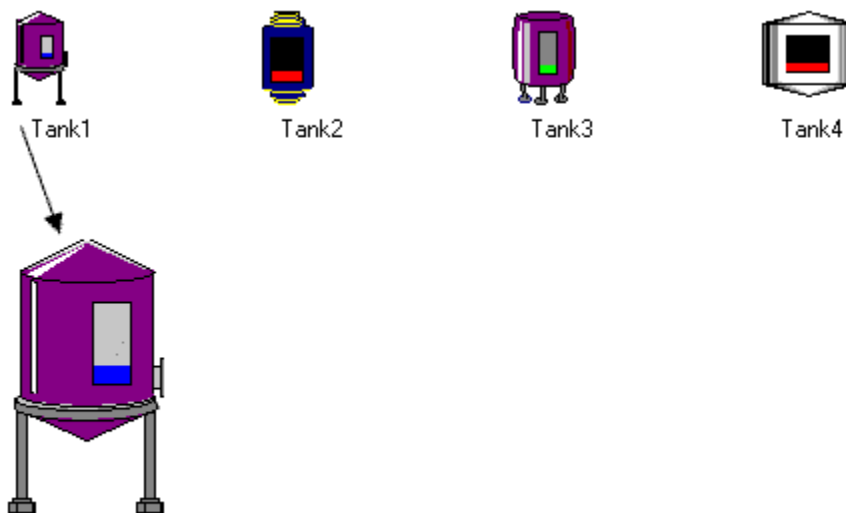
5.4.1.4 Трубы



5.4.1.5 Трубы - модифицированные объекты



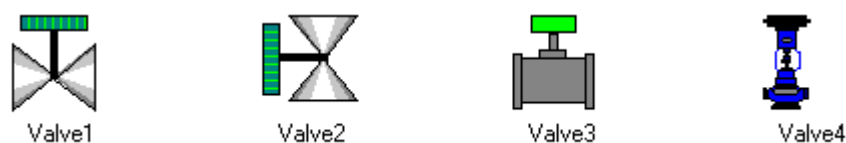
5.4.1.6 Резервуары



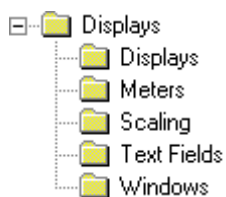
5.4.1.7 Клапаны - модифицированные объекты



5.4.1.8 Клапаны



5.4.2 Дисплеи



5.4.2.1 Дисплеи



8-Bit Display



8-Bit Display + I/O Field



Digital Output

5.4.2.2 Окна



1



2



3



4

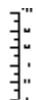


5

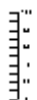


6

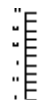
5.4.2.3 Линейки



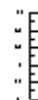
01



02



03



04

5.4.2.4 Текстовые поля



Siemens WinCC



Text



Text: Password Error

5.4.2.5 Измерительные приборы



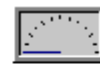
Meter1_0-100



Meter1_Min-Max

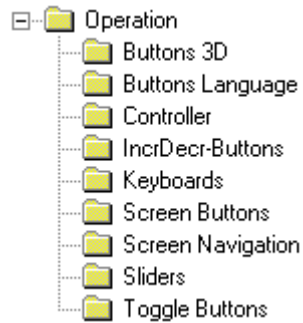


Meter2_Min-Max

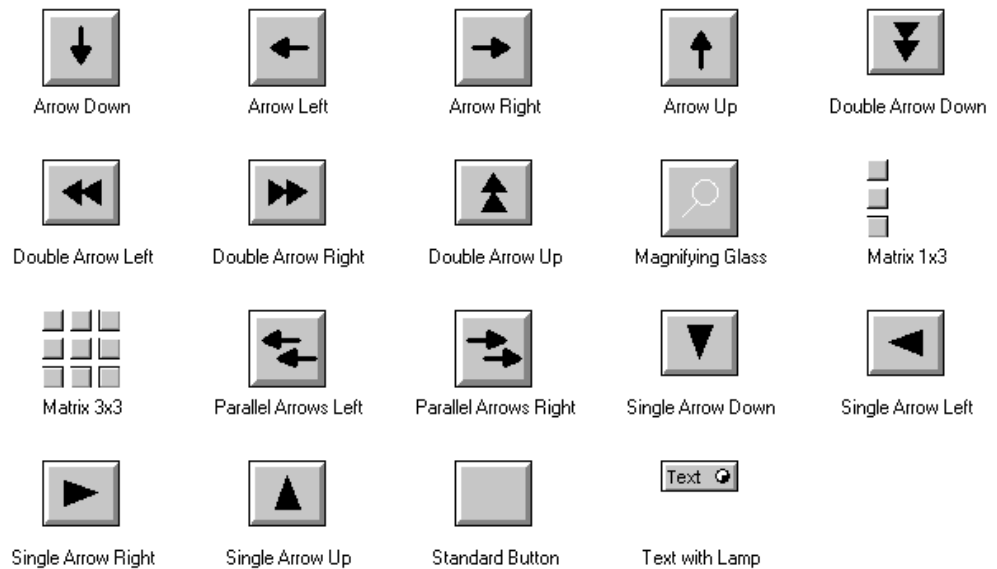


Meter3_Min-Max

5.4.3 Элементы управления



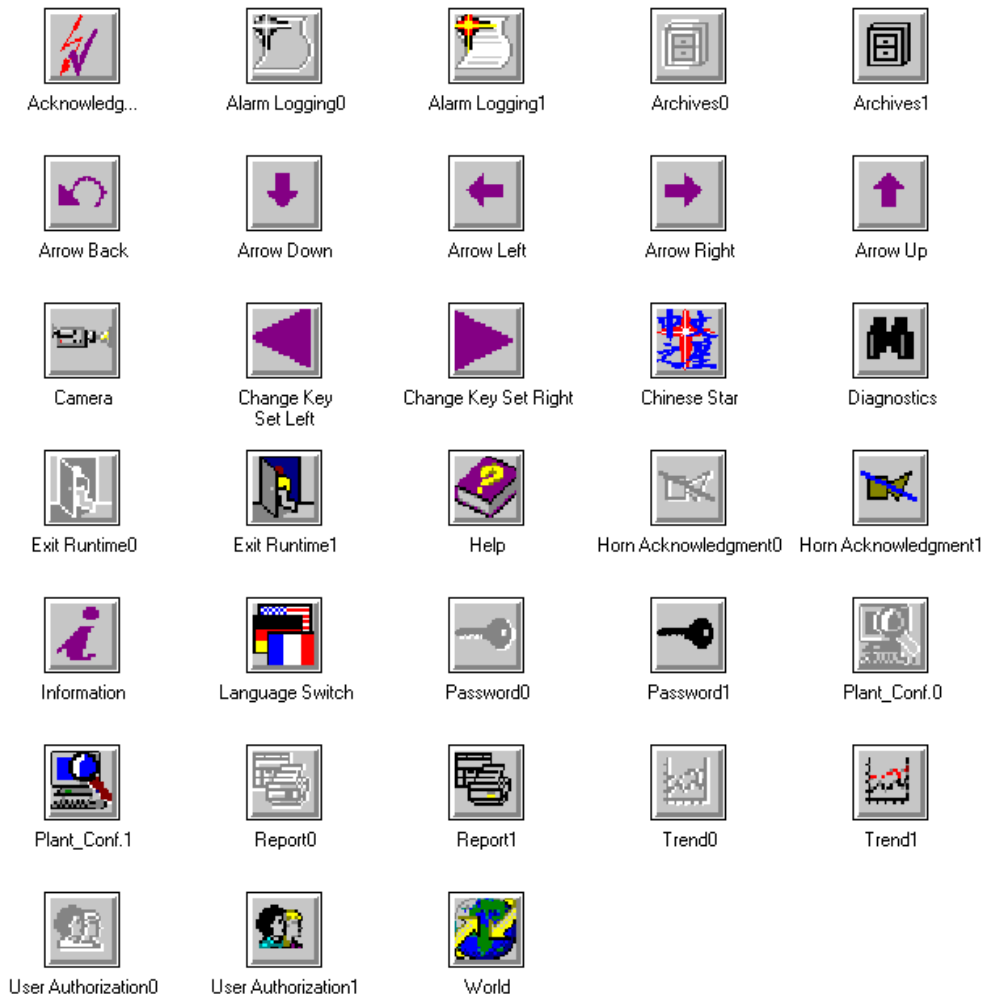
5.4.3.1 3D кнопки



5.4.3.2 Панели управления



5.4.3.3 Кнопки с изображениями



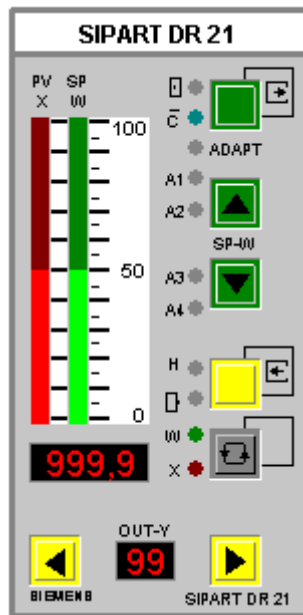
5.4.3.4 Навигация по кадрам



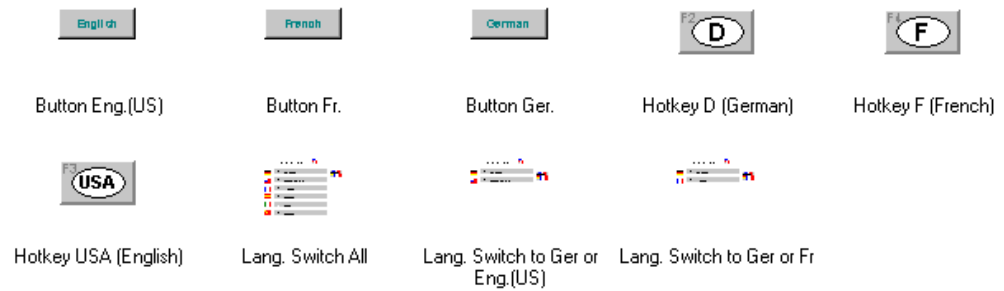
5.4.3.5 Кнопки инкремента/декремента



5.4.3.6 Контроллеры



5.4.3.7 Переключатели языка



5.4.3.8 Клавиатуры



5.4.3.9 Переключаемые кнопки



On_Off_1



On_Off_2



On_Off_3



On_Off_4



On_Off_5



On_Off_6

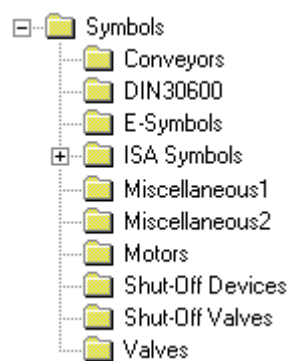


On_Off_7

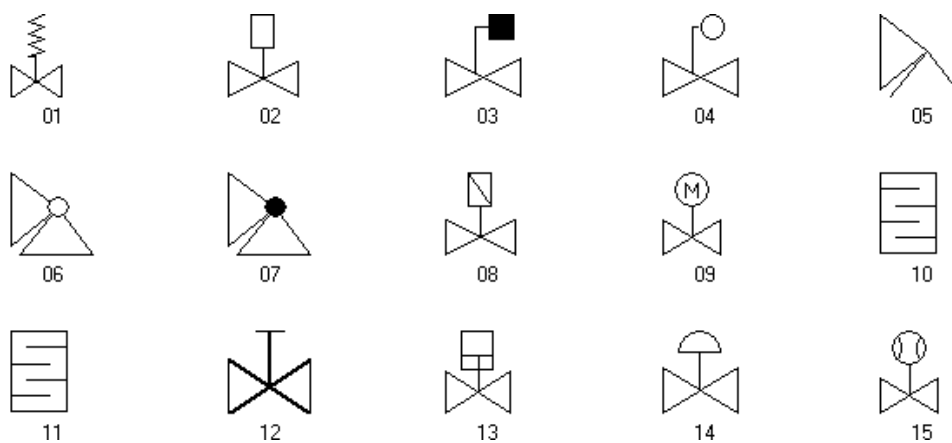


On_Off_8

5.4.4 СИМВОЛЫ



5.4.4.1 Устройства выключения



5.4.4.2 Клапаны выключения



01



02



03



04



05



06



07



08



09



10



11



12



13



14



15



16



17



18



19



20

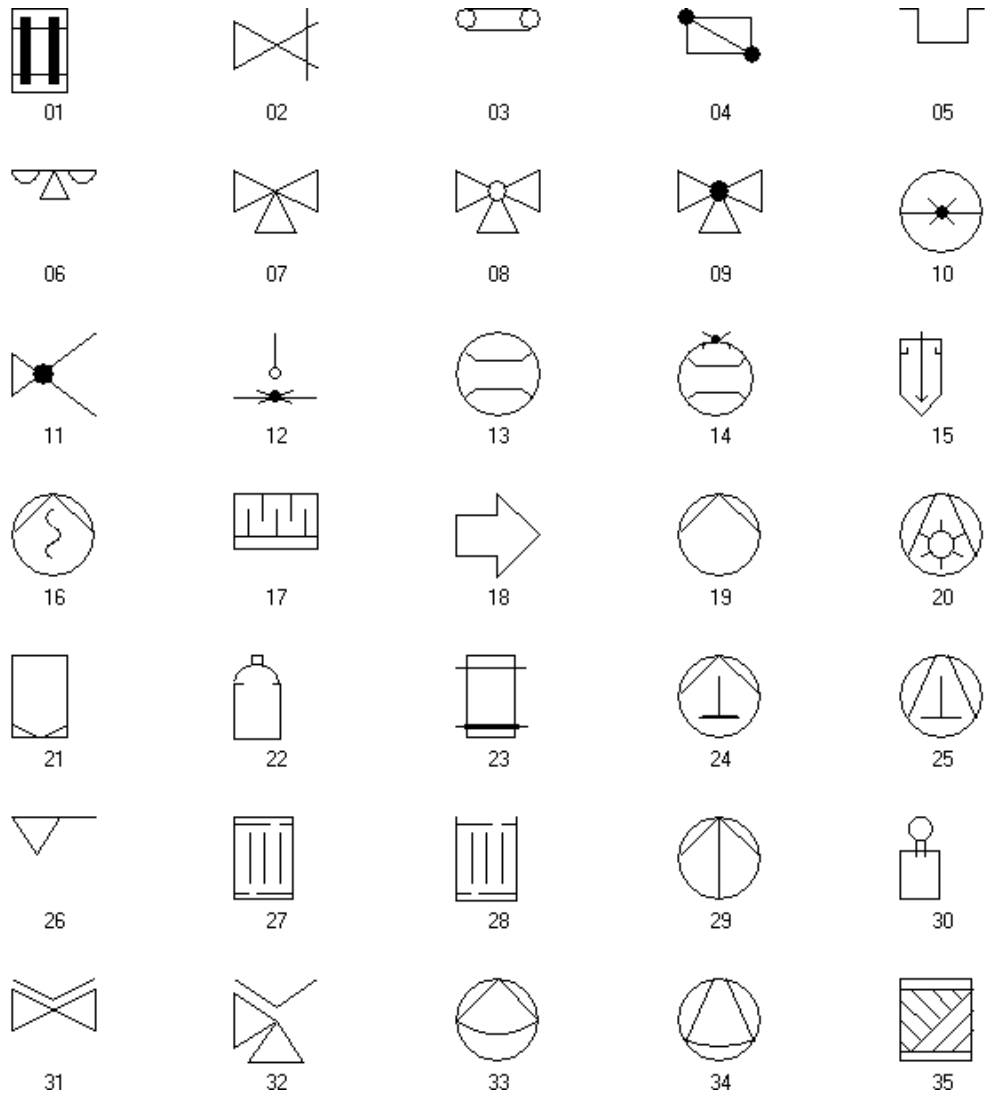


21

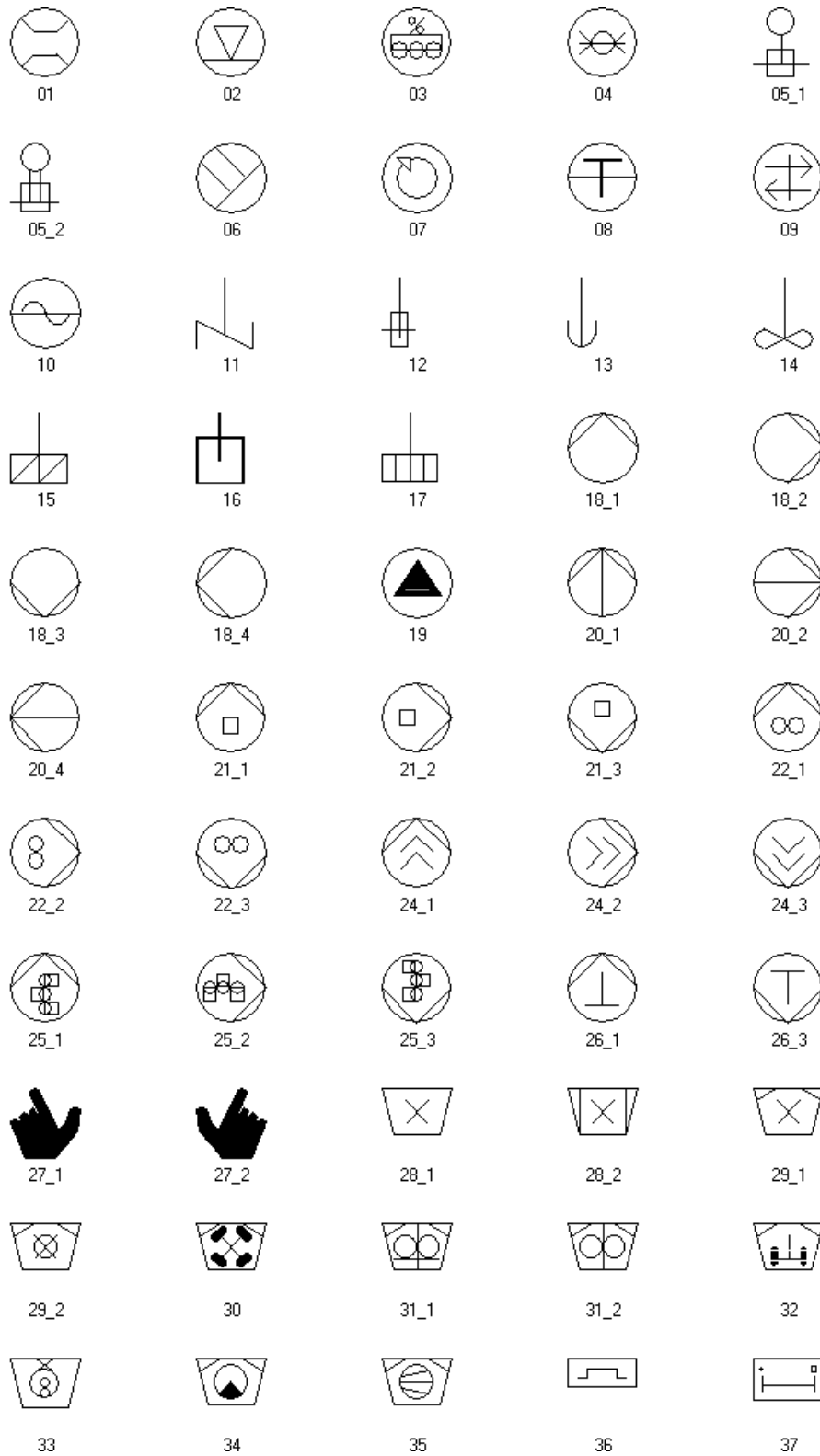


22

5.4.4.3 DIN 30600

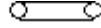


5.4.4.4 Е символы



5.4.4.5 Конвейеры

1



2



3



4



5

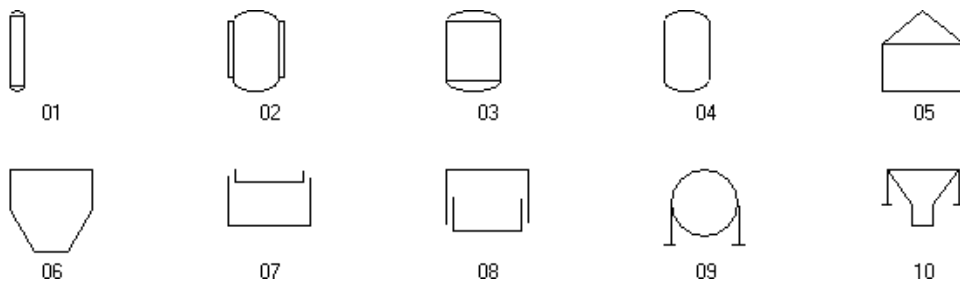


6

5.4.4.6 ISA символы



5.4.4.6.1 isa_s55a



5.4.4.6.2 isa_s55b



1



2



3



4



5



6

5.4.4.6.3 isa_s55c



01



02



03



04



05



06



07



08



09



10

5.4.4.6.4 isa_s55d



01



02



04



04



05



06



07



08



09

5.4.4.6.5 isa_y32a



01



02



03



04



05



06

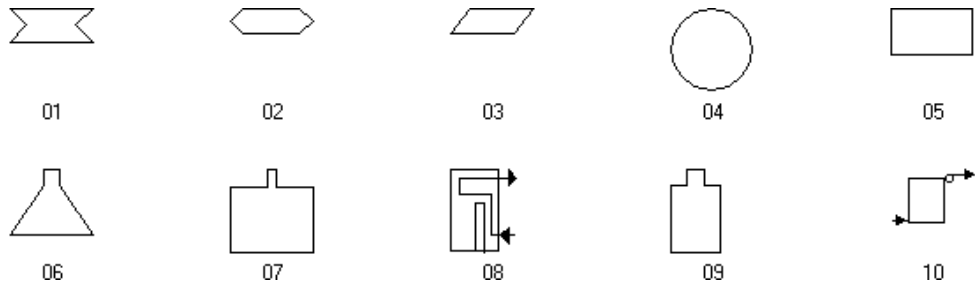


07

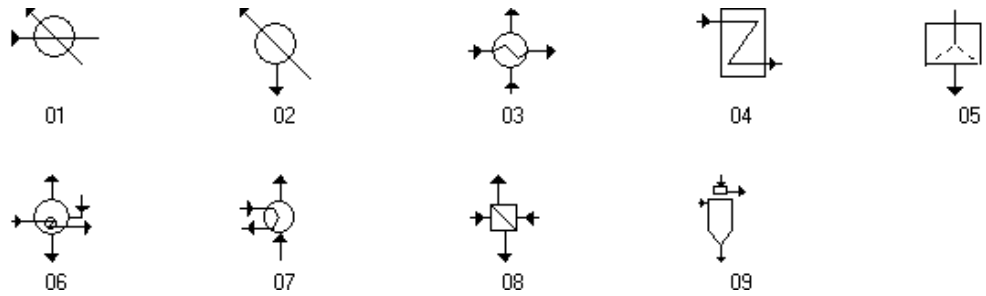


08

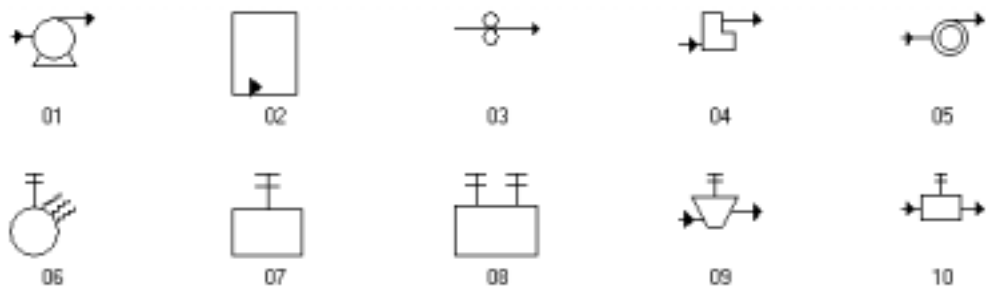
5.4.4.6.6 isa_y32b



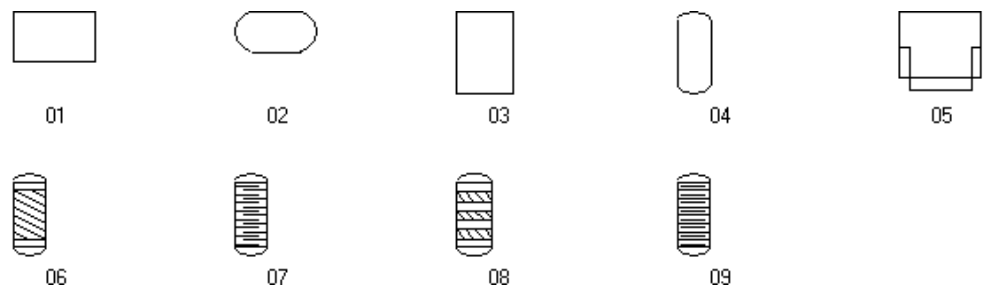
5.4.4.6.7 isa_y32c



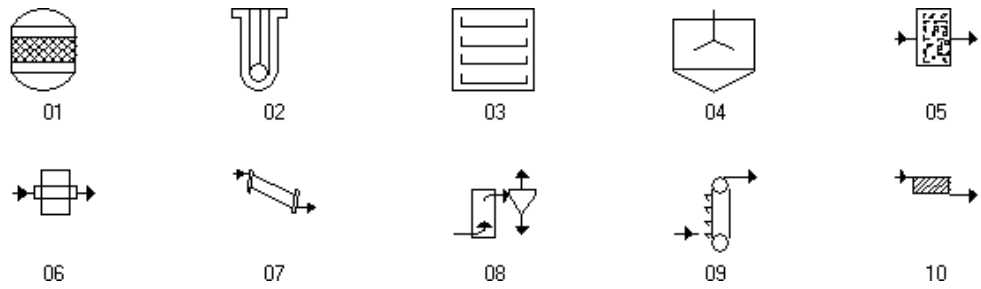
5.4.4.6.8 isa_y32d



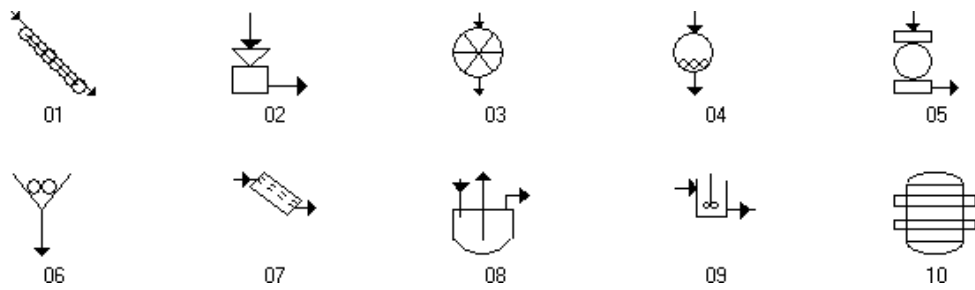
5.4.4.6.9 isa_y32e



5.4.4.6.10 isa_y32f



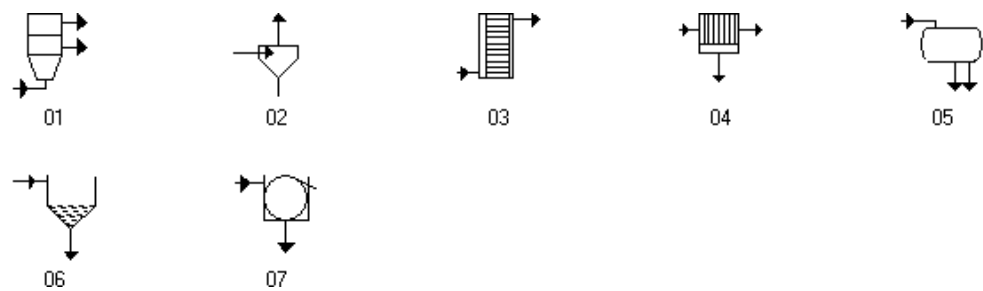
5.4.4.6.11 isa_y32g



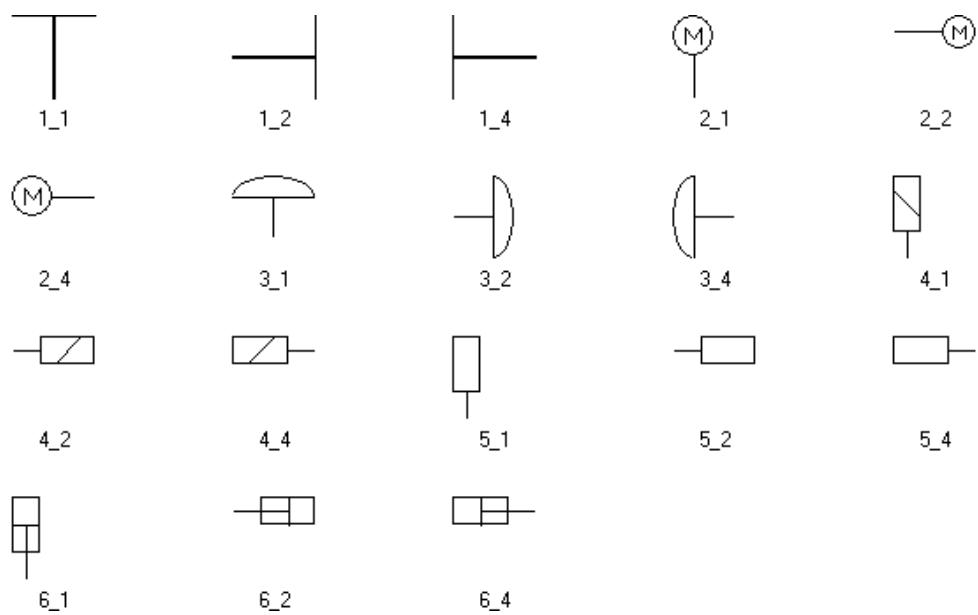
5.4.4.6.12 isa_y32h



5.4.4.6.13 isa_y32i



5.4.4.7 Двигатели



5.4.4.8 Клапаны



01



02



03



04



05



06



07



08



09



10



11



12



13



14



15



16



17



18



19



20



21



22



23



24



25



26



27



28



29



30



31



32



33



34



35



36



37



38



39



40



41



42



43



44



45



46



47



48



49



50



51

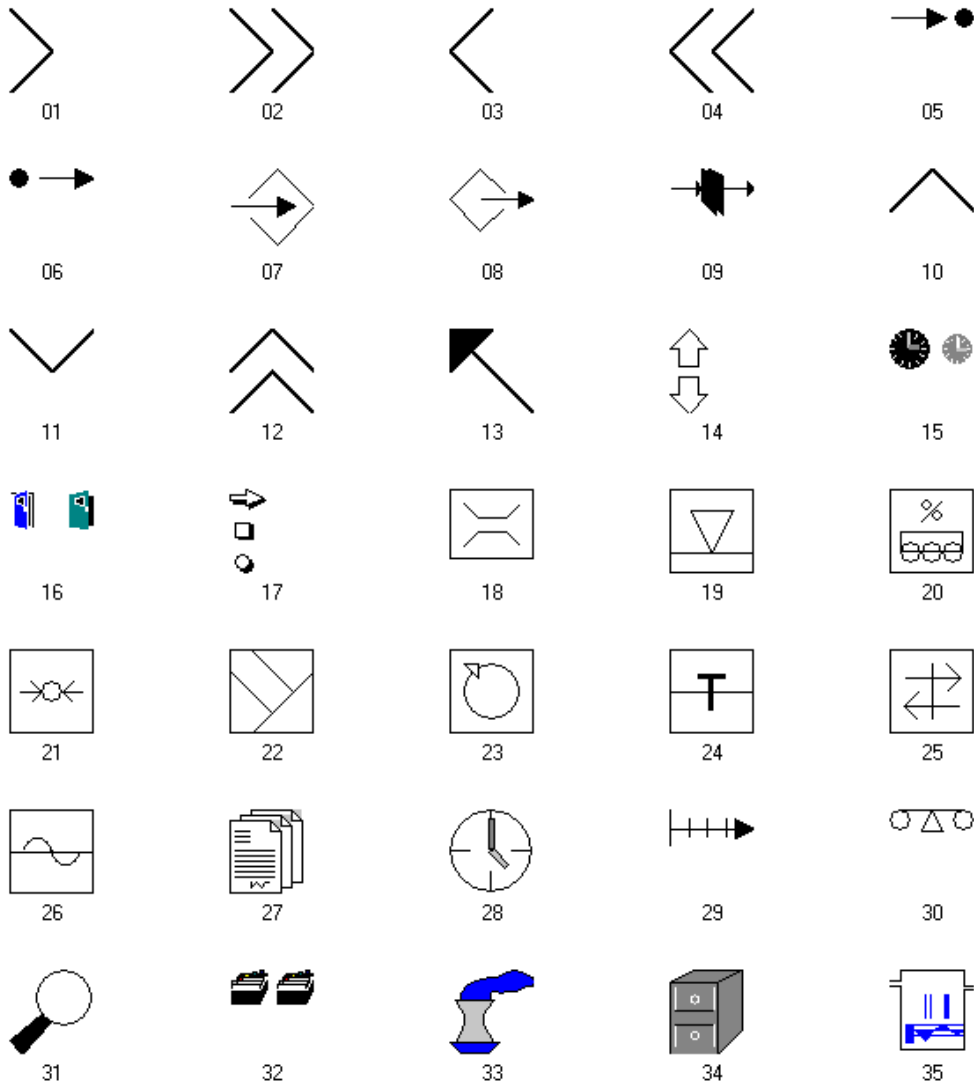


52



53

5.4.4.9 Разное 1



5.4.4.10 Разное 2



01



02



03



04



05



06



07



08



09



10



11



12



13



14



15



16



17



18



19



20



21



22



23



24



25



26



27



28

Index

A

ActiveX, 2-2, 2-4, 3-71, 3-133
ANSI, 4-3
API, 2-5, 3-84, 3-86, 3-135

B

Basic Process Control, 3-17

C

C-API, 2-5
Coros, 3-95

D

DDE, 2-2

E

EN 60073, 3-18
Event
 Function Header at, 4-5

F

Functionality
 Alarm Logging Toolbar, 3-112
Functions
 Header, 4-5

H

HMI, 2-2, 2-6, 3-62
hnStar, 5-72

I

If, 4-54
Informix, 2-6
Ingres, 2-6

M

MS Excel, 3-6, 3-23, 3-88, 3-93, 3-95, 3-98, 5-7
 Чтение данных с помощью MSQuery, 5-7

O

OCX, 2-4, 3-71, 3-108, 3-118, 3-133
 Модуль кадра, 3-118
 Регистрация, 3-56, 3-133
ODBC, 2-2
OLE, 2-2
 Соединения, 3-71
Online, 3-135
 Конфигурация, 3-135
Oracle, 2-6

P

Printf, 4-18

R

Runtime, 3-59
 Данные, 3-59

S

SmartTools, 3-56, 3-71
SQL, 2-5, 2-6, 3-63
 База данных, 3-63
 Инструменты, 3-56
 Программирование, 3-86
Sybase, 2-5, 3-63

U

UNIX, 2-6
UPS, 3-66

V

VDE 0199, 3-18
Visual Basic, 2-2
Visual C++, 2-2

W

While, 4-53
WinCC, 3-83
 API, 3-84
 Version 1.10, 3-98
 Автоматический запуск, 3-62
 Выход, 3-66
 Динамизация, 3-45
 Инструменты, 3-56, 3-71
 Каталог по умолчанию, 3-52
 Концепция управления, 3-16
 Копия проекта, 3-71
 Обработка аварийных сообщений, 3-21
 Передача процедур, 3-83
 Передача тегов, 3-84
 Пользовательский интерфейс, 3-10
 Процедуры, 3-9
 Резервное копирование данных, 3-68
 Среда проекта, 3-58
 Структура, 2-2, 2-4
 Структура папок, 3-58
 Сценарии, 3-9
 Управление тегами, 3-5
 Установки по умолчанию, 3-3
 Файлы протоколов, 3-52
 Цикл обновления, 3-26
Windows, 2-2, 3-19
 Windows NT, 3-19
Wrebuild, 3-56
Wunload, 3-56

A

Аварийное сообщение, 3-21
 В концепции управления, 3-17
 Общая информация относительно спецификации, 3-21
Автозапуск, 3-62
Авторизация, 3-71
Адаптация, 3-25
 Данные для импорта тегов, 3-93
 Обновление кадра, 3-25
 Свойство компьютера, 3-72
Адресация, 3-6
 Косвенная, 3-6, 3-123
Аналоговые значения
 Время, 3-133
Архивация, 3-3, 3-70
 Время архивации, 3-24
 Конфигурация, 3-75

Б

База данных, 2-5
 Восстановление, 3-56
 Редактирование, 3-74
 Резервная копия, 3-68
 Язык запросов, 2-5
Базы данных
 Выборка, 5-17
 Доступ к ISQL, 5-13
 Доступ к MS Access, 5-11
 Доступ к MS Excel, 5-7
Библиотека, 3-70
 По умолчанию, 3-54
 Проект, 3-70, 3-80, 3-82
 Текст, 3-94
Битовая процедура сообщения, 3-21
Битовые операции, 4-39
Буквенный курсор, 3-16, 3-107
Буфер FIFO, 3-16
 Для кадров, 3-16

В

Ввод
 В поля ввода/вывода, 3-106
 Средство, 3-16
 Форматированный, 5-3
Визуализация, 2-2, 3-24
 Станция, 3-24
Внедрение, 3-118
 Из ОСХ, 3-118
Внутренние, 3-31
 Функции, 3-31, 3-53
Возвратная функция, 5-75
 Для обратной связи со статусом задания на запись, 5-75
 Для передачи прочитанной величины процесса, 5-75
Возвращаемое значение, 4-5
Восстановление, 3-56, 3-58
 База данных, 3-56
Временной триггер, 3-41
Временной цикл, 3-31
Временной цикл, 3-25, 3-31
Временные циклы, 3-42
Время исполнения, 3-125, 3-131
 Динамизация, 3-45
 Динамизация для, 3-45
 Курсор, 3-107
 Ограничения при online конфигурации, 3-135
 Спецификации, 3-2
 Управление, 3-107

- Установки, 3-64
- Вход в систему, 3-116
- Выбор, 3-16, 3-35, 3-36, 3-38, 3-79
 - Рисунки для статического отображения, 3-79
 - Сообщений, 3-22
 - Тегов, 3-5
 - Триггер, 3-25
- Выключение, 3-107
 - Курсор времени исполнения, 3-107
- Выполнение
 - Сценариев, 3-41
- Выражения, 4-53
- Выход
 - Из WinCC, 3-66
- Выход из системы, 3-116
- Вычитание, 4-33

Г

- Генерация
 - Нового заголовка, 3-100
 - Новый заголовок, 3-131
- Генерировать, 3-72
 - Новый заголовок, 3-78
 - Создать новый заголовок, 3-72
- Горячая клавиша, 3-103, 3-104, 3-109
- Горячие клавиши, 3-116
- Графика
 - Инструменты, 3-56
- Графический, 3-121
 - Библиотека, 3-121
- Группы, 3-20
 - Группа пользователей, 3-20
 - Группы пользователей, 3-100
 - Группы тегов, 3-5

Д

- Данные
 - База данных, 2-6
 - В базе данных, 3-56
 - Запрос, 3-45
 - Импорт, 3-68
 - Конфигурационные данные, 2-5
 - Обновление, 3-19
 - Передача, 3-74
 - Передача из S5 or S7, 3-84
 - Разделение, 3-58
 - Резервное копирование, 3-70
 - Хранение, 3-22
 - Эксплуатация данных, 2-5
- Данные процесса, 2-5, 3-45

- Действия
 - На открытом кадре, 5-4
- Действующие
 - Окна сообщений, 3-109
 - Окна трендов, 3-113
- Декремент, 4-33
- Деление, 4-33
- Диагностика, 3-88, 4-19, 4-20
 - Область, 5-5
 - Файл для импорта, 3-88
 - Файлы, 3-52
- Диалоговое окно, 3-48
 - Конфигурация, 3-25, 3-48, 3-79
- Диалоговое окно динамики, 3-25, 3-32
- Динамизация, 3-125
 - В WinCC, 3-45
 - Конфигурация, 3-48
 - Модифицированный объект, 3-125
 - Объекты, 3-46
 - Свойства, 3-45
 - События, 3-46
 - Типы, 3-25, 3-32
- Динамика, 3-45
 - Редактирование свойств объекта, 3-45
- Динамический
 - Пример, 3-126
- Динамический мастер, 3-118, 3-124
- Динамическое
 - Соединение с тегами, 3-123
- Документация, 3-2
- Доступ
 - Чтение данных WinCC, 5-11
- Доступ, 3-68, 5-11
 - К базе данных, 3-68, 3-84
 - Права, 3-78

Ж

- Жесткая копия, 3-115

З

- Завершение
 - WinCC, 3-63
- Завершение работы, 3-66
- Задача, 3-63
 - Панель задач, 3-63
 - Смена задачи, 3-31
 - Тестовый вывод, 4-18
 - Форматированная, 5-3
- Задачи, 3-74
 - Команда Проекта, 3-74
- Запрос, 3-31, 3-46

- Данных, 3-45
- Данных из менеджера данных, 3-31
- Код клавиатуры, 3-101
- Событие объекта, 3-46
- Запросы, 3-10
- Запуск, 3-62
 - Автоматический, 3-62
 - Системные сообщения, 3-52
- Защита доступа, 3-16, 3-20, 3-66, 3-100
 - Передача прав, 3-100

И

- Иерархия, 3-16
 - Кадр, 3-105
 - Производственная, 3-103
 - Управление, 3-16
- Измеренное значение, 3-19
 - Обновление, 3-19
 - Передача, 3-100
 - Сбор, 3-76
 - Экспорт, 3-56
- Измеренные значения
 - Архивация, 3-24
- Измеритель, 3-12, 3-125
- Импорт, 3-6, 3-68, 3-84, 3-88
- Инициализация канального устройства, 5-75
 - Параметры для менеджера данных WinCC, 5-75
- Инкремент, 4-33
- Инструментальные средства
 - Импорт/Экспорт тегов, 3-6
- Инструменты, 3-68
 - ОСХ, 3-71
 - SQL, 3-56
 - WinCC, 3-56
 - База данных, 3-68
 - Графика, 3-56
 - Импорт/экспорт Тегов, 3-86
 - Языки, 3-94
- Интеллектуальные инструментальные средства, 3-6
- Интеллектуальные инструменты, 3-86
- Интерфейс, 3-84
 - API, 3-84
- Информация
 - На кадре, 3-10
 - Поиск, 3-4
- Исполнение, 3-88
 - Импорт/экспорт тегов, 3-88

К

- Кадр
 - Выбор, 3-16, 3-109, 3-113, 3-131
 - Иерархия, 3-105
 - Информация на кадре, 3-11
 - Кадр сообщений, 3-110
 - Обновление, 3-24
 - Объект, 3-45
 - Передача, 3-76
 - Размер, 3-14
 - Редактирование цикла, 3-40
 - Структура, 3-118
 - Технология модуля, 3-117
 - Уменьшение размера, 4-12
 - Цикл, 3-29
- Кадры
 - Названия, 3-7
- Канал, 3-71
 - DLL, 3-71, 3-86, 3-93
- Каталог, 3-54
 - Данные в каталоге WinCC, 3-54
 - Структура WinCC, 3-51
- Клавиатура, 3-16
 - Горячие клавиши, 3-115
 - Операции, 3-101
 - Операция, 3-45
 - Управление, 3-16
 - Функциональные клавиши, 3-104
- Клавиши
 - В окне сообщений, 3-110
 - В окне трендов, 3-112
 - Горячая клавиша, 3-104
 - Комбинация клавиш, 3-62
 - Назначение клавиш, 3-108
 - Настройка клавиш, 3-107
 - Область клавиш, 3-13, 3-18
 - Сконфигурированные, 3-94
 - Функциональные клавиши, 3-101
- Клиент, 2-5
- Компьютер, 3-64
 - Горячая клавиша, 3-109
 - Изменение типа, 3-135
 - Папки, 3-61
 - Установки, 3-64
- Конец, 3-108
 - Ввод, 3-108
- Конфигурационные данные, 2-5
- Конфигурация, 3-25, 3-59
 - Данные, 3-59
 - Диалоговое окно, 3-25, 3-35, 3-48, 3-71, 3-79, 3-124
 - Добавление динамики, 3-48
 - Режим, 3-63
 - Спецификации, 3-2

Концепция, 2-6, 3-13
 Резервное копирование данных, 3-66
 Управления, 3-13, 3-16
 Концепция управления, 3-16
 Курсор, 3-16
 Буквенный, 3-16, 3-107
 Таб, 3-16, 3-107
 Курсорные клавиши, 3-108, 3-110

Л

Логика, 4-34
 Сравнение, 4-34

М

Мастер, 3-95
 В модифицированных объектах, 3-124
 Динамики, 3-84
 Обновления кадра, 3-25
 Передача сценариев, 3-135
 Считывание сообщений Sogos, 3-95
 Файлы в каталоге по умолчанию, 3-54
 Чтение тегов S5/S7, 3-84
 Мастер динамики, 3-25, 3-35, 3-47
 Многоклиентская, 3-76
 Многопользовательская, 3-14
 Система, 3-14, 3-66, 3-76
 Модифицированные объекты, 3-130
 В технологии модулей кадров, 3-117
 Передача, 3-80
 Модифицированный объект, 3-80, 3-124, 3-126, 3-131
 Мастер, 3-124
 Модули, 2-3, 3-117
 Заранее сконфигурированные, 3-81
 Интеграция, 2-3
 Технология модулей кадров, 3-117
 Модульность, 2-2
 Мышь, 3-104
 Динамизация событий, 3-46
 Процедура для горячей клавиши, 3-104
 Работа без мыши, 3-101

Н

Направления, 3-93
 Для передачи данных, 3-93

О

Обзор кадров, 3-17
 Область, 5-5
 Временной интервал, 3-112
 Изменения значения, 4-22
 Область значений, 3-47
 Область клавиш, 3-18
 Экрана, 3-13
 Обновление, 3-31
 Типы, 3-31
 Установка опций, 3-24
 Обработка, 3-1
 Данных, 2-2
 Ограничение, 3-76
 Ограничения
 Во время online конфигурации, 3-135
 Во время передачи данных, 3-76
 Название кадра, 3-7
 Название тегов, 3-6
 Однопользовательская, 3-14, 3-74
 Система, 3-14, 3-66, 3-74
 Окна сообщений, 3-76, 3-108, 3-113
 Окна таблиц, 3-76
 Окно, 4-19, 4-20
 Диагностики, 4-19, 4-20
 Смена окна, 3-109
 Цикл, 3-28, 3-31, 3-41
 Окно приложения, 3-76, 3-113
 Операторы, 4-33
 Операции инкремента и декремента, 4-37
 Операционная система, 2-2, 3-19, 3-64, 3-66
 Описание, 4-72
 Определение
 Структуры Си, 4-72
 Определенные пользователем циклы, 3-42
 Определенный пользователем цикл, 3-29
 Опции, 3-17, 3-56, 3-58, 3-105, 3-112
 Основа
 Базовый проект, 3-100
 Для импорта, 3-88
 Основное управление процессом, 3-105, 3-112
 Основной, 3-74
 Основной проект, 3-74
 Основные математические процедуры, 4-35
 Ошибка, 3-58
 Ошибка управления, 3-58
 Поиск, 3-52, 4-12
 Сообщение об ошибке, 3-72

П

Память, 4-45

- Панель управления, 3-108
 - Регистрация аварийных сообщений, 3-108, 3-112
 - Регистрация тегов, 3-112
- Папка
 - Библиотека проекта, 3-82
 - Для автозапуска, 3-62
 - Инструменты для WinCC, 3-86
 - Папка проекта WinCC, 3-58
 - Структура WinCC, 3-22
- Параметры, 3-59
 - Для printf, 4-18
 - Завершение работы, 3-67
 - Название кадра, 3-7
 - Названия проекта, 3-4
 - Названия тега, 3-6
 - по умолчанию, 3-59
 - Разрешения экрана, 3-14
 - Соединение, 3-86
- Пароль, 3-20, 3-66, 4-12
- Платформа, 2-2
- По умолчанию, 3-3
 - Триггер, 3-44
 - Установки, 3-3
 - Язык, 3-54
- Побитовое изображение, 3-79
 - Передача, 3-79
- Подсказки, 3-15
- Подтверждение
 - Сообщения, 3-109
- Пользователь, 2-4, 2-5, 3-10
 - Группы, 3-20, 3-100
 - Передача прав, 3-100
 - Пользовательские временные циклы, 3-28
 - Пользовательские записи, 2-5
- Пользовательская библиотека, 3-105
- Пользовательские архивы, 3-135
- Пользовательский архив, 3-76, 3-100
 - Передача, 3-100
- Пользовательский интерфейс, 2-4, 3-2, 3-10
 - Указание, 3-10
- Понятие
 - Прототип кадра, 3-120
- Последовательный, 3-21
 - Отчет, 3-21
- Права пользователя, 3-20
- Преобразование, 3-76
 - Однопользовательская - многопользовательская, 3-76
- Приложение, 3-45
 - Закрывать, 3-63
 - Интеграция, 2-4
 - Интерфейс (API), 3-84
- Приложение Окно, 2-4
- Приложения Windows, 3-109
- Применение, 2-2
- Пример, 3-126, 3-127
 - Создание, 3-127
- Пример проекта, 3-15, 3-74
 - Разрешение экрана, 3-15
 - Язык, 3-3
- Проверка лицензии, 3-52
- Программа, 3-62
 - Автозапуск, 3-62
 - Для баз данных, 3-56
 - Для передачи данных, 3-86
 - Дополнительные программы, 3-71
 - Собственная, 3-84
- Программный интерфейс, 2-5
- Программы
 - Инструментальные средства, 3-6
- Проект, 3-62
 - Автоматический запуск, 3-62
 - Библиотека, 3-82
 - Глобальные функции, 3-54
 - Копия, 3-71
 - Название, 3-4
 - Низко-эксплуатационный, 3-31
 - Папка, 3-58
 - Передача библиотеки, 3-80
 - Пример проекта, 3-3
 - Резервное копирование, 3-70
 - Создание функций, 4-14, 4-15
 - Трюки реализации, 3-22
 - Функции, 4-12
- Производительность, 3-2, 3-19, 3-47, 3-74
- Протокол, 3-52
 - Файлы, 3-52, 3-61
- Прототип, 3-117, 3-120, 3-124, 3-127
- Процедура
 - Папка, 3-60
 - Передача, 3-83
- Процедуры, 3-24, 4-14
 - В WinCC, 4-4
 - Выбор, 3-38
 - Обновление кадра, 3-24
 - Повторное использование, 4-12
 - Редактирование, 4-12
 - Редактор, 4-12
 - Создание, 4-3, 4-16
 - Указание, 3-9
 - Циклы обновления, 3-31
- Процесс, 3-19
 - Связь, 3-24
 - Соединение, 3-19
 - Теги, 3-123
 - Управление, 3-13
- Прямое соединение, 3-47, 3-105, 3-121, 3-128

Прямое Соединение, 3-25

Р

Радио кнопка, 3-108
 Управление с помощью клавиатуры, 3-108
 Разработка, 2-2
 Разрешение, 3-14
 Экран, 3-14
 Разрешение экрана, 3-15
 Расширение, 3-83
 Для процедур, 3-83
 Файлы WinCC, 3-59
 Регистратор, 3-56
 ОСХ, 3-56
 Регистрация
 ОСХ, 3-133
 OLE, ОСХ, 3-71
 Редактирование
 С триггером тега, 3-32
 Редактировать, 3-29
 Определенный пользователем цикл, 3-29
 Свойства проекта, 3-72
 Состояний формы, 3-45
 Триггер, 3-38
 Цикл кадра, 3-40
 Цикл окна, 3-41
 Редактор, 4-12
 Резервное копирование, 3-66
 Данных WinCC, 3-68
 Концепция, 3-66
 Результат, 4-18
 Вывод, 4-18
 Динамизация, 3-50
 Рекомендация, 3-32
 Для циклов обновления, 3-32
 Решение, 3-62
 Автоматический запуск проекта, 3-62
 Подход, 2-2
 Рисунок
 Изменение, 3-103

С

Свойства канального устройства, 5-78, 5-87
 Online регистрация логических соединений, 5-78, 5-87
 Online регистрация тегов WinCC, 5-78, 5-87
 Без регистрации тегов WinCC, 5-78, 5-87
 Ведомый во времени, 5-78, 5-87
 Ведущий во времени, 5-78, 5-87

Диагностические опции, 5-87
 Диагностические опции, 5-78
 Доступ к удаленным тегам, 5-78, 5-87
 Запись по байтовому адресу, 5-78, 5-87
 Запись по битовому адресу, 5-78, 5-87
 Клиентская функциональность, 5-78, 5-87
 Локальное окно перезапуска, 5-78, 5-87
 Локальный мониторинг активности, 5-78, 5-87
 Обработка значений в байтовом порядке INTEL, 5-78, 5-87
 Редактирование опций каналов, 5-78, 5-87
 Реентерабельность, 5-78, 5-87
 Управление локальными циклами, 5-78, 5-87
 Свойство, 3-31
 Заголовок функции, 4-6
 Объект, 3-26
 Связь, 3-19, 3-31, 3-78
 Влияние на обновление, 3-31
 Интерфейс, 3-71
 Между отдельными задачами, 3-32
 Процесс, 3-24
 Сеть, 3-68
 Символ, 3-54, 4-45
 Для типа динамики, 3-45
 Передача, 3-79
 Правильный в названии проекта, 3-4
 Правильный для названия кадра, 3-7
 Правильный для названия тега, 3-6
 Строка, 4-45
 Хранение, 3-54
 Символ
 Экспорт, 3-82
 Синтаксис, 4-3
 Система, 3-62
 Автоматический запуск, 3-62
 Загрузка, 3-32, 3-44, 3-45
 Многопользовательская, 3-74
 Модули, 2-3
 Однопользовательская, 3-14, 3-74
 Операционная система, 3-19
 Платформа, 2-2
 Программное обеспечение, 3-71
 Размещение, 3-70
 Сообщения, 3-52
 Теги, 3-6, 3-78
 Система шин, 3-19
 Сложение, 4-33
 Событие, 3-46
 Добавление динамики, 3-46
 Контролируемое, 3-19
 Триггер, 3-25, 3-41, 3-44
 Содержимое, 3-59

- Отображение папок проекта, 3-59
- Соединение, 3-19
 - UPS, 3-66
 - Косвенное, 3-118
 - Логическое, 3-85
 - Новый, 3-117
 - Последовательное с CP525, 5-18
 - С процессом, 3-19
 - С тегами, 3-123
 - С тегами процесса, 3-130
 - Список соединений, 3-91
- Создание, 4-128
 - Структуры сообщений, 3-98
 - Файла в сценарии, 4-128
- Сообщение, 3-18
 - Класс сообщения, 3-18
 - Система сообщений, 3-18
 - Список сообщений, 3-109
- Сообщения, 2-5
 - Кадр сообщений, 3-110
 - Классы сообщений, 3-95
 - Передача, 3-94, 3-95
 - Последовательный отчет, 3-21
 - Процедура сообщения, 3-21
 - Резервное копирование, 3-70
 - Системные сообщения, 3-52
 - Считывание, 3-95
 - Указание цветов, 3-18
 - Файл сообщений, 3-96
- Состояние, 3-21
 - Изменение, 3-21
 - Отображение состояний, 3-12
- Специальные символы, 3-7
- Списки назначения, 3-84
- Среда разработки, 3-60, 4-3
- Ссылка, 3-121
 - В кадрах, 3-78
 - В модулях кадра, 3-121
 - В модулях кадров, 3-131
 - Текстовая ссылка, 3-94
- Стандартные
 - Клавиши, 3-105
 - Содержание каталога WinCC по умолчанию, 3-52
 - Установки, 3-25, 3-28
 - Функции, 3-70, 4-12
 - Циклы, 3-32
- Стандартный, 4-3
 - Си, 4-3
- Старт, 3-62
 - Автоматически WinCC, 3-62
- Статус
 - Статус отображения, 3-79
- Строки, 4-45

- Структура, 3-7, 3-17
 - WinCC, 2-2
 - База данных, 3-74
 - Каталоги WinCC, 3-51
 - Концепция управления, 3-17
 - Корпоративные стандарты, 3-2
 - Название кадра, 3-7
 - Папки проекта WinCC, 3-58
 - Связи, 3-130
 - Создание для сообщений, 3-98
 - Структура кадра, 3-118
 - Структура папок, 3-22
 - Тегов, 3-5
 - Текстовые списки, 3-89
 - Управление тегами, 3-5
 - Функции, 4-5
 - Хранения данных, 3-22
- Структуры, 4-72
- Сценарии
 - Редакторы, 4-12
- Сценарии, 3-9, 3-41
 - Выполнение, 3-41
 - Для мастера, 3-54
 - Синтаксис, 4-3
 - Среда разработки, 4-3
 - Язык, 4-1
- Считывание, 3-95
 - Сообщения, 3-95

Т

- Таб курсор, 3-16, 3-107
- Таблица цветов, 5-19
- Табличные окна, 3-112
- Теги, 3-48
 - Архивные теги, 3-114
 - Имена, 4-22
 - Импорт/Экспорт, 3-68
 - Корректные названия, 3-6
 - Передача, 3-84
 - Передача из S5/S7, 3-84
 - Переменные Си, 4-21
 - Симуляция, 3-61
 - Соединение, 3-48
 - Сообщения касающиеся отсутствия, 3-52
 - Структура экспортируемых списков, 3-89
 - Типы, 4-22
 - Триггер, 3-25, 3-29, 3-32, 3-44
 - Указание, 3-5
 - Экспорт, 3-86
- Теги в сценариях, 3-6
- Текст, 3-12
 - Идентификация на экране, 3-12
 - Многоязыковой, 3-94

Отображение, 3-12
 Списки, 3-89
 Текстовые списки, 3-84, 3-108
 Термины, 2-6
 Технологический кадр, 3-46
 Технологический кадр
 Добавление динамики, 3-46
 Работа без мыши, 3-101
 Транзакционно-защищенный, 2-5
 Тренды
 Окна управления, 3-112
 Применить границу, 3-76
 Управляющие окна, 3-109
 Триггер, 3-25
 В диалоговом окне динамики, 3-37
 В процедурах Си, 3-38
 Обновление кадра, 3-25
 Теги, 3-25
 Триггер тега, 3-32
 Управляемый временем, 3-42

У

Указатели, 4-44
 В Си, 4-44
 Умножение, 4-33
 Управление, 3-13, 3-16, 3-19
 Без мыши, 3-101
 Окна трендов, 3-112
 Ошибка управления, 3-11
 С помощью клавиатуры, 3-45, 3-105
 С помощью функциональных клавиш, 3-103
 Событийное, 3-19
 Управление процессом
 Иерархия, 3-16
 Управляющие объекты, 3-107
 Условные, 4-53
 Выражения, 4-53
 Установка, 3-56
 WinCC, 3-71
 Инструменты, 3-56
 Устройство, 3-68
 Для резервного копирования данных, 3-68

Ф

Файлы, 3-58
 В папке по умолчанию, 3-52
 В папке проекта, 3-58
 Функции, 3-84
 API, 3-84

В WinCC, 4-4
 Внутренние, 3-31, 4-14
 Защита доступа, 3-66
 Передача функций проекта, 3-100
 Проекта, 3-59, 4-12
 Редактор, 4-12
 Создание, 4-14, 4-15
 Стандартные, 3-53, 3-100, 4-12
 Структура, 4-5
 Управления, 3-20
 Функциональная клавиша, 3-101
 Функциональность, 3-127
 Модули кадров, 3-127
 Функциональные возможности
 Панель управления системы регистрации аварийных сообщений, 3-114

Ц

Цвет
 Идентификация, 3-12
 Определение, 3-18
 Цвета в проекте, 3-12
 Целевая аудитория, 4-1
 Циклы, 4-53
 Do while, 4-53
 For, 4-54
 While, 4-53
 Циклы обновления, 3-19

Ч

Числа с плавающей точкой, 4-22
 Число, 4-22
 С плавающей точкой, 4-22
 Целые, 4-22
 ЧМИ, 3-19

Ш

Шаблон, 3-3
 Для Ваших проектов, 3-3
 Окна сообщений, 3-109
 Шрифты
 Тип, 3-2
 Шрифт, 3-2
 Размер, 3-2, 3-15
 Цвет, 3-18

Э

Экспорт, 3-6, 3-56, 3-68, 3-81, 3-88
 Сообщения, 3-56
 Теги, 3-86
Экспорт данных, 5-15
 С помощью процедуры Си, 5-15

Я

Язык, 3-94
 Передача, 3-94