

## Глава 1 Введение



В этой главе происходит общее знакомство с Clarion for Windows и сопроводительной документацией.

**“Приглашение” в Clarion for Windows.** Вы только что приобрели продукт, который по мнению TopSpeed Corporation является самым мощным в мире инструментом для разработки Windows приложений.

*Руководстве для пользователя* рассказывает, как создаются приложения с использованием среды Clarion for Windows. Основное внимание уделяется вопросам интерфейса со средой, имеющимся инструментам и шаблонам.

Где искать дополнительную информацию. По Clarion for Windows написано пять книг и обширная контекстная справочная система.

Неограниченная свободная техническая поддержка Clarion for Windows осуществляется по CompuServ Information Service. TopSpeed Corporation предоставляет также платную техническую помощь по телефону.

Добро пожаловать на обзор по разработке Windows приложений.

Что вы найдете в этой книге

Где получить дополнительную информацию

Условные обозначения в документации

Шрифтовые условные обозначения :

Условные обозначения для клавиатуры:

Регистрация продукта

Техническое обслуживание

Факс-система TopSpeed

## **Добро пожаловать на обзор по разработке Windows приложений.**

Добро пожаловать в Clarion for Windows. Вы только что приобрели *наиболее мощный, по мнению TopSpeed Corporation, из всех имеющихся на рынке инструментов разработки приложений Windows!* Теперь вы можете построить изощренное приложение Windows быстрее, чем вы думали это возможно. Эта революционная среда разработки впечатляющим образом увеличит производительность вашей работы. Исполняемые программы, которые вы будете создавать с ее помощью, будут работать также быстро, как программы, изготовленные в результате скучной работы с использованием таких языков, как язык C; кроме того, вы можете без труда связываться практически с любой из существующих баз данных.

Теперь у вас имеются и легко настраиваемая платформа для быстрого создания приложений (RAD), и вся мощь языка Clarion для создания сложных прикладных Windows-программ. Технология *Point and Click (Укажи и нажми)* избавляет вас от сложности и утомительности типичных систем программирования под Windows.

Генератор приложений (Application Generator) строит полностью ориентированные на пользователя Windows-программы за время, существенно меньшее того, которое требуется другим программирующим средам, делая в то же время *ручное кодирование необязательным*. Управляемая шаблонами среда обеспечивает повсеместное многократное использование программного текста без курса интенсивного обучения объектно-ориентированному программированию.

Язык программирования Clarion является мощным и в то же время легким для понимания, ориентированным на бизнес языком программирования четвертого поколения. В сочетании с нашими высокоэффективными драйверами базы данных Clarion for Windows обеспечивает кратчайший цикл разработки и наиболее быстро работающие программы для *вашего* проекта.

- Управляемая шаблонами (template) быстрая разработка приложений  
Заранее написанные стандартные процедуры-шаблоны (template) обеспечивают настраиваемые в соответствии с требованиями пользователя многократно используемые средства для широкого круга функций, таких, как просмотр файлов данных, формы и отчеты. Вам нужно лишь выбрать шаблон из списка и заполнить его. Шаблоны могут быть полностью подогнаны к виду, наиболее отвечающему нуждам вашего приложения. Вы легко можете добавить к своему набору свои собственные или чужие шаблоны.

- Средства форматирования WYSIWYG (Что видите, то и получаете)  
К вашим услугам экранные средства форматирования для редактирования или добавления

оконных и документных элементов к стандартным шаблонам. Легко настраиваемые поля списков, автоматическая связь полей базы данных с вводными полями, выбор действий для стандартных позиций меню - все это вы можете использовать в своем приложении.

- Немедленные результаты - Постройте *свое* приложение за 1 час.

Мастера (Wizard) Clarion for Windows “Мастер Приложений”, “Quick Load” и все прочие процедурные мастера построят словарь, выберут шаблоны, затем создадут приложение для обновления и ведения данных и построения по ним отчетов не позднее чем через час, после того, как вы разорвете упаковку. В третьей главе вы создадите приложение “*Quick Start Tutorial*” (“*Руководство по Быстрому Освоению*”) не написав ни строчки программного текста.

Такие важные процедуры, как проверка данных и поддержка ссылочной целостности базы данных, — задачи, на решение которых тратится много времени на других платформах, в Clarion for Windows программируются *автоматически*. Windows-проекты, обычно требующие на разработку месяцы при использовании других средств, с Clarion for Windows выполняются за существенно меньшее время.

- Разработка на языке четвертого поколения - Си - производительность

В отличие от других RAD платформ, исполняемые модули, построенные с помощью Clarion for Windows, быстры. Технология компиляции TopSpeed создает Windows-программы, которые буквально *летают*.

Язык Clarion обладает мощностью и гибкостью, необходимыми для разработки любого приложения. Это структурированный, компактный, расширяемый язык, который поддерживает все наилучшее в компонентно-ориентированных разработках. Используйте в своем приложении диалоговые элементы Visual Basic (.VBX). Создавайте динамически подключаемые библиотеки, к которым могут обращаться приложения, написанные на других языках.

Для “ручных кодировщиков” - тех, кто пишет приложения с помощью языка Clarion, вместо использования шаблонов - среда разработки обеспечивает богатый инструментарий. Window и Report-форматеры позволят вам переключаться между графическим редактированием окон и отчетов, и редактированием их как структур данных в текстовом режиме. Редактор текста Text Editor обеспечивает цветовое выделение синтаксических единиц. Редактор проекта Project Editor компилирует и связывает модули с использованием *TopSpeed* технологии, а Clarion отладчик программ мирового класса поможет добиться совершенства вашего проекта.

- Независимость от баз данных.

Новые высокоэффективные драйверы базы данных, переписанные как динамически подключаемые библиотеки, поддерживают широко распространенные базы данных и бухгалтерские пакеты, включая форматы Xbase, Paradox, Btrieve и другие. На локальных дисках они показывают гораздо более высокую производительность, чем ODBC драйвер, который используют многие приложения Windows (и который Clarion for Windows также поддерживает).

Храните ваши данные (или преобразовывайте существующие данные) в нашем новом формате файла - TopSpeed для получения невероятно быстрого выполнения. Это, к тому же, эффективно — вы можете запомнить множество файлов (таблиц) данных в одном физическом файле DOS, экономя для конечного пользователя использование механизмов управления файлами.

## **Что вы найдете в этой книге**

*Руководство пользователя (User's Guide)* покажет вам, как создать приложения, используя интегрированную среду разработки Clarion for Windows, с точки зрения разработчика, использующего Генератор приложений. Ниже перечислены разделы данной книги и дается краткое изложение ее содержания:

**Введение** Это глава, которую вы сейчас читаете.

**Установка** Перечисляются требования системы к оборудованию и приводятся инструкции по инсталляции.

**Последовательность разработки** Введение в Clarion. Описываются функциональные части интегрированной среды разработки, а также последовательность шагов, которые необходимо предпринять для создания приложений.

**Редактор словаря данных** Представляет концепцию словаря данных, который содержит определения файлов данных приложения, их поля, ключи и многое другое. При помощи словаря данных Clarion автоматически обеспечивает нужные режимы ввода и контроль достоверности данных, что делает ненужным многие строчки программного текста.

**Генератор приложений** Описывается Генератор приложений, с помощью которого вы

можете автоматически сконструировать процедуры и функции своего приложения. При использовании Генератора приложений выбирайте такие типы процедур, которые наиболее близко соответствуют вашим требованиям, затем доработайте их, создавая выпадающие меню, диалоги или отчеты, определяя данные или создавая формулы и вставки исходного текста.

**Мастера и процедурные шаблоны** Описываются процедурные шаблоны; описываются как их применение, так и способ включения их в ваши приложения.

**Диалоговые, текстовые и распределенные шаблоны** Описываются диалоговые, текстовые и распределенные шаблоны; описываются как их применение, так и способ включения их в ваши приложения.

**Форматер окон (Window Formatter)** Описывается, как использовать Форматер для визуальной разработки окон ваших приложений, включая размер, внешний вид, меню и элементы управления. Форматер окон также позволит вам добавить элементы управления из внешних .VBX библиотек.

**Редактор меню (Menu Editor)** Вы обращаетесь к Редактору меню через Форматер окон. Это позволяет вам создать меню для своего приложения. В этой главе также объясняется, как создавать панели инструментов для ваших приложений.

**Элементы управления (Controls)** Элементы управления - это интерфейсные элементы пользователя, которые вы помещаете в приложение, используя Форматер окон. Они включают поля ввода, кнопки на экране и графические элементы, такие, как массивы битов, например. Эта глава покажет вам, как задать необходимые свойства элемента управления любого из видов.

**Форматер списка** Описывает, как использовать форматер для задания необходимых свойств поля списка. Приводятся инструкции по приданию полю списка таких свойств, как операция drag and drop (перетаски и брось).

**Форматер отчета** Форматер отчета - это основное средство создания отчетов в интегрированной среде разработки. Вы можете поместить элементы управления в отчет, точно так же, как вы помещаете

их в окна. В главе описывается, как процессор печати раздел за разделом генерирует ваш отчет.

<b>Редактор текста</b>	Описывается, как непосредственно редактировать файлы исходных текстов. Текстовый редактор представляет возможность цветового выделения синтаксических конструкции, поиск и замену, а также другие полезные инструменты редактирования программного текста.
<b>Редактор формул</b>	Описывается, как использовать Редактор формул для быстрого генерирования операторов присваивания значений выражений.
<b>Система проекта</b>	Проводит вас через подготовку нового проекта и объясняет компоненты окна дерева проекта. Этот раздел также дает информацию о файлах, которыми необходимо будет укомплектовывать ваши готовые приложения.
<b>Отладчики</b>	Описывается, как использовать Отладчики Clarion for Windows с целью устранения ошибок в вашей программе.
<b>Администратор базы данных</b>	Описывается, как использовать эту утилиту для прямого доступа к файлам данных без необходимости создания специального приложения.
<b>Приложения</b>	Представляют дополнительную информацию по вопросам, имеющим отношение к среде разработки Clarion for Windows но непосредственно в нее не входящим. К ним относятся обсуждение вопросов конструирования окон, спецификации файловых драйверов, коллективной разработки, интерфейса ODBC, и глоссарий.

## Где получить дополнительную информацию

---

Имеются пять книг и обширный контекстный справочник для Clarion for Windows.

✓ Руководство Getting Started содержит в себе два подробных учебника. В первом описывается процесс быстрого создания приложения с использованием утилиты “Quick Start” (быстрый запуск) Мастер (Wizard) Clarion for Windows. Второй дает более подробное знакомство со средой разработки на примере создания приложения “Система Ввода Заказов”.

✓ Руководство пользователя *Users' Guide* - это книга, которую вы в данный момент читаете. Она дает задачно-ориентированное описание среды разработки и ее главных компонент. Она описывает шаблоны, которые поставляются вместе с этим продуктом, и

содержит дополнительные приложения с информацией по таким вопросам, как файловые драйверы Clarion for Windows.

√ Справочник по языку является полным руководством по языку Clarion. Приводится широко иллюстрированное примерами описание всех операторов и функций языка. Справочник организован по функциональным темам.

√ В *Руководстве программиста* описан язык шаблонов Clarion for Windows, проектная система TopSpeed, многоязычное программирование, обращение к функциям API, использование Clarion for Windows в качестве DDE сервера и другие глубокие вопросы.

√ В руководстве пользователя по *Report Writer* описан Clarion Report Writer for Windows — средство разработки документов для конечного пользователя.

√ Гипертекстовый контекстно-зависимый справочник появляется при нажатии клавиши F1, кнопки Help или при выборе одного из пунктов в меню Help. Справка в интерактивном режиме осуществляется через диалоговое окно, предоставляющее вам именно тот раздел, в котором вы нуждаетесь.

**Внимание:** если какая-либо часть интерактивного справочника противоречит напечатанному документу, преимущество отдается интерактивному справочнику.

Корпорация TopSpeed делает все возможное для обновления печатной документации. Однако время, которое нужно издателям, может создать отставание в документации; если мы можем обновить файлы помощи, которые поставляются одновременно с продуктом, печатные материалы вынуждены “подхватывать” это позже.

## ***Условные обозначения в документации***

В документации используются приведенные ниже соглашения о шрифтах и клавиатурных комбинациях .

### **Шрифтовые условные обозначения :**

---

<i>Italics</i>	Показывает, что набирается на клавиатуре, например Type this (Напечатай это).
SMALL CAPS	Указывает нажимаемые клавиши для ввода на клавиатуре, такие как ENTER или ESCAPE.
ALL CAPS	Выделяет ключевые слова языка Clarion. См. также <i>Справочник по языку</i> .



**Boldface** Указывает команды или параметры в выпадающем меню или текст в диалоговом окне. Внимание: этот стиль использует также другой тип шрифта для соответствия жирным символам helvetica, которые Windows использует в качестве системного шрифта.

**LETTER GOTHIC** Используется для диаграмм, списков исходных кодов и для примеров использования операторов исходного текста.

**Совет:** Специальные советы, замечания и предупреждения — сведения, которые не являются самоочевидными из разъяснений к текущему пункту.



Отмечает важные сведения. Если Вы ничего не читали в данной главе, пожалуйста прочтите этот текст.

## **Условные обозначения для клавиатуры:**

---

**F1** Указывает нажатие клавиши. Нажмите и отпустите клавишу F1.

**ALT+X** Показывает комбинацию нажатий клавиш. Удерживая нажатой клавишу ALT, нажмите клавишу X. Затем отпустите обе клавиши.

## **Регистрация продукта**

Прежде чем начать пользоваться Clarion for Windows, заполните и отправьте по почте регистрационную карту, которая пришла с упаковкой. Заполнение этой Business Reply Card дает вам право на ряд важных преимуществ. Будучи зарегистрированными, вы можете пользоваться услугами технического обслуживания TopSpeed, а также будете автоматически получать сообщения о новом продукте и предупреждения об обновлении.

## **Техническое обслуживание**

---

Вы можете получить неограниченное бесплатное техническое обслуживание Clarion for Windows через CompuServe Information Service. Соединившись с CompuServe, наберите GO TOPSPEED. Работники TopSpeed, а также зарегистрированные партнеры TopSpeed по техническому обслуживанию (называемые Командой TopSpeed) своевременно ответят на ваши вопросы. Кроме того, вы получите совет и ответ на вопросы от других пользователей

Clarion for Windows. Мы очень рекомендуем нашим клиентам воспользоваться услугами этой службы.

Фирма TopSpeed предоставляет также платное техническое обслуживание по телефону. Вы можете получить техническую помощь по платному вызову позвонив по телефону (900)884-0444. Имеются различные платные программы технического обслуживания. Чтобы получить более полную информацию, свяжитесь с отделом обслуживания потребителей фирмы TopSpeed по телефону (800) 354-5444 или (305) 785-4555.

### **Факс-система TopSpeed**

---

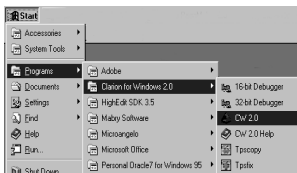
Корпорация TopSpeed также предлагает клиентам доступ через телефон или факс к наиболее популярным техническим и торговым документам. Онлайн документы включают брошюры о продукте, техническую документацию, копии статей, прейскуранты и даже “Что свежестького” о продуктах TopSpeed.

Чтобы заказать нужные документы, наберите (305) 785-4555, нажмите 53 и слушайте инструкции факс-системы. Можете также набрать (305) 785-4556 (Линия стандартного обслуживания TopSpeed) и нажать 1 для связи с системой. Меню интерактивно и удобно для пользователя. На первый случай звонящие могут попросить список имеющихся документов, чтобы просмотреть их перед тем, как сделать свой выбор. Затем вы можете ввести кодовый номер документа и материал будет немедленно доставлен к вам. Вы можете получить доступ к системе непосредственно со своего факс-аппарата или с любого телефонного аппарата с тональным набором.

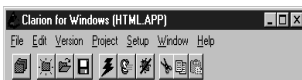
## Глава 2 Установка



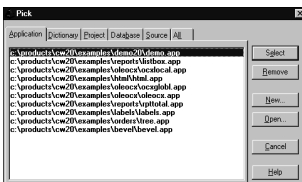
Здесь описан процесс установки и запуска Clarion for Windows и даны первоначальные сведения о перемещении по разделам среды Clarion for Windows.



Первоначальный запуск Clarion for Windows.



Линейка инструментов позволяет быстро перемещаться в среде Clarion for Windows.



Диалог выбора Pick содержит списки последних использованных файлов, которые систематизированы по категориям - приложения, словари, проекты, базы данных, исходные тексты и все вместе.

ТРЕБОВАНИЯ К СИСТЕМЕ  
ПРОГРАММА УСТАНОВКИ  
НАЧАЛО УСТАНОВКИ  
ВАРИАНТЫ УСТАНОВКИ  
ЗАПУСК CLARION FOR WINDOWS  
БЫСТРЫЙ ДОСТУП К ФАЙЛАМ И ФУНКЦИЯМ  
ОСНОВНАЯ ПАНЕЛЬ ИНСТРУМЕНТОВ  
СПИСОК ВЫБОРА

## **Требования к системе**

### **Аппаратные требования**

Среду разработки приложений Clarion вы можете использовать на любой системе, которая отвечает минимальным системным требованиям Microsoft Windows 3.x, Windows 95TM или Windows NT 3.51.

- Для Windows 3.x, рекомендуется 8 мегабайт оперативной памяти.
- Для Windows 95, рекомендуется 12 мегабайт оперативной памяти.
- Для Windows NT 3.51, рекомендуется 16 мегабайт оперативной памяти.
- Минимум от 10 до 25 мегабайт свободного пространства на жестком диске, в зависимости от выбранных опций установки.

Приложения, разрабатываемые вами с помощью Clarion for Windows, будут хорошо работать на компьютерах, которые отвечают только минимальным требованиям для этих операционных систем.

### **Программное обеспечение**

Clarion for Windows работает под Windows 3.x, Windows 95 или Windows NT.

## **Программа установки**

Программа установки, находящаяся на инсталляционной дискете номер один, разворачивает и копирует файлы Clarion for Windows на ваш жесткий диск.

- Для всех целевых операционных систем программа установки предоставляет выбор для установки различных компонентов, например таких, как файлы с демонстрационными программами.
- В Windows 3.x программа установки запрашивает подтверждение перед обновлением оператора PATH в файле AUTOEXEC.BAT для включения каталога Clarion for Windows.
- В Windows 3.x программа установки добавляет в окно Program Manager пиктограммы для среды Clarion IDE, отладчика, справочных файлов и файлов ReadMe.
- В Windows 95 программа установки добавляет пиктограммы среды Clarion,

справочных файлов и файла ReadMe в меню Start > Programs.

## Начало установки

---

□ Чтобы запустить программу установки Clarion for Windows в Windows 3.x

1. Вставьте первую дискету из пакета инсталляционных дискет в дисковод.
2. Из Диспетчера программ, Диспетчера файлов или другой командной программы, способной запускать программы, выберите File > Run.
3. Наберите A:\SETUP (или B:\SETUP) в диалоге Run, затем нажмите кнопку ОК.

Программа установки обеспечивает экран заставки и некоторую текстовую информацию. Она запросит желаемые варианты установки.

□ Чтобы запустить программу установки Clarion for Windows в Windows 95:

1. Вставьте первую дискету из пакета инсталляционных дискет в дисковод.
2. Из меню Start выберите Settings > Control Panel.
3. Выберите Add/Remove Programs, затем нажмите кнопку Install.

Мастер (Wizard) Windows 95 будет направлять вас в течение всего процесса инсталляции.

## Варианты установки

---

После начала установки вы увидите экран, показывающий перечень вариантов.

1. Выберите варианты установки, отмечая компоненты, которые вы хотите установить, и нажмите кнопку ОК.
2. Определите целевой дисковод и имя каталога, затем нажмите кнопку ОК.

Стандартно используется C:\CW20. Однако программа установки запишет компоненты среды Clarion в подкаталог BIN, указанного каталога. Программа SETUP инсталлирует главные компоненты среды Clarion в подкаталог BIN того каталога, который вы определили в диалоге.

Программа установки Clarion for Windows инсталлирует все файлы системы в

указанный каталог и его подкаталоги. Она не устанавливает файлы в какие-либо другие каталоги.

Во время инсталляции индикатор выполнения будет следить за процессом копирования файлов программой установки.

3. Выберите YES или NO, когда программа установки запросит разрешения на изменение PATH.

Для Windows 3.x среда Clarion for Windows требует, чтобы подкаталог BIN был указан в PATH. При выборе NO, файл AUTOEXEC.BAT должен быть отредактирован вручную.

Еще одно единственное изменение в системных файлах заключается в том, что при первом запуске Clarion For Windows добавляет свой собственный раздел к WIN.INI.

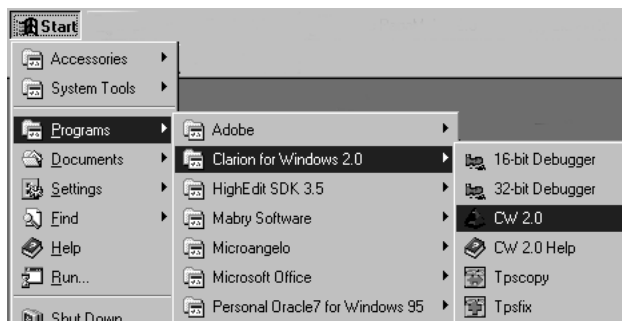
4. Выберите YES или NO, когда программа установки спрашивает нужно ли вы показывать файл ReadMe.

Если файл не прочитан сразу, его пиктограмму можно найти в группе Диспетчера программ (PROGRAM MANAGER) (или меню Start III Programs), которую создает программа установки. Мы рекомендуем прочитать его сразу после того, как программа установки скопирует все файлы.

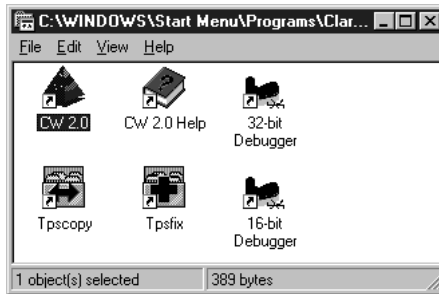
5. Нажмите кнопку ОК после завершения программы установки.

## Запуск Clarion For Windows

Для запуска Clarion For Windows перейдите в Clarion for Windows с помощью меню Start



или найдите пиктограмму Clarion For Windows в программной группе Clarion For Windows и дважды щелкните на ней мышью (если специально не оговорено, речь идет о левой кнопке мыши):



Появится готовая к работе интегрированная среда разработки Clarion for Windows.

- В следующей главе имеется краткий путеводитель по разделам среды.

## **Быстрый доступ к файлам и функциям**

Вероятно первое, что бросается в глаза после запуска Clarion For Windows - это пиктограммы на панели инструментов. Панель инструментов предоставляет кратчайший путь к наиболее часто используемым командам, которые управляют файлами и / или проектами. Кроме того, имеется еще одна группа “ускорителей”, размещенная в меню Clarion, которая также описывается в данном разделе.

## **Основная панель инструментов**

---

Основная панель инструментов включает следующие пиктограммы средств среды разработки:



Кнопка Выбор Эта кнопка открывает список выбора для всех типов файлов (подробнее в следующей главе). Нажатие кнопки эквивалентно выбору из меню File III Pick.



Кнопка Новый Эта кнопка вызывает диалоговое окно New, которое позволяет вам создать новое приложение, файл с исходными текстами на Clarion, словарь данных, файл проекта или текстовый файл. Нажатие этой кнопки эквивалентно выбору из меню File > New.





Кнопка Открыть Эта кнопка вызывает стандартное диалоговое окно Open (Открыть файл), которое позволяет “гулять” по дереву каталогов в поисках необходимого файла. Нажатие этой кнопки эквивалентно выбору из меню File > Open.



Кнопка Сохранить Эта кнопка сохраняет текущий файл любого типа. Сразу после запуска Clarion for Windows эта кнопка не действует, так как еще нет открытых файлов. Нажатие этой кнопки эквивалентно выбору из меню File > Save.



Кнопка Построить Эта кнопка компилирует и связывает текущий проект или приложение. Окно Make показывает подробности протекающих процессов. Нажатие этой кнопки эквивалентно выбору из меню Project > Make.



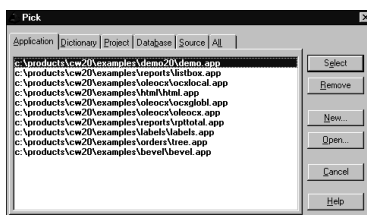
Кнопка Запустить Эта кнопка запускает компиляцию и редактирование связей текущего проекта или приложения (если текущие установки включают Auto Make Before Run - автоматическое построение перед прогоном), а затем запускает его. Нажатие этой кнопки эквивалентно выбору из меню Project > Run.



Кнопка Отладка Эта кнопка запускает компиляцию и редактирование связей (если текущие установки включают Auto Make Before Run) текущего проекта или приложения и загружает отладчик Clarion for Windows. Нажатие этой кнопки эквивалентно выбору из меню Project > Debug.

## Список выбора

Диалоговое окно Pick содержит в разделенном на категории Application (Приложение), Dictionary (Словарь базы данных), Project (Проект), Database (База данных), Source (исходный текст) и All (Все) списковом поле, перечень последних использованных файлов.



В каждой из категорий выводится список до двадцати последних использованных файлов соответствующего типа:

Диалоговое окно **Pick** содержит следующие кнопки:

<b>Select</b>	Открывает текущий выбранный файл.
<b>Remove</b>	Удаляет текущий выбранный файл из списка выбора.
<b>New</b>	Позволяет создать новый файл.
<b>Open</b>	Позволяет открыть отсутствующий в списке выбора файл.

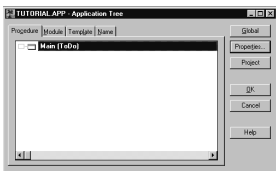
## Глава 3 Последовательность разработки



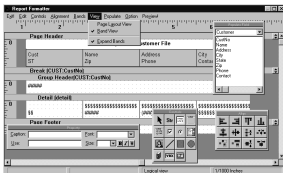
Здесь дается обзор того, как Генератор приложений соединяет в единое целое язык Clarion и все составные части Среды разработки приложений.



Словарь данных содержит описание базы данных, включая описания файлов, ключей, индексов, драйверов, взаимосвязей и др.



Генератор приложений на основании шаблонов, которые выбираются из регистра шаблонов, создает процедура за процедурой исходный текст приложения.



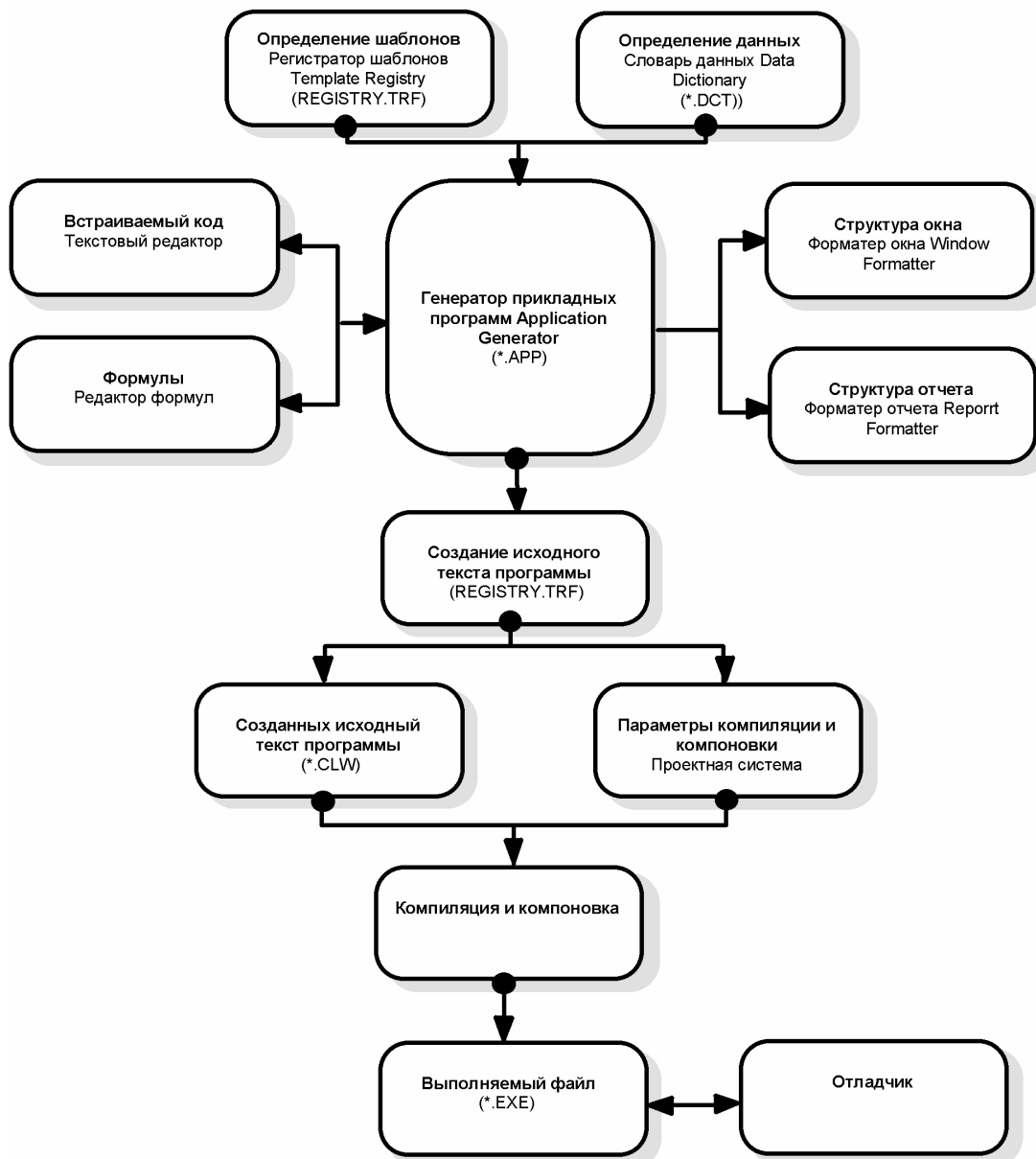
Окна и документы приложения разрабатываются с помощью визуальных инструментов проектирования.



Практически из каждого окна в Clarion for Windows имеется доступ к обширной контекстно зависимой справочной системе. Для этого нужно нажать кнопку Help или клавишу F1.

СХЕМА ЭТАПОВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ  
ПРОГРАММИРОВАНИЕ В CLARION  
ШАБЛОНЫ И ГЕНЕРАЦИЯ ИСХОДНЫХ ТЕКСТОВ  
ПРОЦЕСС РАЗРАБОТКИ  
СРЕДА РАЗРАБОТКИ CLARION  
РЕДАКТОР СЛОВАРЯ ДАННЫХ  
ГЕНЕРАТОР ПРИЛОЖЕНИЙ  
ФОРМАТЕР ОКНА  
ФОРМАТЕР ОТЧЕТА  
РЕДАКТОР ТЕКСТА  
РЕДАКТОР ФОРМУЛ  
ПРОЕКТНАЯ СИСТЕМА  
ОТЛАДЧИК  
КОНТЕКСТНЫЙ СПРАВОЧНИК

### Схема этапов разработки приложения



## ***Программирование в Clarion***

Clarion является бизнес - ориентированным языком программирования четвертого поколения (4GL), специально предназначенным для быстрой разработки приложений (RAD).

Clarion for Windows - это полная Среда разработки, которая поможет выполнить все, начиная от разработки своего собственного словаря данных до генерации исходной программы Clarion, поставки кода многократного использования, и управления компилированием, связыванием и распространением файлов.

Реализованный в Clarion for Windows язык Clarion полностью автоматизирует всю программистскую “кухню” взаимодействия с Windows, которую многие языки программирования оставляют на совести программиста.

В язык встроена независимость от драйвера файла: Clarion for Windows располагает DLL -драйверами для большинства популярных форматов PC-баз данных, а также другими отдельно поставляемыми драйверами.

### **Шаблоны и генерация исходных текстов**

---

Clarion for Windows генерирует исходные тексты несколькими способами. Исходные текста генерируют визуальные конструкторы: Форматер окон, Форматер отчетов, Форматер списков, Редактор формул. Исходные текста генерируют Мастера (Wizard): Мастер приложений, Мастер быстрого начала, процедурные Мастера. И, наконец, исходный текст генерируют шаблоны: процедурные шаблоны, вместе с диалоговыми, текстовыми и распределенными шаблонами.

Генератор приложений Clarion управляется шаблонами.

Шаблоны представляют собой сценарии кодогенерации. Обычно шаблон запрашивает некоторые сведения, а затем на их основе создает подходящий набор операторов исходного текста.

Разнообразные шаблоны этого пакета могут генерировать все, от отдельных операторов до законченных процедур и даже целые приложения. Отдельные сгенерированные тексты подходят друг к другу как взаимозаменяемые готовые детали.

Шаблоны обладают многими достоинствами объектно-ориентированного программирования, например многократным использованием кода, не требуя изучения заумного объектно-ориентированного языка.

Регистр шаблонов (REGISTRY.TRF) хранит заранее написанный выполняемый код и структуры данных, которые могут быть настроены в соответствии с требованиями разрабатываемого приложения или вкусами программиста. Можно модифицировать стандартные Clarion-шабоны, и сохранить результат в регистре шаблонов. Можно также добавлять чужие шаблоны и использовать их в дополнение к шаблонам Clarion и одновременно с ними. Можно писать собственные шаблоны. Язык шаблонов описан в руководстве программиста и в контекстной справочной системе.

### **Процедурные шаблоны**

Процедура представляет собой хранящийся набор операторов на языке Clarion, которые, собственно, и решают задачу. Процедурный шаблон является интерактивным инструментом, который запрашивает информацию у разработчика, а затем генерирует исходный текст для процедуры, настроенный именно на ту задачу, которую разработчик должен выполнить.

Clarion for Windows располагает большим выбором предварительно написанных процедурных шаблонов, с помощью которых можно быстро разрабатывать приложения для баз данных. В руководстве Getting Started демонстрационная программа “Quick Start Tutorial” демонстрирует несколько таких процедурных шаблонов, другие примеры процедур можно найти в руководстве “Hands on Tutorial”. Сначала выбирается процедурный шаблон, который генерирует программу, наиболее близкую к поставленной задаче, а затем все окончательно настраивается под требования пользователя с помощью других средств среды. Процедуры могут включать в себя такие элементы, как окна просмотра записей и оконные формы для редактирования отдельных записей.

Для включения процедуры в новое приложение в регистре выбирается соответствующий процедурный шаблон, который используется для добавления новой процедуры к файлу .APP (файл, где Генератор приложений хранит все процедуры). Если процедура обслуживает содержащее меню окно, процедуры меню автоматически добавляются к приложению и помечаются “To Do”.

Обычный способ настройки процедуры в соответствии с требованиям пользователя - это вызов одного из форматоров - форматора окна или форматора отчета, и добавление, например, нового окна или нового диалогового элемента. Форматоры - это инструменты визуального проектирования. Например, для того, чтобы поместить командную кнопку в верхний правый угол диалогового окна, вам необходимо выбрать эту кнопку на панели инструментов (TOOLBOX), и затем щелкнуть мышью в выбранном месте проектируемого окна.

Другим способом приведения процедуры в соответствие требованиям пользователя является добавление вставок исходного текста в тело будущей программы. По запросу программиста Генератор приложений показывает дерево, содержащее главные действия до, во время и после выполнения процедуры, и все события, которые могут быть связаны с окном или с вводными полями в окне данной процедуры. Можно выбрать точное место

для встраивания кода, а затем написать его вручную или использовать для его генерации текстовые шаблоны.

### **Шаблоны диалоговых элементов**

Диалоговыми элементами является практически все, что видно в окне или отчете. Например, поле флажков, кнопка, поле ввода и поле списка - все это является диалоговыми элементами.



Диалоговые шаблоны создают как эти элементы, так и программу для управления ими. Исходная программа, которую генерируют шаблоны может, например, загрузить данные из файла в структуру QUEUE (очередь), затем показать данные в списковом поле.

**Замечание:** обычно использовать диалоговый шаблон удобнее, чем просто диалоговый элемент.

### **Текстовые и распределенные шаблоны**

Текстовые шаблоны размещают сгенерированный исходный текст в указанных точках вставки. Например, шаблон InitiateThread генерирует один единственный оператор START для запуска процедуры, в качестве нового процесса в MDI приложении.

Распределенные шаблоны генерируют исполняемый исходный текст, обеспечивающие функциональные свойства, связанные скорее с процедурой, чем с конкретным диалоговым элементом. Текст может генерироваться в ряде предопределенных точек вставки. Например, шаблон DateTimeDisplay может показать в окне время и/или дату. Обычно каждый из них содержит экранные инструкции о том, как использовать в приложении его функциональные возможности.

## **Процесс разработки**

На верхнем уровне для разработки приложения нужно проанализировать ситуацию, а затем разработать и реализовать решение.

В анализе нужно идентифицировать и выделить данные и процедуры, которые их обрабатывают.

Для реализации хорошего решения нужно обеспечить эффективные, с точки зрения стоимости, методы реализации тех процедур, которые были идентифицированы во время анализа.

Clarion for Windows разработан для облегчения реализации эффективных процедур обработки данных. Каждая компонента среды играет свою специфическую роль в процессе реализации.



## Среда разработки Clarion

Среда разработки содержит семь основных функциональных частей, причем все они доступны друг из друга. При использовании Генератора приложений кнопки в различных диалогах ведут в другие части среды. Схема этапов разработки приложения в начале данной главы показывает, как эти части взаимодействуют друг с другом и с регистром шаблонов. В центре всего процесса разработки находится Генератор приложений.

В этом разделе дается описание каждой части в том порядке, в котором встретит каждую из них программист, пользующийся генератором приложений. Каждая часть имеет собственное диалоговое окно, через которое программист определяет функциональные возможности будущего приложения. Затем по команде Генератор приложений с помощью шаблонов, дополненных вставками исходного программного текста, создает исходный текст программы приложения. После этого проектная система компилирует и собирает приложение.

Программирование Clarion for Windows - это во многом “прогулка наедине” по ряду диалоговых окон. Нет обязательной последовательности, в которой вы должны “наполнять” диалоги, хотя некоторые являются предпосылкой для других. Далее предлагается обычный порядок, который уже доказал свою эффективность.

### Редактор словаря данных

---



Словарь данных (файл .DCT), обслуживаемый редактором словаря, содержит описание базы данных, включая ее файлы, ключи, индексы, драйверы базы данных, поля, описания полей, отношений, правила проверки содержимого полей и правила обеспечения целостности базы и др. Это первый файл, создаваемый при конструировании приложения.

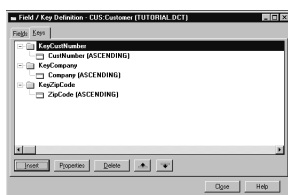
Можно создать определения файла с нуля или импортировать определения из существующих файлов данных.

Другие части среды разработки используют информацию из словаре, чтобы дать, например, возможность легко помещать поля данных в диалоговые окна, проектируемые для конечного пользователя. Генератор приложений создает программный текст для всех операторов, обращающихся к файлам базы данных, руководствуясь в значительной мере сведениями получаемыми им из словаря данных. А Мастер приложений даже создает полнофункциональное приложение, основываясь исключительно на содержании словаря данных.

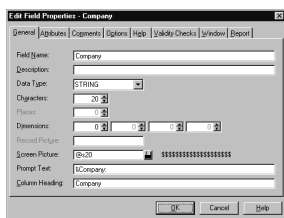
Новый словарь создается командой File> New, затем выбирается страничка Dictionary и нажимается кнопка Create. Откроется диалоговое окно Dictionary. В этом окне описываются файлы данных приложения и их алиасы (псевдонимы имен файлов). Кнопки вызывают диалоги New File Properties, New File Alias и New Relationship.



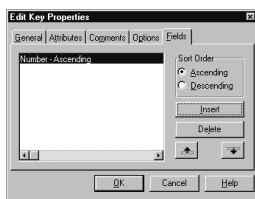
В диалоге New File Properties по очереди определяются имена и драйверы для всех файлов данных. Эта диалоговое окно позволяет также указать некоторые дополнительные параметры файлов, такие как Threaded (процесс), который определяет, что каждый использующий файл процесс получает свой собственный буфер записи. Это полезно при использовании MDI интерфейса.



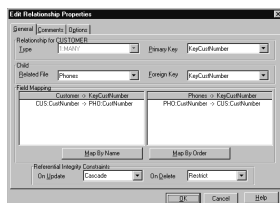
В диалоговом окне Field/Keys Defenition (определение поля/ключей) нажатием кнопки Insert (вставка) определяются поля, ключи и индексы файлов. Вся информация в этом диалоге организована иерархически.



В диалоговом окне New Field Properties (свойства нового поля) определяются поля, их размеры и типы данных. Можно также предварительно задать некоторые специфические свойства полей, такие, как, например, автоматическое выравнивание текста. Можно также вернуться к предыдущему диалогу определения ключей и взаимосвязей.



Диалоговое окно Key Components (компоненты ключа) предназначено для определения компонент ключа. Clarion for Windows автоматически строит правильный ключ, даже если объявлен ключ, содержащий поля разных типов. Из этого диалога можно вернуться к диалогу Dictionary (словарь) для определения взаимосвязей.



Взаимосвязи между файлами задаются в диалоговом окне New Relationship Properties (свойства нового отношения). С помощью диалоговых элементов этого окна можно установить требования ссылочной целостности базы данных (RI). После того, как большая часть словаря определена, нужно сохранить словарь и перейти к файлу .APP.

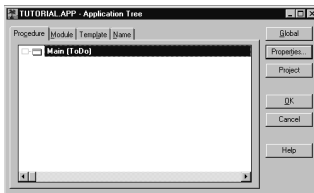
## Генератор приложений

Генератор приложений генерирует исходный текст приложения, процедура за процедурой, основываясь на заранее разработанных шаблонах, выбираемых из регистра шаблонов. Это позволяет добавлять глобальные и локальные переменные и настраивать процедуры под требования пользователя с помощью инструментов визуального проектирования и вставок исходного текста.

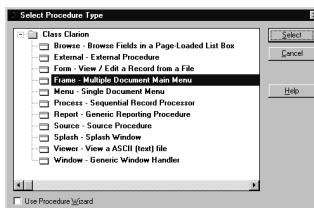
Генератор приложений обеспечивает доступ к другим частям интегрированной среды разработки для разработки внешнего вида и функциональных возможностей окон, меню, отчетов и других элементов пользовательского интерфейса.



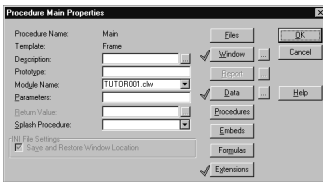
Разработка нового приложения начинается с помощью команды File > New и выбора типа файла Application (приложение). В диалоговом окне Application PropertiesXE “(свойства приложения)” задаются основные параметры приложения - имя приложения, файл словаря данных, имя HELP- файла и шаблон приложения. После этого Генератор приложений создает .APP файл и показывает дерево приложения.



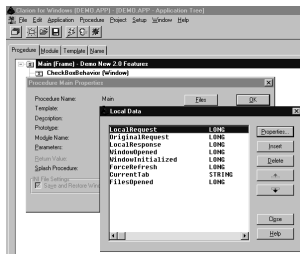
Обзор и сопровождение приложения выполняется в диалоговом окне Application Tree (дерево приложения). Здесь иерархически в виде дерева представлены все процедуры приложения, а те из них, которые еще должны быть определены, отмечены как “To Do”. Для определения глобальных переменных и других параметров нужно нажать кнопку Global.



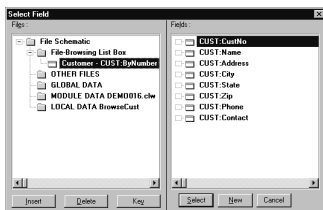
Функциональный тип “To Do” процедур задается в диалоговом окне Select Procedure Type (выбор типа процедуры). В списке представлены доступные типы процедурных шаблонов, такие, как таблица или форма. Кнопка Select выбирает один из них и открывает диалог Procedure Properties (свойства процедуры).



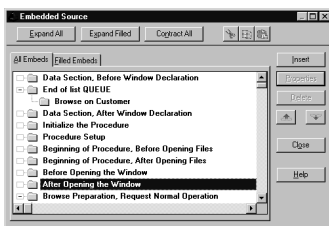
Диалоговое окно Procedure Properties - это координирующий центр для всех прочих диалогов, которые позволяют адаптировать типовые предоставляемые шаблонами процедуры для выполнения необходимых работ так, как это требуется. Для определения локальных переменных нужно нажать кнопку Data (данные).



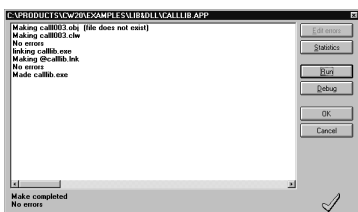
Локальные переменные и порядок, в котором программа будет их инициализировать, определяются в диалоге Data. Определение имени переменной, типа, размера и т. п. производится в диалоговом окне, идентичном диалогу New Field Properties, которое открывается по нажатию кнопки Insert.



Диалог Select Fields (выбор полей) открывается кнопкой Files. В нем выбираются файлы, ключи, псевдонимы (алиасные имена), структуры VIEW (виртуальные файлы) и поля, которые будет использовать процедура и ее диалоговые элементы.



Кнопка Embed (вставка) вызывает диалоговое окно Embedded Source (встраиваемый исходный текст). Оно позволяет встраивать фрагменты исходного текста перед, во время или после выполнения процедуры или в те места процедуры, которые ответственны за обработку событий, связанных с окном или его вводными полями.



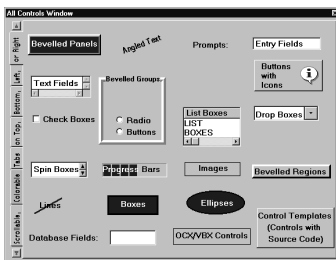
Выбрав точку вставки нужно нажать кнопку Insert (вставить).

После настройки процедурного шаблона в соответствии с требованиями приложения с использованием формatera окон, формatera отчетов и текстового редактора, нужно вернуться к окну Application Tree и сгенерировать текст программы. Затем он компилируется собирается и выполняется!

## Форматер окна

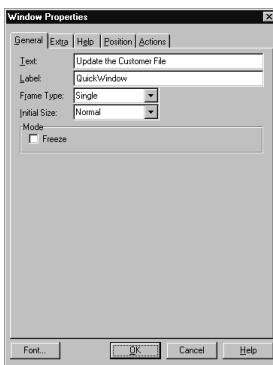
Все, что видит конечный пользователь - окна и диалоговые элементы приложения - визуально проектируется в форматере окна. Он автоматически формирует программный текст для всех элементов, которые были визуально спроектированы на экране.

При использовании Генератора приложений форматер окна вызывается нажатием кнопки Window в диалоге Procedure Properties.

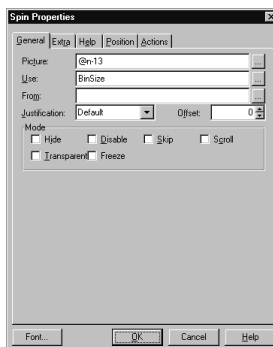


Форматер окон показывает изображение разрабатываемого окна. Щелчком мыши на панели инструментов выбирается нужный диалоговый элемент, а затем вторым щелчком он размещается в нужном месте окна.

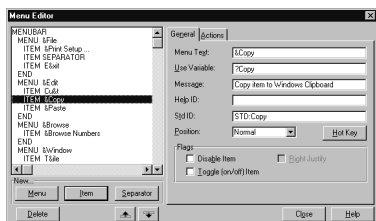
С помощью команды Preview! (Предварительный просмотр), можно увидеть окно точно в том виде, в котором оно предстанет перед конечным пользователем.



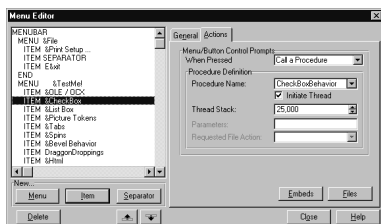
Каждое окно и каждый диалоговый элемент окна имеют связанные с ними диалоговые окна, которые определяют их поведение, а в случае вводного поля, и их содержание. Щелчок правой кнопкой мыши над диалоговым элементом или окном открывает доступ к их диалогам свойств. Диалоговое окно Window Properties (свойства окна) определяет базовые характеристики окна, такие как тип окна, наличие системного меню, заголовок, размер, форма, полоса прокрутки и т. д.



Типичный диалог свойств диалогового элемента задает такие параметры, как метка, размер, форма, цвет, шрифт и, а для полей ввода - связанную с диалоговым элементом переменную для хранения его содержимого.

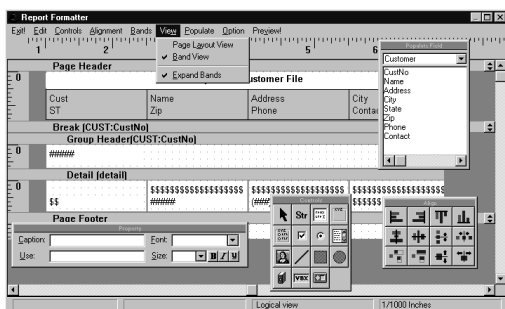


Для вызова редактора меню нужно выполнить команду Menu > New Menu. Здесь редактируется текст меню, с помощью кнопки Item (пункт) добавляются новые пункты и задается их функциональность, либо выбором выполняемой процедуры, либо указанием на стандартную процедуру Windows, например Cut, Copy или Paste.



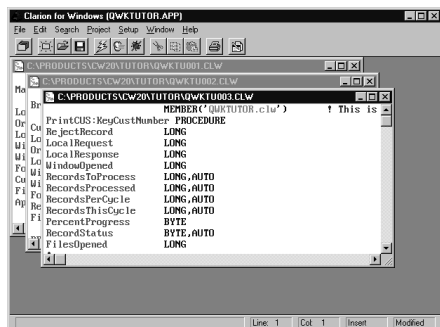
На страничке Actions задается связь пункта меню с вызовом процедуры, которая будет выполняться, когда пользователь выберет в меню эту команду.

## Форматер отчета



Форматер отчета взаимосвязан с Генератором приложений, во многом так же, как и форматер окна. Диалоговые элементы размещаются на полосах отчета. Во время работы программы процессор печати обрабатывает записи файлов, управляет переводом страниц, обрабатывает окончание групп, размещает верхние и нижние колонтитулы в полном соответствии с проектным описанием.

## Редактор текста

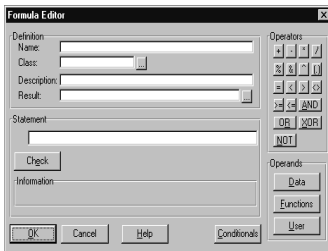


Редактор текста - это полнофункциональный текстовый редактор программиста, который используется для написания исходного текста программы вручную.

Наиболее вероятно, что при использовании Генератора приложений, Редактор текста будет вызываться преимущественно для создания вставок выполняемых процедурой действий. Или же можно написать от начала и до конца целиком все приложение.

Текстовый редактор представляет возможность выделять цветом синтаксические конструкции программы, облегчая тем самым идентификацию различных операторов языка Clarion. Он также располагает всеми возможностями для поиска и замены текста и всеми остальными стандартными средствами редактирования.

## Редактор формул



Редактор формул помогает быстро генерировать и сопровождать простые и сложные операторы присваивания. Он обеспечивает синтаксический контроль и мгновенный доступ ко всем переменным функциям и операциям, которые могут использоваться в операторах присваивания.

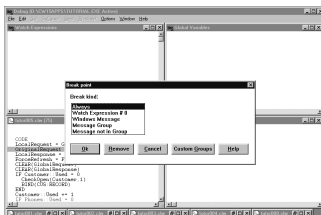
## Проектная система

Генератор приложений создает для приложения файл проекта. Файл проекта задает режимы компилятора и редактора связей, такие, как необходимость включения отладочного кода, выбор методов оптимизации исполняемого кода, внешние драйверы файлов и т. д.

Дерево проекта показывает файлы исходной программы, библиотеки и другие внешние файлы, участвующие в процессе компиляции и сборки приложения. Задание нужных режимов делается с помощью кнопки Properties (свойства). При работе с Генератором приложений файл проекта автоматически поддерживается Генератором приложений.

## Отладчик

Отладка программы обычно требует прогона программы и ее периодической остановки для проверки текущих значений различных переменных. Отладчик состоит из ряда окон, в которые выводятся исходные тексты, текущие значения переменных, имена активных в данный момент процедур и многое другое.



В проектной системе устанавливается режим включения в .EXE файл отладочной информации, а затем запускается отладчик с помощью нажатия кнопки Debug в диалоге Compile Results или выбором команды Project > Debug.

Самый простой способ отладки приложения заключается

в идентификации той части программы, которая предположительно вызывает ошибку, и в установке точки прерывания в этой части программы.

Затем делается прогон программу, и отладчик приостановит ее в указанной точке для проверки значений переменных. Это поможет в выявлении ошибки и добиться совершенства приложения!

## Контекстный справочник

---

Практически из каждого окна в Clarion for Windows имеется доступ к обширной контекстно зависимой справочной системе. Для этого нужно нажать кнопку Help или клавишу F1.



Раздел Common Questions содержит указания и примеры решения типичных задач. Текст примера можно вырезать и вставить непосредственно в текст программы. Выберите Help > Contents и нажмите кнопку Common Questions.

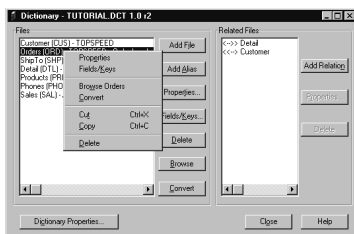
Справочная система Clarion for Windows состоит из основного и дополнительных справочных файлов. В них можно осуществлять поиск по индексам и ключевым словам. Для поиска по индексам и ключевым словам в дополнительных справочниках откройте их непосредственно в Диспетчере программ.



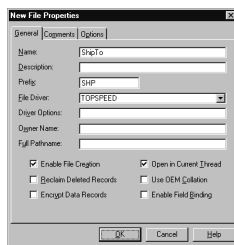
## Глава 4 Редактор словаря данных



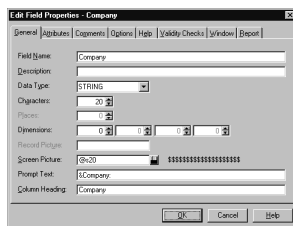
В этой главе описано создание Словаря базы данных. Это определение файлов данных приложения, их полей данных, ключей, правил проверки вводных полей, режимов ввода, связей между файлами и требований ссылочной целостности.



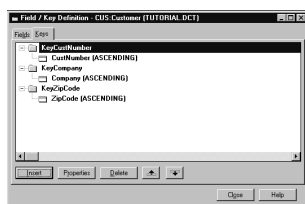
При создании Словаря должны быть определены все файлы, образующие базу данных. См. раздел добавление файлов к словарю.



Для каждого файла задается имя и выбирается драйвер файла. См. диалог New File Properties (свойства нового файла).



Далее создаются поля файлов и определяются их свойства, включая стандартные диалоговые элементы ввода и условия проверки. См. Добавление и модификация полей.



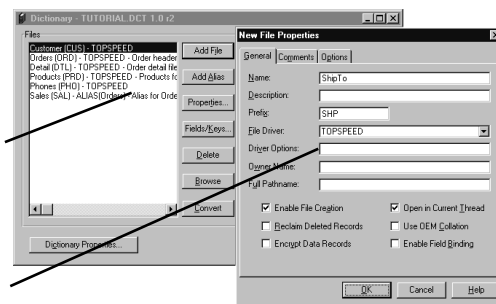
Сортировка записей основывается на содержимом отдельных полей файла или на их комбинации, что и определяется при задании ключей. См. Добавление и модификация ключей.

ЧТО ТАКОЕ СЛОВАРЬ ДАННЫХ  
ПРЕИМУЩЕСТВА ИСПОЛЬЗОВАНИЯ СЛОВАРЯ ДАННЫХ  
ФУНКЦИИ РЕДАКТОРА СЛОВАРЯ  
РАЗРАБОТКА СЛОВАРЯ И БАЗЫ ДАННЫХ  
НОРМАЛИЗАЦИЯ  
КЛЮЧИ  
РЕЛЯЦИОННЫЕ ОПЕРАЦИИ  
РЕДАКТОР СЛОВАРЯ ДАННЫХ  
СОЗДАНИЕ СЛОВАРЯ ДАННЫХ  
ОТКРЫВАЯ РЕДАКТОР СЛОВАРЯ  
ДОБАВЛЕНИЕ В СЛОВАРЬ ФАЙЛОВ  
НЕСКОЛЬКО СЛОВ О БЫСТРОЙ ЗАГРУЗКЕ QUICK LOAD  
ИМПОРТ ОПРЕДЕЛЕНИЙ ФАЙЛОВ  
СВОЙСТВА НОВОГО ФАЙЛА  
ДОБАВЛЕНИЕ В СЛОВАРЬ ПСЕВДОНИМА ФАЙЛА  
ДОБАВЛЕНИЕ И ИЗМЕНЕНИЕ ПОЛЕЙ  
ОПРЕДЕЛЕНИЕ СВОЙСТВ ПОЛЯ  
ДОБАВЛЕНИЕ И ИЗМЕНЕНИЕ КЛЮЧЕЙ  
ЗАДАНИЕ СВОЙСТВ КЛЮЧА  
КОМПОНЕНТЫ КЛЮЧА  
ДОБАВЛЕНИЕ И ИЗМЕНЕНИЕ СВЯЗЕЙ  
УСТАНОВКА ТРЕБОВАНИЙ ПО ОБЕСПЕЧЕНИЮ ССЫЛОЧНОЙ  
ЦЕЛОСТНОСТИ  
УПРАВЛЕНИЕ СЛОВАРЕМ  
КОПИРОВАНИЕ И ВСТАВКА  
ВЕРСИИ СЛОВАРЯ  
СРЕДСТВА НАСТРОЙКИ РЕДАКТОРА СЛОВАРЯ

Эта глава описывает создание словаря данных. Шаблоны и Генератор приложений генерируют операторы языка Clarion для выполнения широкого круга функций, основываясь в значительной мере на том, как сконструирован словарь базы данных. В этой главе объясняется:

- Что такое словарь данных и для чего он предназначен.
- Как структура проектируемых файлов данных и их параметры, которые задаются в диалогах словаря, влияют на эффективность хранения данных приложения. Здесь же дается краткое изложение теории реляционных баз данных.
- Как организовать проверку вводимых данных и установить режимы ввода данных для конечного пользователя.
- Как задать стандартные связанные с данными диалоговые элементы. Clarion даже позволяет задать в словаре данных такие элементы управления, как прокручиваемые поля или специальные списковые поля. Генератор приложений автоматически разместит эти диалоговые элементы в создаваемых для приложения вводных окнах.

Диалог Dictionary показывает все файлы базы данных, включая виртуальные файлы и псевдонимы файлов.



Каждый файл данных определяется в отдельном диалоге File Properties

## Что такое словарь данных

Словарь данных - это центральное хранилище сведений о файлах приложения и о полях этих файлов. Здесь, в том числе, хранится информация о том, где и как будут храниться данные на диске, а также о том, как данные будут представлены конечному пользователю в отчетах и на экране компьютера.

Файл словаря (.DCT) хранит:

- ✓ имена файлов
- ✓ описания файлов
- ✓ структуры файлов
- ✓ ключи файлов
- ✓ индексы файлов
- ✓ связи между файлами
- ✓ типы полей данных
- ✓ описания полей
- ✓ тексты всплывающих пояснений к полям
- ✓ условия проверки полей
- ✓ шаблоны ввода полей
- ✓ стандартные диалоговые элементы для представления полей в окнах и отчетах
- ✓ сообщения в строке состояния
- и многое другое.

## Преимущества использования словаря данных

---

Преимуществом централизованного хранения всей этой информации является то, что такой подход экономит огромное количество времени при разработке и сопровождении приложений. Кроме того, это придает приложению большую последовательность и обеспечивает конечному пользователю более короткие сроки обучения. Это происходит благодаря тому, что поля, файлы, взаимосвязи, правила проверки и т. д. определяются только один раз, вместо того, чтобы определять их при каждом использовании в приложении.

Хранящаяся в словаре данных информация определяет стандартный метод работы с данными. При ее задании в словаре данных формируются стандартные методы обращения с каждым файлом и с каждым полем, которые применяются каждый раз, когда используется данный файл или поле. Это означает, что метод работы с данными разрабатывается только один раз, вне зависимости от того, сколько раз поле используется в приложении, и сколько приложений пользуются этим словарем, однако в каждом конкретном случае есть возможность модифицировать этот стандартный метод.

## Функции редактора словаря

---

Ниже приводится перечень основных функций, выполняемых редактором словаря и диалогов, в которых осуществляется их выполнение.

- ☐ Манипуляции файлами и связями файлов в диалоге Dictionary (словарь).
- ☐ Выбор драйвера файла и назначение имени и места расположения файла данных в диалоге Edit (или New) File Properties (редактирование свойств (или свойства нового) файла).
- ☐ Манипуляции полями и ключами в диалоге Field/Key Definition (определение поля/ключа).
- ☐ Определение каждого из полей и их типов данных в диалоге Edit (или New) Field Properties (редактирование свойств (или свойства нового) поля).
- ☐ Определение связей между файлами в диалоге Edit (или New) Relationship Properties (редактирование свойств (или свойства новой) связи между файлами).

## Разработка словаря и базы данных

В этом разделе дается краткий обзор теории реляционных баз данных. Тщательное планирование и организация базы данных в самом начале разработки приложения может привести к созданию более эффективного приложения, не говоря уже о сэкономленных часах перепроектирования в более позднее время.

Реляционная модель имеет дело с тремя аспектами управления данными: структурой, целостностью и манипуляцией. Для наших целей мы обсудим три практических требования, которые возникают в связи с этими аспектами - это нормализация данных, ключи и реляционные операции.

### Нормализация

---

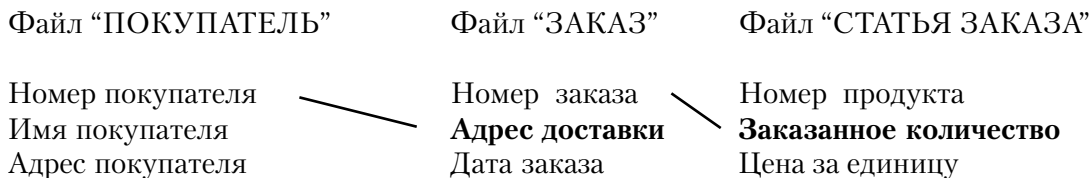
В своем простейшем виде нормализация означает, что каждый элемент данных запоминается в базе только один раз. Чтобы избежать дублирования внутри базы данных, в хорошо спроектированной базе данные разносятся в отдельные взаимосвязанные файлы.

В качестве примера рассмотрим простейшую систему ввода заказов, которой нужно хранить следующие данные:

Номер покупателя  
Имя покупателя  
Адрес покупателя  
Адрес отгрузки  
Номер заказа

Номер продукта  
Заказанное количество  
Цена за единицу продукта

Эти данные могли бы целиком храниться в записях одного файла (один заказ - одна запись - пер.), но это было бы очень неэффективно (если только покупатели не повторяются). В повторном заказе покупателя дублировались бы все сведения о покупателе. Чтобы исключить это дублирование, необходимо разделить данные на отдельные файлы:



Это организует данные в логическую схему и устраняет дублирование. Процесс задания связи записей в одном файле с записями в другом файле требует добавления дополнительных полей по крайней мере к двум файлам, с тем, чтобы файлы хранили общие значения. Это будет обсуждаться далее.

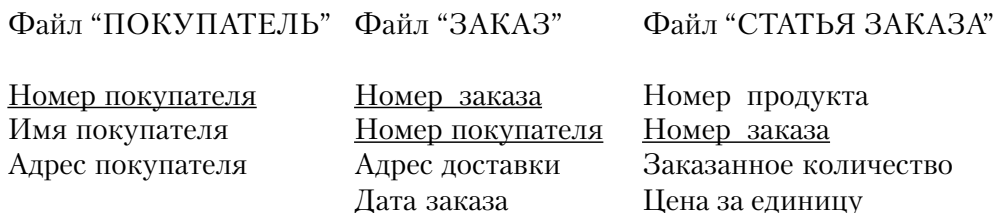
Строгая реляционная теория требует, чтобы:

- База данных состояла из одной или большего числа таблиц, которые примерно соответствуют файлам словаря данных Clarion.
- Таблица состояла из названий колонок (которые на уровне файлов мы будем называть полями) и некоторого количества (возможно, равного нулю) строк (записей).
- Каждая запись содержала в точности одно значение для каждого поля.

## Ключи

---

Чтобы в системе ввода заказов, приведенной выше, связать записи в файлах покупателя, заказа и статей заказа, можно было бы добавить по одному полю к двум файлам, как показано ниже:



В реляционной теории баз данных утверждается:

- Первичный ключ должен существовать для каждой таблицы. Первичный ключ - это уникальное поле или уникальная комбинация полей. Первичный ключ не должен принимать нулевое или пустое значение.
- Внешний ключ может соответствовать первичному ключу другой таблицы. Если таблица “А” содержит внешний ключ, который совпадает с первичным ключом таблицы “В”, то каждое значение ключа в таблице “В” должно быть равным одному из значений первичного ключа в таблице “А” или быть пустым.

В приведенном выше примере Номер Покупателя - это первичный ключ (могут быть два “Джона Смита”, но не два заказчика #1001’й). Поле Номера Заказчика добавляется к другому файлу как внешний ключ.

Можно определить три типа соотношений между файлами:

✓ **Один-к-многим.** Одна запись в одном файле связана со многими записями в другом файле. В вышеприведенном примере один Покупатель может быть связан со многими записями в файле заказов (каждый покупатель может делать много заказов). В бизнес-приложениях это одно из наиболее распространенных отношений. Оно носит также название отношения Родитель-Потомок.

✓ **Один-к-одному.** В точности одна запись в одном файле связана с одной записью в другом файле. Это наиболее подходит для случая, когда один файл может иметь или не иметь данные в некоторых полях. Если бы все эти поля были в одном файле, то пространство диска было бы потрачено на пустые поля.

Если бы в вышеприведенном примере адрес доставки редко отличался бы от адреса заказчика, вы бы могли поместить его в другой файл.

✓ **Многие-к-многим.** Несколько записей одного файла связаны с несколькими записями другого файла. Чтобы применить этот случай к нашему примеру, предположим, что система ввода заказов была ориентирована на пользователя, который одновременно и покупает детали и производит из них продукцию. Если какой либо продукт (детали или комплектующие) мог бы быть использован для производства многих других продуктов и наоборот, то два дополнительных файла могли бы выглядеть так:

#### **Файл деталей**

Номер детали  
Описание детали

#### **Файл продукта**

Номер продукта  
Описание продукта

## Реляционные операции

---

Теория реляционных баз данных выделяет набор операторов для манипулирования данными. Три операции, которые рассматривают теоретики для систем реляционных баз данных, это Выбор (Select), Проекция (Project) и Объединение (Join). Система не обязана явно поддерживать соответствующие операторы языка. Главное, чтобы в рамках системы реализовывались соответствующие функции. С теоретической точки зрения таблица состоит просто из набора заголовков - колонок (или полей) плюс произвольное (возможно, нулевое) количество строк (записей) данных.

- Операция Выбор извлекает подмножество строк из данной таблицы. Другими словами, выделяет подмножество записей, которое удовлетворяет некоторому условию.
- Проекция выделяет подмножество столбцов (полей) в данной таблице. Другими словами, подмножество указанных полей, из которого затем удаляются повторяющиеся записи (пример ниже).
- Реляционное Объединение объединяет вместе две таблицы. Результат - образование новой, более обширной таблицы.

Выбор (не путать с оператором “Select” в SQL) обеспечивает средства для просмотра таблицы и извлечения записи или записей. База данных должна быть способна оценить условие по информации в отдельной записи, изолированно, не обращаясь к другим записям. В нашем примере выделение записи или записей (рассматривая все файлы), которые отвечают условию “Номер Заказчика = 100”, является примером реляционного выбора.

**Проекция** выделяет уникальные значения полей. В вышеприведенном примере предполагается, что в файле статей заказа имеется многократное дублирование номеров продуктов. Проекция файла “Статья заказа” по полю “Номер продукта” дает новую таблицу всех продуктов, заказанных заказчиком (не обязательно совпадающих со всеми сделанными продуктами) в файле продукта. Все проданные продукты будут иметь одно и только одно вхождение в список.

**Объединение:** Вернемся к примеру: для того, чтобы работать со всеми возможными комбинациями деталей и продуктов, необходимо построить некоторое особое соотношение между этими двумя файлами. Решением является определение третьего файла, называемого файлом “Объединение”. Этот файл создает два отношения Один-к-Многим. Отношения между тремя файлами будут определены:

Файл деталей:

Номер детали(Первичный ключ)



Описание детали

Файл ДеталиПродукта:

Номер детали (1-ый компонент первичного ключа и внешний ключ)

Номер продукта (2-й компонент первичного ключа и внешний ключ)

Используемое количество

Файл продуктов:

Номер продукта (Первичный ключ)

Описание продукта

Файл ДеталиПродукта имеет многокомпонентный первичный ключ и два внешних ключа. Отношение между файлом “Детали” и файлом ДеталиПродукта. - Один-к-Многим. Отношение между файлом “Продукты” и файлом ДеталиПродукта - это также отношение Один-к-Многим. Это делает файл объединения “посредником” между двумя файлами с отношением Многие-к-Многим.

Обычно файл объединения содержит дополнительную информацию. В данном примере поле “Используемое количество” логически принадлежит к файлу ДеталиПродукта.

## Редактор словаря данных

---

Язык Clarion поддерживает три аспекта управления данными, с которыми имеет дело реляционная теория баз данных. Редактор словаря - это инструмент планирования структуры и целостности базы данных, инструмент двух “правил” реляционной модели. Редактор словаря позволяет вам “предварительно сконструировать” некоторые реляционные операции, определенные в теории баз данных; остальными операциями занимаются операторы в языке Clarion.

- Редактор словаря позволяет легко задать подходящую структуру базы данных путем определения файлов, полей и соотношений.
- Редактор словаря позволяет легко планировать как первичные, так и внешние ключи базы данных в соответствии с правилами целостности реляционной модели.
- Редактор словаря позволяет без реализовать требования ссылочной целостности данных, автоматически поддерживая синхронизацию связанных файлов за счет “каскадных” изменений в файлах и за счет “запрета” таких изменений и удалений, которые бы вызвали взаимную несовместимость файлов.

## Создание словаря данных

Далее приводится сценарий процесса создания словаря данных, т.е. определения файлов, полей, ключей и связей файлов. Этот сценарий содержит лишь основные шаги разработки. Многие возможности описываемых здесь диалоговых окон словаря данных оставляются в их стандартном первоначальном состоянии и даются основные описания того, что делают диалоговые поля в редакторе словаря. Все подробности касающиеся этих возможностей и других элементов диалога рассматриваются более полно в оставшейся части этой главы.

□ Определите файлы базы данных и поля каждого файла:

1. Выберите в меню среды File>New, затем выберите страничку Dictionary.

2. Укажите путь (папку) и File Name (имя файла) для файла словаря данных, затем нажмите кнопку Create (создать).

Появляется диалог Dictionary.

3. Нажмите кнопку Add File (добавить файл), затем, в ответ на вопрос, хотите ли вы воспользоваться Quick Load (быстрой загрузкой), нажмите кнопку No.

Появится диалог New File Properties (свойства нового файла). Для использования Quick Load для добавления файлов к словарю данных, посмотрите краткое описание этой процедуры в расположенном далее разделе О процедуре Quick Load.

4. На страничке General (общий) введите Name (имя), Prefix (префикс) и выберите для файла данных File Driver (драйвер файла), затем нажмите ОК чтобы закрыть диалог.

5. Нажмите кнопку Fields/Keys (поля/ключи) чтобы открыть диалог Field/Key Definition (определение полей/ключей)

6. На страничке Fields (поля) нажмите кнопку Insert (вставить).

Появится диалог New Field Properties (свойства нового поля).

7. На страничке General (общий) введите имя поля в поле Name (имя), выберите Data Type (тип данных) и задайте длину в символах.

8. На страничке Validity Checks (проверка данных) и выберите способ проверки данных данного поля.

9. На страничке Window (окно) задайте, в каких диалоговых элементах данное поле и

его приглашение будут появляться в окнах и диалогах приложения.

10. На страничке Report (отчет) задайте, как данное поле будет выглядеть в напечатанном отчете.

11. Чтобы закончить с данным полем и перейти к следующему нажмите кнопку ОК.

Снова появится готовый к вводу следующего поля диалог New Field Properties.

12. Повторите для остальных полей файла шаги с 7 по 11.

По завершению каждого из полей появляется диалог New Field Properties , готовый для задания следующего поля.

13. Для возврата в диалог Field/Key Definition в пустом диалоге New Field Properties, который появляется после добавления последнего поля, нажмите кнопку Cancel (отменить).

☐ Определите ключи файлов:

1. Находясь на страничке Keys (ключи), нажмите кнопку Insert (вставить) для того, чтобы открыть диалог New Key Properties (свойства нового ключа).

2. В позиции General (общий) наберите Key Name (имя ключа).

3. На страничке Fields (поля) нажмите кнопку Insert (вставить), после чего появится диалог Insert Key Components (вставить компоненты ключа).

Ключ состоит из одного или нескольких полей файла. Таким образом диалог Insert Key Components позволяет задать, какие поля войдут в ключ.

4. Выделите в списке нужное поле, щелкнув над ним мышью, а затем нажмите кнопку Select (выбрать).

Снова нажмите кнопку Insert (вставить), если к ключу нужно добавить дополнительные поля.

5. Для всех остальных ключей файла повторите шаги с 2 по 4.

6. Для возврата в диалог Field/Key Definition нажмите кнопку Cancel (отменить).

7. Для возврата в диалог Dictionary нажмите кнопку Close (закрыть).

Повторите вышеприведенные последовательности формирования файлов, полей и

ключей для всех остальных файлов базы данных.

☐ Определите связи между файлами:

1. Выберите файл для которого нужно задать связи с другими файлами и нажмите кнопку Add Relation (добавить связь с файлом) в группе Related Files диалога Dictionary.

2. Выберите тип (Type) из выпадающего списка тип связи.

3. Выберите Related File (связанный файл) из выпадающего списка.

Это другой файл в соотношении.

4. Из соответствующего выпадающего списка выберите Primary Key (первичный ключ) для первоначального файла и Foreign Key (внешний ключ) для связанного файла.

5. Нажмите кнопку Map by Name (сопоставить по именам) или Map by Order (сопоставить по порядку) для установления связи между первичным и внешним ключами.

6. Для возвращения в диалог Dictionary нажмите кнопку OK.

7. Выберите File>Save As для сохранения файла .DCT.

## **Открывая редактор словаря...**

Обычно создание словаря данных является первым шагом на пути создания приложения. Следовательно, это первое с чем приходится иметь дело в меню интегрированной среды разработки.

☐ Чтобы открыть Редактор словаря для создания нового файла словаря:

1. Выберите в меню среды File> New, затем выберите страничку Dictionary (словарь).

2. Укажите путь (папку) и File Name (имя файла) для файла словаря данных, затем нажмите кнопку Create (создать).

☐ Чтобы открыть Редактор словаря для редактирования существующего файла словаря:

1. Выберите File> Open, затем выберите страничку Dictionary (словарь).

2. Измените при необходимости устройство и путь и укажите файл словаря, который нужно открыть. Выполните двойной щелчок мышью над его именем в списке File Name (имя файла) или выберите его и нажмите кнопку Open (открыть).

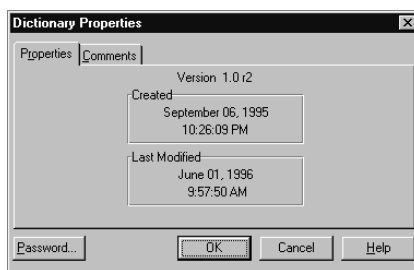
Один и тот же словарь данных можно использовать более чем для одного приложения. Приложение, однако, может иметь только один словарь данных.

**Совет:** Clarion for Windows автоматически распознает и преобразует формат словари данных Clarion for DOS 3.007 (и выше). Он импортирует все атрибуты, за исключением атрибутов размеров экранного поля для мемо-полей. Кроме того, так как в Clarion for Windows требования стандартной реляционной модели несколько усилены, некоторые соотношения не могут быть полностью преобразованы из-за более строгого контроля ошибок.

☐ Для того, чтобы добавить текстовое описание к словарю данных:

1. Нажмите кнопку Dictionary Properties (Свойства словаря) в нижней части диалогового окна.

2. На страничке Comments (комментарии) наберите описание в отведенном для этого месте.



Описание служит только для удобства и никак не влияет на приложение. Оно полезно в ситуациях, когда разработка проекта продолжается другими программистами или когда приходится возвращаться к проекту через длительный промежуток времени.

☐ Для того, чтобы добавить пароль к словарю данных:

1. Нажмите кнопку Password (Пароль).

2. Когда появится диалоговое окно Password Validation (задание пароля), наберите пароль в отведенном для этого месте и нажмите кнопку ОК.

3. Когда появится диалоговое окно Password Verification (проверка пароля) наберите тот же пароль и нажмите кнопку OK.



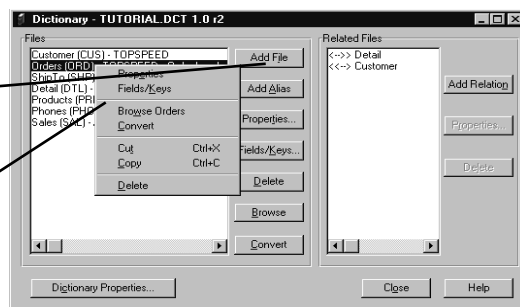
Пароль поможет защитить разработанную базу данных от несанкционированного доступа.

## Добавление в словарь файлов

Основной функцией словаря является определение файлов данных приложения. Определите файлы, добавляя их в левой части диалогового окна Dictionary. Любая из двух кнопок “Add” (добавить) справа от списка позволит сделать добавление к списку.

Для добавления нового файла нажмите кнопку Add File

Чтобы вызвать всплывающее меню операций, нажмите над именем файла или над взаимосвязью правую кнопку мыши.



□ Для добавления файла к списку файлов данных нажмите кнопку Add File (добавить файл). Это вызовет появление диалогового окна New File Properties. Другой способ создания файла состоит в импорте его определения из существующего файла данных с помощью операции File > Import File

□ Чтобы добавить к списку псевдоним файла, нажмите кнопку Add Alias (добавить псевдоним). Это вызовет появление диалога New Alias (Новые псевдонимы). См. далее раздел Добавление псевдонимов файла.

Строка заголовка диалогового окна содержит имя текущего файла словаря. Выберите имя файла в левой части окна и затем нажмите нужную кнопку справа от списка файлов для вызова соответствующего диалогового окна.

## Несколько слов о быстрой загрузке Quick Load

---

При нажатии кнопки Add File (добавить файл) вам будет предложена возможность использования быстрой загрузки Quick Load для добавления файла к словарю данных. Быстрая загрузка дает вам возможность задавать только самую основную информацию о файле и его полях, заполняя все остальные необходимые атрибуты стандартными значениями. Быстрая загрузка особенно полезна для быстрого создания работающего приложения, которое позднее может быть настроено более аккуратно.

Напротив, если выполнено подробное планирование и спецификация проекта, может оказаться предпочтительным добавлять файлы не применяя быструю загрузку, а используя преимущества предоставляемые всевозможными атрибутами полей и файлов, поддерживаемых словарем данных Clarion с самого начала. Например, словарь данных поддерживает проверку ввода данных, но при использовании быстрой загрузки проверка по умолчанию отсутствует. В следующем разделе предполагается, что для добавления файла быстрая загрузка не используется.

См. *обучение работе с Quick Load в руководстве Быстрый старт.*

## Импорт определений файлов

---

Редактор словаря позволяет быстро добавлять в Словарь описания файлов, создавая описания файлов на основе существующих файлов.

1. Находясь в диалоге Dictionary, выберите в меню File > Import File.

Появляется диалог Select File Driver (выбор файлового драйвера).

2. Выберите в выпадающем списке файловый драйвер и нажмите кнопку ОК.

Выберите драйвер того файла, описание которого Вы создаете. Возникает диалог Open File (открыть файл).

3. Нажмите кнопку с многоточием (...) и выберите файл используя стандартный диалог открытия файла.

4. Нажмите кнопку ОК, чтобы закрыть диалоги.

Редактор словаря создает описание файла и возникает диалог Edit File Properties (редактирование свойств файла).

5. Внесите необходимые изменения в определение нового файла и нажмите кнопку ОК.

В словарь добавляется файл и определения его полей и ключей.

## Свойства нового файла

Характеристики нового файла базы данных определяются в диалоговом окне New File Properties (свойства нового файла). Этот диалог позволяет добавить данных к списку файлов базы данных новый файл и выбрать для него драйвер файла.

Как только файл появится в списке, можно объявлять его поля, ключи, устанавливать его связи и задавать другие свойства данных. Генератор приложений будет использовать данные этого диалогового окна при создании структуры FILE и операций ввода вывода, в соответствии с требованиями приложения.

### General (Главные, общие)

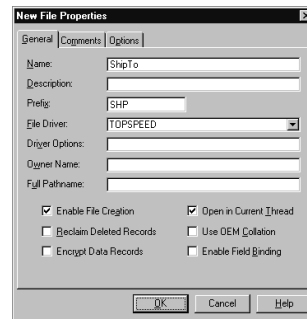
**Name (Имя)** Введите имя файла данных так, как оно должно появляться в тексте программы. Оно является меткой структуры FILE. Задавайте допустимое имя в соответствии с правилами Clarion (см. Справочник языка). Clarion при необходимости автоматически сократит это имя. Имя для соответствующего ему DOS файла (см. ниже Full Pathname) может отличаться от этого имени.

**Совет:** можно задать имена файлов так, чтобы воспользоваться преимуществом путей Novell.

1. Снабдите имя файла префиксом “!Glo:.” Например, !Glo:Customer.
  2. Затем создайте в программе переменную с тем же именем за исключением знака восклицания: Glo:Customer.
  3. Включите следующий текст в точку встраивания CheckOpen Setup:  
Glo:Customer = ‘Server/Vol:\dir\Cust.btr’
- Теперь файл может быть открыт без выделения ему буквы устройства.

**Description (Описание)** Введите строку описания файла. Clarion автоматически показывает это описание в некоторых диалогах, облегчая распознавание содержания файла.

**Prefix (Префикс)** Немедленно после ввода имени файла данных, Clarion автоматически выделит три первые буквы имени и использует их в качестве префикса меток при обращении к файлу. Можно ввести вручную другой префикс





(максимальная длина префикса 14 символов).

Префикс позволяет приложению различать одноименные переменные в различных файловых структурах. Например, поле, называемое Invoice (Накладная) может существовать в двух различных файлах: Orders (Заказы) и Sales (Продажи). Устанавливая уникальные префиксы для Заказов (ORD) и Продаж (SAL), приложение может обращаться к этим полям как ORD:INVOICE и SAL:INVOICE. В Clarion 2.0 префикс не обязателен. Подробности вы найдете в разделе Синтаксис квалификации полей в Справочнике языка.

## File Driver

**(Драйвер файла)** Определите тип файла данных: TopSpeed, Clarion, Btrieve, ASCII и т. д. При использовании Генератора приложений Clarion for Windows автоматически подключает библиотеки используемых драйверов файлов базы данных. В приложении Драйверы файлов обсуждаются относительные достоинства каждого из драйверов.

Следует иметь в виду, что конкретные драйверы файлов могут отличаться друг от друга в части поддержки некоторых атрибутов, которые можно добавить к структуре FILE в этом диалоговом окне.

## Driver Options

**(Необязательные параметры драйвера)** В этом поле можно указать дополнительные параметры для используемого драйвера. Этим передаются дополнительные инструкции драйверу файла. Содержимое поля Options соответствует второму параметру атрибута DRIVER структуры FILE, известному также как “строка драйвера”. Приложение Драйверы файлов содержит дополнительную информацию.

## Owner name

**(Имя владельца)** Необязательное поле, содержащее пароль для доступа к файлу. Возможность парольной защиты данных зависит от используемого драйвера файла. Это добавляет атрибут OWNER (ВЛАДЕЛЕЦ) к оператору FILE. Для большинства файловых систем нужно также включить шифрование (Encrypt).

Шифрование файла означает, что только ваше приложение будет способно читать этот файл. Однако это не означает, что приложение будет автоматически запрашивать пароль у конечного пользователя. Более того, доступ к данным в этом файле будет закрыт для всех других систем просмотра файлов.

При использовании драйвера ODBC укажите в этом поле через запятую имя источника данных, идентификатор пользователя и пароль. Для

получения дополнительной информации по данному вопросу смотрите приложение  
Использование ODBC.

### Full Pathname

#### (Полный путь)

Введите полное имя файла данных. Можно опустить расширение файла - Clarion добавит правильное расширение в зависимости от выбранного драйвера файла. Здесь задается значение атрибута NAME.

Если вы оставите это поле пустым, Clarion автоматически по умолчанию добавит первые восемь букв указанного в поле NAME имени к текущему пути.

При использовании драйвера TopSpeed, можно хранить несколько таблиц в одном физическом файле - разделите имя файла и таблицы с помощью разделителя “\!”, как, например, в TUTORIAL\!ORDERS. Это ссылка на таблицу ORDERS в файле TUTORIAL.TPS. Для получения более полной информации см. также приложение Драйверы файлов.

**Внимание:** Первые восемь символов имени таблицы должны быть уникальными для всех таблиц, хранящихся в одном физическом файле. Пример: используйте \!EmpPayroll и \!EmpMaster, а не \!EmployeePayroll и \!EmployeeMaster.

При использовании драйвера ODBC для определения FILE, подобного Microsoft Access, который также может хранить несколько таблиц в одном файле, поместите имя файла в это поле. Обычно имя физического файла, который содержит таблицу, имеется в списке файла ODBC.INI; диспетчер драйвера ODBC дает информацию об этом драйверу. Для получения более полной информации см. приложение Использование ODBC.

**Совет:** Для определения имени переменной вместо действительного имени файла поместите имя переменной в это поле вслед за восклицательным знаком (!). Например: !FileNameVar.

### Enable File Creation

#### (Разрешение создать файл)

Необязательное указание, что приложение должно создать файл данных, если он не существует во время выполнения. Это добавляет атрибут CREATE к оператору FILE.

### Reclaim Deleted Records

#### (Использование пространства

#### удаленных записей)

Эта возможность зависит от файлового драйвера. Она указывает, что приложение может использовать освободившееся при уничтожении записей пространство файла. Иначе новые записи

добавляются к концу файла. Это добавляет атрибут RECLAIM к оператору FILE.

### **Encrypt Data Records**

**(Шифровать записи данных)** Включает режим шифрования файла. Необходимо также указать Owner Name (см. выше). Это добавляет атрибут ENCRYPT к оператору FILE.

### **Open in Current Thread**

**(Открыть в текущем процессе)** Дополнительно можно указать, что каждый процесс в приложении, использующий этот файл, должен резервировать память для своего собственного отдельного буфера записи. Это важно для использования в MDI приложениях и улучшает управление файлом. По умолчанию шаблоны Clarion автоматически добавляют атрибут THREAD к каждой структуре FILE.

### **Use OEM Collation**

**(Использование OEM упорядочивания)**

Атрибут OEM определяет, что файл FILE содержит написанную не на английском языке строку данных. Эти строки автоматически переводятся из данных набора символов OEM ASCII, содержащихся в файле, в набор символов ANSI для изображения в Windows. Все строковые данные в записи автоматически переводятся из набора символов ANSI в набор символов OEM ASCII прежде, чем запись переносится на диск.

Специфический набор символов OEM ASCII, используемый при перекодировке, поступает с кодовой страницы DOS, загруженной файлом COUNTRY.SYS. Это делает файл данных специфичным для языка, используемого для этой кодовой страницы, и означает, что данные не могут быть использованы на компьютере с иной загруженной кодовой страницей.

### **Enable Field Binding**

**(Разрешать использование поля в динамических выражениях)** Дополнительно можно указать, что все переменные структуры RECORD могут быть использованы в динамических выражениях (с помощью BIND и EVALUATE) во время выполнения приложения. Компилятор отведет место в памяти для хранения полного наименования Prefix.Name каждой переменной. Иначе он будет использовать свою собственную внутреннюю ссылку для каждой переменной. Таким образом, признак BINDABLE увеличивает количество памяти, необходимой для приложения.

## Comments (Комментарии)

**Comments (Комментарии)** Откройте страничку Comments чтобы ввести дополнительное описание файла размером до 1000 символов.

## Options (Необязательные параметры)

### **Do Not Auto-Populate This File**

**(Не генерировать процедуры просмотра и редактирования)**

Отметка в этом поле запрещает мастеру (Wizard) приложения генерировать для этого файла процедуры просмотра и редактирования.

### **]User Options (Дополнительные**

**параметры пользователя)** Текст, набранный в этом поле, доступен для любых шаблонов и утилит, которые обрабатывают этот файл. Синтаксис этого текста определяется соответствующими шаблонами и утилитами.

Чтобы в любой момент изменить свойства файла, выделите имя файла на списке диалога Dictionary, а затем нажмите кнопку Properties или нажмите над именем файла правую кнопку мыши и выберите Properties во всплывающем меню.

## ***Добавление в словарь***

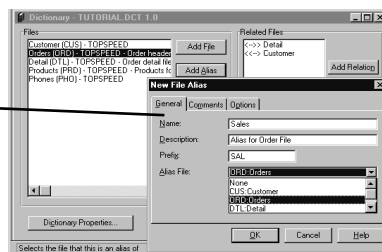
Псевдоним создает дополнительную ссылку для файла без дублирования файла на диске. Псевдоним файла можно добавить, если только файл уже имеется в списке словаря. В диалоговом окне Dictionary нажмите кнопку Add Alias для вызова диалогового окна New File Alias (новый псевдоним файла).

Создание псевдонима, который далее может использоваться как обычный файл.

Введение псевдонимов файлов ценой некоторых системных накладных расходов дает несколько дополнительных преимуществ:

- Псевдонимы позволяют установить множественные отношения между файлами.

Строгая теория реляционных баз данных утверждает, что файл может иметь одновременно только одну реляционную связь с другим файлом. Псевдонимы позволяют “законно” обходить это ограничение.



- Псевдонимы предоставляют второй буфер файла для того же файла.

Этим можно воспользоваться для организации дополнительного механизма просмотра файла, а также “параллельно” работающих вводных форм. Это особенно полезно для MDI-приложений.

- Используется дополнительная память и ресурсы.

Любой драйвер файла, использующий внешние ключевые файлы, требует дополнительного дескриптора файла для каждого псевдонима. Например, файл с тремя внешними ключами и тремя псевдонимами требует шестнадцать дескрипторов: по одному для “первого” файла данных и каждого из трех его ключей, а также дополнительные четыре для каждого псевдонима. При использовании псевдонимов рекомендуется выбирать драйвер файла, который запоминает ключи вместе с данными, такие как TopSpeed или Btrieve.

**Совет:** При использовании псевдонимов файл должен быть открыт в режиме SHARE (разделяемый).

Диалоговое окно New File Alias содержит следующие странички и поля:

#### **General (Главные, общие)**

**Name (Имя)** Введите “имя” файла данных, которое будет использоваться в программе для обращения к файлу. Имя должно быть правильной меткой Clarion.

**Description (Описание)** Введите строку описания для псевдонима файла. Clarion автоматически показывает это описание в некоторых диалогах.

**Prefix (Префикс)** Стандартно Clarion будет использовать в качестве префикса первые три буквы Name. Можно задать собственный префикс размером до 14 символов.

#### **Alias File (Псевдоним файла)**

Выберите файл из выпадающего списка. Это оригинальный файл, на который “ссылается” псевдоним. Выпадающий список содержит только те файлы, которые были ранее определены с использованием команды Add File в диалоговом окне Dictionary.

#### **Comments (Комментарии)**

**Comments (Комментарии)** Откройте страничку Comments чтобы ввести дополнительное

описание файла

размером до 1000 символов.

### Options (Необязательные параметры)

#### **Do Not Auto-Populate This File Alias**

**(Не генерировать процедуры просмотра и редактирования)**

Отметка в этом поле запрещает мастеру (Wizard) приложения генерировать для этого псевдонима файла процедуры просмотра и редактирования.

#### **User Options (Дополнительные параметры пользователя)**

Текст, набранный в этом поле, доступен для любых шаблонов и утилит, которые обрабатывают этот псевдоним файла. Синтаксис этого текста определяется соответствующими шаблонами и утилитами.

Чтобы впоследствии изменить свойства псевдонима файла, выделите его имя в списке диалога Dictionary, а затем или дважды щелкните мышью на нем, или нажмите кнопку Properties.

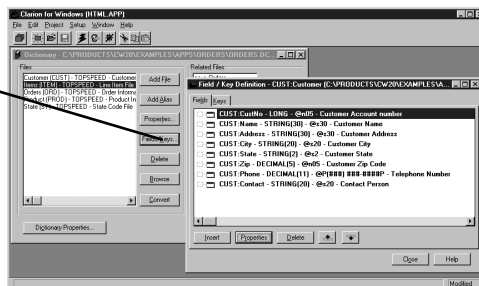
Вы можете отредактировать поля и ключи для псевдонима, нажав кнопку Fields/Keys. Диалоговое окно Field/Key Definition предоставляет список полей и ключей для оригинального файла. Любые выполненные изменения изменят оригинал.

## **Добавление и изменение полей**

После того, как определен файл, можно задавать его поля. Выберите файл в списке файлов диалогового окна Dictionary и нажмите кнопку Fields/Keys или нажмите над именем файла правую кнопку мыши и выберите Fields/Keys во всплывающем меню. Если был выбран псевдоним, Редактор словаря автоматически покажет поля в оригинального файла. Любые изменения впоследствии изменят и оригинальный файл и псевдоним.

Для добавления или редактирования полей нажмите кнопку Fields/Keys


Диалоговое окно Field/Key Definition содержит две странички. Страничка Fields (поля) (левая) показывает список полей, страничка Keys (ключи) (правая) показывает список ключей.



☐ Чтобы добавить новое поле, выберите страничку Fields (поля), затем нажмите кнопку Insert (вставить).

☐ Для изменения существующего поля выберите имя поля и нажмите кнопку Properties (свойства).

☐ Для удаления существующего поля выберите имя поля и нажмите кнопку Delete (удалить).

☐ Чтобы передвинуть выбранное поле в пределах списка полей, нажмите кнопки  Эта процедура перегруппирует метки полей в пределах структуры FILE.

При добавлении или изменении поля используются диалоговые окна New Field Properties или Field Properties, которые появляются при нажатии на кнопки Insert или Properties, соответственно.

## **Определение свойств поля**

---

Диалоговые окна NewField Properties и Field Properties позволяют задать атрибуты и свойства полей.

Эти диалоговые окна идентичны диалогам для определения и редактирования переменных памяти. Все атрибуты языка Clarion, которые можно задать для поля файла, также применимы и к переменным памяти. Однако имеется несколько дополнительных атрибутов, которые могут быть указаны только для глобальных или локальных переменных памяти.

Интегрированная среда разработки Clarion for Windows использует одно и то же диалоговое окно для определения полей и переменных памяти, и поэтому не нужно изучать два отдельных диалога. Те элементы, которые относятся к атрибутам, применимым только к переменным памяти, выключены при определении полей файла.

Редактор словаря позволяет быстро добавлять поля одно за другим. Каждый раз, когда завершается описание очередного поля и закрывается диалоговое окно New Field Properties, вновь открывается пустое диалоговое окно New Field Properties, для определения следующего поля. Нажмите Cancel (отмена), когда пустое диалоговое окно появится после завершения описания последнего поля, чтобы вернуться к диалогу Field/Key Definition.

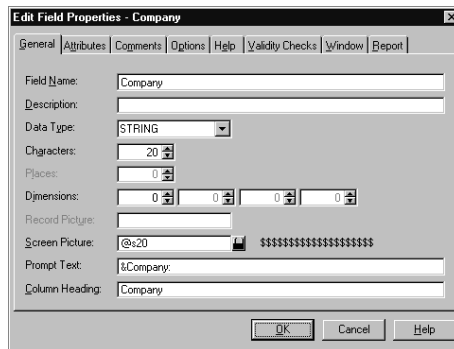
### **General (Главные, общие)**

☐ Чтобы задать имя поля, введите правильную Clarion метку в поле Field Name.

Требования к синтаксису имен полей могут слегка меняться в соответствии с выбранным драйвером файла.

□ Чтобы добавить текстовое описание поля, введите его в поле Description (описание). Описание появится в поле диалогового окна Field Properties. См. далее Comments. Диалоговое окно Field Properties обеспечивает централизованный доступ к широкому спектру атрибутов и свойств поля.

□ Для назначения типа данных поля, выберите его из выпадающего списка Data Type (Тип данных). Clarion поддерживает следующие типы полей, которые определяют, как данные будут храниться в файле на диске и как к ним будет организован доступ в оперативной памяти. Типы полей полностью соответствуют типам переменных Clarion и к ним добавляются мемо-поля и поля с шаблонами. Состав доступных типов полей определяется выбранным драйвером файла.



**STRING** Строка символов фиксированной длины, обычно не более 65,520 символов, что определяется драйвером файла.

**MEMO** Текстовое поле переменной длины, длиной не более 65,536 символов. Чтобы указать, что поле может содержать двоичные данные, нужно отметить поле Binary (двоичный). Дополнительную информацию о том, как каждый драйвер хранит мемо-поля, см. в приложении Драйверы файлов.

**PICTURE** Обеспечивает хранение строки символов “по шаблону”. PICTURE не является особым типом данных, он объявляет поле типа STRING с длиной и форматом PICTURE. Введите в поле шаблона соответствующий шаблон хранения данных. См. в описании языка полный список имеющихся шаблонов с примерами.

**CSTRING** Заканчивающаяся нулем строка символов, максимальной длиной до 65,520 символов. Соответствует строчному типу данных в языке C и полю типа “ZString” в Btrieve.

**PSTRING** Строка символов переменной длины с лидирующим индикатором длины строки, максимальной длиной до 255 символов. Соответствует строчному типу данных в Паскале и полю типа “LString” в Btrieve.

**BYTE** Может содержать беззнаковое целое число от 0 до 255.



<b>SHORT</b>	Может содержать целое число от -32,768 до 32,767.
<b>USHORT</b>	Может содержать целое число от 0 до 65,535.
<b>LONG</b>	Может содержать целое число от -2,147,483,648 до 2,147,483,647.
<b>ULONG</b>	Может содержать целое число от 0 до 4,294,967,295.
<b>DATE</b>	Соответствует типу поля “Дата” в Btrieve.
<b>TIME</b>	Соответствует полю типа “Время” в Btrieve.
<b>SREAL</b>	Может содержать вещественное число между $0+1.175494535e-38$ и $0+3.40282347e+38$ . Соответствует формату Intel 8087 short real.
<b>REAL</b>	Может содержать вещественное число между $0+2.225073858507201e-308$ и $0+1.79769313496231e+308$ . Соответствует формату Intel 8087 long real.
<b>BFLOAT4</b>	Вещественное число между $0+5.87747e-39$ и $0+1.7014118346e+38$ . Соответствует четырехбайтовому формату одинарной точности Microsoft BASIC.
<b>BFLOAT8</b>	Может содержать вещественное число между $0+5.87747e-39$ и $0+1.7014118346e+38$ . Соответствует восьмибайтовому формату двойной точности Microsoft BASIC.
<b>DECIMAL</b>	Может содержать вещественное число между -9 9 9 , 9 9 9 , 9 9 9 , 9 9 9 , 9 9 9 , 9 9 9 , 9 9 9 , 9 9 9 , 9 9 9 и 9,999,999,999,999,999,999,999,999,999,999 в упакованном десятичном формате. Имеет точность в 31 разряд. Должна быть задана по крайней мере одна “позиция” слева от десятичной точки.
<p><b>Совет:</b> Десятичный тип обычно дает наилучшую общую производительность при выполнении математических вычислений. Компилятор оптимизирует выполнение операции путем умножения величин на степень десяти перед обработкой. Это значительно повышает производительность в системах без математического сопроцессора не снижая точности вычислений.</p>	
<b>PDECIMAL</b>	Может содержать вещественное число между -999,999,999,999,999,999,999,999,999,999 и 9,999,999,999,999,999,999,999,999,999,999 в упакованном десятичном формате. Имеет точность в 31 разряд. Должна быть задана по крайней мере одна “позиция” слева от десятичной точки.

## GROUP

Составная структура данных, которая содержит другие поля с различными типами данных. Это соответствует STRUCT в языке C. Введите метку группы в поле Field Name. Каждое последующее диалоговое окно New Field Properties будет определять элементы внутри группы.

**Совет:** Для перемещения полей в группу и из группы используйте кнопки  и .

## BLOB

Может содержать двоичные данные переменной длины, длиной свыше 64К. Подобно полям memo, BLOB поля (Binary Large Objects - большие двоичные объекты) всегда имеют переменную длину, которая не указывается. BLOB поля зависят от драйвера базы данных и в настоящее время поддерживаются только драйвером TopSpeed.

☐ Для создания ссылочной переменной отметьте поле Reference (ссылка). Ссылочная переменная хранит ссылку на другую переменную, которая включает, но не ограничивается адресом переменной в памяти. Это поле доступно только при определении переменных памяти. Дополнительную информацию можно получить в Описании языка.

☐ Для того, чтобы задать длину строкового поля, введите число в поле Characters (символов). Если поле содержит десятичное число, определите позицию десятичной точки в поле Places.

☐ Для того, чтобы объявить массив и определить размерность массива, введите значения размерностей в полях Dimensions (размерности). Можно задать до четырех измерений массива. Общий размер массива не должен превышать 65,520 байт.

☐ Чтобы указать для символьного поля шаблон хранения записи, укажите шаблон форматирования в поле Record Picture (шаблон записи).

☐ Чтобы задать шаблон для вывода на экран, введите шаблон форматирования в поле Screen Picture (экранный шаблон). Когда Генератор приложений будет создавать для этого поля диалоговые элементы окна или отчета, этот шаблон будет использоваться в качестве стандартного для изображения данного поля.

☐ Кнопка с изображением замка рядом с полем Screen Picture обозначает, что шаблон данного поля может быть “заблокирован” от изменений, даже при изменении типа поля. Чтобы запретить экранный шаблон, нажмите на замок.

☐ Для задания стандартной подсказки для поля введите символьную строку в поле Prompt Text (текст подсказки). Генератор приложений при размещении поля в окне помещает этот текст в связываемую с этим полем подсказку PROMPT.

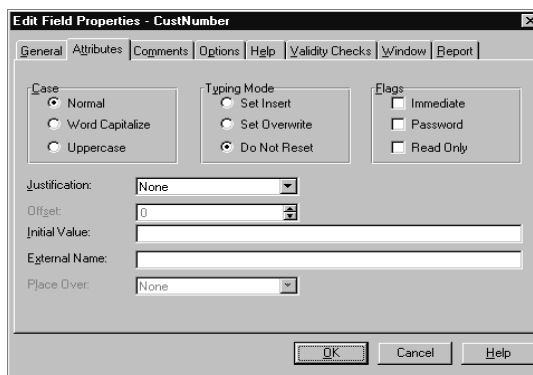
**Совет:** Для того, чтобы указать, что с диалоговым элементом не должно быть связано никаких подсказок, очистите поле **Prompt Text**, а затем нажмите кнопку **Reset Controls** на страничке **Window**.

□ Для задания стандартного заголовка столбца в отчете, для данного поля, введите имя столбца в поле **Column Heading** (заголовок столбца). Генератор приложений использует это имя при создании отчета.

### Attributes (Атрибуты).

□ Чтобы задать атрибут регистра символов для диалоговых элементов, представляющих данное поле, отметьте в группе **Case** один из вариантов - **Normal** (обычный регистр), **Word Capitals** (слова с большой буквы) или **Uppercase** (все в верхнем регистре). Генератор приложений добавит атрибуты **CAP** или **UPR** к описанию диалогового элемента поля ввода.

□ Чтобы задать атрибут режима ввода с клавиатуры для диалоговых элементов, представляющих данное поле, отметьте в группе **Typing Mode** один из вариантов - **Set Insert** (режим вставки), **Set Overwrite** (режим замещения), **Do Not Reset** (не сбрасывать). Генератор приложений добавит атрибуты **INS** или **OVR** к описанию диалогового элемента поля ввода.



□ Чтобы задать режим автоматической немедленной генерации событий в данном поле, отметьте в группе **Flags** (флаги) поле **Immediate** (немедленно). Генератор приложений добавит атрибут **IMM** к описанию диалогового элемента поля ввода.

□ Чтобы задать атрибут сокрытия данных для диалоговых элементов, представляющих данное поле, отметьте в группе **Flags** (флаги) поле **Password** (пароль). Генератор приложений добавит атрибут **PASSWORD** к описанию диалогового элемента

поля ввода. Когда конечный пользователь вводит данные в поле, имеющее атрибут PASSWORD, набираемые символы отображаются на экране в виде звездочек.

□ Чтобы задать атрибут запрещения ввода для диалоговых элементов, представляющих данное поле, отметьте в группе Flags (флаги) поле Read only (только читать). Генератор приложений добавит атрибут READONLY к описанию диалогового элемента поля ввода.

□ Чтобы задать способ выравнивания данных для диалоговых элементов, представляющих данное поле, выберите из выпадающего списка Justification (выравнивание) необходимый тип выравнивания. Генератор приложений добавит к описанию диалогового элемента поля ввода один из атрибутов LEFT (влево), RIGHT (вправо), CENTER(по центру) или DECIMAL (десятичное). Атрибут LEFT выравнивает влево самый левый знак. Атрибут RIGHT выравнивает вправо самый правый знак. Атрибут CENTER центрирует средний знак. Атрибут DECIMAL выравнивает вправо десятичную точку (что скрывает все знаки справа от нее).

□ Чтобы задать величину втяжки для диалоговых элементов, представляющих данное поле, задайте величину втяжки в поле Offset (смещение). Если поле выравнивается влево, втяжка двигает левый знак в обратном направлении - вправо. Если поле выравнивается вправо, втяжка двигает правый знак в обратном направлении - влево. Если используется десятичное выравнивание, втяжка двигает десятичную точку влево, проявляя десятичные знаки, которые в противном случае были бы скрыты. При центрировании положительная втяжка смещает центральный знак вправо, отрицательная - влево. Генератор приложений использует это значение как параметр для атрибута LEFT, RIGHT, CENTER или DECIMAL диалогового элемента. Единица измерения - диалоговые единицы.

□ Чтобы задать стандартное начальное значение поля укажите ее в поле Initial Value (начальное значение. Указание начального значения для поля базы данных приводит к генерации оператора присваивания.

✓ Символьный литерал, заключенный в одинарные кавычки генерирует:  
STATE:Code = 'FL'

✓ Для числовых типов данных число без кавычек генерирует:  
CUS:CreditLine = 1000

✓ Метка глобальной или локальной переменной процедуры генерирует:  
USER:Preference = MyGlobalPreference

✓ Функция генерирует:  
INV:Date = Today ( )

Указание начального значения для глобальных переменных, переменных модуля или локальных переменных процедуры генерирует оператор объявления данных:

- √ Символьный литерал без кавычек генерирует:  
MyString    STRING('LITERAL')
- √ Для числовых типов данных число без кавычек генерирует:  
MyNumber   LONG(100)

**Совет:**    **Функции и переменные можно использовать для инициализации долей базы данных, но нельзя использовать в качестве начальных значений для глобальных переменных, переменных модуля или локальных переменных процедуры. При инициализации долей базы данных литералами должны использоваться кавычки, а для инициализации переменных в памяти кавычки не применяются.**

■ Чтобы задать для поля внешнее имя укажите его в поле External Name (внешнее имя). Это требуется в тех случаях, когда метка поля в программе отличается от имени поля в файле данных; например, можно обращаться к полю через ODBC-подключение к базе данных, содержащей имена полей, которые не являются правильными метками в Clarion. Укажите здесь имя поля, как оно задано в файле данных. Это задает атрибут NAME поля.

■ Чтобы объявить поле как переопределение, выберите из выпадающего списка Place Over (переопределение) имя другого поля. Это позволит текущему полю переопределить область памяти, отведенную для другого поля. Это добавляет к декларации поля атрибут OVER.

■ Выпадающий список Storage Class (класс хранения) доступен только при определении переменных памяти. Для переменных памяти выбирается один из атрибутов:

**DEFAULT** не добавляет никаких атрибутов.

Переменная размещается в стеке, что означает, что для каждого нового экземпляра процедуры выделяется память для переменной.

**EXTERNAL - LOCAL** добавляет атрибут EXTERNAL.

Указывает, что переменная определена во внешней библиотеке и, следовательно, память для переменной в программе не выделяется.

**EXTERNAL - DLL** добавляет атрибуты EXTERNAL и DLL (dll\_mode).

Указывает, что переменная определена во внешней DLL. dll\_mode - это переключатель, указывающий активен ли атрибут DLL или нет. Атрибут DLL необходим для EXTERNAL переменных в 32-разрядных приложениях.

**STATIC** добавляет атрибут **STATIC**.

Переменная размещается в памяти статически, а не в стеке, что обеспечивает “постоянство” выделенной памяти для каждого нового экземпляра процедуры.

**THREAD** добавляет атрибут **THREAD**.

Переменная размещается в памяти статически и отдельно для каждого выполняемого в программе процесса. Таким образом значение переменной зависит от исполняемого процесса.

Эти атрибуты управляют распределением памяти для переменных. Подробная информация в приводится в Описании языка.

### **Comments (Комментарии)**

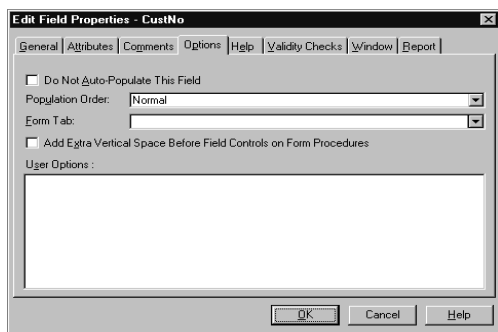
Для добавления комментария или текстового описания выберите страничку **Comments**, позволяющую ввести до 1000 символов.

### **Options (Необязательные параметры)**

☐ Чтобы заставить Мастеров (Wizard) Clarion исключать это поле из полей просмотра, форм и отчетов, отметьте флажок **Do Not Populate This Field** (не размещать это поле)

☐ Чтобы указать, где Мастера (Wizard) Clarion должны размещать это поле в полях просмотра, формах и отчетах, используйте выпадающий список **Population Order** (порядок размещения). **Normal** размещает поля в том порядке, в котором они появляются в Словаре данных. Все поля **First** (первый) размещаются прежде всех **Normal** и **Last** (последний) полей. Все поля **Last** размещаются после всех полей **First** и **Normal**.

☐ Чтобы указать Мастерам (Wizard) Clarion на какой страничке свойств должно находиться данное поле, используйте выпадающий список **Form Tab**.



☐ Чтобы указать Мастерам (Wizard) Clarion на необходимость дополнительного вертикального отступа перед этим полем в процедурах типа форма, отметьте флажок **Add Extra Vertical Space Before Field Controls** (добавить вертикальный отступ перед диалоговым элементом поля).

☐ Чтобы указать дополнительные параметры разработчика, введите в свободной форме текст в поле **User Options** (параметры

пользователя). Текст, набранный в этом поле, доступен любым шаблонным утилитами, которые обрабатывают этот файл. Синтаксис этих параметров задается используемыми утилитами.

### **Help (Помощь)**

□ Чтобы задать идентификатор справки для диалоговых элементов, представляющих данное поле, задайте имя темы помощи в поле HLP. Генератор приложений добавит атрибут HLP к описанию диалогового элемента поля.

□ Чтобы задать текст сообщения, появляющегося в строке состояния для диалоговых элементов, представляющих данное поле, наберите в поле Message (сообщение) текст сообщения. Когда диалоговый элемент, связанный с данным полем получает фокус, в строке состояния появляется это сообщение, при условии, что в приложении имеется строка состояния. Генератор приложений добавляет атрибут MSG к описанию диалогового элемента поля.

□ Для задания всплывающего сообщения для диалоговых элементов, представляющих данное поле, введите текст сообщения в поле Tool Tip. Когда связанный с мышью курсор, неподвижно находится над данным полем, в всплывающем поле непосредственно ниже мыши выводится текст этого сообщения. Генератор приложений добавляет атрибут TIP к описанию диалогового элемента поля.

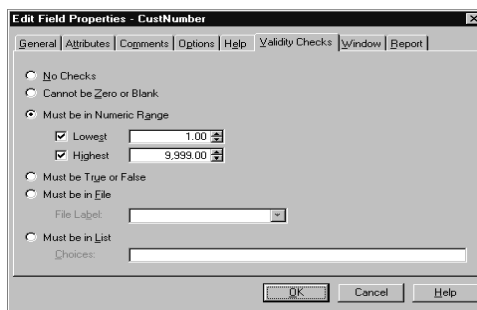
### **Validity Check (Проверка правильности)**

Чтобы задать проверку введенных данных по завершении поля, выберите в диалоговом окне Field Properties страничку Validity Checks (проверка правильности) а затем укажите нужный режим проверки данных, щелкнув мышью над одной из радиокнопок.

Использование странички проверки правильности данных для задания границ диапазона возможных значений поля.

Генератор приложений использует информацию из окна Validity Checks при создании и обслуживании диалоговых элементов. Когда пользователь завершает работу с очередным полем и перемещает фокус к другому элементу управления, или нажимает в диалоге ввода кнопку ОК, приложение выдает предупреждающий звуковой сигнал и, если введены неправильные данные, возвращает фокус обратно на этот диалоговый элемент.

**Совет:** Устанавливая проверку правильности данных, обеспечьте пользователя справочным сообщением в строке состояния. Например, если указано, что числовое



поле должно содержать величину между 1 и 50, поместите сообщение типа “Введите число от 1 до 50” в поле Message (см. выше раздел Help (Помощь)).

Проверка правильности данных ограничивает ввод данных выбранными требованиями:

☐ Для отключения проверки правильности данных выберите No Checks. Это стандартный выбор.

☐ Чтобы потребовать от пользователя обязательного ввода данных в поле без установления каких-либо дополнительных критериев, выберите Cannot be Zero or Blank (не может быть ноль или пусто). Генератор приложений добавит атрибут REQ к описанию диалогового элемента поля.

Поля с атрибутом REQ на страничках диалога ведут себя иначе, чем в одностраничных диалогах. Так как у пользователя есть возможность вовсе не открывать вторую и последующие страницы, то для обеспечения ввода в обязательные поля на закрытых страницах требуются дополнительные меры:

Разместите все обязательные поля на первой странице и задайте атрибут REQ и для страницы и для полей, или

организуите диалог типа Мастер, чтобы пользователь был вынужден открыть все страницы, или

в начале или в конце процедуры добавьте программный текст, который выберет все странички с обязательными полями и задайте атрибут REQ для этих страниц и обязательных полей на них.

☐ Для задания диапазона допустимых значений вводимых данных выберите Must be in Numeric Range (должно быть в числовом диапазоне). Затем введите две величины в поля Lowest (нижняя граница диапазона) и Highest (верхняя граница диапазона).

Вводя значение только нижней или только верхней границы диапазона, можно задать диапазон с открытой границей.

☐ Для ввода логических бинарных значений (yes/no, true/false, on/off) выберите Must be True or False (должно быть истинно или ложь) Must be True or False (Должны быть истинны или ложны)”. Этот вариант лучше всего работает с типом данных BYTE и экраным диалоговым элементом типа флажок (check box).

**Совет:** Если для числового поля указывается требование Must be True or False, то стандартно используется диалоговый элемент типа флажок (check box).



☐ Для проверки присутствия введенного значения в связанном файле выберите **Must be in File** (должно содержаться в файле). Допустимые варианты ввода появятся в выпадающем списке **File Label** (метка файла) только в том случае, если предварительно была задана связь с одним или несколькими другими файлами. См. далее в этой главе раздел **Добавление и модификация связей**.

☐ Для проверки наличия введенного значения в списке выберите **Must be in List** (должно быть в списке). Затем задайте список возможных значений в поле **Choices** (варианты) в формате “Вариант1|Вариант 2|Вариант 3”. Разделите варианты вертикальной чертой (|) (на клавиатуре обычно SHIFT+\\).



**Совет:** Если вы собираетесь предоставить конечному пользователю выбирать из ограниченного количества вариантов в поле списка, выпадающего списка или радиокнопками, укажите список этих вариантов, отделяя один вариант от другого вертикальной чертой.

### “Окно” (Window)

Страничка **Window** (Окно) в диалоговом окне **Field Properties** (свойства поля) используется для указания того, как данное конкретное поле будет представлено пользователю в оконной среде. Помните, что это назначение является стандартным методом представления. Определяя его здесь, в Словаре данных, задается стандартный метод представления поля, который будет использоваться всякий раз, когда поле размещается в окне **Clarion**. Это означает, что метод представления нужно сконструировать лишь один раз, независимо от того, сколько раз приложение будет использовать данное поле, и независимо от того, как много приложений используют данный словарь; однако сохраняется возможность в каждом конкретном случае модифицировать это стандартное представление.

☐ Для изменения стандартных параметров вводных диалоговых элементов и их полей подсказок

**Совет:** Задавая характеристики диалоговых элементов в этот момент, можно сэкономить время впоследствии. Каждое приложение, которое использует этот словарь и каждая процедура в приложении будут автоматически форматировать элементы управления как это задано в Словаре данных. Если не задавать формат диалоговых элементов здесь, в словаре данных, и если диалоговый элемент требует специального форматирования, впоследствии, при проектировании приложения, это придется делать для каждой процедуры в отдельности.

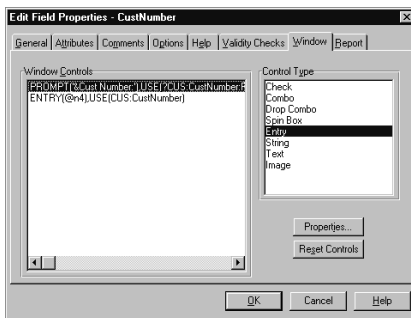
1. Выберите в диалоговом окне **Edit Field Properties** (редактирование свойств поля) страничку **Window** (Окно).

2. Выберите в списке **Control Type** тип используемого диалогового элемента. В списке

Window Controls (диалоговые элементы окна) немедленно будет приведен в соответствие с выбранным типом.

3. Выберите в списке Window Controls диалоговый элемент (PROMPT, ENTRY, TEXT, SPIN и т.д.), а затем нажмите кнопку Properties (свойства). Откроется диалог свойств для выбранного диалогового элемента. Используя этот диалог, задайте положение, размер, цвет, шрифт, текст, режим и т. д. диалогового элемента. См. раздел Диалоговые элементы и их свойства.

### Предварительное форматирование диалогового элемента ENTRY



□ Для восстановления стандартных характеристик диалоговых элементов нажмите кнопку Reset Controls (сброс диалоговых элементов).

**Совет:** Для того, чтобы указать, что с диалоговым элементом не должно быть связано никаких подсказок, очистите поле Prompt Text на страничке General, а затем нажмите кнопку Reset Controls на страничке Window.

### “Отчет” (Report)

Эта страничка действует точно также, как страничка Window. Задавая здесь свойства диалоговых элементов отчета, мы определяем какие диалоговые элементы будут использованы при проектировании отчетов Clarion.

## **Добавление и изменение ключей**

Ключи и индексы файлов в базы данных создаются и редактируются в диалоговом окне Field/Keys Definition (определение поля/ключа) Словарь данных будет генерировать корректное объявление структуры FILE, основываясь на сведениях, получаемых из заполненных диалоговых полей.

Ключи и индексы определяют порядок сортировки в каждом из файлов. В зависимости

от используемого драйвера файла, ключ может находиться или внутри файла или существовать как внешний файл.

Ключи, в отличие от индексов, автоматически обновляются при добавлении, изменении или уничтожении записей. Приложение Драйверы файлов содержит дополнительную информацию о том, как каждый драйвер файла поддерживает ключи или индексы.

Индексы, как правило, существуют по отношению к индексируемым ими файлам как внешние файлы. Помните, что для каждого внешнего ключа или индексного файла в DOS требуется отдельный блок управления файлом. Индексные файлы не обновляются автоматически. Индексы обновляются оператором BUILD.

Динамический индекс позволяет объявить индексный файл без указания в словаре данных составляющих его полей. Приложение во время исполнения программы должно задать поле (поля) ключа, в качестве второго параметра оператора BUILD. Приложение может заново построить тот же индексный файл позднее, задав другие поля в качестве компонент ключа.

Основные шаги создания ключа следующие:

1. Выберите файл из списка в группе Files диалогового окна Dictionary (словарь) и нажмите кнопку Field/Keys (поле/ключи).
2. В диалоговом окне Field/Keys Definition (определение поля/ключа) выберите страничку Keys (ключи) чтобы перенести фокус на список Keys.
3. Выделите ключ (если таковой существует), затем нажмите кнопку Insert (вставить). Появится диалоговое окно New Key Properties (свойства нового ключа).
4. Укажите правильную метку Clarion в поле Key Name (имя ключа).
5. Дополнительно можно задать Description (описание). Это описание проявится затем в различных диалоговых полях, в том числе в диалоговом окне File Definition (определение файла).
6. Выберите страничку Attributes (атрибуты) и отметьте все флажки, которые нужны для данного ключа.
7. Если необходимо, укажите правильное имя файла в DOS в поле External Name (внешнее имя) если система файла нуждается в этом.  
Clarion автоматически добавит подходящее расширение.
8. Выберите закладку Fields (поля), затем нажмите кнопку Insert (вставить).

Появится диалоговое окно Insert Key Component (вставить компоненту ключа).

9. Выберите поле двойным щелчком мыши в списке, это перенесет его имя на страничку Fields (поля), которая показывает поля, которые будут компонентами нового ключа.

10. Нажмите кнопку ОК, чтобы закрыть диалоговое окно New Key Properties.

Снова появится диалоговое окно New Key Properties для создания еще одного ключа.

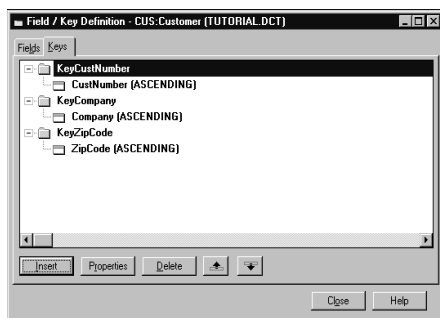
11. Повторите шаги с 4 по 10 для создания остальных ключей этого файла.

12. Закончив добавление ключей, нажмите Cancel (отменить) для того, чтобы закрыть диалоговое окно New Key Properties и вернуться в диалоговое окно Field/Keys Definition.

По окончании этого процесса все ключи появятся друг за другом на древовидной диаграмме на страничке Keys (ключи) вместе с выстроенными по порядку полями, являющимися их компонентами.

Для изменения ключа выберите ключ и нажмите кнопку Properties (свойства) в диалоговом окне Field/Key Definition (определение поля/ключа). Появится диалоговое окно Edit Key Properties (редактирование свойств ключа). Если была выбрана компонента ключа, то наверху будет находиться страничка Fields (поля). Если был выбран ключ, то наверху будет находиться страничка General (общие). В следующем разделе Установка свойств ключа описаны возможности этого диалогового окна.

Диалог Field/Keys Defenition показывает поля и ключи для одного файла.



## Задание свойств ключа

Следующие странички и поля на них появляются в диалоговых окнах New Key Properties и Edit Key Properties. здесь задаются атрибуты ключа.

### General (Главные, общие)

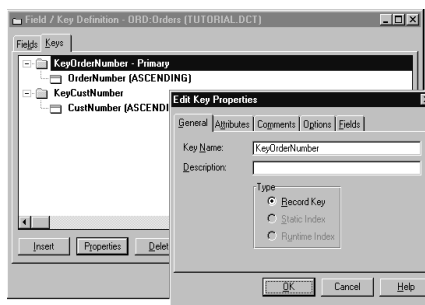
**Key Name (Имя ключа)** Чтобы задать для ключа метку Clarion, введите в этом поле правильную метку Clarion.

**Совет:** Помните, что вы не можете дать ключу то же имя, что и одному из полей в пределах структуры RECORD. Одно общепринятое правило - используйте имя поля и слово “key”, как например в LastNameKey.

**Description (Описание)** Чтобы поместить в словарь текстовое описание ключа данных, наберите его в этом поле. Описание появится в таких диалогах, как диалог File Definition (определение файла). Если вы собираетесь использовать в приложении много ключей, рекомендуем заполнить это поле.

**Type (Тип)** Чтобы определить ключ записи, статический индекс или динамический индекс, выберите одну из радиокнопок в группе Type (тип). Имейте в виду, что ключи записи автоматически обновляются при всяком добавлении изменении или удалении записей. Индексы не обновляются автоматически, и требуют для этого оператора BUILD.

Параметры Static Index (статический индекс) и RunTime Index (динамический индекс) недоступны, когда отмечено поле Require Unique Value (требуется уникальное значение) на страничке Attributes (атрибуты), так как индексы всегда допускают дублирование.



### Attributes (Атрибуты)

**External Name (Внешнее имя)** Чтобы, при необходимости, указать имя DOS файла для внешнего ключа, введите допустимое имя DOS файла в этом поле. Clarion автоматически добавит надлежащее расширение. Генератор приложений добавляет атрибут NAME к оператору KEY. Некоторые файловые системы требуют внешнего имени. Дополнительную информацию вы можете найти в приложении Драйверы базы данных.

**Require Unique Value**

(Требуется уникальное значение) Чтобы не допустить появления множества записей с повторяющимися значениями в их ключах, отметьте это поле. Этот параметр

актуален только для ключей и недоступна для индексов. Генератор приложений опускает атрибут DUP в оператор KEY. (В оригинале речь ошибочно идет о несуществующем атрибуте NODUP. - Ред.)

**Primary Key (Первичный ключ)** Для объявления текущего ключа в качестве первичного ключа, отметьте это поле. Генератор приложений добавит к описанию ключа атрибут PRIMARY (первичный). Это может потребоваться для некоторых драйверов файлов. Более полную информацию вы можете получить в приложении Драйверы базы данных.

**Auto Number (Автонумерация)** Чтобы потребовать от Генератора приложений создания кода для автонумерации записей, отметьте это поле.

Определение первичного ключа



### Case Sensitive

**(Чувствительность к регистру клавиатуры)** Отметьте это поле, чтобы записи были отсортированы с учетом регистра символов. При создании или обновлении ключа все заглавные буквы будут предшествовать строчным аналогично их позиции в таблице ASCII. Генератор приложений опускает атрибут NOCASE в операторе KEY.

### Exclude Empty Keys

**(Исключить пустые ключи)** Для исключения из ключевого файла записей с нулевыми или пустыми значениями в компонентах данного ключа, отметьте это поле. Генератор приложений добавит атрибут OPT к оператору KEY.

**Внимание:** Первичный ключ должен быть уникальным и не должен содержать пустых значений. Выбор параметра Primary Key имеет тот же самый эффект, что и отметка полей Require Unique Values и Exclude Empty Keys одновременно.

### Comments (Комментарии)

Выберите страничку Comments (комментарии) для ввода описания ключа размером до

1000 символов.

### Options (Необязательные параметры)

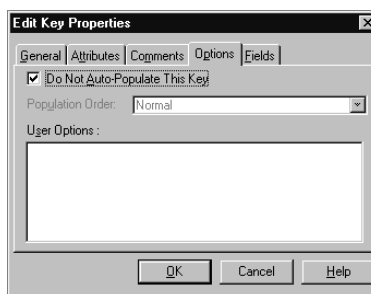
#### **Do Not Auto-Populate This File**

**(Не генерировать процедуры просмотра и процедуры модифицирования)** Отметьте это поле, чтобы дать указание Мастеру (Wizard) Clarion не генерировать окна просмотра и отчеты на основе данного ключа.

#### **Population Order**

##### **(Порядок заполнения)**

Чтобы установить, на каком месте Мастер (Wizard) Clarion поместит данный ключ на страничках просмотра и в меню отчетов, воспользуйтесь выпадающим списком Population Order (порядок заполнения). Позиция Normal размещает ключи в том порядке, как они появляются в Словаре данных. Все First (первые) ключи помещаются перед ключами Normal и Last. Все ключи Last помещаются после полей First и Normal. (В оригинале ошибочно речь идет о полях. - Ред.)



**User Options (параметры пользователя)** Введите здесь дополнительные параметры для шаблонных утилит, которые обрабатывают этот ключ. Текст, набранный в этом поле, становится доступным для любых шаблонов утилит, обрабатывающих этот ключ. Синтаксис определяется шаблонами утилит.

### **Компоненты ключа**

На страничке Fields (поля) диалогового окна Edit Key Properties (редактирование свойств ключа) задайте компоненты ключа (поле или поля сортировки). В ключ можно включить более одного поля. При определении ключа из нескольких полей можно комбинировать различные типы данных. Можно указать различный порядок по разным полям: одно поле - по возрастанию, другое - по убыванию, однако возможность смешанного порядка сортировки зависит от типа драйвера файла. Более полная информация имеется в разделе Драйверы базы данных.

## Key Fields List

### (Список полей ключа)

Чтобы добавить к ключу поля или компоненты, нажмите кнопку Insert (вставить). Появившийся список Insert Key Components (вставить компоненты ключа) покажет доступные поля. Чтобы поместить имя поля в Key Fields List, щелкните над ним дважды мышью.

## Sort Order

### (Порядок сортировки)

Чтобы указать последовательность сортировки компонент вашего ключа, выберите либо радиокнопку Ascending (по возрастанию), либо Descending (по убыванию).

## Component Order

### (Порядок компонент)

Можно изменить порядок компонент составного ключа. Чтобы передвинуть компоненту вверх по списку, выберите ее в Key Fields List (список полей ключа), затем нажмите кнопку  $\uparrow$  (или нажмите CTRL+  $\uparrow$  ). Чтобы передвинуть компоненту вниз по списку, выберите ее в списке Key Fields List , затем нажмите кнопку  $\downarrow$  (или нажмите CTRL+  $\downarrow$  ).

## Добавление и изменение связей

Связи между файлами задаются в диалоговом окне New (или Edit) Relationship Properties (свойства нового (или редактирование свойств) соотношения). Связи текущего выбранного файла появятся в списке Related Files (связанные файлы) в правой части диалогового окна Dictionary (словарь).

Основные шаги определения отношения:

1. Выберите файл в списке Files в левой части диалогового окна Dictionary.

2. Нажмите кнопку Add Relation (добавить отношение).

Появится диалоговое окно New Relationship Properties (свойства нового соотношения) New Relationship Properties (Свойства новых отношений) .

3. Выберите тип соотношения из выпадающего списка Type: 1: Many (один к многим) 1\:\: Many (Один к многим) ” , или Many:1 (многие к одному) Many\:\:1 (Многие к одному) ” . Соотношение 1: Many определяет связь, когда одна запись файла связана со многими записями в другом файле. Например, файл покупателей содержит по одной записи для каждого из покупателей, в то время как файл заказов будет содержать множество записей для каждого покупателя.

В этом примере не важно, с какого файла начать. Если сначала выбран файл покупателей, то связь будет типа 1: Many, а если сначала выбран файл заказов, то нужно указать связь типа Many:1.

Метка, расположенной ниже этого поля группы, при этом изменится на Child



(дочерний)Child (дочернее)” , или на Parent (родительский), соответственно.

4. Выберите из выпадающего списка Related File (связанный файл).

Записи в этих двух файлах имеют нечто общее, что их связывает: это номер покупателя. Например номер покупателя может быть 629, и заказы этого покупателя тоже должны иметь номер покупателя 629. Таким образом номер покупателя является ключом для этой связи файлов.

5. Выберите для первого файла из выпадающего списка в правой верхней части диалогового окна ключ Primary Key (первичный) или Foreign Key (внешний).

Clarion автоматически изменят метку этого списка (либо Primary Key, либо Foreign Key) в соответствии с типом выбранного отношения.

Primary Key всегда является однозначным (уникальным) ключом файла. В нашем примере в файле покупателей должен быть только один покупатель с номером 629, иначе сведения о покупателях с номерами 629 перепутаются. Поэтому номер покупателя является первичным ключом в файле покупателей.

Foreign Key не должен быть уникальным, но его значения должны соответствовать значениям первичного ключа в другом файле. Когда покупатель номер 629 сделает несколько заказов, в файле заказов появится несколько записей для покупателя с номером 629. Поэтому ключ по номеру покупателя в файле заказов тоже является ключом в этом соотношении, но ключом внешним - Foreign Key.

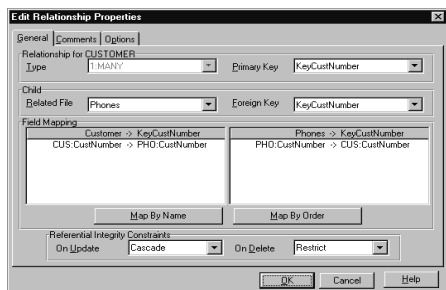
6. Если необходимо, то выберите ключ Primary Key или Foreign Key для связанного файла из выпадающего списка непосредственно под предыдущим списком.

7. Нажмите кнопку Map by Name (соответствие по имени) для задания связи между двумя ключами через одноименные поля.

Списки Field Mapping (соответствие полей)Field Mapping (Схема полей)” показывают связи, установленные между двумя файлами.

Этот шаг по заданию соответствия необходим потому, что ключи файлов не всегда бывают определены одинаковым образом. Например ключ Key\_CustNumber в файле покупателей может состоять из полей CustNumber, в то время, как в файле заказов ключ может содержать поля CustNumber и OrderDate (дата заказа) (пример изменен - ред.). Задание соответствия гарантирует, что многокомпонентные ключи будут правильно обрабатываться.

Задание отношения один к многим, включая схему соответствия полей и требования ссылочной целостности.



8. Дополнительно в группе Referential Integrity Constraints (требования ссылочной целостности) можно указать требования по обеспечению целостности данных путем выбора их в выпадающих списках On Update (при обновлении) и On Delete (при удалении).

См. раздел об обеспечении ссылочной целостности.

9. Нажмите кнопку ОК.

Вновь определенное соотношение появится в диалоговом окне Dictionary.

## Установка требований по обеспечению ссылочной целостности

Устанавливая требования по обеспечению ссылочной целостности данных в Словаре данных, можно указать Генератору приложений, как должна быть устроена программа в части выполнения обновлений и удалений при работе со связанными файлами данных.

Требование ссылочной целостности данных состоит в том, что внешний ключ обязательно должен иметь соответствующее значение в первичном ключе. Это создает некоторые потенциальные проблемы, когда пользователь желает изменить или удалить запись первичного ключа.

Диалоговое окно New Relationship Properties (характеристики нового соотношения) позволяет указать, как исполняемая программа должна обрабатывать ситуации, когда обновляется или удаляется одна из нескольких связанных записей.

**No Action (Никаких действий)** Указывает Генератору приложений, что для поддержания ссылочной целостности не требуется никакой программной поддержки.

**Restrict (Ограничить)** Указывает Генератору приложений, что он не должен позволять конечному пользователю удалять или изменять записи, если их данное используется во внешнем ключе. Например, если пользователь попытается изменить значение первичного ключа, генерируемая программа проверяет наличие во внешнем ключе этого значения. Если будет установлено соответствие, то внесение изменений не разрешается.

**Cascade (Привести в соответствие)** Указывает Генератору приложений, что он должен обновить или удалить запись внешнего ключа. Например, если пользователь изменяет величину первичного ключа, то генерируемая программа изменяет все

совпадающие значения во внешнем ключе. Если пользователь удаляет родительскую запись, то программа удаляет все дочерние записи.

**Совет: Шаблоны обеспечивают поддержку целостности данных на все уровни связей, как это определено в Словаре данных.**

**Clear (Очистить)** Указывает Генератору приложений, что он должен очистить поля во внешнем ключе.

## **Управление вашим словарем**

Редактор словаря предоставляет некоторые возможности, помогающие в управлении словарями данных.

- Вы можете копировать и вставлять определения файлов и их полей из одного файла словаря в другой.
- Редактор словаря предлагает встроенный диспетчер версий, который позволяет документировать любые изменения, когда вносятся значительные изменения. Это также позволяет возвращаться к старым версиям, так что можно “отменить” все изменения.
- Редактор словаря имеет обычные настроечные параметры, которые, например, позволяют определить стандартный драйвер файла.

## **Копирование и вставка**

---

Вы можете использовать команды Copy (копировать) и Paste (вставить) для копирования определений файлов и их полей из одного словаря в другой. Чтобы сделать это:

1. Откройте файл словаря.
2. Выберите файл из списка Files в диалоговом окне Dictionary.
3. Выполните Edit > Copy или нажмите комбинацию клавиш CTRL + C.
4. Откройте второй файл словаря.
5. Выполните Edit > Paste или нажмите комбинацию клавиш CTRL + V.

После вставки появляется диалоговое окно New File Properties (свойства нового файла). Можно изменить определение файла требуемым образом. После нажатия кнопки ОК, файл появится в диалоговом окне второго словаря.

Копирование и вставка полей из одного файла в другой работает точно так же, за

исключением того, что вместо диалоговых окон Dictionary нужно открыть диалоговые окна Field/Keys Definition. Ограничением является то, что целевой файл должен поддерживать тип оригинального поля, иначе команда Paste не будет работать.

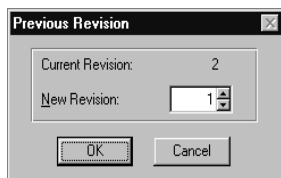
**Совет:** Можно скопировать элементы словаря данных Data Dictionary, такие как файл или поле, в буфер промежуточного хранения, а затем, используя текстовый редактор, вставить соответствующие операторы на языке Clarion.

## Версии словаря

Редактор словаря автоматически помещает в файл словаря внутренний номер версии. Новый словарь автоматически начинается с версии 1.0. Номер версии/реvisions можно увидеть в строке заголовка диалогового окна Dictionary. Диалоговое окно Dictionary Properties (свойства словаря) показывает также дату и время первоначального создания, а также дату и время последнего изменения.

Номер версии нужно повышать вручную, каждый раз, когда в словарь вносятся значительные изменения, например, начиная работу с версией #2 приложения в диалоговом окне Dictionary, выполните Version > Checkpoint. Номер revisions (r.#) добавляется к строке заголовка. Номер revisions увеличивается с установкой каждой новой “точки фиксации”.

Чтобы вернуться к предыдущей версии, выполните Version > Revert. Укажите номер revisions для отката путем выбора его в прокручиваемом списке диалога Previous Revision.



## Средства настройки Редактора словаря

В диалоговом окне Dictionary Options (параметры словаря) можно изменить некоторые стандартные параметры Редактора. Для обращения к этому окну выполните Setup > Dictionary Options.

### File Options (Параметры файлов)

☐ Для того чтобы задать для новых файлов стандартный драйвер базы данных, выберите его из выпадающего списка Default Driver.

☐ Чтобы список файлов Files диалогового окна Dictionary выводился в алфавитном порядке, отметьте пункт Sort dictionary files alphabetically.



☐ Чтобы сделать атрибут THREAD (выделение отдельного буфера RECORD для каждого процесса) стандартным для новых файлов, отметьте пункт Default THREAD Attribute.

☐ Чтобы видеть описания файла в списке Files диалогового окна Dictionary, отметьте пункт Display File Description.

☐ Чтобы видеть драйверы файла в списке Files диалогового окна Dictionary, отметьте пункт Display File Driver.

### **Field Options (Параметры поля)**

☐ Для того чтобы задать, что описания полей, которые вводятся при определении файла, будут также служить текстом для атрибута Message (сообщение в строке состояния), отметьте поле Assign Description to Message (задать сообщение используя описание).

☐ Чтобы видеть описание поля в диалоговом окне Field/Key Definition (определение поля/ключа), отметьте поле Display Field Description (вывод описания поля).



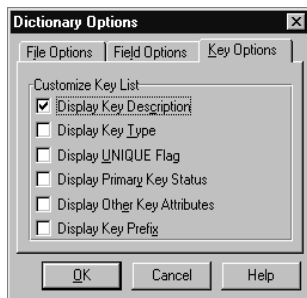
☐ Чтобы видеть тип поля в диалоговом окне Field/Key Definition (определение поля/ключа), отметьте поле Display Field Type (вывод типа поля).

☐ Чтобы видеть шаблон вывода поля в диалоговом окне Field/Key Definition (определение поля/ключа), отметьте поле Display Field Picture (вывод шаблона поля).

### **Key Options (Параметры ключей)**

☐ Чтобы видеть описание ключа в диалоговом окне Field/Key Definition (определение поля/ключа), отметьте поле Display Key Description (вывод описания ключа).

☐ Чтобы видеть тип ключа в диалоговом окне Field/Key Definition (определение поля/ключа), отметьте поле Display Key Type (вывод типа ключа).



☐ Чтобы видеть признак уникальности в диалоговом окне Field/Key Definition (определение поля/ключа), отметьте поле Display UNIQUE Flag (вывод атрибута уникальности).

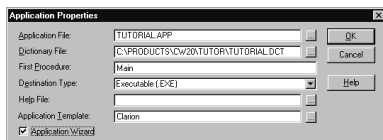
☐ Чтобы видеть признак первичного ключа в диалоговом окне Field/Key Definition (определение поля/ключа), отметьте поле Display Primary Key Status (вывод атрибута первичного ключа).

☐ Чтобы видеть остальные атрибуты ключа в диалоговом окне Field/Key Definition (определение поля/ключа), отметьте поле Display Other Key Attributes (вывод остальных атрибутов ключа).

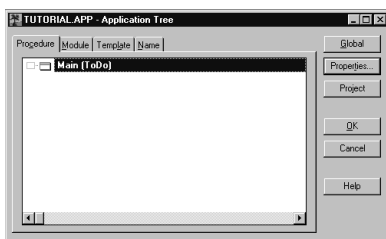
## Глава 5 Использование Генератора Приложений



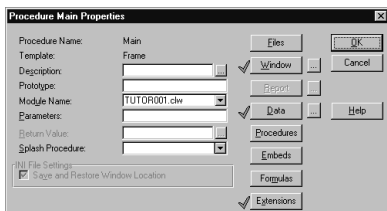
Генератор Приложений создает исходный текст, на основе выбранных из Регистра шаблонов и соответствующим образом настроенных процедурных шаблонов. По мере необходимости Генератор Приложений автоматически вызывает другие компоненты среды.



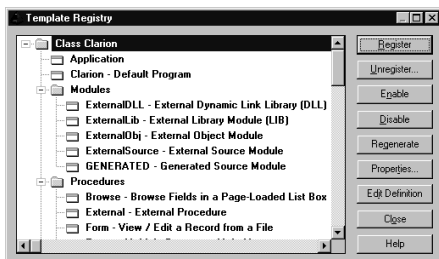
Разработка начинается с определения .APP файла, в котором будут храниться процедуры разрабатываемого приложения и их параметры.



Дерево приложения показывает список процедур. Те процедуры, которые еще не наполнены конкретным содержанием, помечаются типом “To Do”.



Диалоговое окно “Procedure Properties” (спецификации процедур) действует подобно координирующему центру, через который вызываются различные инструменты интегрированной среды разработки для специализации процедур. Это окно может содержать также некоторые дополнительные поля для различных параметров.



Шаблоны можно настраивать по своему желанию, добавлять или удалять. Используя Регистр шаблонов можно регенерировать .TPL файлы.

При использовании Генератора приложений, создаются процедуры, для решения основных задач разрабатываемого приложения, описывается как решаются эти задачи и какие окна, диалоги и отчеты получает при этом конечный пользователь. Управляемый шаблонами процедур из Регистра шаблонов, Словарем данных и предоставленной разработчиком информацией, Генератор приложений генерирует исходные тексты приложения, создавая операторы языка Clarion, необходимые для решения задач наиболее эффективным способом.

В этой главе описано, как заполнить все диалоги Генератора приложений, чтобы он мог начать генерацию исходного текста:

- Как установить режимы генератора приложений.
- Как начать разработку нового приложения и создать .APP файл.
- Как добавлять в приложение глобальные и локальные переменные.
- Как добавлять процедуры в “Application Tree”.
- Как добавлять в процедуры вставки исходного текста для получения требуемой функциональности.
- Как сопровождать шаблоны и файл REGISTRY.TRE.



## Установка режимов Генератора приложений.

Диалог Application Options (режимы Генератора приложений) позволяет указать стандартные свойства как для каждого вновь создаваемого приложения, так и для текущего приложения. Для доступа к диалогу нужно выполнить команды Setup ? Application Options. Диалог содержит четыре страницы: Application (приложение), Registry (регистр), Generation (генерация) и Synchronization (синхронизация).

### Application (приложение)

Эта страничка позволяет стандартизовать разнообразные свойства приложения.

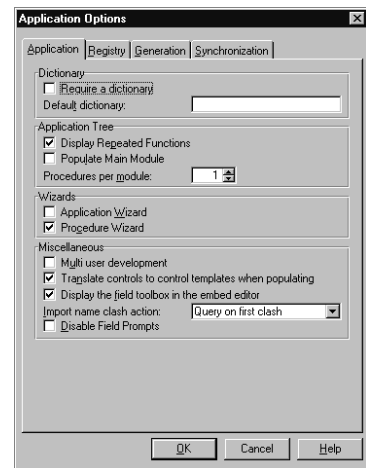
Require a dictionary

(словарь обязателен) Каждое новое приложение должно использовать Словарь данных.

Default dictionary

(стандартный словарь)

Задаёт спецификацию файла стандартного словаря данных, которая появляется в диалоге Application Properties каждый раз при создании нового приложения. Перед закрытием диалога можно выбрать другой словарь.



Совет: Один единственный словарь может использоваться для создания множества приложений.

Display repeated functions

(повторный вывод процедур)

Указывает, должен ли Генератор приложений повторно выводить процедуры, на которые имеются ссылки более, чем в одном месте дерева приложения, или просто пометить их как “Expanded above” (раскрыто выше).

Populate main module

(размещение в главном модуле)

Генератор приложений будет размещать процедуры в главном модуле. При выключенном режиме Генератор приложений размещает в главном модуле только глобальные данные и текст главного модуля. Все остальные процедуры размещаются в других файлах.

Procedures per module  
(процедур на модуль)

Указывает количество процедур, которые генератор приложений записывает в каждый исходный модуль. Это влияет на время компиляции при включенном режиме Conditional Generation (условная генерация). Например, если установлена одна процедура на модуль, то при каждой последующей компиляции будут переделываться только те процедуры, которые изменились со времени последней компиляции. Обратная сторона этого в том, что требуется больше дискового пространства. Обычно чем меньше число, тем быстрее.

Application Wizard  
(Мастер приложения)

Задаёт стандартное значение для переключателя Application Wizard в диалоге Application Properties при создании нового приложения. Мастер приложения строит законченное приложение на основе словаря данных.

Procedure Wizard  
(Мастер процедур)

Задаёт стандартное значение для переключателя Procedure Wizard в диалоге Select Procedure Type при создании новой процедуры. Атрибут в этом поле активизирует Мастера, который поможет создать процедуру просмотра, форму или отчет.

Multi user development

(коллективная разработка) Открывает Словарь данных и Регистр шаблонов в режиме только для чтения, что позволяет многим разработчикам работать с одним и тем же словарем данных. См. также приложение Стратегия разработки и развертывания.

Translate controls to control templates when populating

(подставлять при размещении шаблоны диалоговых элементов) Включите этот режим, чтобы формater окон предлагал список шаблонов диалоговых элементов при размещении диалогового элемента. Этот режим рекомендуется для начинающих работать с Clarion for Windows.

Display the field toolbox in the embrd editor

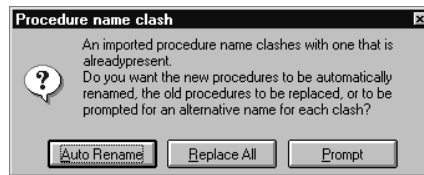
(показать в редакторе вставок инструментальное окно со списком полей) Включите этот режим, чтобы показать в редакторе вставок инструментальное окно со списком полей. Это позволяет выбирать имена переменных и полей в списке инструментального окна.

### Import name clash action

(действия при коллизии имен во время импорта) Указывает, как Генератор приложений при импорте разрешает коллизии имен из файла приложения и имен уже имеющихся процедур. Можно выбрать одно из:

### Query on first clash

(Запрос при первом конфликте) Когда возникает первый конфликт, Генератор приложений запрашивает, какие действия следует предпринять в этом и всех последующих случаях.

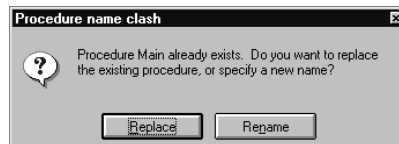


Выберите Auto Rename (автоматическое переименование), Replace All или Prompt (запрос).

Auto Rename переименовывает все импортируемые процедуры, которые вызывают конфликт имен, добавляя к именам последовательные номера.

Replace All заменяет все существующие процедуры на импортируемые процедуры с такими же именами.

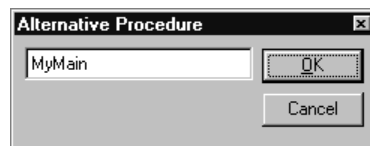
Prompt выводит запрос о необходимых действиях для каждого конфликта.



Выберите между Replace (заменить) и (переименовать).

Replace заменяет существующую процедуру на импортируемую процедуру с таким же именем.

Rename запрашивает имя для переименования импортируемой процедуры.



Ask for alternative (запросить варианты)

Для всех процедур с одинаковыми именами Генератор приложений будет запрашивать имена для переименования импортируемых процедур.

Auto Rename (автоматическое переименование)

Для всех процедур с одинаковыми именами Генератор приложений переименовывает все импортируемые процедуры, которые вызывают конфликт имен, добавляя к именам последовательные номера.

Replace previous

(заменить предыдущие)

Для всех процедур с одинаковыми именами Генератор приложений заменяет все существующие процедуры на импортируемые процедуры.

Disable Field Prompts

(отключить подсказки к полям)

Указывает, что подсказки полей, которые генерируются шаблонами не должны появляться на страничках Actions. Это не отменяет генерацию подсказок диалоговыми шаблонами.

## **Registry (Регистр)**

На этой страничке задаются режимы, имеющие отношение к сопровождению и использованию Регистра шаблонов (Template Registry) - REGISTRY.TRF. См. далее в этой главе Установка режимов Регистра шаблонов.

## **Generation (генерация)**

На этой страничке задаются режимы, имеющие отношение к генерации исходных текстов.

Conditional Generation

(условная генерация)

Указывает, что должны перекомпилироваться только те модули, которые изменились после последней компиляции.

Debug generation

(отладочная генерация)

Включает и выключает запись отладочного протокола Генератора приложений и задает имя файла протокола - Debug Filename. В случае фатальной ошибки Генератора приложений этот протокол позволит службе технической поддержки TopSpeed установить причины возникновения ошибки.

Generation Message

Указывает сколько и каких сообщений будет выводиться во время генерации исходных текстов. Можно выбрать No Messages (нет сообщений), Module Names only (только имена модулей)

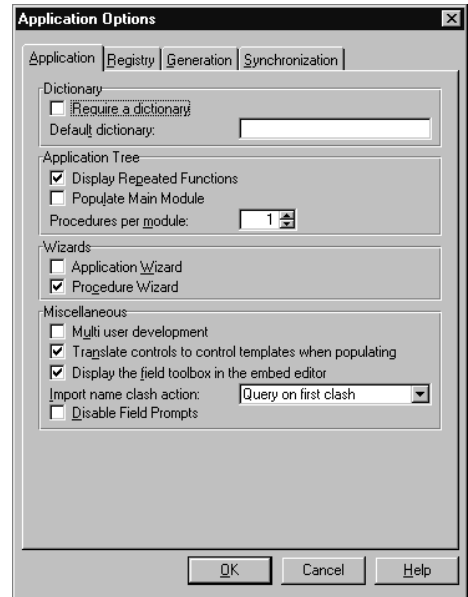
(#MESSAGE строка 1), Module and Procedure Names (имена модулей и процедур) (#MESSAGE строки 1 - 2) или All Messages (все сообщения) (#MESSAGE строки 1 - 3). См. #MESSAGE в Руководстве программиста.

### **Synchronization (синхронизация)**

На этой страничке задается когда и как записанные в Словарь данных атрибуты диалоговых элементов применяются в процедурах и диалоговых элементах приложения. См. также Меню Application - Synchronize, и Диалоговые элементы и их свойства - Общие атрибуты диалоговых элементов.

#### **Synchronize Application**

when opened (синхронизировать при открытии приложения) Отметьте этот режим, чтобы применять при каждом открытии приложения. Сбросьте отметку, чтобы применять атрибуты только по явной команде. См, далее Меню Application - Synchronize.



#### **Synchronize Window definitions**

(синхронизировать определения Window) Отметьте этот режим, чтобы применять словарные атрибуты к структурам WINDOW во время полной синхронизации приложения. Сбросьте отметку, чтобы пропускать структуры WINDOW.

#### **Synchronize Report definitions**

(синхронизировать определения Report) Отметьте этот режим, чтобы применять словарные атрибуты к структурам REPORT во время полной синхронизации приложения. Сбросьте отметку, чтобы пропускать структуры REPORT.

#### **Update controls for variables**

(Обновить диалоговые элементы для переменных памяти) Отметьте этот режим, чтобы применять диалоговые атрибуты переменных памяти к связанным с ними диалоговым элементам. Сбросьте отметку, чтобы пропускать диалоговые элементы, связанные с переменными памяти.

### Primary attributes only

(только основные атрибуты) Отметьте этот режим, чтобы применять только основные диалоговые атрибуты. Основные атрибуты, это те атрибуты, которые задаются в диалоге Field Properties. Они включают Field Name, Characters (длина), Screen Picture, Prompt Text, Column Heading, Case (UPR, CAP), Typing Mode (INS, OVER), Flgs (IMM, PASSWORD, READONLY), Justification, Initial Value, Help Ids, Messages, Tool Tips и Validity Checks. Вторичные атрибуты, это те, которые задаются нажатием кнопок Properties на страничках Window и Report диалога Field Properties.

Совет: Отметьте режим Primary attributes only для ускорения процесса синхронизации, особенно, если синхронизация проводится при каждом открытии приложения.

### Clear HELP,MSG,TIP if omitted in dictionary

(очищать атрибуты HELP,MSG,TIP, если они не указаны в словаре) Отметьте этот режим, чтобы переопределить связанные с диалоговыми элементами атрибуты подсказок (указанные в формате окна) на пустые значения из словаря данных. Сбросьте отметку, чтобы атрибуты сохранялись независимо от их наличия в словаре.

### Allow control types to change

(разрешить изменение типа диалоговых элементов) Отметьте этот режим, чтобы использовались новые типы диалоговых элементов. Например SPIN может быть заменен на ENTRY.

Внимание: При изменении типа элемента может возникнуть “висящие” вставки текста. Например, у SPIN имеется точка вставки NewSelection, а у ENTRY - нет. Повисшие вставки нужно вручную перенести в подходящее место.

### Allow conversion from list to drop list

(разрешить преобразование list в drop list) Отметьте этот режим, чтобы разрешить преобразование list в drop list.

Clear all other attributes if omitted in dictionary (очищать все остальные атрибуты, если они не указаны в словаре)

Отметьте этот режим, чтобы переопределить все связанные с диалоговыми элементами атрибуты (указанные в формате окна), кроме HELP,MSG и TIP, на пустые значения из словаря данных. Сбросьте отметку, чтобы атрибуты сохранялись независимо от их наличия в словаре. Отметка этого режима разблокирует кнопку More, которая позволяет указать режим независимо для каждого из атрибутов.

- More (еще) Нажмите кнопку, чтобы для каждого из атрибутов разрешить переопределение атрибутов, указанных в форматере окна, на пустые значения из словаря данных, или запретить переопределение вне зависимости от наличия атрибута в словаре. Это атрибуты Font, Alert, Tally, Cursor, Key, Icon и Colors.
- Dictionary can override size  
(словарь может переопределять размеры) Отметьте этот режим, чтобы разрешить преобладание словарных атрибутов размера, над заданными в форматере окна. Размеры элементов могут изменяться, когда для них заданы стандартные размеры и изменяется текст элемента, или когда в словаре явно указаны размеры, которые отличаются от заданных в форматере окна.
- Ignore Freeze attribute setting  
(игнорировать установленный атрибут Freeze) Отметьте этот режим, чтобы разрешить применение словарных атрибутов к элементам с атрибутом #Freeze. Сбросьте отметку, чтобы эти элементы не затрагивались. См. Диалоговые элементы и их свойства - Общие атрибуты диалоговых элементов - Установка режимов диалогового элемента.
- Refreeze frozen control after synchronize  
(переустановить #Freeze после синхронизации) Отметьте этот режим, чтобы переустановить #Freeze после синхронизации. Сбросьте отметку, чтобы снять атрибут #Freeze после синхронизации.
- Update field formatting  
(обновить форматы полей) Отметьте этот режим, чтобы разрешить применение словарных атрибутов к ФОРМАТ-ным строкам структур LIST (т.е. выравнивание). Или, другими словами, форматировать поля списка в соответствии с атрибутами Словаря данных. Сбросьте отметку, чтобы оставить LIST структуры без изменений.
- Update column headers  
(обновить заголовки колонок) Выберите из выпадающего списка режим приведения в соответствие заголовков колонок в списках и требований Словаря данных. Выберите одно из:  
Always (Всегда) Генератор приложений всегда использует словарный заголовок, даже, если он пустой.  
If present in dictionary  
(Если указано в словаре) Генератор приложений использует словарный заголовок, только если он не пустой, иначе используется заголовок структуры LIST.  
Window and Dictionary

(Окно и Словарь) Генератор приложений использует словарный заголовок, только если оба заголовка не пустые, иначе используется заголовок структуры LIST.

Never (Никогда) Генератор приложений никогда не использует словарный заголовок. Всегда используется заголовок структуры LIST.

Display warning if could not

synchronize (Вывести предупреждение, если синхронизация невозможна) Отметьте этот режим, чтобы при возникновении предупреждающих сообщений они выводились в диалоговом окне. Предупреждения возникают при изменении размеров элемента

Add report entry when controls change size

(Добавить в протокол строки для изменивших размер элементов) Отметьте этот режим, чтобы в протоколе появлялись предупреждающих сообщения о изменениях размеров элементов. При выключенном режиме сообщения не формируются.

Filename for report

(файл протокола) Укажите имя файла протокола для вывода предупреждающих сообщений. Очистка этого поля подавляет формирование протокола.



## Создание .APP файла.

Первым шагом в создании нового приложения (после создания Словаря данных) является создание .APP файла. .APP файл хранит процедуры, данные и другие составные части, которые заданы для приложения.

Совет: Полезно создавать новый каталог для каждого разрабатываемого приложения, так как всякий раз при открытии .APP файла, Clarion For Windows будет использовать каталог, в котором расположен .APP файл, как рабочий каталог.

1. Выберите в диспетчере файлов File? Create Directory или File? New? Folder, введите имя создаваемого подкаталога и нажмите ОК (или используйте командную строку DOS и команду Mkdir).

2. Запустите Clarion и выберите File?New.

Появится диалоговое окно New.

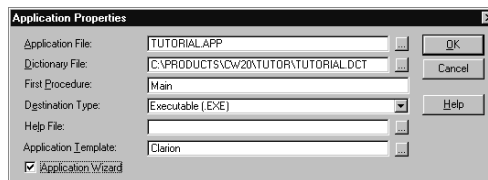
3. На страничке Application (приложение) (щелчком мыши над ней) уберите флажок в поле Quick Start Wizard с помощью щелчка мыши над ним.

Так как Quick Start Wizard описывается в руководстве Быстрое начало, здесь он обсуждаться не будет.

4. Пользуйтесь выпадающим списком Drives (Дисководы) и полем Folders (Папки) для продвижения к каталогу вашего приложения, затем нажмите кнопку Create (Создать).

Появится диалоговое окно Application Properties (Свойства приложения). Это диалоговое окно даст вам возможность определить существенные файлы для приложения.

Определение нового .APP файла под названием TUTORIAL.APP, с использованием Словаря данных TUTORIAL.DCT.



5. Введите имя для .APP файла в поле Application File (Файл приложения).

Имя вводится в соответствии с правилами DOS для имен файлов (должна быть также правильная метка Clarion, более подробно вы можете узнать об этом в Справочнике языка). CLARION автоматически добавит к имени расширение .APP. Clarion будет использовать путь, который вы определили в диалоговом окне New (Новый). Или вы можете нажать кнопку с многоточием (...) для того, чтобы определить путь к вашему .APP файлу.

6. В поле Dictionary File (Файл словаря) наберите имя файла словаря данных, которым будет пользоваться приложение (расширение .DCT) или нажмите кнопку с многоточием (...) для выбора файла словаря через диалоговое окно Select Dictionary (Выберите словарь).

Процедура создания словаря данных описывается в предыдущей главе. Диалоговое окно Select Dictionary - это стандартный диалог открытия файла (Open File).

Совет: Если в диалоговом окне Application options в поле Require Dictionary (требуется словарь) вы отмените необходимость обязательного использования словаря данных, Application Generator не будет требовать словаря данных для генерации приложения. Описание диалогового окна Application options приводится ниже.

7. Вы можете, если хотите, дать первой процедуре Вашего приложения произвольное имя взамен стандартного имени MAIN.

Сделать это вы можете, набрав новое имя процедуры в поле First Procedure (Первая процедура).

8. Выберите из предлагаемого системой выпадающего списка Destination Type тип генерируемого компилятором модуля для Вашего приложения: либо исполняемый Executable (.EXE), либо библиотечный Library (.LIB), либо динамически загружаемую библиотеку Dynamic Link Library (.DLL).

Совет: Использование .LIB и .DLL для разбиения больших приложений на модули позволит значительно сократить стоимость разработки и сопровождения: компиляция и сборка только небольшой части приложения сокращает время разработки, а вызов набора общих функций из единственного источника означает сопровождение лишь единственного экземпляра программного кода.

Можно разрабатывать фрагмент приложения как .EXE, а по завершении переделать его в .DLL.

См. Стратегии разработки и развертывания.

Замечание: Нужно устанавливать Application's Destination Type, а не Project Target Type, потому что Генератор Приложений и шаблоны при генерации текста анализируют Application's Destination Type, а не Project Target Type.

9. В поле Help File (справочный файл) введите имя .HLP файла для Вашего приложения или, используя кнопку с многоточием (...) или выберите файл через диалоговое окно Select Help File .

Генератор приложений требует, чтобы в этот момент уже существовал файл .HLP, но Вы можете объявлять несуществующие темы, так что файл может быть “фиктивным”.

Если Вы указываете справочный файл в текущем каталоге, приложение будет искать файл сначала в в текущем каталоге, затем в системном каталоге, а затем по каталогам, перечисленным в PATH. Полная спецификация файла не хранится.

Однако, если указать файл в другом каталоге, то формируется и сохраняется полная спецификация файла, и приложение будет разыскивать файл по указанному пути.

Вы отвечаете за создание стандартного Windows файла .HLP, который содержит контекстные строки символов и ключевые слова, которые вы вводите как необязательные атрибуты HLP для различных элементов управления и диалоговых окон. Существует много продуктов разных производителей программного обеспечения, которые помогут вам сделать это.

Выберите тип Application Template (Шаблон приложения).

Вы можете использовать стандартный шаблон (Clarion), или, нажав кнопку с многоточием (...), выбрать другой набор. Шаблон приложения управляет процессом кодогенерации.

11. Уберите флажок в поле Use Application Wizard (Использование Мастера приложения) щелкнув над ним мышью.

Отметка в этом поле заставляет Генератор приложений использовать ваш словарь для создания всего работающего приложения. В этой главе мы построим приложение без использования Мастера приложения.

12. Нажмите кнопку ОК .

CLARION For Windows создаст .APP файл и затем выведет на экран в диалоговом окне Application Tree(дерево приложения) дерево вновь создаваемого приложения.

## Обзор: Создание приложения

С момента создания .APP файла вы ведете разработку Вашего приложения через последовательность диалогов с системой. Когда вы создаете различные “меню” и “панели инструментов” своего приложения они вызывают Указанные Вами процедуры. Генератор Приложений автоматически добавляет эти процедуры к дереву приложения, как процедуры типа “TO DO”. Выбирая подходящие шаблоны, вы наполняете их “функциональностью”.

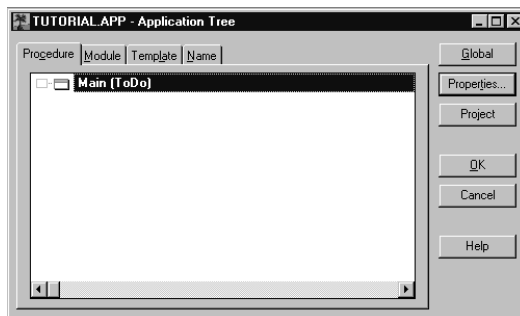
Помните, что шаблоны являются кодогенераторами, которые Вы должны снабдить информацией о настройке генерируемого кода. Для наполнения шаблонов информацией о внешнем виде Вашего приложения, как его увидит конечный пользователь, используются формтеры окон и отчетов (Window Formatter и Report Formatter).

Далее следует обзор, который иллюстрирует задачи, которые вы обычно решаете при построении приложения с помощью Генератора Приложений. Более подробную информацию даст Вводный курс в руководстве Getting Started.

Начальным пунктом этой экскурсии является диалоговое окно Application Tree. После завершения диалога в окне Application Properties новое дерево приложения содержит единственную процедуру типа (TO DO), с именем MAIN (или другим, если вы меняли имя). Наша экскурсия предполагает, что словарь данных уже существует.

Диалоговое окно Application Tree для нового приложения. Оно содержит одну MAIN процедуру типа (TO DO)

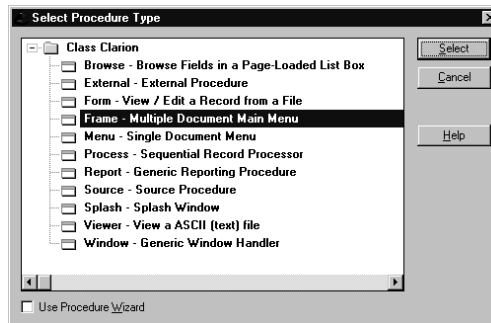
- ☐ Определите тип процедуры MAIN .
  - ☐ Добавьте команды меню и их “ToDo” процедуры
- ☐ Определите “ToDo” процедуры
  - ☐ Выберите подходящий шаблон для генерации каждой из процедур
  - ☐ Укажите файлы, используемые каждой процедурой
  - ☐ Добавьте локальные переменные
  - ☐ Используйте Формтер Окона для разработки своих окон
- ☐ Генерируйте исходный код и постройте приложение
- ☐ Оттестируйте приложение нажатием на кнопку Run в диалоге Compile Results.



### Определение процедуры Main.

Нажмите кнопку Properties (свойства) для вызова диалогового окна Select Procedure Type (выбор типа процедуры). Вам будет предоставлен для выбора список типов процедур, доступных в зарегистрированном в настоящее время регистре шаблонов.

Выберите тип процедуры Frame для MAIN из диалогового окна Select Procedure Type и нажмите кнопку Select.



Процедура шаблона Frame обычно является наилучшей отправной точкой для типичного приложения, в котором используются различные MDI - дочерние окна для предоставления разнообразных способов обзора данных и форм. Эта шаблонные процедуры содержат основу MDI-окон, которые уже включают полностью функциональные стандартные меню окон типа File, Edit и Help

Появляется диалоговое окно Procedure Main Properties (свойства основных процедур). Обыкновенно, каждая процедура уже содержит стартовые установки по умолчанию для таких элементов, как окно приложения, базовая структура меню, к которой вы можете сделать добавления, отчеты и др. Эти процедуры по умолчанию спроектированы для реальных применений, имеются в виду такие применения, как, например, формы (форма - это просто окно, которое показывает одиночную запись и дает возможность изменить запись) для обновления записей в базе данных. При разработке конкретного приложения вы можете переработать эти процедуры-заготовки в соответствии с вашими требованиями.

### Добавьте команды меню и их “ToDo” процедуры.

Нажмите кнопку Window, чтобы получаете доступ к форматуру окна. Когда появится Форматер окна, перейдите непосредственно к Menu Editor (Редактор меню): выберите Menu ? Menu Editor. Появится диалоговое окно Menu Editor. Более подробное описание редактирования меню смотрите в главе Создание меню и линеек инструментов.

Для добавления нового пункта меню нажмите кнопку Item. Затем выберите страничку

Actions (Действия) чтобы указать, какую процедуру или программу данный пункт меню будет вызывать, когда его выберет конечный пользователь. После того, как Вы введете имя процедуры, Генератор приложений добавит новую “ToDo” процедуру к дереву процедур приложения.

При создании MDI - приложения вам необходимо отметить переключатель Initiate Thread (Инициализация Процесса).

Нажмите кнопку Close для закрытия Menu Editor, сохраняя сделанные вами изменения. Нажмите меню Exit! для выхода из формatera окна с сохранением изменений.

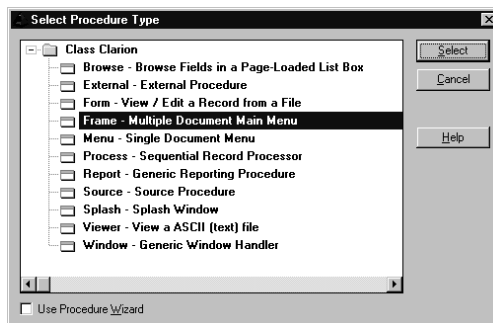
### **Определите “ToDo” процедуры.**

Чтобы сделать это, выберите первую процедуру “To Do” в дереве приложения и нажмите кнопку Properties (Свойства). Пункт “ToDo” - это процедура или процедуры, названные вами с помощью Menu Editor .

### **Выберите подходящий шаблон для генерации каждой из процедур.**

Выберите тип процедуры из диалогового окна Select Procedure Type (Выбор типа процедуры) и нажмите кнопку Select. В этой точке вы можете, например, выбрать для процедуры шаблон просмотра таблиц, который показывает записи в поле списка.

Выберите тип процедуры из списка имеющихся шаблонов процедур.



Если Вы отметите переключатель Use Procedure Wizard, то Мастера Таблиц, Форм и Документов запросят у Вас всю необходимую для создания процедуры информацию. Иначе это нужно будет сделать вручную.

### **Укажите файлы, используемые каждой процедурой.**

В диалоговом окне Procedure Properties нажмите кнопку Files, чтобы открыть диалоговое окно File Schematic Definition (Определение схемы файлов) и затем выберите файлы и ключи, которые будут использоваться в процедуре. Для более подробной инструкции см. ниже раздел Схема файлов. Добавляя файлы к схеме файлов, Вы открываете доступ к ним

в разрабатываемой процедуре .

### **Добавьте локальные переменные.**

Нажмите кнопку Data (Данные) в диалоговом окне Procedure Properties. В разделе Данные процедуры это описано более подробно. По существу Вы объявляете каждую переменную так же, как определяете поле в файле данных.

### **Используйте Форматер Окна для разработки своих окон.**

В диалоговом окне Procedure Properties нажмите кнопку Window; когда на экране появится Форматер окна и прообраз будущего окна. В зависимости от выбранного шаблона в окне будут размещены те или иные диалоговые элементы. Все, что появляется в окне, является элементами управления, включая кнопки, поля списка, поля флажков, spin поля, поля ввода данных и т.д. Выберите в окне элемент управления, затем Edit?Properties и Edit?Actions для указания того, что и как должен делать элемент управления. Более полную информацию вы найдете в главе Использование диалоговых, текстовых и распределенных шаблонов.



Применяйте диалоговые шаблоны (меню Populate) для размещения “заготовленных” элементов, которые являются полнофункциональными диалоговыми элементами в сочетании с обслуживающим их программным текстом, который обеспечивается Генератором приложений. Например, диалоговый шаблон BrowseBox создает списковый элемент и обслуживающий его программный текст, который загружает и прокручивает список. Программный текст можно изменять в Форматере Окна с помощью диалога свойств управляющего элемента.

Применяйте словарные поля (меню Populate) для размещения элементов “требующих некоторой настройки”, т. е. полей ввода, которые автоматически связываются с полями Словаря Данных или с переменными памяти.

Применяйте чистые диалоговые элементы (меню Control) для размещения диалоговых элементов “сделай сам”, т. е. только диалоговых элементов без обслуживающего их кода.

### **Генерируйте исходный код и постройте приложение**

Для генерации, компиляции и редактирования связей приложения нажмите кнопку Make (символ молнии) на панели инструментов для генерации исходных текстов программы, компиляции и сборки приложения. Генератор приложений автоматически обеспечивает информацию для компиляции и сборки приложения.

### **Оттестируйте приложение нажатием на кнопку Run в диалоге Compile Results.**

Испытайте приложение, нажав кнопку Run (Запуск) в диалоге Compile Results

(компиляция результатов).

После испытания вашей первой процедуры вы можете перейти к добавлению других процедур к вашему приложению, к встраиванию в текст программы вставок исходного кода, т. е. к добавлению в приложение новых функциональных возможностей.

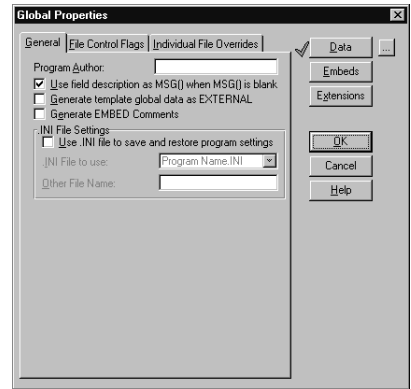


## Определение глобальных характеристик

Можно задать целый набор параметров, которые будут относиться ко всему приложению. Можно также объявить переменные памяти, которые будут общими для всей программы. Эти “глобальные” настройки выполняются в диалоговом окне Global Properties.

### Глобальные переменные

Все глобальные переменные приложения должны быть объявлены перед оператором CODE в вашем модуле PROGRAM (более подробно об этом в Справочнике языка). Проще всего сделать это с помощью диалогового окна Global Properties.



Чтобы получить доступ к диалоговому окну Global Properties (Глобальные свойства), перейдите к диалоговому окну Application Tree (Дерево приложения) и нажмите кнопку Global.

Чтобы добавить глобальные переменные, в диалоговом окне Global Properties нажмите кнопку Data

- Чтобы добавить новую переменную к списку:
  1. Нажмите кнопку Insert.
  2. Заполните диалоговое окно New Field Properties (свойства новых полей).

Диалоговое окно New Field Properties - это тот же самый диалог, который вы видели, когда добавляли поля к файлу словаря данных. В этом окне вы можете установить тип данных, длину, метку для переменной и т.д. Смотрите выше раздел Добавление или изменение полей.

3. Для закрытия диалогового окна New Field Properties нажмите кнопку OK .

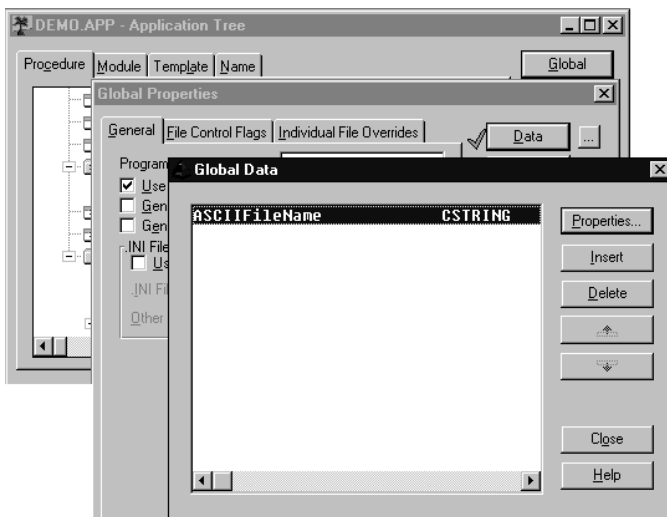
4. Нажмите кнопку Close , чтобы закрыть диалоговое окно Global Data.

- Для изменения типа данных или метки переменной:

1. Выделите переменную в списке диалогового окна Global Data.  
2. Нажмите кнопку Properties.  
3. Сделайте любые нужные вам изменения в диалоговом окне Edit Field (редактирование полей) Properties и нажмите кнопку OK.

4. Нажмите кнопку Close, чтобы закрыть диалоговое окно Global Data.

- ☐ Чтобы передвинуть переменную вверх в списке, выделите ее и затем нажмите кнопку  $\uparrow$
  - ☐ Для перемещения переменной вниз по списку выделите ее и нажмите кнопку  $\downarrow$
- Список переменных. типы переменных и другие их свойства задаются в диалоге Field Properties, который открывается кнопкой Properties.



## Общие характеристики приложения

На страничке General в диалоговом окне Global Properties имеются следующие параметры:

Program Author

(Автор программы) Запишите здесь права, которыми вы гордитесь.

Use field description as MSG() when MSG() is blank

(Использование описание поля как MSG когда MSG пусто)

Сообщает генератору приложений о необходимости использовать описание поля в словаре данных, как параметр атрибута MSG, когда никакой другой параметр не определен. Иными словами, описание поля становится по умолчанию сообщением строки статуса для каждого поля в вашем приложении.

Generate global data as EXTERNAL

(Генерировать глобальные данные с атрибутом EXTERNAL)

Добавляет атрибут EXTERNAL (Внешний) к вашим определениям глобальных переменных (см. Справочник языка). Это означает, что ваша программа будет полагаться на внешнюю

библиотеку при распределении памяти для этих переменных и при определении их как общих переменных так, чтобы ваша программа могла получить доступ к ним.

**Замечание:** Если Вы создаете приложение, которое состоит более, чем из одной, построенной Генератором Приложений DLL, Вы должны отметить “Generate global data as EXTERNAL” во всех приложениях, за исключением одного. См. Стратегии разработки и развертывания.

#### Generate EMBED Comments

(Генерировать комментарии к вставкам)

Указывает генератору приложений на необходимость окружить ваши вставки комментариями, которые облегчают их поиск внутри сгенерированной исходной программы.

## Поддержка файла .INI

---

Шаблоны Clarion поддерживают .INI файл (стандартный инициализационный файл Windows). Это ASCII файл, хранящий настройки приложения в перерыве между сеансами.

Одно из применений файла .INI это сохранение установленного пользователем положения окна до следующего сеанса. Многие из стандартных процедур Clarion обеспечивают это при условии, что включена поддержка .INI файла.

☐ Чтобы включить автоматическую поддержку файла .INI:

1. Выберите закладку General в диалоговом окне Global Properties.
2. Отметьте поле Use .INI file to save and restore program settings (Используйте файл .INI для сохранения и восстановления программных установок)ХЕ “

3. Укажите имя .INI файла

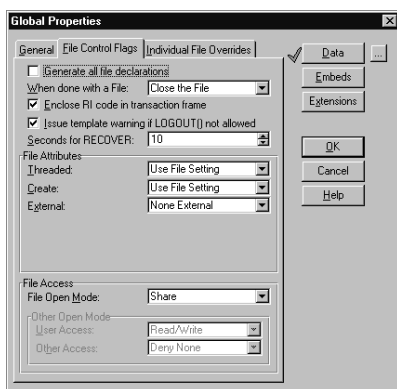
Чтобы задать то же имя файла, что и .EXE, но с расширителем .INI, выберите из выпадающего списка .INI file to use вариант ProgramName.INI. Чтобы указать другое имя, выберите в выпадающем списке Other и затем заполните поле Other File Name (Другое имя файла).

4. Нажмите кнопку ОК для закрытия диалогового окна Global Properties.

Если в вашем приложении возникнет необходимость изменить содержимое .INI файла, вы можете воспользоваться функцией PUTINI. Чтобы прочитать значение из .INI файла, воспользуйтесь функцией GETINI. Подробности вы найдете в Справочнике языка.

**Совет:** Если ваше приложение требует хранить десятки или даже сотню переменных от сеанса к сеансу, не помещайте их все в .INI файл, создайте управляющий файл и воспользуйтесь обычным вводом-выводом. Выборка переменных из .INI файла процедура относительно медленная. Кроме того, если вы хотите скрыть информацию от конечного пользователя, имейте в виду, что .INI файл это текстовый файл и, следовательно, легкодоступен.

## Определение других глобальных характеристик - Управление файлом



Диалоговое окно Global Properties позволяет вам переопределять некоторые параметры Словаря Данных. Можно также определить, каким образом процедуры будут иметь доступ к файлам. Можно уточнять атрибуты файла для всех файлов сразу или индивидуально. Чтобы получить доступ к этим характеристикам, выберите закладку File Control Flags (Параметры управления файлами).

Generate all file declarations

( Генерировать описания всех файлов) Флажок в этом поле указывает Генератору Приложений о необходимости включить в программу описание всех файлов из словаря данных, независимо от того, использованы они или нет в данном приложении. Описывая все эти файлы, вы можете сослаться на данные файлы в любом самодельном исходном коде, который вы могли добавить к вашему приложению.

When done with a File

(Когда закончена работа с файлом) Определяет, закрывает ли приложение автоматически каждый файл когда работа с ним завершена.

**Совет:**

Один из способов создания “хорошо ведущего себя “ в Windows приложения - это не “стричь наголо” ресурсы системы. Один из ограниченных ресурсов - это блоки управления файлом (file handlers). Вы можете “вернуть обратно” неиспользуемые блоки, указав в выпадающем списке Close the File (Закрывать файл).

Enclose RI code in transaction

frame (Включение RI кода в рамку транзакции) Разрешает операцию отката в случае,

если обновление не происходит во время операции поддержания ссылочной целостности.

**Совет:** Если все файлы в цепи отношений используют одну и ту же файловую систему и файловая система поддерживает транзакции, а вы не хотите использовать транзакции для RI кода, Вы должны очистить поле флажков для каждого файла в Individual File Override (переопределения для индивидуальных файлов) и в Global Settings (глобальные установки).

Issue template warning if LOGOUT() not allowed

(Шаблон должен выдать предупреждение если LOGOUT() невозможен) Когда в Словаре Данных содержится драйвер файла, который не поддерживает функцию LOGOUT (используемую в процедурах проверки ссылочной целостности), отметка в этом поле формирует предупреждение во время компиляции.

Threaded

(Для процессов)

Это требуется для MDI процедур просмотра и обновления данных в файле, чтобы предотвратить конфликты с буферами записи, когда конечный пользователь меняет фокус от одного процесса к другому.

Use File Setting

(Применить параметры словаря) Применяет атрибут THREAD в соответствии со Словарем Данных.

All Threaded

(Все для процессов) Добавляет атрибут THREAD к каждому файлу.

None Threaded

(Ни один для процессов) Опускает атрибут THREAD у всех файлов.

Create (Создать)

Определяет, будет ли приложение создавать файл данных, если его не существует. Добавляет атрибут CREATE к структуре FILE.

Use File Setting

(Применить параметры словаря) Применяет атрибут CREATE в соответствии со Словарем Данных.

Create All (Создавать все)

Добавляет атрибут CREATE к каждому файлу.

Create None

(Не создавать ни один) Опускает атрибут CREATE у всех файлов.

External (Внешний) Определяет, добавит ли Генератор Приложений к структуре FILE атрибут EXTERNAL. Атрибут EXTERNAL указывает, что память для буфера записи файла выделяется во внешней библиотеке. См. Руководство по языку.

### None External

(Ни один External) Опускает атрибут EXTERNAL у всех файлов.

All External (Все External) Добавляет атрибут EXTERNAL к каждому файлу и позволяет указать, Declaring module (объявлен в модуле) или All files are declared in another .APP (все файлы объявлены в другом приложении).

Замечание: Применения EXTERNAL при совместного использовании файла в нескольких модулях (.LIB, .DLL или .EXE), только в одном модуле файл должен быть объявлен без атрибута EXTERNAL. Это обеспечит единственный буфер записи для файла и все битбблиотеки и .EXE будут использовать одну и ту же память при обращении к полям этого файла.

### Declaring module

(Объявлен в модуле) Имя файла (без расширения) MEMBER модуля, содержащего объявление файла без атрибута EXTERNAL. Если файл объявлен в PROGRAM модуле, поле должно быть пустым.

All files are declared in another .APP

(Все файлы объявлены в другом приложении) Отметка в этом поле указывает Генератору приложений, что файлы объявлены в другом приложении (а не при ручном кодировании). Генератор Приложений добавит атрибут EXTERNAL к переменной File:Open (которая управляет вызовом процедуры CheckOpen), гарантируя правильные моменты открытия/закрытия файлов, что обеспечивает целостность буферов данных.

### File Open mode

(Режим открытия файла) Указывает, как приложение открывает файлы. См. Руководство по языку.

Open Открывает файл для чтения/записи с запретом записи для всех остальных.

Share Открывает файл для чтения/записи ничено не запрещая всем остальным.

Other (Другой) Указывает на специфическую комбинацию прав доступа к файлу.

### User Access

(для приложения) Укажите Read Only (только чтение), WriteOnly (только запись), или

Read and Write (чтение и запись).

Other Access

(для всех остальных) Укажите Deny None (нет запретов), Deny All (запрещено все), DenyRead (запрет чтения), DenyWrite(запрет записи) или AnyAccess (режим совместимости с FCB)

### **Individual File Overrides (Индивидуальные переопределения файла)**

Выберите закладку Individual File Overrides чтобы переопределить словарные значения параметров для отдельных файлов. Выберите файл, для которого нужно изменить значения параметров и нажмите кнопку Properties.

Параметры на этой станции зеркально соответствуют параметрам странички File Control Flags (Параметры управления файлами) и ведут себя точно также, за двумя исключениями:

- Указанные значения применяются только к выбранному файлу.
- Для каждого параметра имеется дополнительный вариант: Use Default (стандартно). Этот вариант устанавливает атрибуты, как указано в File Control Flags (Параметры управления файлами).

### **Точки вставок в окне глобальных характеристик**

Через точки вставок диалоговое окно Global Properties (Глобальные характеристики) обеспечивает доступ к глобальной MAP структуре программы, к разделу глобальных данных приложения, к разделу, который открывает все файлы, и к разделу, который обрабатывает ошибки, встречающиеся при открытии файлов. Можно также вставить текст в файл экспорта (.EXP), известный также под названием файла определений модуля (Module Defenition File), и в список поставляемых модулей приложения.

Чтобы получить доступ к этим точкам, нажмите кнопку Embeds (вставки) в диалоге Global Properties (Глобальные характеристики). Как и для любой другой точки встраивания текста в них можно написать свой собственный программный код, вызывать процедуру или использовать текстовый шаблон. Генератор приложений, при генерировании исходного текста, размещает Ваш текст или обращается к указанной процедуре на строке исходного кода, следующей за точкой, которую Вы выбрали в диалоговом окне Embedded Source (Вставка исходного текста).

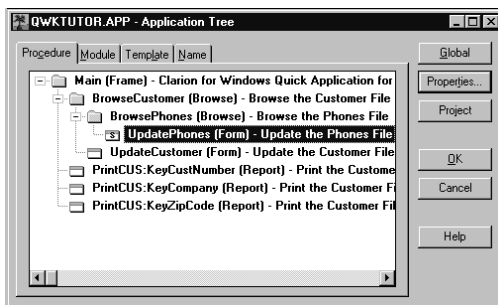
Смотрите в разделе вставки исходного текста более полную информацию о вставке в приложение исходного текста.

## Добавление процедур к вашему приложению

Процедура - это последовательность инструкций - операторов на языке Clarion, которые выполняют задачу. Когда программа выполняет задачу, такую, как например, печать отчета - это исполняемая программа. Первая процедура, которую выполняет ваше приложение, называется по умолчанию “Main” (главная) . Вы можете назвать вашу первую процедуру, как угодно, но когда эта глава ссылается на первую процедуру, мы будем ссылаться на процедуру под названием “Main”. Все другие процедуры отвечают от “Main” - одна процедура может вызвать другую.

Диалоговый элемент с древовидной структурой (или элемент структурной схемы (outline control)) в диалоговом окне Application Tree (Дерево приложения) иллюстрируют ветвление процедур от корня “Main”. Это дерево дает схеме логической структуры приложения. Знак доллара \$ на пиктограмме процедуры означает, что процедура содержит вставки(Embeds).

Дерево приложения показывает процедуры, которые вы создаете, когда добавляете элементы меню, команды из панели инструментов или процедуры, вставленные как исходный текст. Каждая новая процедура помечается “To Do” и сопровождается вашими комментариями.



## Определение типа процедуры

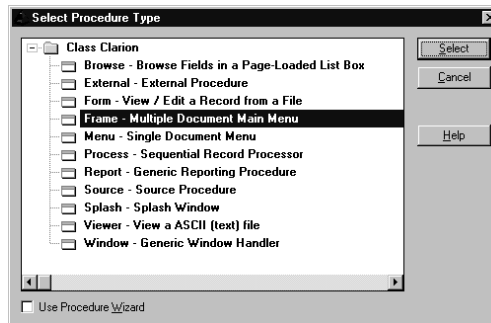
После появления процедуры в дереве приложения, первым шагом является выбор типа процедуры из списка типов процедур, имеющих в диалоговом окне Select Procedure Type (Выбор типа процедуры).

- Чтобы открыть диалоговое окно Select Procedure Type, выберите любую “ToDo” процедуру в диалоговом окне Application Tree (Дерево приложения), затем нажмите кнопку Properties или выберите Edit?Properties. Вы можете также щелкнуть дважды мышью над процедурой.

- Чтобы определить тип процедуры из предлагаемого набора процедур, выделите



необходимый вам тип в диалоговом списке Select Procedure Type, затем нажмите кнопку Select (выбрать). Вы можете дважды щелкнуть мышью над типом процедуры. Если Вы выбрали процедуру Browse, процедуру Form или процедуру Report и отметили поле флажков Use Procedure Wizard (Использование Мастера процедур), появится диалог соответствующего Мастера, который проведет Вас по всем этапам определения свойств процедуры. Если вы не отметили поле флажков Use Procedure Wizard, появится диалоговое окно Procedure Properties.



Если Вам придется позднее изменить тип уже разработанной процедуры, выберите Procedure ? Change Template Type. Появится диалоговое окно Select Procedure Type, так что вы сможете выбрать другой тип процедуры. Если новый тип процедуры не поддерживает некоторые структуры - такие как меню - которые вы определили в предыдущем типе процедуры, вы можете “потерять” некоторые другие ранее определенные структуры. Следовательно, будьте осторожны при изменении типа процедуры.

## Определение характеристик процедуры

После того, как вы выбрали тип процедуры, вы можете определить характеристики процедуры - эти характеристики включают:

- описание данной процедуры
- прототип процедуры
- модуль, содержащий исходную программу
- параметры, передаваемые процедуре
- значение, которое возвращает функция
- использование .INI файла
- файлы, к которым имеет доступ процедура
- окно, показываемое процедурой, включая его размер, форму, вид и функциональность
- генерируемый процедурой документ
- элементы данных (поля и переменные), используемых процедурой
- процедуры, вызываемые данной процедурой
- специфический встроенный в процедуру исходный текст

- формулы, используемые данной процедурой
- распределенные шаблоны, которые расширяют функциональные возможности процедуры

Вам нет необходимости определять каждую характеристику для каждой процедуры. Во многих случаях подходящим является стандартное определение характеристик. Когда стандартная характеристика уже задана, около ее командной кнопки появляется зеленая галочка. Например, шаблон процедуры просмотра содержит предопределенное окно, поэтому для процедур, сгенерированных с помощью этого шаблона рядом с кнопкой Window появляется зеленая галочка.

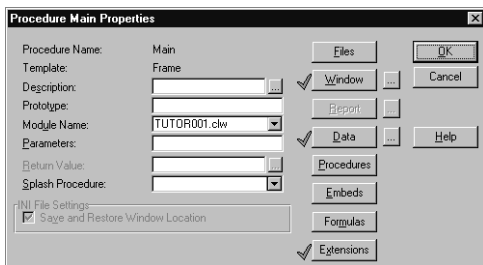
Для задания своих характеристик пользуйтесь диалоговым окном Мастера или диалоговым окном Procedure Properties, а также его подчиненными диалоговыми окнами и форматерами. Диалоги Мастеров используют стандартные значения для многих характеристик процедур, а затем шаг за шагом запрашивают информацию, необходимую для завершения процедуры. Диалоговые окна Procedure Properties позволяют более гибко определять характеристики процедуры.

Данный раздел будет в первую очередь посвящен диалоговым окнам Procedure Properties. Обсуждаемые в данном разделе характеристики одинаковы во всех диалоговых окнах Procedure Properties. Вы определяете эти свойства путем заполнения полей ввода и использования командных кнопок в диалоговых окнах Procedure Properties.

В последующих главах обсуждаются дополнительные свойства процедур, их диалоги и различные процедурные шаблоны, которые используют их.

□ Для задания свойств процедуры:

На дереве приложения выберите процедуру и нажмите кнопку Properties для получения доступа к диалоговому окну Procedure Properties. Или дважды щелкните мышью над процедурой или щелкните правой кнопкой мыши над процедурой и выберите Properties из всплывающего меню.



Диалог Procedure Properties. Другие процедуры могут включать дополнительные подсказки для установления дополнительных функциональных возможностей.

Description (Описание)

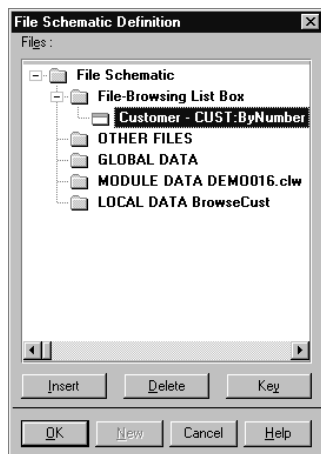
Краткое описание процедуры. Появляется в виде короткого текста вслед за именем процедуры в диалоговом окне Application Tree.

Нажмите кнопку с многоточием (...), чтобы ввести или отредактировать более длинное описание (до 1000 символов)

.

- Prototype (Прототип)** Позволяет ввести специфический прототип процедуры, который Генератор Приложений помещает в MAP структуру приложения. Если ничего не указано, Генератор Приложений обеспечит правильный прототип для выбранного процедурного шаблона. См. Прототипы процедур в Руководстве по языку и Прототипы и передача параметров ниже. Например: (SHORT, LONG, STRING), LONG
- Module Name (Имя модуля)** Выберите из выпадающего списка файл модуля (.CLW), который будет содержать исходный текст процедуры. По умолчанию генератор приложений строит имена модулей из первых пяти символов имени .APP файла и трехзначного номера для каждого модуля. Можно указать другие имена модулей, выбрав в меню tApplication?Insert Module. Например: MYAPP001.CLW
- Parameters (Параметры)** Укажите параметры, которые передаются процедуре. Параметры - это разделенных запятыми необязательный список меток, выражений или литералов. Список заключается в скобки. В исходном тексте процедуры Вы должны обеспечить функциональность этих параметров. См. Прототипы процедур в Руководстве по языку и Прототипы и передача параметров ниже. Например: (CustID, PI\*2, 'SMITH')
- Return Value (Возвращаемое значение)** Укажите переменную, которая получает возвращаемое функцией значение (функции возвращают значение, а процедуры - не возвращают). См. Прототипы процедур в Руководстве по языку и Прототипы и передача параметров ниже.
- Save and Restore Window Location (Сохранение и восстановление положения окна)** Отметьте этот режим, чтобы процедура с помощью .INI файла сохраняла и восстанавливала положение окна.

## Файлы процедуры



Данные файла (данные, хранящиеся в файлах приложения) доступны для любой процедуры всего приложения, однако Генератору Приложений нужно указать, какие файлы будут использованы, чтобы он мог построить программу для чтения файла.

☐ Нажмите кнопку Files (Файлы) в диалоговом окне Procedure Properties (характеристики процедуры).

Чтобы добавить файл данных к схеме файлов, выберите соответствующую строку в списке Files и нажмите кнопку Insert. В диалоговом окне Insert File (Вставить файл) выберите файл. Первый файл, который вы добавите для элемента управления, будет “первичным” файлом. Все остальные - вторичными.

Чтобы удалить элемент из схемы файлов, выделите его и нажмите кнопку Delete.

Чтобы установить последовательность доступа к записям файла, выберите файл в списке Files и нажмите кнопку Key. Затем в диалоговом окне Change Access Key (Изменить ключ доступа) выберите необходимый вам ключ.

## Окна процедуры

☐ Нажмите кнопку Window (Окно) в диалоговом окне Procedure Properties (характеристики процедуры).

Форматер окна Window Formatter дает вам возможность визуально определить размер, форму, меню, элементы управления и функциональные характеристики для окна разрабатываемой вами процедуры. Доступ к Window Formatter вы получите, нажав кнопку Window в диалоговом окне Procedure Properties или из дерева приложения, выбрав процедуру, щелкая на ней правой кнопкой мыши и затем выбирая Window из всплывающего меню. Дополнительную информацию о том, как определить ваше окно, вы найдете в главе Использование форматера окна.

☐ Нажмите кнопку с многоточием (...) около кнопки Window (Окно) в диалоговом окне Procedure Properties (характеристики процедуры).

Совет: Кнопка с многоточием (...) рядом с кнопкой Window позволяет отредактировать исходный текст объявления структуры Window. В исходном тексте Вы

увидите некоторые не относящиеся к языку ключевые слова, такие, как #LINK или #ORIG. Не удаляйте и не меняйте их. Генератор приложений использует их для построения исходных текстов, но не включает их в генерируемый текст.

## **Документы процедуры**

---

□ Нажмите кнопку Report (Документ) в диалоговом окне Procedure Properties (характеристики процедуры).

Форматер документа Report Formatter дает вам возможность визуально определить размер, форму, меню, элементы управления и функциональные характеристики для документа разрабатываемой вами процедуры. Доступ к Report Formatter вы получите, нажав кнопку Report в диалоговом окне Procedure Properties или из дерева приложения, выбрав процедуру, щелкая на ней правой кнопкой мыши и затем выбирая Report из всплывающего меню. Дополнительную информацию о том, как определить документ, вы найдете в главе Использование форматера документов.

□ Нажмите кнопку с многоточием (...) около кнопки Report (Документ) в диалоговом окне Procedure Properties (характеристики процедуры).

Совет: Кнопка с многоточием (...) рядом с кнопкой Report позволяет отредактировать исходный текст объявления структуры Report. В исходном тексте Вы увидите некоторые не относящиеся к языку ключевые слова, такие, как #LINK или #ORIG. Не удаляйте и не меняйте их. Генератор приложений использует их для построения исходных текстов, но не включает их в генерируемый текст.

## **Данные процедуры**

---

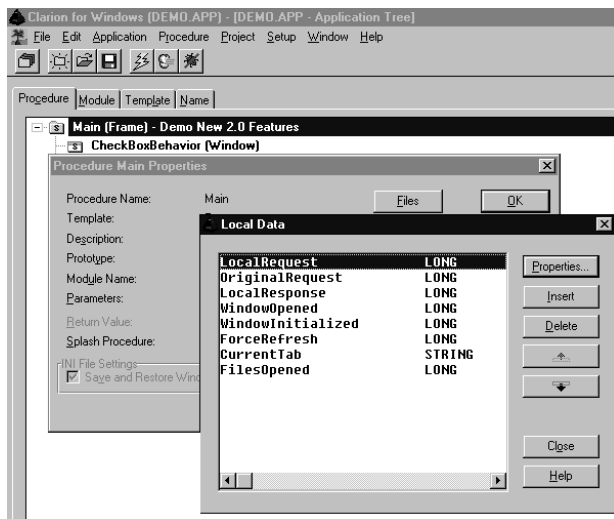
Процедуры могут иметь доступ к нескольким классам данных. Сюда входят данные файла и его полей, данные GLOBAL (глобальные), данные MODULE (модуля), данные LOCAL (локальные).

Локальные данные объявляются в секции данных процедуры и доступ можно получить только в пределах той процедуры, где данные были определены. Данные модуля объявляются в секции данных модуля. Модуль - это просто файл исходного текста, который может содержать несколько процедур. К данным модуля можно получить доступ из любой процедуры, содержащейся в том исходном файле, где определены данные модуля. К глобальным данным можно получить доступ из любой процедурой приложения. Более подробно об этом рассказано в разделе Объявления данных и размещение памяти в Справочнике языка.

## Локальные данные

□ Нажмите кнопку Data (Данные) в диалоговом окне Procedure Properties (характеристики процедуры).

Откроется диалог Local Data (Локальные данные). В списке будут показаны все ранее объявленные локальные переменные.



Для объявления нового данного нажмите кнопку Insert (Добавить).

Появляется диалог New Field Properties (Свойства нового поля). Напечатайте имя переменной, укажите ее тип, укажите остальные атрибуты, включая атрибуты для окна. См. Словарь Данных - Добавление и модификация полей.

□ Нажмите кнопку с многоточием (...) около кнопки Data (Данные) в диалоговом окне Procedure Properties (характеристики процедуры) чтобы отредактировать TXA текст объявления данных. См. Руководство программиста о подробностях TXA формата.

## Данные модуля

□ Нажмите кнопку Data (Данные) в диалоговом окне Module Properties (характеристики модуля).

Данные модуля объявляются точно также, как локальные данные за одним исключением - нужно выбрать не процедуру, а модуль. Чтобы определить данные модуля (переменные памяти, доступные процедурам, находящимся в одном модуле):

1. В диалоговом окне Application Tree (Дерево приложения) выберите страничку Module.
2. Отметьте модуль (папку), а не процедуру.
3. Нажмите кнопку Properties, чтобы вызвать диалоговое окно Module Properties или щелкните правой кнопкой мыши и выберите Data из всплывающего меню.

4. Нажмите кнопку Data.

Появится диалоговое окно Module Data.

5. Нажмите кнопку Insert (Вставить).

Появится диалоговое окно Field Properties. Это то же самое диалоговое окно, как и то, в котором вы устанавливаете свойства поля в словаре данных. Подробности смотрите в разделе Использование словаря данных и в разделе, предшествующем данному.

## **Вызовы других процедур**

---

☐ Нажмите кнопку Procedures (Процедуры) в диалоговом окне Procedure Properties (характеристики процедуры).

Это имена процедур, которые добавляются к дереву приложения.

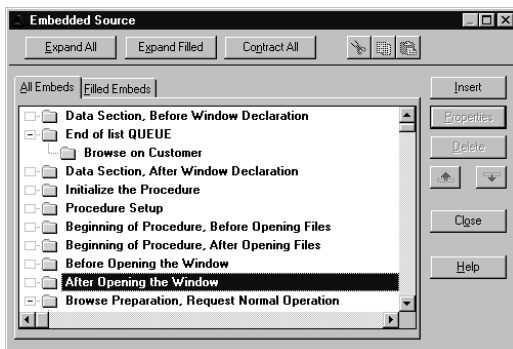
Одни процедуры могут вызывать другие процедуры. Вызовы процедур, указанные в шаблонных диалогах добавляются к дереву приложений автоматически, однако Генератор Приложений не может “увидеть” вызовы процедур из вставок исходного текста.

Чтобы добавить к дереву приложения процедуры, которые вызываются из вставок исходного текста, нажмите кнопку Procedures (Процедуры) для доступа к диалоговому окну Called Procedures (Вызываемые процедуры). Чтобы добавить вызываемую процедуру, нажмите кнопку Insert (Вставить) и введите имя процедуры в следующем диалоговом окне. Чтобы удалить процедуру, нажмите кнопку Delete. Дополнительные кнопки дают вам возможность изменить порядок перечисленных вызываемых процедур.

Совет: Назначение кнопки Procedures - вставить вызов другой процедуры и добавить эту, вызываемую процедуру к дереву приложения. Процедуры, которые вызываются другим способом, автоматически добавляются к дереву приложения.

## Встраивание исходного текста (Embedded Source)

□ Нажмите кнопку Embeds (Точки вставки) в диалоговом окне Procedure Properties (характеристики процедуры).



Вставки исходного текста в процедуру предоставляют Вам все возможности по настройке процедур. Можно определить или создать последовательности операторов для выполнения перед, во время и после процедуры. Вы можете сами написать программный код или использовать текстовый шаблон, который сгенерирует необходимый текст. Далее подробно рассматривается процесс встраивания исходного текста.

Как только вы сделали вставку, процедура помечается в дереве приложения знаком доллара (\$).

Для того, чтобы эффективно встраивать исходные тексты, необходимо разбираться в том, как устроен сгенерированный текст в который Вы будете добавлять свои вставки. Чтобы освоиться с типичными приемами программирования Clarion шаблонов, изучите по справочной системе Пример исходного текста Browse процедуры и Пример исходного текста Form процедуры (Выберите Help (Справка), нажмите кнопку Search (Поиск)Б затем введите example в первое поле странички Index (Индекс)). Например переменные GlobalRequest и GlobalResponse используются для передачи информации между процедурами, а АСCEPT-цикл используется для обработки действий пользователя на основе продвижения от поля к полю.

Генератор приложений добавит Ваш текст к генерируемому им тексту именно в той точке, которую Вы определили для его размещения.

Следующая программа была сгенерирована процедурным шаблоном Window для иллюстрации стандартных точек вставки. Каждая точка для возможных вставок отмечена комментарием (знак восклицания начинается комментарий), указывающим метку точки вставки в том виде, как она появляется в диалоговом окне Embedded Source.



Обратите внимание, что Вы можете при необходимости добавлять собственные точки вставки. См. #EMBED в Руководстве программиста или в справочной системе. Руководство программиста поставляется в сомтаве Professional Edition и Enterprise Edition Clarion for Windows.

ShowEmbeds      PROCEDURE

LocalRequest     LONG,AUTO

OriginalRequest   LONG,AUTO

LocalResponse    LONG,AUTO

WindowOpened     LONG

WindowInitialized LONG

ForceRefresh     LONG,AUTO

CurrentTab        STRING(80)

LastName          String(20)

! Embed Point: Data Section, Before Window Declaration

window            WINDOW('Caption'),AT(.,260,100),GRAY

ENTRY(@s20),AT(108,25),GRAY

BUTTON('Ok'),AT(130,72),USE(?Ok)

END

! Embed Point: Data Section, After Window Declaration

CODE

! Embed Point: Initialize the Procedure

LocalRequest = GlobalRequest

OriginalRequest = GlobalRequest

LocalResponse = RequestCancelled

ForceRefresh = False

CLEAR(GlobalRequest)

CLEAR(GlobalResponse)

! Embed Point: Procedure Setup

! Embed Point: Beginning of Procedure, Before Opening Files

! Embed Point: Beginning of Procedure, After Opening Files

! Embed Point: Before Opening the Window

OPEN(window)

WindowOpened=True

! Embed Point: After Opening the Window

! Embed Point: Preparing Window Alerts

! Embed Point: Preparing to Process the Window

ACCEPT

! Embed Point: Accept Loop, Before CASE EVENT() handling

CASE EVENT()

! Embed Point: CASE EVENT() structure, before generated code

OF EVENT:AlertKey

! Embed Point: Window Event Handling (AlertKey)

OF EVENT:PreAlertKey

```
! Embed Point: Window Event Handling (PreAlertKey)
OF EVENT:CloseWindow
! Embed Point: Window Event Handling (CloseWindow)
OF EVENT:CloseDown
! Embed Point: Window Event Handling (CloseDown)
OF EVENT:OpenWindow
! Embed Point: Window Event Handling (OpenWindow)
IF NOT WindowInitialized
    DO InitializeWindow
    WindowInitialized = True
END
SELECT(?LastName)
OF EVENT:LoseFocus
! Embed Point: Window Event Handling (LoseFocus)
OF EVENT:GainFocus
! Embed Point: Window Event Handling (GainFocus)
ForceRefresh = True
IF NOT WindowInitialized
    DO InitializeWindow
    WindowInitialized = True
ELSE
    DO RefreshWindow
END
OF EVENT:Suspend
! Embed Point: Window Event Handling (Suspend)
OF EVENT:Resume
! Embed Point: Window Event Handling (Resume)
ELSE
! Embed Point: Other Window Event Handling
! Embed Point: CASE EVENT() structure, after generated code
END
! Embed Point: Accept Loop, After CASE EVENT() handling
! Embed Point: Accept Loop, Before CASE FIELD() handling
CASE FIELD()
! Embed Point: CASE FIELD() structure, before generated code
OF ?LastName
! Embed Point: Control Handling, before event handling (?LastName)
CASE EVENT()
OF EVENT:Accepted
! Embed Point: Control Event Handling, before generated code
! Embed Point: Control Event Handling, after generated code
OF EVENT:Rejected
! Embed Point: Control Event Handling, before generated code
```

```

    ! Embed Point: Control Event Handling, after generated code
OF EVENT:Selected
    ! Embed Point: Control Event Handling, before generated code
    ! Embed Point: Control Event Handling, after generated code
ELSE
    ! Embed Point: Other Control Event Handling (?LastName)
END
! Embed Point: Control Handling, after event handling (?LastName)
OF ?Ok
    ! Embed Point: Control Handling, before event handling (?Ok)
CASE EVENT()
OF EVENT:Accepted
    ! Embed Point: Control Event Handling, before generated code
    Do SyncWindow
    ! Embed Point: Control Event Handling, after generated code
OF EVENT:Selected
    ! Embed Point: Control Event Handling, before generated code
    ! Embed Point: Control Event Handling, after generated code
ELSE
    ! Embed Point: Other Control Event Handling (?Ok)
END
    ! Embed Point: Control Handling, after event handling (?Ok)
    ! Embed Point: CASE FIELD() structure, after generated code
END
! Embed Point: Accept Loop, After CASE FIELD() handling
END
DO ProcedureReturn
!-----
ProcedureReturn ROUTINE
! Эта процедура обеспечивает общую точку выхода для всех
! сгенерированных шаблоном подпрограмм.
! Сначала нужно закрыть все открытые процедурой файлы.
! Затем должно быть закрыто окно, открытое в этой процедуре.
! Затем формируется значение GlobalResponse, для передачи вызывающей процедуре
! информации о характере завершения этой процедуры.
! И наконец возвращаем управление вызывающей процедуре.
! Embed Point: End of Procedure, Before Closing Files
! Embed Point: End of Procedure, After Closing Files
! Embed Point: Before Closing the Window
IF WindowOpened
    CLOSE(window)
END
! Embed Point: After Closing the Window

```

```

! Embed Point: End of Procedure
IF LocalResponse
  GlobalResponse = LocalResponse
ELSE
  GlobalResponse = RequestCancelled
END
RETURN

```

!-----

#### InitializeWindow ROUTINE

```

!|
!| Эта подпрограмма используется для подготовки к работе всех диалоговых элементов.
!| Она должна вызываться в процедуре только один раз.
!|
! Embed Point: Window Initialization Code
DO RefreshWindow

```

!-----

#### RefreshWindow ROUTINE

```

!|
!| Эта подпрограмма используется для актуализации всех диалоговых элементов.
!|
! IF window{Prop:AcceptAll} THEN EXIT.
! Embed Point: Refresh Window routine, before lookups
! Embed Point: Lookup Related Records
! Embed Point: Refresh Window routine, after lookups
! Embed Point: Refresh Window routine, before DISPLAY()
DISPLAY()
ForceRefresh = False

```

!-----

#### SyncWindow ROUTINE

```

!|
!| Эта подпрограмма используется перед вызовом прочеудр с помощью кнопок и меню
!| для извлечения всех записей, на которые указывают диалоговые элементы окна.
!|
! Embed Point: Sync Record routine, before lookups
! Embed Point: Lookup Related Records
! Embed Point: Sync Record routine, after lookups

```

!-----

! Embed Point: Procedure Routines

Встраивание исходного текста в процедуру

1. Нажмите кнопку Embeds (Вставки) в окне Procedure Properties для вызова диалогового окна

### **Embedded Source (Вставка исходного текста).**

Диалоговое окно Embedded Source перечисляет все точки в процедуре, в которые вы можете вставить свой собственный программный текст. Выберите вывод всех точек или лишь заполненных точек.

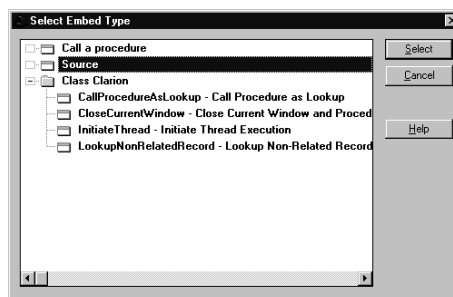
Совет: Для встраивания текста, связанного с определенным диалоговым элементом, откройте форматор окна, щелкните правой кнопкой по этому элементу, и выберите Embeds во всплывающем меню. В списке окажутся только те точки вставки, которые имеют отношение к выбранному диалоговому элементу.

1. Выберите точку, в которой нужно вставить программный текст, и нажмите кнопку Insert.

Появится диалоговое окно Select Embed Type. Имеется четыре способа создания вставок: ручное кодирование с помощью текстового редактора, вызов другой процедуры, вставка текстового шаблона или вставка текста на языке шаблонов. В каждой точке можно делать любое количество вставок любого типа.

☐ Для того чтобы вручную написать вставку с помощью текстового редактора:

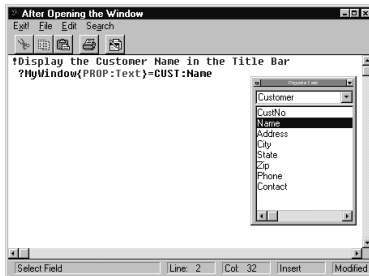
1. Выберите пункт SOURCE в диалоговом окне Select Embed Type (Выбор типа вставок).



Диалог Select Embed Type позволяет вам вызвать процедуру, добавить свой собственный исходный текст или использовать текстовый шаблон.

2. Нажмите кнопку Select (Выбрать) для вызова текстового редактора с пустым окном исходного текста.

Вызывается текстовый редактор. На экране присутствует плавающее инструментальное окно Populate Fields, в котором можно выбирать имена переменных и полей. Просто щелкните над полем, чтобы вставить его в позицию курсора.



3. Напишите текст своей программной вставки.

Совет: Не забудьте использовать интерактивную помощь для объяснений и примеров синтаксиса и техники языка Clarion. Копируйте и вставляйте прямо из примеров помощи!

4. Выберите Exit!

5. Выберите Yes в ответ на вопрос о необходимости сохранения вашего текста.

6. Нажмите кнопку Close для закрытия диалогового окна Embedded Source.

☐ Чтобы вызвать процедуру:

1. Выберите в диалоговом окне Select Embed Type, пункт Call a Procedure (Вызвать процедуру).

Для ввода имени вызываемой процедуры откроется диалог с названием точки вставки.



2. В поле Procedure to Call введите имя новой процедуры или выберите существующую процедуру из выпадающего списка

Ввод нового имени процедуры приводит к тому, что Генератор Приложений объявляет новую процедуру типа “To Do” и подключает ее к дереву приложения. Если процедура с таким именем уже существует, Генератор Приложений предполагает, что вы собираетесь вызвать ее и не добавляет новую “To Do”-процедуру.

Функциональные возможности новых процедур задаются отдельно. См. выше раздел Определение характеристик процедур.

3. Нажмите кнопку ОК , чтобы закрыть диалоговое окно.

☐ Использование текстового (Code) шаблона для создания текста точки вставки.

1. Выберите диалоговом окне Select Embed Type текстовый шаблон вставки и нажмите кнопку Select.

Текстовые шаблоны это пункты, объединенные в папки класса. См. описание включенных в поставку Clation текстовых шаблонов в главе Диалоговые, текстовые и распределенные шаблоны.

Это приведет к появлению диалогового поля Prompts for..(подсказки для..) (заголовок включает имя выбранного программного шаблона).

2. Прочитайте инструкции и объяснения в диалоге.

Каждый текстовый шаблон включает текст объяснения, как его использовать, и как заполнить необходимые поля ввода или установить необходимые режимы.

3. Заполните поля или выберите режимы в диалоговом окне Prompts for...

4. Нажмите кнопку ОК, чтобы закрыть окно

### **Управление вставками текста**

Диалоговое окно Embedded Source имеет кнопки ? и ? для изменения порядка следования вставок; исполнение происходит в том порядке, в котором они появляются в списке. Имеются также кнопки Delete и Properties и кнопки Вырезать, Копировать и вставить для работы со вставками.

☐ Чтобы вырезать (Cut) и вставить (Paste) (или скопировать и вставить) вставку исходного текста из одной точки вставки в другую:

1. В диалоговом окне Embedded Source (Вставки исходного текста) выделите строку в древовидной диаграмме.

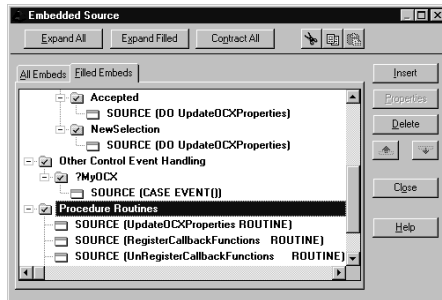
Выделение строки точки встраивания (пиктограмма папки) выбирает все вставленные в этой точке вставки исходные тексты для последующих операций вырезания и вставки. Выделение одной позиции вставки исходного текста выбирает только эту позицию.

2. Нажмите кнопки Cut (Вырезать) или Copy (копировать).

3. Снова выделите строку в иерархической диаграмме.

4. Нажмите кнопку Paste (Вставка).

Используйте кнопки вырезания, копирования и вставки для переноса одной или более позиций встраивания из одной точки вставки в другую.



## Формулы в процедуре

- Нажмите кнопку Formula в диалоговом окне Procedure Properties.

Это позволит Вам создать выражение и сохранить результаты в переменной памяти или в поле файла. Затем вы можете поместить результат в окно или документ.

Редактор формул может создавать простые операторы присваивания и сложные условные структуры. О подробностях процесса формирования выражений см. главу Использование редактора формул.

## Использование в процедуре распределенных шаблонов

- Нажмите кнопку Extensions в диалоговом окне Procedure Properties.

Здесь показаны как диалоговые, так и распределенные шаблоны процедуры. Эти шаблоны обеспечивают выполнение дополнительных функций в основных процедурных шаблонах. Диалоговые шаблоны позволяют создавать и управлять диалоговыми элементами. Например, с помощью диалогового шаблона может быть добавлен список для просмотра файла.

Распределенные шаблоны обеспечивают выполнение дополнительных функций, которые не привязаны к какому либо диалоговому элементу. Например с помощью распределенного шаблона может быть обеспечен вывод даты и времени. Дополнительная информация об этих шаблонах приводится в главе Диалоговые, текстовые и распределенные шаблоны.



## Прототипы и передача параметров

При вызове процедур и функций может возникнуть необходимость в передаче параметров, возврате значений или в том и другом. Передачу параметров и возврат значений можно указать (прототипировать) в диалоге *Procfdure Properties*.

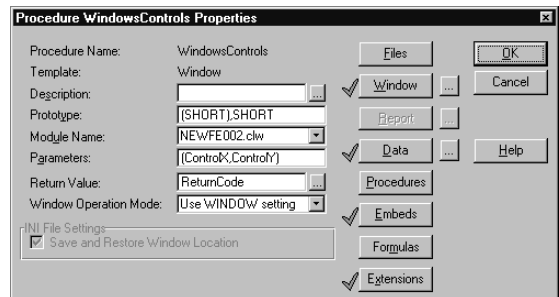
**Замечание:** В языке Clarion единственное различие между процедурой и функцией состоит в том, что функция возвращает значение, а процедура не возвращает.

Для передачи параметров в процедуру или функцию необходимо сделать следующее.

1. Добавить в структуре MAP программы к прототипу процедуры список типов передаваемых параметров.
2. Добавить в операторах PROCEDURE или FUNCTION список имен передаваемых параметров.
3. Передать параметры при вызове процедуры или функции.

## Добавление параметров к прототипу процедуры

Для переопределения оператора прототипа, который генерируется в MAP структуре программы, используется поле *Prtototipe* в диалоге *Procfdure Properties*. Прототип объявляет все, что должна знать вызывающая программа для вызова процедуры или функции, включая типы данных всех параметров. См. Прототипы процедур в Руководстве по языку.



Например, для генерации следующего прототипа в MAP структуре

```
MAP
...
MODULE('NEWFE002.CLW')
  WindowsControls(SHORT,SHORT),SHORT
END
...
END
```

напечатайте в поле *Prtototipe* (SHORT,SHORT),SHORT.

Обратите внимание, что весь текст из поля Prototype, включая скобки, добавлен к прототипу процедуры. Слова в скобках определяют типы передаваемых процедуре или функции параметров. Слово, которое стоит после скобок определяет тип возвращаемого функцией значения. См. Прототипы процедур в Руководстве по языку. Пример см. в Элементы диалога и их свойства - Элементы OLE с OCX.

## **Добавление параметров к оператору PROCEDURE или FUNCTION**

Напечатайте (ControlX,ControlY) в поле Parametrс диалога Procedure Properties, чтобы получить следующий оператор процедуры:

```
WindowsControls PROCEDURE(ControlX,ControlY)
...
```

Снова весь текст из поля Parametrс, включая скобки, добавляется к оператору PROCEDURE или FUNCTION.

Нажмите кнопку с многоточием (...) около поля Return Value в диалоге Procedure Properties, чтобы выбрать или определить переменную, содержащую возвращаемое функцией значение и сгенерировать следующий программный текст. Обратите внимание, что из-за наличия возвращаемого значения слово

PROCEDURE было заменено на FUNCTION. Обратите также внимание, что сгенерированная теперь подпрограмма ProcedureReturn возвращает значение, указанное в поле Return Value: ReturnCode.

```
WindowsControls FUNCTION(ControlX,ControlY)
...
CODE
...
ProcedureReturn ROUTINE
  IF WindowOpened
    CLOSE(window)
  END
  IF LocalResponse
    GlobalResponse = LocalResponse
  ELSE
    GlobalResponse = RequestCancelled
  END
POPBIND
RETURN(ReturnCode)
```

Совет: Вы должны добавить встраиваемый текст для присваивания подходящего значения переменной возврата.

Пример см. в Элементы диалога и их свойства - Элементы OLE с OCX.

## Вызов процедур и функций с параметрами

Используйте для передачи параметров в вызываемую процедуру или функцию страничку Actions свойств вызывающего диалогового элемента (пункт меню, кнопка, и т.д.).

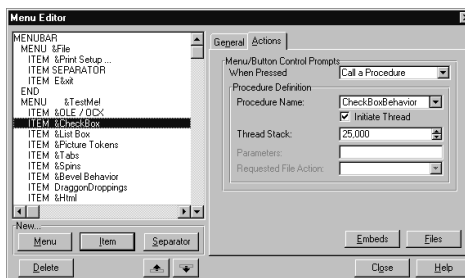
### Передача параметров в процедуру

1. Откройте страничку Actions свойств вызывающего диалогового элемента.
2. В выпадающем списке When Pressed (При нажатии) выберите Call a Procedure (Вызов процедуры).
3. В выпадающем списке Procedure Name (Имя процедуры) выберите имя вызываемой процедуры.

Если Вы еще не определили эту процедуру, впечатайте ее имя. Новую процедуру можно определить позже.

Совет: В операторе START нельзя передавать параметры процедурам и функциям, поэтому не следует отмечать Initiate Thread, если нужно передать параметры. Отметка у Initiate Thread сгенерирует оператор START для запуска нового процесса.

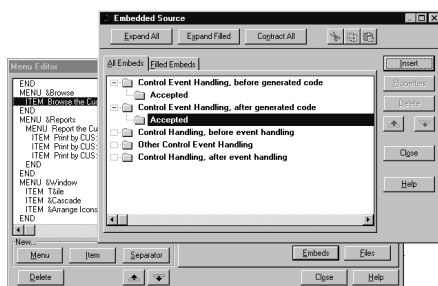
4. В поле Parametrs напечатайте через запятую список передаваемых параметров. Параметр может быть литералом, выражением или именем переменной.



## Получение возвращаемого функцией значения

Хотя функцию можно вызвать с помощью Call a Procedure со странички Actions, такой метод не позволит получить возвращаемое значение. Поэтому для получения возвращаемого значения следует использовать вставку исходного текста. Далее показан один из возможных способов вызова функции с помощью диалогового элемента, однако функцию можно вызвать множеством способов.

1. Откройте страничку Actions свойств вызывающего диалогового элемента.
2. В выпадающем списке When Pressed (При нажатии) выберите No Special Action (Никаких специальных действий).
3. Нажмите кнопку Embeds.
4. В диалоге Embedded Source выберите точку Accepted для Control Event Handling, after generated code и нажмите Insert (добавить).
5. В диалоге Select EmbedType выберите SOURCE.
6. В текстовом редакторе запишите вызов функции, например так:  
`ReturnCode = WindowsControls(0,ControlY)`



7. Выберите в меню Exit!, и сохраните вставку.

## Работа с деревом приложения

В диалоговом окне Application Tree (Дерево приложения) Clarion предоставляет вам меню и закладки для обслуживания дерева, а также для обеспечения различных представлений дерева. Меню, используемые для обслуживания дерева, - это меню Edit (Редактирование), меню Application (Приложение) и меню Procedure (Процедура).

### Использование меню Edit

---

При активном диалоге Application Tree вы можете выполнить следующие команды из меню Edit:

- |  |  |
|--|--|
| Properties (свойства)                            | Вызывает диалоговое окно Procedure Properties для выбранной процедуры. Эквивалент кнопки Properties.   |
| Window (Окно)                                    | Вызывает для выбранной процедуры форматер окна. Эквивалент кнопки Window в диалоге Procedure Properties.   |
| Report (Документ)                                | Вызывает для выбранной процедуры форматер документа. Эквивалент кнопки Report в диалоге Procedure Properties.  |
| Data (Данные)                                    | Вызывает для выбранной процедуры диалог Local Data. Эквивалент кнопки Data в диалоге Procedure Properties.   |
| Embeds (Вставки текста)                          | Вызывает для выбранной процедуры диалог Embedded Source. Эквивалент кнопки Embeds в диалоге Procedure Properties.  |
| Extensions (Диалоговые и распределенные шаблоны) | Вызывает для выбранной процедуры диалог Extension and Control Templates, который позволяет добавлять к процедуре распределенные шаблоны и производить настройку параметров диалоговых и распределенных шаблонов. Эквивалент кнопки Extensions в диалоге Procedure Properties. См. дополнительную информацию об этих шаблонах в главе Диалоговые, текстовые и распределенные шаблоны.   |
| Source (Исходный текст)                          | Открывает файл исходного текста процедуры, при условии, что была проведена генерация исходных текстов. Любые изменения, которые Вы внесете здесь в исходные тексты будут уничтожены при очередной генерации текстов приложения, однако иногда бывает чрезвычайно полезно взглянуть на текст в целом, и обычно бывает полезно в качестве эксперимента вручную изменить исходный текст, прежде чем внести постоянные изменения с помощью инструментов Генератора Приложений. |
| Find (найти)                                     | Позволяет вам искать процедуру по имени. Это может быть очень полезно в большом приложении с десятками процедур. Наберите строку символов для поиска в диалоговом окне Search for Procedure (поиск процедуры).   |
| Find next (найти следующую)                      | Позволяет вам искать следующее вхождение   |

процедуры, используя ту же самую поисковую строку, как и в предыдущем поиске. Если вы ранее не проводили поиск, появится диалоговое окно Search for Procedure.

Edit by Name

(редактирование по имени) Позволяет вызывать диалоговое окно Procedure Properties через окно Edit Procedure by Name (редактировать процедуру по имени), в котором необходимо ввести имя редактируемой процедуры. Это может быть очень полезно в большом приложении со многими процедурами.

Delete (уничтожить)

Удаляет код, характеристики и т.д. текущей выбранной процедуры. Процедура остается позицией ToDo в дереве приложения, так как процедура еще вызывается другой процедурой. Чтобы удалить процедуру из дерева приложения, вы должны удалить все ссылки на эту процедуру.

## Использование меню Application

При активированном диалоговом окне Application Tree (Дерево приложения) вы можете выполнить следующие команды из меню Application:

Properties (Свойства приложения)

Вызывает диалоговое окно Application Properties для изменения установок файла .APP.

Global Properties

(Глобальные свойства)

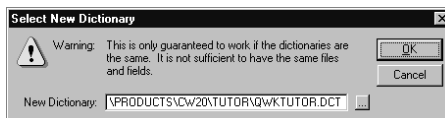
Вызывает диалоговое окно Application Properties. Эквивалентно использованию кнопки Global в окне Procedure Properties.

Change Dictionary

(сменить словарь)

Позволяет использовать новый словарь данных для приложения. Наберите имя файла в диалоговом окне Select New Dictionary или нажмите эллиптическую кнопку(...) для выбора нового файла словаря из окна открытия файлов Open File.

Если ваша процедура уже ссылается на поля данных в одном словаре, а структурный префикс ФАЙЛА и имена полей ЗАПИСИ в новом словаре те же, тогда диалоговое окно Select New Dictionary дает предупреждающее сообщение.



Insert Module

(вставить модуль)

Определяет новый MODULE для генерируемой исходной программы. Вы можете также задать внешний .LIB или .OBJ

файл, чтобы добавить их к проекту.

### Template Utility

(Утилита шаблона) Дает вам возможность работать с любыми шаблонами утилит, которые зарегистрированы. Вы можете написать свои собственные утилиты или купить их у продавцов программного обеспечения. Используйте эту команду для старта любых Мастеров Clarion.

### Redistribute Procedures

(Перераспределить процедуры) Перераспределяет процедуры среди модулей в порядке, в котором они уже появляются. Число процедур, содержащихся в каждом модуле, определяется назначаемыми вами Procedures Per Module (Процедур на модуль). Эта утилита сразу действует на дерево приложения, однако сами файлы модуля исходного текста не изменятся до тех пор, как исходный текст не будет генерирован в следующий раз.

### Repopulate Modules

(Перенаполнить модули) Перераспределяет процедуры по модулям, пытаясь разместить в одном модуле процедуры, которые вызывают друг друга. Число процедур, содержащихся в каждом модуле, определяется назначаемыми вами Procedures Per Module (Процедур на модуль). Эта утилита сразу действует на дерево приложения, однако сами файлы модуля исходного текста не изменятся до тех пор, как исходный текст не будет генерирован в следующий раз.

### Renumber Modules

(Перенумеровка модулей) Автоматически перенумеровывает модули. Это полезно тогда, когда были удалены пустые модули. Обратите, пожалуйста, внимание, что эта позиция сразу влияет на дерево приложения, однако сами файлы модуля исходного текста не изменятся до тех пор, как исходный текст не будет генерирован в следующий раз.

### Delete Empty Modules

(Удалить пустые модули) Удаляет из дерева приложения модули, которые опустели в результате изменений или удалений приложения. Эта позиция меню сразу изменяет дерево приложения, однако файлы модулей на вашем дисковом не удаляются.

### Delete Empty Libraries

(Удалить пустые библиотеки) Удаляет те библиотеки из дерева приложения, которые стали пустыми в результате изменений и удалений приложения. Эта опция меню сразу изменяет дерево приложения, однако файлы библиотеки на вашем дисковом не удаляются.

## Использование меню Procedure

При активном диалоговом окне Application Tree (Дерево приложения) вы можете выполнить следующие команды из меню Procedure (Процедура):

**New (добавить процедуру)** Добавляет процедуру, не связанную с деревом процедур. То же самое делает клавиша Insert.

**Synchronize (синхронизация)** Применяет стандартные спецификации диалоговых элементов, указанные в Словаре Данных, включая Screen Picture, Prompt, Heading, Case, Typing Mode, Flags, Justification, Initial Value, Help Ids, Messages, Tool Tips, Validity Checks и т. д. ко всем диалоговым элементам процедуры, за исключением следующего.

Не изменяется тип элемента и его положение. Размеры элемента могут изменяться, если установлены в "Default". Пустые атрибуты словаря не замещают заполненные атрибуты поля (т.е. Tool Tips). Элементы, размещенные диалоговыми шаблонами сохраняют атрибуты, заданные шаблонами. Явно замороженные элементы не изменяются.

**Rename (переименовать процедуру)** Позволяет изменить имя выбранной процедуры. Наберите новое имя в диалоге Rename.

**Copy (Копировать)** Дает вам возможность копировать текущую выбранную процедуру. Наберите новое имя в диалоговом поле New Procedure (Новая процедура).

**Synchronize (синхронизация)** Применяет стандартные спецификации диалоговых элементов, указанные в Словаре Данных, включая Screen Picture, Prompt, Heading, Case, Typing Mode, Flags, Justification, Initial Value, Help Ids, Messages, Tool Tips, Validity Checks и т. д. ко всем диалоговым элементам процедуры, за исключением следующего.

Не изменяется тип элемента и его положение. Размеры элемента могут изменяться, если установлены в "Default". Пустые атрибуты словаря не замещают заполненные атрибуты поля (т.е. Tool Tips). Элементы, размещенные диалоговыми шаблонами сохраняют атрибуты, заданные шаблонами. Явно замороженные элементы не изменяются. О замораживании элементов см. Диалоговые элементы и их свойства - Общие свойства диалоговых элементов.

**Change Module (изменить модуль)** Позволяет передвигать выбранную процедуру из одного модуля в другой. Место назначения вводится в диалоговом



окне Select Destination Module.

Ваше приложение может выполняться немного быстрее, если вы сгруппируете процедуры, которые обычно исполняют вместе, в один и тот же модуль.

Change Template

(изменить шаблон) Позволяет изменить тип процедуры для выбранной процедуры. Выберите новый шаблон процедуры в диалоговом окне Select Procedure Type.

## Использование страниц просмотра

---

Диалоговое окно Application Tree (Дерево приложения) дает вам возможность расположить ваши процедуры в четырех видах упорядочивания. Тип упорядочивания вы можете изменить выбирая нужную страничку.

Procedure (по процедурам) Показывает процедуры в логическом порядке, вставляя каждую процедуру под вызывающей ее процедурой. Это просмотр, принятый по умолчанию и в комбинации с легендами “To Do” для неопределенных еще процедур это лучший просмотр для определения того, что еще остается сделать для завершения приложения.

Module (по модулям) Каждая процедура лежит под именем файла модуля, в который помещается сгенерированный для этой процедуры текст.

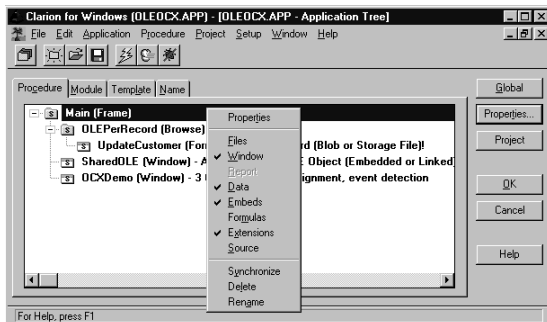
Совет: Эта страничка представляет единственное место, откуда можно попасть в точки вставки текста на уровне модуля и в диалог Default Program Properties (стандартные свойства программы)

Template (по шаблонам) Перечисляет процедуры в соответствии с типом шаблона процедуры.

Name (по имени) Дает отсортированный по алфавиту список процедур. Это бывает иногда удобно для больших проектов.

## Всплывающее меню

В диалоговом окне Application Tree (Дерево приложения) щелчок правой кнопкой мыши на любой процедуре дает доступ к следующим командам. Это всплывающее меню обеспечивает быстрый доступ к свойствам процедуры.



**Properties (свойства)** Вызывает диалоговое окно Procedure Properties для выбранной процедуры. Эквивалент кнопки Properties.

**Files (файлы)** Вызывает диалоговое окно File Schematic Definition для выбранной процедуры. Эквивалент кнопки Files в диалоге Procedure Properties..

**Window (Окно)** Вызывает для выбранной процедуры форматер окна. Эквивалент кнопки Window в диалоге Procedure Properties.

**Report (Документ)** Вызывает для выбранной процедуры форматер документа. Эквивалент кнопки Report в диалоге Procedure Properties.

**Data (Данные)** Вызывает для выбранной процедуры диалог Local Data. Эквивалент кнопки Data в диалоге Procedure Properties.

**Embeds (Вставки текста)** Вызывает для выбранной процедуры диалог Embedded Source. Эквивалент кнопки Embeds в диалоге Procedure Properties.

**Formulas (Формулы)** Вызывает редактор формул для выбранной процедуры. Эквивалент кнопки Formulas в диалоге Procedure Properties.

**Extensions (Диалоговые и распределенные шаблоны)** Вызывает для выбранной процедуры диалог Extension and Control Templates, который позволяет добавлять к процедуре распределенные шаблоны и производить настройку параметров диалоговых и распределенных шаблонов. Эквивалент кнопки Extensions в диалоге Procedure Properties. См. дополнительную информацию об этих шаблонах в главе Диалоговые, текстовые и распределенные шаблоны.

**Source (Исходный текст)** Открывает файл исходного текста процедуры, при условии, что была проведена генерация исходных текстов. Любые изменения, которые Вы внесете здесь в исходные тексты будут уничтожены при очередной генерации текстов приложения,

однако иногда бывает чрезвычайно полезно взглянуть на текст в целом, и обычно бывает полезно в качестве эксперимента вручную изменить исходный текст, прежде чем внести постоянные изменения с помощью инструментов Генератора Приложений.

**Synchronize (синхронизация)** Применяет стандартные спецификации диалоговых элементов, указанные в Словаре Данных, включая Screen Picture, Prompt, Heading, Case, Typing Mode, Flags, Justification, Initial Value, Help Ids, Messages, Tool Tips, Validity Checks и т. д. ко всем диалоговым элементам процедуры, за исключением следующего.

Не изменяется тип элемента и его положение. Размеры элемента могут изменяться, если установлены в “Default”. Пустые атрибуты словаря не замещают заполненные атрибуты поля (т.е. Tool Tips). Элементы, размещенные диалоговыми шаблонами сохраняют атрибуты, заданные шаблонами. Явно замороженные элементы не изменяются. О замораживании элементов см. Диалоговые элементы и их свойства - Общие свойства диалоговых элементов.

**Delete (уничтожить)** Уничтожает выбранную процедуру.

**Rename (переименовать процедуру)** Позволяет изменить имя выбранной процедуры. Наберите новое имя в диалоге Rename.

## Обслуживание ваших шаблонов

Файлы шаблонов (\*.TPL) управляют генератором приложений. Каждый шаблон процедуры содержит родовой или “модельный” код. Шаблоны интерактивны - они обрабатывают информацию, определяемую вами, когда вы проектируете приложение в интегрированной среде разработки. Clarion обращается к файлу шаблонов дважды:

Прежде чем создавать ваше приложение Clarion предобрабатывает файл шаблонов и запоминает информацию в файле REGISTRY.TRF. Предобработка происходит только тогда, когда генератор приложений выявляет новый или измененный шаблон.

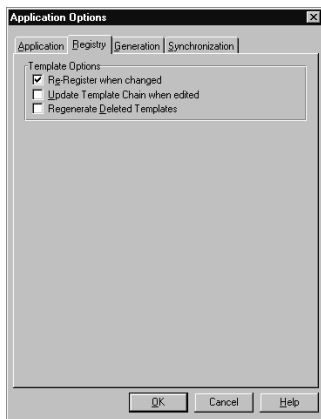
- Когда генератор приложений предобрабатывает файл шаблонов, он составляет список всей той информации, которой вы должны обеспечить каждую из процедур. Он также определяет точки, в которых вы сможете сделать ваши вставки программного текста для окончательной настройки процедур.

Во время генерации программы Генератор Приложений использует информацию, представленную Вами в процессе проектирования, информацию из словаря данных и файла .APP, а затем обрабатывает ее вместе с операторами на языке шаблонов и символами в REGISTRY.TRF файле для генерации текста вашей программы.

- Каждый набор шаблонов может содержать множество шаблонов процедур, которые вы используете для создания процедур в вашем приложении. Прежде чем вы сможете использовать шаблон, он должен быть помещен в регистр шаблонов.

## Настройка регистра шаблонов

Для установки опций регистра шаблонов выберите Setup ? Application Options. Появится диалоговое окно Generator Options. Выберите страничку (tab) Registry.



Программный код шаблона может быть разделен на многие файлы, обычно это .TPW и .TPL. Clarion использует эти файлы для получения одного логического набора шаблонов (REGISTRY.TRF) предназначенного для создания приложений. Рассматривайте файлы .TPW и .TPL как источники или резервные копии ваших шаблонов, а файл REGISTRY.TRF как вашу рабочей копии.

Страничка Registry позволяет вам установить, каким образом файлы языка шаблонов и один логический набор шаблонов должны будут управлять генерацией вашего приложения. Эти опции рассчитаны главным образом на программистов, которые создают свои собственные файлы шаблонов или модифицируют шаблоны принятые по умолчанию.

☐ Чтобы ваши шаблоны автоматически перерегистрировались, когда генератор приложений обнаруживает изменение в шаблонах, т.е. когда файл .TPW или .TPL изменяется и вы хотите, чтобы изменение было скопировано или зарегистрировано в файл REGISTRY.TRF, отметьте поле флажков Re-Register when changed (перерегистрировать если изменено).

☐ Чтобы файлы шаблонов обновлялись автоматически при внесении изменений в Template Registry, т.е. если вы хотите, чтобы изменения в файле REGISTRY.TRF отражались в файлах .TPW или .TPL, установите флажок в поле Update Template Chain when changed (обновить цепь шаблонов, если есть изменения).

☐ Чтобы определить, что генератор приложений должен регенерировать файлы .TPL и .TPW из REGISTRY.TRE, в случае если файлы должны быть удалены, установите флажок в поле Regenerate Deleted Templates (регенерировать удаленные шаблоны).

## Открытие Регистра шаблонов

---

Регистр шаблонов содержит список всех шаблонов (и их процедур), доступных вам при построении приложения. Для создания приложения вы должны иметь по крайней мере один зарегистрированный набор (класс) шаблонов.

Хотя шаблоны стандартного класса Clarion (CW.TPL) зарегистрированы заранее, если вы должны перерегистрировать его, или если вы хотите зарегистрировать чужой набор шаблонов, вы можете сделать это следующим образом:

### 1. Выберите Setup ? Template Registry.

Появится диалоговое окно Template Registry, которое содержит дерево ваших шаблонов и связанных в них процедур, а также группу командных кнопок для управления регистром.

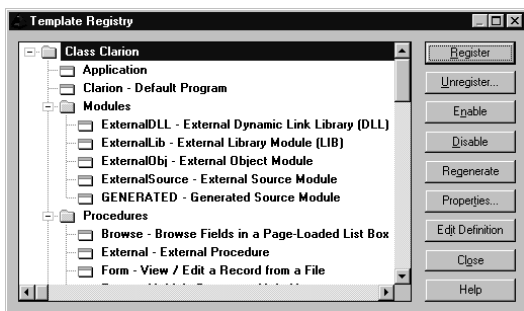
### 2. Нажмите кнопку Register.

Появится диалоговое окно Template File. Это окно позволяет вам выбрать шаблоны для регистрации. Оно содержит список файлов типа - \*.TPL (List Files of Type) . По умолчанию подкаталог или папка, содержащий шаблоны, имеет имя "C:\CW20\TEMPLATE".

### 3. Выберите файл CW.TPL (или чужой файл .TPL) и нажмите кнопку ОК.

В результате вы регистрируете набор шаблонов (класс), делая их тем самым доступными для использования.

## Обслуживание регистра шаблонов



Диалоговое окно Template Registry предоставляет набор командных кнопок обеспечивающих управление регистром:

- |                            |   |
|----------------------------|---|
| Unregister (дерегистрация) | Эта кнопка удаляет высвеченный в данный момент класс (набор) шаблонов из REGISTRY.TRF   |
| Enable                     | Эта кнопка делает доступными выделенный в данный момент класс (набор) шаблонов или шаблонную процедуру (если вы ранее выполнили для них команду disable). |
| Disable                    | Эта кнопка делает недоступными для вашего приложения выделенный класс шаблонов или шаблонную процедуру.   |
| Regenerate                 | Эта кнопка регенерирует файл .TPL для выделенного в данный момент класса шаблонов.  |

Регистр шаблонов работает двояким образом. Помимо того, что он позволяет вам добавить шаблоны и процедуры к выбранной форме, вы можете также произвести настройку ваших шаблонов. Инструкции по редактированию шаблона в описании кнопки Properties.

Когда вы изменяете свойства вашего шаблона, все изменения шаблона запоминаются в файле REGISTRY.TRF эффективно устанавливая новые величины по умолчанию для вашего приложения. При нажатии кнопки ReGeneration генератор приложений читает REGISTRY.TRF файл и затем переписывает файл .TPL, который изначально содержал первоначальный текст шаблона. Это позволяет вам настроить шаблон для ваших конкретных нужд и затем передать копию измененного файла .TPL другому программисту для регистрации.

- |  |   |
|--|---|
| Properties (Свойства)                        | Эта кнопка открывает доступ к диалоговому окну Template Procedure Properties. Нажмите кнопку Global Data для редактирования содержащихся в этом шаблоне глобальных данных, принятых по умолчанию. |
| Edit Definition (редактирование определения) | Эта кнопка открывает текст шаблона (.TPL) в текстовом редакторе.  |

Если отмеченная в данный момент позиция в дереве регистра шаблонов является модулем, текстовое окно открывается на первой строке кода на языке шаблона для #MODULE. Если текущая выделенная позиция в дереве регистра шаблона - процедура, оно открывается на первой строке кода на языке шаблона для #PROCEDURE. Более подробную информацию о формате и синтаксисе языка шаблона вы можете найти в Справочнике языка шаблонов.

Когда Вы следующий раз откроете приложение все сделанные изменения будут перерегистрированы или нет в соответствии с настройками Регистра в диалоге Application Options.

Открывает файл контекстной помощи для шаблонов.

Закрывает диалоговое окно Template Registry.

Close (заккрыть)

## Команды импорта / экспорта приложений

Когда генератор приложений активен, меню File содержит дополнительные команды для присоединения процедур из других приложений. Процесс экспорта управляется путем использования формата экспортного файла, предназначенного помочь присоединить процедуры, написанные с помощью других версий Clarion'a. Процесс импорта может непосредственно обрабатывать формат файла .APP, в котором генератор приложений хранит создаваемое приложение.

В меню File появляются следующие команды :

**Import from App (импорт из APP)** Позволяет выбрать файл .APP из диалогового окна Select Application to Import From (выбор приложения для импорта из). После того, как вы сделали свой выбор и нажали кнопку ОК, генератор приложений добавляет процедуры из выбранного файла .APP в дерево приложения.



**Внимание:** Убедитесь перед импортированием, что имеются резервные копии исходного и целевого приложений. Clarion for Windows 2.0 в процессе конвертации преобразует исходный файл в формат 2.0. Будучи конвертированными в формат 2.0 файлы исходного текста не будут более пригодны для использования ранними версиями Clarion for Windows.

**Import Text (импортирование текста)** Импортирует процедуры, определенные в файле .TXA (ASCII версия файла приложения .APP), созданного с помощью команды Export Text (экспорт текста) (см. дальше). Вы получите приглашение переименовать или переместить процедуры с конфликтами имен.

**Export Text (экспорт текста)** Позволяет создать из данного приложения текстовый файл типа .TXA (ASCII версия вашего файла .APP).

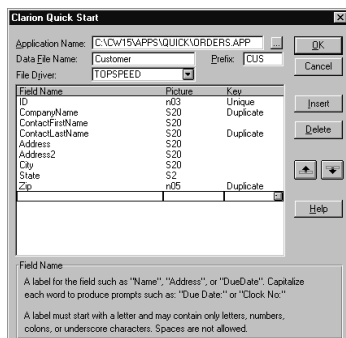
**Selective Export (селективный экспорт)** Позволяет создать текстовый файл типа .TXA, содержащий только выбранную процедуру.



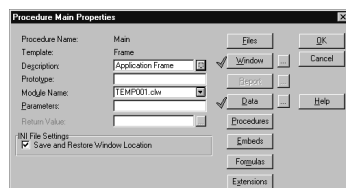
## Глава 6 Мастера и шаблоны процедур



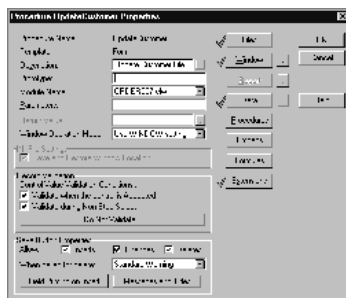
В этой главе Вы найдете руководство по использованию Мастеров процедур и шаблонов процедур.



Мастер быстрого старта является одним из тех инструментов, которые превращают Быструю разработку приложений в действительно быструю.



Шаблон Frame обеспечивает каркас для приложений, использующих многодокументный интерфейс.



Шаблон просмотра обеспечивает заранее определенное окно с прокручиваемым списком и кнопками обновления данных для вызова процедуры Form (обновление).



Шаблон отчета позволяет вам создавать формы отчета, используя Форматер отчета и включить их в ваше дерево процедур.

**Мастера (WIZARDS)****Мастер быстрого старта****Мастер приложения****Мастер процедуры просмотра****Мастер формата****Мастер отчета****Использование опций мастера****Шаблоны процедур****Компонентно-ориентированные шаблоны****Шаблон Window****Шаблон Frame****Шаблон Menu****Шаблон исходной программы Source****Шаблон процесс (Process)****Шаблон External****Шаблон Browse****Шаблон Form****Шаблон Report****Шаблон Viewer**

## WIZARDS (МАСТЕРА)

Clarion for Windows дает разработчикам WIZARDS (МАСТЕРА) - мощные шаблоны утилит, которые дают вам возможность создать процедуры Просмотр, Форма или Отчет путем простого ответа на несколько кратких вопросов. Вы даже можете с помощью мастера создать полное приложение из какого-то существующего словаря!

Опции, которые вы заранее устанавливаете в словаре данных, обеспечивают дополнительное управление процедурами, которые создают мастера. Более подробная информация имеется в Оптимизации для Мастеров.

### Мастер быстрого старта

---

С помощью мастера быстрого старта вы можете создать словарь данных и рабочее приложение без необходимости кодирования.

Просто определите файл данных - и мастер быстрого старта создает полное приложение Windows. Весь процесс занимает менее пяти минут! Ваше приложение будет содержать процедуру форма для обновления файла, процедуру просмотра с многими ключами и столько отчетов, сколько ключей имеет файл данных.

Вам нужно лишь определить поля для одиночного файла. Для каждого поля вы обеспечиваете имя, шаблон его изображения и информацию о ключе. Это образует словарь данных. Мастер быстрого старта создает приложение, основанное на этом словаре. Как только вы назначили все опции, кнопка ОК генерирует файл .APP и загружает процедуры в диалоговое окно Дерево приложения.

Чтобы использовать Quick Start Wizard (Мастер быстрого старта):

1. Если необходимо, выберите в диспетчере файлов File>Create Directory или в Windows 95 Explorer выберите File >New>Folder, наберите имя и нажмите ОК.

2. В Clarion for Windows выберите File> New.

Появится диалоговое окно файла New.

3. Выберите закладку (tab) Application.

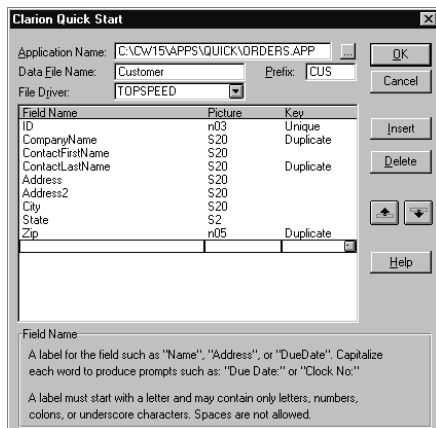
4. Наберите имя для файла .APP в поле Application Field.

Наберите допустимое DOS-имя файла. Clarion автоматически добавляет расширение .APP.

5. Отметьте поле Use Quick Start Wizard (Использование мастера быстрого старта) под

списком файлов, затем нажмите кнопку Create (Создать).

Откроется диалоговое окно Quick Start Wizard. Это диалоговое окно даст вам возможность определить файл для приложения.



## Application Name

(Имя приложения) DOS-имя файла для файла .APP. Мастер быстрого старта будет использовать то же самое имя файла (с расширением .DCT) для файла словаря данных.

Если необходимо, нажмите эллиптическую кнопку (...) для изменения каталога и наберите имя файла в диалоговом поле Open File. Рабочий каталог, в котором будут генерированы все файлы исходного текста, зависит от того, где находится файл .APP.

## Data File Name

(Имя файла данных) Наберите допустимое DOS-имя файла (расширение не нужно) для файла данных.

Prefix (Префикс) Когда вы по клавише tab выходите из поля Data File Name (Имя файла данных), это поле автоматически заполняется первыми тремя буквами имени файла данных. Если необходимо, наберите в этом поле собственный префикс (до 14 букв).

Префикс позволяет вашему приложению делать различие между одинаковыми именами переменных, встречающимися в различных файловых структурах. Поле, называемое Счет, может существовать в одном файле данных, называемом Заказы, а также в другом, называемом Продажи. Путем установления уникального префикса для Заказов (ORD) и Продаж (SAL) приложение может различать два поля как ORD:счет и SAL:счет.

File Driver (Драйвер файла) Установите тип файла данных. При использовании генератора приложений Clarion for Windows автоматически компонует правильную библиотеку драйверов файлов базы данных. Обсуждение относительных

преимуществ каждого драйвера вы найдете в приложении Драйверы базы данных.

Помните, что отдельные драйверы файлов могут отличаться в поддержке некоторых атрибутов, которые вы добавляете к структуре FILE в этом диалоговом поле.

**Field Name (Имя поля)** Чтобы поименовать каждое поле, наберите допустимую метку Clariion в поле Name. Правильные имена файла могут слегка отличаться в зависимости от драйвера файла.

**Picture (Шаблон изображения)** Определите в поле Picture (Шаблон изображения) шаблон изображения принимаемый по умолчанию. Шаблон изображения, наряду с выбранным драйвером файла, определяет тип данных, которые Мастер быстрого старта использует для данного поля. Когда генератор поля создает элементы управления окна и отчета для данного поля, данный шаблон определяет вид в котором содержимое этого поля появляется на экране или в отчете

**.Key (Ключ)** Эта позиция определяет, нужно ли создавать ключ используя это поле как компонент и, если да, то какой тип ключа. Назначая Unique (Уникальный), вашему приложению будет обеспечено, что каждая запись будет иметь уникальное значение. Duplicate (дублирование) устанавливает ключ, который допускает более одной записи с той же самой величиной в компоненте ключа.

Мастер быстрого старта создает процедуру просмотра с многими ключами и отчеты для каждого назначаемого вами ключа.

Мастер быстрого старта дает вам возможность поименовать каждое поле, одно за другим, путем нажатия “стрелки вниз” в для добавления новой позиции к списку.

Нажмите “стрелку вниз” в или клавишу tab для перехода к следующему полю.

**Insert (Вставить)** Эта кнопка дает вам возможность вставить новое пустое поле поверх текущего выбранного поля.

**Delete (Удалить)** Эта кнопка дает вам возможность удалить текущее выбранное поле.

**Move Up (Передвинуть вверх)** Эта кнопка дает вам возможность передвинуть текущее выбранное поле на одну позицию вверх в списке полей.

**Move Down (Передвинуть вниз)** Эта кнопка дает вам возможность передвинуть текущее выбранное поле на одну позицию вниз в списке полей.

6. После того, как вы определили все нужные вам поля, нажмите кнопку ОК.

Мастер быстрого старта создаст ваше приложение и покажет дерево приложения.

## Мастер приложения

Этот мастер создает полное приложение из существующего словаря. Он создает главную процедуру, содержащую меню с позициями, вызывающими все подчиненные процедуры, которые он создает. Он также создает процедуры “Просмотр”, “Отчет” и “Форма” (обновление) для каждого назначенного файла.

Для систем файлов на основе SQL Мастер приложения также генерирует код для подключения пользователя к SQL системе при первоначальной загрузке программы, затем снова использует информацию о пользователе для каждого файла, к которому получен доступ.

Чтобы использовать Мастер приложения:

1. Если необходимо в диспетчере файлов выберите File>Create Directory или в Windows 95 Explorer выберите File>New>Folder, наберите имя и нажмите ОК.

2. В Clarion for Windows выберите File>New.  
Появится диалоговое окно New.

3. Выберите закладку (tab) Application.

4. Наберите имя для файла .APP в поле Application File (Файл приложения). Не пользуйтесь Quick Start Wizard (Мастер быстрого старта), очистите поле флажков под списком файлов (см. Использование Мастера быстрого старта).

Наберите допустимое DOS-имя файла. Clarion автоматически добавляет расширение .APP.

Появится диалоговое поле Application Properties (Свойства приложения). Этот диалог дает вам возможность определить существенные для приложения файлы.

5. Введите имя .DCT-файла приложения, которое приложение будет использовать в поле Dictionary File (Файл словаря) или нажмите эллиптическую кнопку (...) для того, чтобы выбрать файл в диалоговом окне Select Dictionary (Выбрать словарь).

6. Если необходимо переименуйте процедуру First (первая) или примите умолчание - Main (Главная).

Мастер приложения (Application Wizard) будет использовать имя, которое вы дали, при создании начальной процедуры.

7. Выберите Destination Type (Тип адресата) из выпадающего списка.

Это определяет тип объектного файла для вашего приложения. Выберите из Executable (.EXE), Library (.LIB) или Dynamic Link Library (.DLL).

8. Если необходимо наберите имя для файла .HLP приложения в поле Help File (Файл помощи) или воспользуйтесь эллиптической кнопкой (...) для выбора файла в диалоговом окне Open File (Открыть файл).

Генератор приложения не требует, чтобы в этой точке существовал файл .HLP. До поры вы можете оставить поле пустым а затем позднее заполнить.

Генератор приложения позволяет вам поименовать темы помощи в вашем приложении без определения того, что файл помощи существует. Вы отвечаете за создание файла .HLP, который содержит контекстные строки и ключевые слова, вводимые вами по необходимости в качестве HLP атрибутов для различных элементов управления и диалогов.

9. Примите по умолчанию шаблон Clarion в поле Application Template (Шаблон приложения).

Выбранный шаблон приложения управляет генерацией программы.

10. Отметьте поле Use Application Wizard (Использование мастера приложения) чтобы воспользоваться мастером для создания полного приложения, основанного на выбранном словаре и нескольких даваемых вами ответах.

11. Нажмите кнопку ОК.

Появится диалоговое окно Application Wizard (Мастер приложения).

12. Ответьте на вопрос(ы) в каждом диалоговом окне, затем нажмите кнопку Next (Следующий). В последнем окне работает кнопка Finish (Окончание). Если вы удовлетворены своими ответами, нажмите кнопку Finish.

Мастер приложений создает файл .APP на основе словаря и данных вами ответов, затем он показывает диалоговое окно Application Tree (Дерево приложения) для нового вашего приложения.

Вы можете управлять некоторыми опциями мастера в словаре данных назначая опции для файлов, ключей, полей или отношения (смотрите Оптимизация для Мастеров).

## **Мастер процедуры просмотра**

---

Этот мастер создает Процедуру просмотра с многими ключами из существующего определения файла словаря. Поле просмотра сортируется каждым назначаемым вами ключем. Порядок сортировки контролируется Закладками (TAB). Он также создает соответствующие процедуры формата Form (Update) если вы установили, какие обновления разрешаются.

Чтобы воспользоваться Мастером процедуры просмотра (Browse Procedure Wizard):

1. Выделите процедуру ToDo в дереве процедур Procedure Tree и нажмите кнопку ENTER.

Открывается диалоговое окно Select Procedure (Выбрать процедуру).

2. Выберите Browse (просмотр) из списка шаблонов процедур.

3. Отметьте поле флажков Use Procedure Wizard (Использование мастера процедур), чтобы использовать мастер для создания процедуры, основанной на выбранном файле словаря и нескольких даваемых вами ответах.

4. Нажмите кнопку Select (выбрать).

Откроется диалоговое окно Browse Procedure Wizard (Мастер процедуры просмотра) .

5. Ответьте на вопрос(ы) в каждом диалоге, затем нажмите кнопку Next (следующий). На последнем диалоговом окне работает кнопка Finish (окончание). Если вы удовлетворены своими ответами, нажмите кнопку Finish.

Мастер процедуры просмотра создает процедуру(ы), основанную на словаре файла и данных вами ответах, затем показывает диалоговое окно Procedure Properties (свойства процедур) для вашей новой процедуры.

Вы можете контролировать некоторые опции мастера в словаре данных назначая опции для файлов, полей, ключей и отношений (смотрите Оптимизация для Мастеров).

## **Мастер формы**

---

Данный мастер создает процедуру “Форма” обновления на основе существующего определения файла словаря.

Чтобы воспользоваться Мастером процедуры формы (Form Procedure Wizard):

1. Выделите процедуру ToDo в дереве процедур Procedure Tree и нажмите кнопку ENTER.

Открывается диалоговое окно Select Procedure (Выбрать процедуру).

2. Выберите Form (формат) из списка шаблонов процедур.

3. Отметьте поле флажков Use Procedure Wizard (Использование мастера процедур) чтобы использовать мастер для создания процедуры, основанной на выбранном файле словаря и нескольких даваемых вами ответах.



4. Нажмите кнопку Select (выбрать).

Откроется диалоговое окно Form Procedure Wizard (Мастер процедуры формы) .

5. Ответьте на вопрос(ы) в каждом диалоге, затем нажмите кнопку Next (следующий). На последнем диалоговом окне работает кнопка Finish (окончание). Если вы удовлетворены своими ответами, нажмите кнопку Finish.

Мастер процедуры формы создает процедуру(ы), основанную на словаре файла и данных вами ответах, затем показывает диалоговое окно Procedure Properties (свойства процедур) для вашей новой процедуры.

Вы можете контролировать некоторые опции мастера в словаре данных назначая опции для файлов, полей, ключей и отношений (смотрите Оптимизация для Мастеров).

## Мастер отчета

---

Данный мастер создает процедуру отчета из существующего определения файла словаря.

Чтобы воспользоваться Мастером процедуры отчета (Report Procedure Wizard):

1. Выделите процедуру ToDo в дереве процедур Procedure Tree и нажмите кнопку ENTER.

Открывается диалоговое окно Select Procedure (Выбрать процедуру).

2. Выберите Report (отчет) из списка шаблонов процедур.

3. Отметьте поле флажков Use Procedure Wizard (Использование мастера процедур) чтобы использовать мастер для создания процедуры, основанной на выбранном файле словаря и нескольких даваемых вами ответах.

4. Нажмите кнопку Select (выбрать).

Откроется диалоговое окно Report Procedure Wizard (Мастер процедуры отчета).

5. Ответьте на вопрос(ы) в каждом диалоге, затем нажмите кнопку Next (следующий). На последнем диалоговом окне работает кнопка Finish (окончание). Если вы удовлетворены своими ответами, нажмите кнопку Finish.

Мастер процедуры отчета создает процедуру(ы), основанную на словаре файла и данных вами ответах, затем показывает диалоговое окно Procedure Properties (свойства процедур) для вашей новой процедуры.

Вы можете контролировать некоторые опции мастера в словаре данных назначая опции для файлов, полей, ключей и отношений (смотрите Оптимизация для Мастеров).

## Мастер печати словаря

---

Этот мастер печатает описание существующего файла словаря с различными уровнями подробностей для файлов, полей, ключей и связей. Вы можете печатать на принтер или в файл.

Для того, чтобы воспользоваться Мастером печати словаря:

1. Откройте приложение, использующее данный словарь.
2. Выберите из меню Application > Template Utility.  
Появится диалоговое окно Выбор утилиты (Select Utility).
3. Отметьте Печать словаря (Dictionary Print) и нажмите кнопку Выбрать (Select).  
Появится диалоговое окно Мастер печати словаря (Dictionary Print Wizard).
4. Ответьте на вопрос(ы), появляющийся в каждом диалоге, а в заключение нажмите кнопку Следующий (Next).

После первого диалога становится доступной кнопка Окончание (Finish). Нажмите теперь кнопку Окончание для того, чтобы напечатать всю информацию, имеющуюся для всех файлов, полей, ключей и связей.

Имеется другой вариант действий. Пройдите через все диалоги и выберите требуемые файлы, а также уровень подробностей при печати (Все, Часть или Ничего) для различных компонент словаря.

## Оптимизация для Мастеров

---

Опции Мастера в Редакторе Словаря Данных обеспечивают более высокую степень управления функциональными возможностями мастера. Мастер использует опции, назначенные для файла, поля, ключа или псевдонима при создании процедур. Кроме того, Мастера используют имена файлов, полей, ключей и псевдонимы, а также описания, для составления текста, располагающегося в меню, на строках заголовка, в закладках и т.д.

### Опции файла

Do Not Auto-Populate This File

(не генерировать процедуры просмотра и процедуры модифицирования)

Указывает мастеру пропустить этот файл при создании первичных процедур просмотра или процедур отчета.

User Options (Опции пользователя) Опции пользователя предназначены для того, чтобы дать вам возможность представить информацию для использования набором покупных шаблонов. Опции пользователя разделены запятыми, т.е. каждый ввод отделяется запятой.

Следуйте инструкциям, поставленным с вашими набором шаблонов расширения.

### **Опции псевдонима**

Do Not Auto-Populate This File (не генерировать процедуры просмотра и процедуры модифицирования) Указывает мастеру пропустить этот файл с псевдонимом при создании первичных процедур просмотра или процедур отчета

User Options (Опции пользователя) Опции пользователя предназначены для того, чтобы дать вам возможность представить информацию для использования набором покупных шаблонов. Опции пользователя разделены запятыми, т.е. каждый ввод отделяется запятой.

Следуйте инструкциям, поставленным с вашими набором шаблонов расширения

### **Опции поля**

Do Not Auto-Populate This File (не генерировать процедуры просмотра и процедуры модифицирования) Указывает мастеру пропустить это поле при создании первичных процедур формата, просмотра или процедур отчета.

Population Order (Порядок заселения) Устанавливает порядок, в котором мастера заселяют поля. Выберите Normal (нормальный), First (первый) или Last (последний) в выпадающем списке. Мастера заселяют в этом порядке: все поля указанные первыми, затем все поля, указанные как нормальные, и наконец все поля, указанные как последние.

Form Tab (Закладка формата) Указывает ТАВ (закладку), на которой мастера заселяют поле. Наберите надпись для ТАВ (закладки) или выберите ранее созданную вами закладку из выпадающего списка. Это позволит вам направить мастера к групповым полям так, как вы этого желаете.

Add Extra Vertical Space Before Field Controls on Form (Добавить вертикальный промежуток перед элементами управления поля на форматах) Отметьте это поле для указания мастерам о необходимости добавить вертикальный промежуток между элементом управления этого поля и поля, заселенного выше него.

User Options (Опции пользователя) Опции пользователя предназначены для того, чтобы дать вам возможность представить информацию для использования набором покупных шаблонов. Опции пользователя разделены запятыми, т.е. каждый ввод отделяется запятой.

Следуйте инструкциям, поставленным с вашими набором шаблонов расширения

### **Опции ключа**

#### **Do Not Auto-Populate This File**

не генерировать процедуры просмотра и процедуры модифицирования)

Указывает мастеру пропустить этот ключ при создании первичных процедур просмотра или процедур отчета.

#### **Population Order**

(Порядок заселения) Устанавливает порядок, в котором мастера заселяют ключи. Выберите Normal (нормальный), First (первый) или Last (последний) в выпадающем списке. Мастера заселяют в этом порядке: все ключи указанные первыми, затем все ключи, указанные как нормальные, и наконец все ключи, указанные как последние.

#### **User Options**

(Опции пользователя) Опции пользователя предназначены для того, чтобы дать вам возможность представить информацию для использования набором покупных шаблонов. Опции пользователя разделены запятыми, т.е. каждый ввод отделяется запятой.

Следуйте инструкциям, поставленным с вашими набором шаблонов расширения

### **Опции отношений**

User Options (Опции пользователя) Опции пользователя предназначены для того, чтобы дать вам возможность представить информацию для использования набором покупных шаблонов. Опции пользователя разделены запятыми, т.е. каждый ввод отделяется запятой.

Следуйте инструкциям, поставленным с вашими набором шаблонов расширения

### **Оптимизация Словаря для Мастеров**

При создании процедур Мастера извлекают информацию из вашего Словаря Данных. Понимание того, как Мастера используют информацию словаря, даст вам информацию, необходимую для получения желаемых результатов.

### **Наименования Полей и Ключей**

Редактор словаря создает элементы управления по умолчанию для каждого поля. Знание того, как он использует предоставляемую вами информацию, помогает создать правильные элементы, устанавливаемые по умолчанию. Имя поля используется для создания приглашения по умолчанию и Заголовка столбца. Если вы используете смешанный случай, такой, например, как FirstName, заголовок столбца и приглашение по умолчанию создаются

как два слова - в данном случае: First Name .

Мастера создают многоключевую процедуру просмотра, которая использует описания ключей в качестве текста, показываемого на закладках. Если описание отсутствует, используется имя ключа. Добавление описаний позволяет Мастерам создавать закладки в том виде, в котором вы их хотите получить.

### **Описания полей**

Описания полей присваиваются атрибуту MSG элемента управления в окне по умолчанию. Атрибут MSG обеспечивает текст, который появляется в строке состояния данного приложения. Если описания внесены в словарь, это устраняет необходимость назначать сообщение для каждого размещаемого элемента управления.

Сообщение, показываемое в строке состояния, описывает действующий элемент управления. Наличие дополнительной информации улучшает функциональные характеристики вашего приложения.

### **Использование назначаемых по умолчанию элементов управления окна**

Редактор словаря создает тип элемента управления по умолчанию для каждого типа данных. Мастера используют этот назначенный по умолчанию элемент управления при создании процедур.

Например, строка символов STRING становится элементом управления ввода. В особых случаях вы можете захотеть иметь иной тип управления. Например, в случае номера клиента, который автоматически увеличивается, вы никогда не захотите, чтобы пользователь изменил его. В этом случае вы должны модифицировать назначаемое по умолчанию управление окна и выполнить его в виде строки символов (STRING).

Другим примером может быть поле, которое содержит конечный список для выбора значения поля. В этом случае вы можете создать ниспадающий список в качестве назначаемого по умолчанию элемента управления окна и определить правильный список для выбора при проверке данных.

## Шаблоны процедур

Этот раздел посвящен описанию шаблонов процедур Clarion for Windows. Здесь также будут упомянуты несколько шаблонов элементов управления, которые являются предметом рассмотрения следующего раздела.

Язык шаблонов Clarion обеспечивает поддержку для компонентно-ориентированного подхода к разработке. Когда вы создаете процедуру, вы начинаете с шаблона процедуры. Это генерирует исходную программу для инициализации процедуры, исполнения ее и окончания. Clarion for Windows предоставляет вам богатый ассортимент шаблонов процедур, которые генерируют код для обращения к- большому числу разнообразных задач приложений.

### Компонентно-ориентированные шаблоны

Исходная программа, генерируемая шаблонами процедур, обычно содержит определение малого числа переменных для поддержания управления поведением окна, а также цикл ACCEPT и операторы CASE для обработки событий, характерных для данного окна и полей, содержащихся в нем:

```
ACCEPT
CASE EVENT ( )
... (Стандартный код для поддержки окна)
END
CASE FIELD ( )
... (Стандартный код для обработки событий, связанных с полем)
END
END
```

Шаблоны процедур содержат также стандартные приглашения для всех кнопок, полей ввода или полей флажков, которые вы добавляете в окно процедуры. Вы получаете доступ к этим приглашениям через диалоговые окна Действий для любого из этих элементов управления.

Элемент управления полем ввода, например, поступает с дополнительными приглашениями, помогающими вам быстро использовать его как справочное поле. Другой пример - это поле флажка, которое обеспечивает дополнительные подсказки, так что вы можете обновить переменные или спрятать/открыть элементы управления когда конечный пользователь отмечает или очищает поле.

Таким образом, шаблон процедуры действует как контейнер, автоматически обеспечивающий поддержку для дополнительных слоев функциональных возможностей шаблона.

## Шаблоны процедур и шаблоны управления

Многие из шаблонов процедур содержат также шаблоны элементов управления. Шаблоны элементов управления генерируют код для определения и сопровождения конкретного элемента управления, включая загрузку данных в элемент управления и выгрузку их оттуда. Например, шаблон процедуры просмотра является в действительности родственным шаблоном процедуры окна, содержащим шаблоны управления Поле просмотра (BrowseBox) и Кнопки обновления просмотра (BrowseUpdateButtons)!

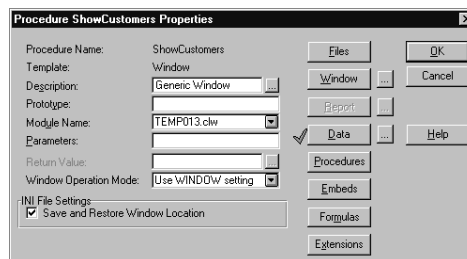
В зависимости от того, какие шаблоны - процедур или управления - вы добавляете к вашему приложению, точка вставки появляется в диалоговом окне Вставляемый источник (Embedded Source), а вместе с этим становится доступным новый набор шаблонов управления. Остальные разделы данной главы посвящены описанию шаблонов процедур, которые поставляются вместе с Clarion for Windows, и содержат руководство по заполнению опций и сообщений, которые реализуют их.

## **Шаблон Window**

Шаблон функционирует как чистая грифельная доска, на которой вы можете создать свою собственную процедуру окна любого вида. Нажмите кнопку Window в диалоговом окне Procedure Properties для создания вашего нового окна.

Для шаблонов управления и элементов управления, которые вы размещаете, шаблоны полей (на них есть ссылка в шаблоне процедуры окна) добавляют точки вставок для событий, которые они генерируют. Кнопка Embeds даст вам возможность добавить соответствующий код после размещения вами элементов управления в окне.

Единственные “заранее определенные” элементы шаблонов, к которым вы можете иметь доступ через диалог Procedure Properties - это локальные переменные, создаваемые данным шаблоном, которые исполняемая программа создала путем использования шаблонов для передачи данных к процедуре вызова и обратно от нее. Они “управляют” окном и процедурами, отслеживая, открыто ли окно и не нужны ли процедуры в ответ на глобальное событие.



Программа, генерируемая данным шаблоном, обеспечивает функционирование создаваемого вами окна. Она содержит цикл ACCEPT для окна и структуру CASE для обработки любых событий, связанных с окном или полями в окне.

**Совет:** Чтобы сдублировать окно, созданное для другого приложения или процедуры без копирования процедуры целиком, скопируйте декларацию WINDOW из другого текста исходной программы, затем вставьте ее в редактор окна, доступный для разрабатываемой процедуры. Предупреждение: это не относится к окнам, элементы управления которых были размещены шаблонами управления!

В дополнение к обычным командным кнопкам Свойств процедуры (Procedure Properties) шаблон процедуры Окно содержит еще две кнопки, дающие вам дополнительные возможности:

### Window Operation Mode

(режим работы окна)

Дает вам возможность менять свойства окна, который вы выбрали в форматере окна. Вы можете установить тип окна: Normal, MDI (добавляющий атрибут MDI- интерфейс многих документов), или Modal (добавляющий атрибут MODAL к структуре WINDOW). Вы можете также выбрать Use Window Setting (выбор установки окна), что является позицией по умолчанию. Это устанавливает, что вы не хотите отвергать Window Properties (свойства окна).

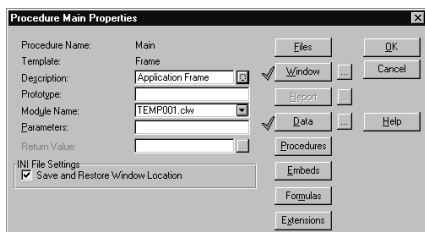
### INI File Settings

(установки файла INI)

Устанавливает, что вы хотите сохранить позицию окна (местоположение) в файле приложения .INI, когда конечный пользователь закрывает его. Для поддержки этого вы должны сделать доступным использование файла .INI в диалоге Global Properties (общие свойства).

## Шаблон Frame

Этот шаблон обеспечивает головное MDI (Интерфейс многих документов) окно, содержащее заранее определенное меню оболочки. Это меню в свою очередь обеспечивает такие часто используемые функции, как команда выхода Exit, стандартные команды редактирования и управления окном и буфером обмена.



При создании MDI - приложения головное окно должно быть главной процедурой. Вы начинаете новые процессы (threads) исполнения для каждого дочернего



окна MDI, которое вы хотите видеть появившимся внутри головного окна. Кнопки действий для элементов управления или позиция меню обеспечивают поле флажков для выбора начала новой цепи исполнения, вы можете также использовать шаблон программы инициализации процесса `InitiateThread Code`.

Заранее определенное головное окно содержит меню программной оболочки со следующими командами:

Файл - Установка печати и Выход;

Редактирование - Вырезать, Копировать и Вставить;

Окно - Мозаика, Каскад и Упорядочить значки;

Помощь - Содержание, Как пользоваться помощью и Как найти справку о чем-то.

Каждая команда из заранее установленного меню реализует стандартное (STD) поведение окон. Библиотеки поддержки выполняют все функции автоматически, так что вам не нужно составлять никаких программ для этих команд меню.

Шаблон имеет стандартные точки вставки, а кроме того, дополнительные точки вставки, которые необходимы вам для связи кода с данным выбором любой команды меню, или в том случае, если линейка выделения “выпадает на” или “выделяет” какую-либо из команд меню. Если вы добавляете Линейку инструментов, точки вставки добавляются соответственно для каждого элемента управления, помещаемого на линейку инструментов.

В дополнение к обычным командным кнопкам Свойств процедуры (Procedure Properties) шаблон процедуры Frame содержит следующее:

### Splash Procedure

(Процедура Заставка) Вызывает процедуру, которая появляется после открытия рамки приложения, но прежде чем будет сгенерировано какое-либо событие пользователя.

Обычно процедура Заставка обеспечивает визуальные или звуковые фанфары (или и то и другое) для вашей программы. Экран заставки может показать хорошо узнаваемый символ или знак, а эта узнаваемость повысит уровень комфорта пользователя и может служить рекламой вашей программы. Кроме того, это отвлекает внимание пользователя от зачастую скучной процедуры загрузки и инициализации программы.

Более подробную информацию об этом вопросе вы можете найти в Шаблоне заставки.

### INI File Settings

(установки файла INI) Устанавливает, что вы хотите сохранить позицию окна (местоположение) в файле .INI вашего приложения, когда конечный пользователь закрывает его. Для поддержки этого вы должны сделать доступным использование файла .INI в диалоге Global Properties (общие свойства).

## Шаблон заставки Splash

---

Шаблон заставки генерирует программу для показа окна с изображением и неким текстом. Окно закрывается автоматически через установленный промежуток времени. Кроме того, вы можете позволить пользователю при желании закрыть это окно в любое время, для чего он должен щелкнуть на нем мышью.

Процедура Frame (головное окно) может по желанию вызывать процедуру заставки. Более полную информацию вы можете получить в разделе Шаблон Frame. Есть и другой способ вызова. Вы можете вызвать процедуры заставки с помощью встроенной исходной программы. Смотрите для этого Генератор Приложения - Встроенная исходная программа.

Обычно процедура Заставка обеспечивает визуальные или звуковые фанфары (или и то и другое) для вашей программы. Экран заставки может показать хорошо узнаваемый символ или знак, а эта узнаваемость повысит уровень комфорта пользователя и может служить рекламой вашей программы. Кроме того, это отвлекает внимание пользователя от зачастую скучной процедуры загрузки и инициализации программы.

Шаблон заставки предоставляет собой заранее определенное окно с изображением и некоторые текстовые строки на панели с трехмерными скосами.

В дополнение к обычным командным кнопкам диалоговое окно Свойства процедуры (Procedure Properties) шаблона заставки содержит еще и следующие приглашения:

### Display Time

(Время демонстрации в секундах) Определяет максимальный интервал времени, в течение которого демонстрируется окно.

### Close when the user clicks on the splash window

(Закреть, когда пользователь щелкнет мышью на окне заставки) Если вы отметите данное поле, то тем самым дадите возможность пользователю закрыть окно в любой момент с помощью щелчка мыши на этом окне.

## Шаблон Menu

---

Этот шаблон обеспечивает SDI (интерфейс одного документа) окно.

Все приглашения подобны приглашениям шаблона головного MDI- окна. Заранее определенное окно содержит только одно меню (File, файл), в свою очередь имеющее единственную команду (Exit, выход).

## Шаблон исходной программы Source

Шаблон процедуры исходной программы обеспечивает элегантный и простой способ добавления к вашему приложению программы, написанной вручную. Он обеспечивает две точки, в которых можно встроить вашу программу: секцию данных и секцию программ.

Шаблон просто объявляет процедуру, управляет различными опционными параметрами, размещает вставки объявления данных в секцию данных, начинает секцию CODE и затем помещает любую исполняемую встроенную программу в секцию CODE:

... (локальные данные)

CODE

... (ваша вставка программного текста)

Шаблон исходной программы предоставляет обычные командные кнопки в диалоговом окне Procedure Properties (свойства процедур). Смотрите раздел Генератор Приложения - Добавление процедуры к вашему приложению. В качестве примера применения шаблона исходной программы смотрите раздел Элементы управления и их свойства - Элементы управления техники OLE с ОСХ.

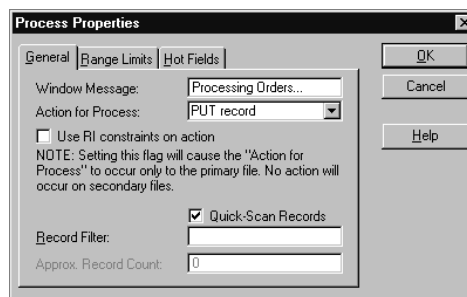
## Шаблон процесса (Process)

Шаблон процедуры обработки записей читает записи из файла данных и выполняет некоторую обработку каждой записи. Вы можете определить фильтр или диапазон записей, для которых необходимо выполнить обработку. Заранее определенное окно содержит индикатор течения процесса, показывающий конечному пользователю, какой процент операции завершен.

В дополнение к обычным командным кнопкам диалоговое окно Свойства процедуры (Procedure Properties) шаблона процесса содержит еще и следующие приглашения:

Process Properties

(Свойства процесса) Эта кнопка обеспечивает доступ к свойствам процесса.



### **Закладка “Общая” (General)**

#### Window Message

(Сообщение окна) Текст для показа на окне состояния процесса.

Action (действие) Это поле позволяет вам установить, что обработка заключается в изменении (PUT) записей или в удалении записей. Вы можете подключить программу в точку вставки Activity for each Record (транзакция для каждой записи), которую добавляет шаблон элемента управления.

Use RI constraints on action (использование ограничений ссылочной целостности в отношении действия)

Отметьте это поле, чтобы сформировать вызов стандартных процедур обновления Ссылочной целостности (Referential Integrity RI), генерируемых шаблонами. Это включает ограничения ссылочной целостности RI, определенные в вашем словаре данных. Очистите это поле для того, чтобы генерировать простые PUT (ВЫВОДИТЬ) или DELETE (УДАЛИТЬ) в зависимости от того, какое было выбрано Действие для процесса (Action for Process).

#### Quick-Scan Records

(Быстрый доступ к записям) Устанавливает режим буферизации доступа к файлам ODBC, ASCII, DOS или BASIC. Эти драйверы файлов читают буфер (не запись), обеспечивая быстрый доступ. В среде многих пользователей эти буферы надежны не на все 100% для последующего доступа, так как другой пользователь может изменить файл между обращениями. В качестве защиты драйвер перечитывает буферы перед доступом к каждой записи. Чтобы не допустить перечитывание включите QUICKSCAN.

#### Record Filter

(фильтр записей) Наберите выражение чтобы ограничить процесс обработкой только до тех записей, которые удовлетворяют фильтру. Это отфильтровывает все годные записи. Когда фильтр записей используется вместе с ограничениями диапазона, рассматриваться будут только те записи, которые попадают в назначенный диапазон..

#### Approx Record Count

(приблизительное число записей) Это число используется индикатором течения процесса, который появляется на время процесса.

### **Закладка “Пределы диапазона” (Range Limit)**

Эта закладка (tab) доступна только тогда, если вы назначаете ключ для файла в Схеме Файлов.

#### Range Limit Field

(Поле, ограниченное по диапазону) В соответствии с Range Limit Type (типом предела диапазона) назначает для включения в данный процесс запись или

группу записей. Для выбора поля нажмите эллиптическую (...) кнопку. Предел диапазона зависит от ключа; сгенерированная программа будет использовать оператор SET, чтобы найти первую правильную запись.

#### Range Limit Type

(тип предела диапазона) Определяет тип предела диапазона, который следует применить. Выберите один из появляющихся в ниспадающем списке.

#### Current Value

(текущее значение) Ограничивает ключ текущим значением Поля, ограниченного по диапазону.

#### Single Value

(одно значение) Дает вам возможность ограничить ключ одним значением. Определите переменную, содержащую эту величину, в появляющемся поле Range Limit Value.

#### Range of Values

(диапазон величин) Дает вам возможность назначить верхний и нижний пределы. Определите переменную, содержащую эти пределы, в полях Low Limit (нижний предел) и High Limit (верхний предел).

#### File Relationship

(отношение файлов) Дает вам возможность выбрать ограничивающий диапазон файл из отношения один-к-многим. Это ограничивает процесс таким образом, что он включает только те дочерние записи, которые соответствуют текущей записи в родительском файле. Например, если ваш отчет был Списком заказов, вы можете ограничить процесс только теми заказами, которые относятся к текущему заказчику (в файле Заказчик).

### **Закладка “Горячие поля” (Hot Fields)**

Когда вы выбираете закладку (tab) горячие поля (Hot Fields), вы можете выбрать поле (или поля) для добавления к VIEW. При прокручивании файла генерированная исходная программа читает данные для этих полей из VIEW, а не с диска. Это оптимизирует работу. Элементы Первичного ключа и текущего ключа всегда включены во VIEW, так что их не нужно вставлять в список Hot Field. Любое поле, используемое в вычислении или фильгровании, должно присутствовать во VIEW.

Кроме того через это диалоговое окно, вы можете определить поля как (BIND). Вы должны связывать любое поле, используемое в фильтре.

### **Шаблон External**

Шаблон внешней процедуры объявляет, что процедура содержится во внешней библиотеке (\*.LIB только) или в объектном файле. Генератор приложения не пишет никакой программы. Система проекта связывает внешний файл как модуль. Более подробная информация содержится в разделе Стратегии разработки и развертывания.

После выбора внешнего шаблона из диалогового окна Select Procedure Type (выбор типа процедуры), выберите OBJ или LIB из диалога Select Module Type (выбор типа модуля).

Выберите имя файла внешней библиотеки или объектного файла из выпадающего списка в поле Module Name (имя модуля). Появляются только те внешние модули, которые уже включены в проект, так что если ваш модуль не появляется, добавьте сначала новый модуль. Чтобы добавить модуль в генераторе приложений, выберите Application>Insert Module.

Если необходимо, введите объявления параметров в поле Prototype (прототип). Более подробную информацию вы можете найти в разделе Генератор Приложения - Макетирование и Передача параметров.

## Шаблон Browse

---

Шаблон просмотра данных состоит из нескольких шаблонов элементов управления, которые добавляют к шаблону Окно поле списка, кнопки обновления, кнопку выбора и кнопку закрытия окна. Диалог File Schematic Definition (определение схемы файлов) автоматически присоединяет выбранные вами файлы к шаблону управления BrowseBox (просмотр данных).

Кроме того, сгенерированная программа включает подпрограмму Refresh Window (обновление окна), которая выполняет поиски связанных записей, обновление полей списка там, где это необходимо, и получает текущие данные для элементов управления для показа их в окне.

В дополнение к обычным командным кнопкам диалоговое окно Procedure Properties (свойства процедуры) шаблона просмотра содержит следующие дополнительные приглашения:

### Update Procedure

(процедура обновления) Наберите имя новой процедуры или выберите имеющуюся процедуру из ниспадающего списка. Если вы назовете новую процедуру, Генератор приложения автоматически добавит ее к Дереву приложения (Application Tree ).

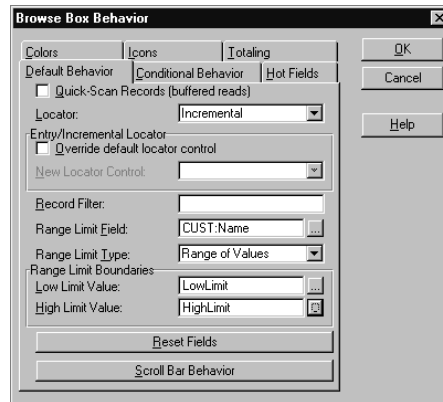
### Allow Edit via Popup

(разрешить редактирование через всплывающее меню) Отметьте это поле для активации всплывающего меню в тот момент, когда пользователь щелкнет правой клавишей мыши на поле списка. Всплывающее меню вызывает процедуру обновления для Вставки, Изменения или Удаления записи.

## Browse Box Behavior

(поведение поля просмотра)

Эта кнопка обеспечивает доступ к диалоговому окну, составленному из закладок, в котором вы можете установить опции для управления поля просмотра.



### Закладка “Поведение по умолчанию” (Default Behavior)

Эта закладка содержит приглашения, которые управляют поведением поля просмотра по умолчанию.

#### Quick-Scan Records

(Быстрый доступ к записям) Устанавливает режим буферизации доступа к файлам ODBC, ASCII, DOS или BASIC. Эти драйверы файлов читают буфер за время (не запись), обеспечивающее быстрый доступ. В среде многих пользователей эти буферы надежны не на все 100% для последующего доступа, так как другой пользователь может изменить файл между доступами. В качестве защиты драйвер перечитывает буферы перед доступом к каждой записи. Чтобы не допустить перечитывание включите QUICKSCAN.

**Locator (Локатор)** Локатор дает пользователю возможность искать особые записи в поле списка без необходимости прокручивать вручную весь список. Локатор доступен только при просмотре файла в Порядке ключей (т.е. установлен КЛЮЧ в Схеме файлов). Сделайте свой выбор типа локатора в ниспадающем списке:

**Step (шаг)** Назначает локатор типа одного удара клавиши без необходимости поля ввода. Когда поле просмотра получает фокус и пользователь набирает символ, поле списка перемещается к первой записи, ключевое поле которой начинается с этого символа (или следующего старшего символа, если нет ни одного значения ключа, начинающегося с данного символа локатора). Повторный набор того же самого символа перемещает список к следующей записи в списке с ключевым полем, начинающимся с этого символа.

**Entry (Ввод)** Назначает локатор поля ввода, который активизируется, когда поле

ввода принято. Атрибут USE поля ввода является первым свободным компонентом первичного ключа файла. Свободная компонента - это компонента, которая не ограничена по диапазону до одиночной величины.

Когда конечный пользователь помещает в поле ввода один или более символов, а затем нажимает TAB или выбирает другой элемент управления на экране, поле списка перемещается к ближайшей совпадающей записи.

#### Incremental

(инкрементальный) Назначает локатор поля ввода, который активизируется тогда, когда меняется поле ввода. Атрибут USE поля ввода является первым свободным компонентом первичного ключа доступа к файлу. Свободная компонента - это компонента, которая не ограничена по диапазону до одиночной величины.

Когда конечный пользователь помещает в поле ввода один или более символов, поле списка перемещается к ближайшей совпадающей записи после каждого отдельного удара по клавише.

None (отсутствует) Локатор не назначается.

#### Override default locator control

(замена элемента управления Локатор, назначенного по умолчанию) Отметьте это поле для установки вручную элемента управления для локатора с помощью ниспадающего списка New Locator Control (Новый элемент управления Локатор). Это оказывается полезным тогда, когда на одном и том же окне поля просмотра имеется более одного поля ввода.

#### New Locator Control

(Новый элемент управления Локатор) Выберите элемент управления для использования в качестве локатора из ниспадающего списка.

Record Filter (фильтр записей) Наберите выражение для ограничения состава поля просмотра только теми записями, которые соответствуют выражению фильтра. Вы должны связать (BIND) любое поле файла, которое используется в выражении фильтра. Закладка "Горячие поля" (Hot Fields) дает вам возможность связать поля.

#### Range Limit Field

(Поле ограниченное диапазоном) Вместе с полем Range Limit Type (тип предела диапазона) устанавливает запись или группу записей для включения в поле списка. Наберите имя поля путем нажатия эллиптической (...) кнопки. Предел диапазона зависит от ключа; генерируемая исходная программа будет использовать оператор SET, чтобы найти правильную запись. Это будет возможно только тогда, когда вы назначите ключ KEY для файла в схеме файлов.

#### Range Limit Type

(тип предела диапазона) Назначает применяемый тип предела диапазона. Выберите нужный из следующего выпадающего списка.

Current Value (текущее значение) Ограничивает ключ текущим значением Range Limit Field (поля ограниченного диапазона).



**Single Value (единственное значение)** Дает вам возможность ограничить ключ до единственного значения. Назначьте переменную, содержащую эту величину, в поле Range Limit Value (величина предела).

**Range of Values**

(диапазон величин) Дает вам возможность установить верхний и нижний пределы. Назначьте переменную, содержащую пределы, в поля Low Limit (нижний предел) и High Limit (верхний предел).

**File Relationship**

(отношение файлов) Дает вам возможность выбрать ограничивающий диапазон файл из отношения 1:Many (Один-к-многим). Это ограничивает окно просмотра только показом тех дочерних записей, которые соответствуют текущей записи в родительском файле. Например, если ваше поле просмотра было списком заказов, вы могли ограничиться показом только заказов данного покупателя (в файле Покупатель).

**Reset Fields button**

(кнопка переустановки полей) Нажатие этой кнопки вызывает показ списочного поля, которое позволяет добавить Reset Fields. Если величина какого-либо поля в Reset Fields списке меняется, поле просмотра обновляется.

**Scroll Bar Behavior button**

(кнопка поведения полосы прокрутки) Нажатие этой кнопки вызывает показ диалогового окна, в котором вы можете определить, каким образом работает полоса прокрутки.

**Scroll Bar Behavior**

(поведение полосы прокрутки) Определяет характер работы полосы прокрутки. Выберите Fixed Thumb (фиксированный указатель) и Movable Thumb (подвижный указатель) из выпадающего списка.

**Key Distribution**

(распределение ключа) Эта позиция определяет распределение точек полосы прокручивания. Выберите одно из двух заранее назначенных распределений (алфавитное (Alpha) или по фамилиям (Last Names)) или свое собственное (Custom) или рабочее (Runtime) из ниспадающего списка.

**Alpha (алфавитный порядок)** Определяет 100 равномерно распределенных в алфавитном порядке точек.

**Last Names (Фамилии)** Определяет 100 точек, распределенных как обычные фамилии, типичные для Соединенных Штатов. Если ключ доступа отсортирован по цифровым данным, вам следует использовать распределения Custom и Runtime.

**Custom**

(По определению пользователя) Дает вам возможность назначить свои собственные точки.

**Run-time**

(По определению во время выполнения программы) Читает первую и последнюю записи и вычисляет значения для 100 равномерно распределенных в этом

промежутке точек.

### Custom Key Distribution

(распределение ключей, задаваемое пользователем) Дает вам возможность установить контрольные точки для распределения вдоль линейки прокрутки (полезно тогда, когда у ваших данных асимметричное распределение). Вставьте величины для каждой точки в список. Строчные константы должны быть заключены в одинарные кавычки (‘ ’).

### Runtime Distribution

Parameters (параметры распределения во время исполнения) Дает вам возможность назначить тип символов, учитываемых при определении точек распределения. Это годится только тогда, когда Свободным Ключевым Элементом является STRING или CSTRING. Отметьте в полях типы символов, которые вы хотите включить в рассмотрение.

## **Закладка “Условное поведение” (Conditional Behavior)**

Эта закладка содержит списочное поле, которое дает вам возможность определить характерное поведение, основанное на неких условиях. Добавьте условия к списку путем нажатия кнопки Insert. Появляется диалоговое окно, в котором вы определяете условие и желаемое поведение, когда это условие выполнено.

Во время работы эти условия оцениваются и используется реакция на условие, соответствующая первому выполняющемуся условию в списке.

В этом диалоговом окне вы можете установить:

Condition (условие) Любое допустимое выражение.

Key to Use

(используемый ключ) Если необходимо, используйте ключ для того, чтобы определить порядок сортировки поля просмотра, когда это условие выполняется.

Остальные поля и кнопки те же самые, как и в закладке поведения по умолчанию (Default behavior tab).

## **Закладка “Горячие поля” (Hot Fields).**

Когда вы выбираете закладку горячего поля Hot Fields tab, вы можете выбрать поле (или поля) для сохранения их “живым” в очереди QUEUE. При прокручивании через файл генерированная исходная программа читает данные для этих полей из QUEUE а не с диска. Это ускоряет обновление спискового поля.

Выбор “горячих” полей дает вам возможность также поместить элементы управления поля файла вне поля просмотра, которые обновляются каждый раз, когда другая запись

выбирается в поле списка. Элементы первичного ключа и текущий ключ всегда включены в QUEUE, так что их не нужно вставлять в список горячих полей.

Это диалоговое окно дает вам возможность связывать (BIND) поле. Вы должны связать любое поле файла, которое используется в выражении фильтра или как поле для суммы.

### **Закладка “Суммирование” (Total)**

Эта закладка содержит поле списка, которое позволяет вам определить суммарные поля для поля просмотра. Нажмите кнопку INSERT для добавления суммарных полей.

Появится диалоговое окно, в котором вы можете определить суммарные поля для BrowseBox (поле просмотра).

Total Target Field

(итоговое объектное поле) Переменная для хранения итога. Это может быть локальная, глобальная или модульная переменная. Вы можете также использовать файловое поле; однако вам нужно писать программу для обновления файла данных.

Total Type (Тип итога) Выберите Count, Sum или Average из выпадающего списка. Count подсчитывает число записей. Sum добавляет величины к полю итога. Average определяет арифметическое среднее поля для итога.

Field to Total

(поле для итога) Поле, которое должно быть суммировано или усреднено. Этот поле недоступно, когда итоговое поле - типа Count.

Total Based On

(Итог основан на) Выберите Each Record Read (читать каждую запись) или Specified Condition (оговоренное условие) из выпадающего списка. Это устанавливает, рассматривать ли каждую запись или только те, которые отвечают некоторым критериям фильтрования.

Total Condition

(Условие суммирования) Требуемое условие при использовании итога, основанного на специфическом условии. Вы можете использовать любое допустимое выражение.

Закладка “Цвета” (Color) Эта закладка доступна только, если вы отметите поле флажков Color Cells (ячейки цвета) в формате поля списка. Оно показывает список полей, которые были установлены для окрашивания.

Чтобы назначить цвета, выделите желаемое поле и нажмите кнопку Properties (свойства).

### **Настройка цветов**

Это диалоговое окно позволяет вам установить цвета переднего плана и для нормальных;

строк поля списка и отдельно для выбранных строк (Определение цвета осуществляется отдельно для каждой колонки ).

Ниже раздела цветов по умолчанию имеется список условного назначения цветов. Чтобы добавить условие и выбрать специальные цвета для показа поле при выполнении условия, нажмите кнопку INSERT.

При работе эти условия оцениваются и используются цвета для первого выполненного условия в списке.

### **Закладка “Пиктограммы” (Icons)**

Эта закладка доступна только, если вы отметите поле флажка Icons (пиктограммы) в форматере поля списка. Оно показывает список полей, которые были установлены для показа Пиктограммы. Чтобы выбрать пиктограмму, выделите желаемое поле и нажмите кнопку Properties (свойства).

Настройка Пиктограмм

поля просмотра

Это диалоговое окно позволяет дает вам возможность выбрать для поля пиктограмму по умолчанию.

Default Icon

(пиктограмма по умолчанию)

Пиктограмма по умолчанию. Вы можете выбрать стандартную пиктограмму или файл пиктограмм (.ICO) на диске.

Conditional Icon Usage

(условное использование пиктограммы)

Ниже раздела пиктограммы по умолчанию имеется список условного использования пиктограммы. Чтобы добавить условие и выбрать специальные пиктограммы для показа при выполнении условия, нажмите кнопку INSERT. Во время работы эти условия оцениваются и используется пиктограмма для первого выполненного в списке условия.

## **Шаблон Form**

---

Шаблон формы генерирует программу показа и обновления одиночной записи из файла.

Шаблон формы обеспечивает заранее определенное окно с шаблоном управления Кнопка сохранения, а также полем для вывода сообщения о характере обработки записи. Смотрите раздел Шаблоны управления, программы и расширения - Кнопка сохранения, в котором содержатся подробности о их приглашениях и характере действия. В настоящей процедуре шаблон управления Кнопка сохранения управляет файлом I/O (Ввод-вывод). Диалоговое окно File Schematic Definition (Определение схемы файлов) автоматически присоединяет выбранные вами файлы к шаблону управления Кнопка сохранения.

В дополнение к обычным командным кнопкам диалоговое окно Procedure Properties (свойства процедуры) для шаблона формы содержит также следующие приглашения:

INI Settings (установки INI) Устанавливает, что вы хотите сохранять положение окна (местоположение) в файле .INI приложения, когда пользователь закрывает его. Для поддержки этого вы должны открыть доступ к использованию файла .INI в диалоговом окне Global Properties (общие свойства).

Шаблон элемента управления SaveButton (кнопка “сохранение”) предоставляет следующие приглашения:

Allow

(разрешается) Отметьте любую комбинацию трех полей для назначения разрешенных операций файла I/O (Ввод-вывод):

Inserts

(вставки) Генерирует программу для управления вставкой записи (ADD).

Changes

(изменения) Генерирует программу для управления изменением записи (PUT).

Deletes

(удаления) Генерирует программу для управления удалением записи (DELETE).

When called for Delete

(при вызове для удаления) Позволяет выбрать, что показывается, когда эта процедура вызвана для удаления записи.

Standard Warning

(стандартное предупреждение) Показывает поле сообщений, приглашающее подтвердить удаление.

Show Form

(показывать форму) Показывает форму.

Automatic Delete

(автоматическое удаление) Позволяет записи быть удаленной без ее показа или без приглашения для подтверждения удаления.

After successful insert

(после успешной вставки) Выбирает режим вставки по-одной-за-раз или режим с повторением.

Return to caller

(возврат к вызову) Генерирует RETURN для возврата в вызвавшую процедуру вслед за успешной вставкой. В результате получается режим вставки по-одной-за-раз.

Insert another record

(вставка следующей записи) Не генерирует RETURN для возврата в вызвавшую процедуру после успешной вставки. В результате это приводит к режиму с повторением вставки.

Ask the user before adding another record

(запрашивать пользователя перед добавлением следующей записи) Не генерирует автоматически RETURN для возврата в вызвавшую процедуру вслед за

успешной вставкой, а запрашивает пользователя, добавлять ли следующую запись.

### Field Priming on insert

(снабжение поля информацией по умолчанию при вставке) Эта процедура дает вам возможность снабдить значениями по умолчанию поля в новой записи. Это значение перекрывает любую первоначальную величину, назначенную в словаре данных. Вы можете выбрать поле и поставить начальное значение в диалоговом окне Field Priming (заполнение поля информацией).

### Messages and Titles

(сообщения и заголовки) Войдите в диалоговое окно, чтобы выбрать сообщения и их местоположение для показа, нажав для этого кнопку Messages and Titles.



### Insert Message

(сообщение при добавлении) Определяет текст сообщения, когда процедура вызывается для добавления записи. Текст по умолчанию - "Record will be added" (запись будет добавлена).

### Change Message

(сообщение при изменении) Определяет текст сообщения, когда процедура вызывается для изменения записи. Текст по умолчанию - "Record will be changed" (запись будет изменена).

### Delete Message

(сообщение при удалении) Определяет текст сообщения, когда процедура вызывается для удаления записи. Текст по умолчанию - "Record will be deleted" (запись будет удалена).

### On Aborted add/change

(о прерванных добавлениях/изменениях) Определяет действие, которое надо предпринять, когда пользователь нажимает кнопку Cancel во время добавления или модификации записи. Вариантами выбора из ниспадающего списка являются: Offer to save changes (предложение сохранить изменения), Confirm Cancel (подтвердите отмену обновления) или Cancel without confirming (прервать без подтверждения).

### Location of Message

(местоположение сообщения) Устанавливает, где показывать сообщение. Выберите None/Window Control (нигде/ элемент управления окна) для показа

сообщения в элементе управления. Выберите Title Bar (строка заголовка) или Status Bar (строка статуса). Если необходимо, выберите место строки статуса в поле Status Bar Section (выбор места на строке статуса).

Display Record Identifier on the Title Bar

(показать идентификатор записи в строке заголовка) Дает вам возможность добавить строку символов в конец заголовка на строке заголовка.

Record Identifier

(идентификатор записи) Устанавливает строку символов для добавления ее в конец строки заголовка, которую вы можете использовать для идентификации записи. Наберите строку символов в поле Record Identifier. Для использования переменного имени предварите его знаком восклицания (!).

Шаблон расширения ValidateRecord (проверка правильности записи) добавляет приглашения для Control Value Validation Conditions (условия проверки правильности величины).

Validate when the control is Accepted

(проверять когда принято) Устанавливает, что проверка правильности происходит тогда, когда элемент управления генерирует EVENT:Accepted (событие принято) в тот момент когда пользователь завершает или перемещает фокус с данного поля.

Validate during NonStop Select

(проверять правильность во время безостановочного отбора) Устанавливает, что проверка правильности происходит тогда, когда какая-либо величина связанная с элементом управления изменяется если окно в режиме AcceptAll (Non-Stop) находится в фокусе.

Do Not Validate

(не проверять правильность) Открывает диалоговое окно Do Not Validate...., которое дает вам возможность вставить поля в список; эти поля будут исключены из проверок на правильность.

## Шаблон Report

---

Нажмите кнопку Report (отчет), чтобы определить ваш отчет. Заранее нет установленного отчета, если вы не использовали Мастер отчета для генерации данной процедуры.

Шаблон процедуры отчета генерирует программу для прочтения файла данных и обновления элементов управления в секции отчета DETAIL для каждой записи. Вы можете установить фильтр или диапазон записей, для которых надо выполнить операцию. Заранее определенное окно содержит индикатор, показывающий конечному пользователю процент выполнения процедуры отчет.

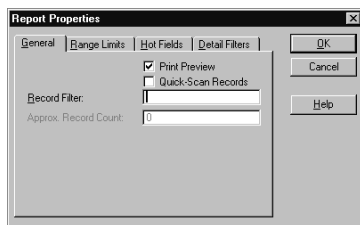
В дополнение к обычным командным кнопкам диалоговое окно Procedure Properties шаблона отчета содержат следующие дополнительные приглашения:

## Window Message

(сообщение окна) Назначает сообщение для показа в окне индикатора выполнения процесса.

## Report Properties

(свойства отчета) Эта кнопка открывает диалоговое окно, в котором вы назначите опции для данного отчета



### Закладка “Общая” (General)

#### Print Preview (просмотр

перед печатью) Будучи отмеченным, поле флажков Print Preview устанавливает, что конечный пользователь видит отчет в режиме просмотра перед печатью. Конечный пользователь может после этого распечатать отчет или отменить распечатку.

#### Quick-Scan Records

(Быстрый доступ к записям) Определяет режим буферизации доступа к файлам ODBC, ASCII, DOS и BASIC. Эти драйверы файлов читают буфер по одному (не запись), делая возможным быстрый доступ. В среде многих пользователей эти буферы не на 100% надежны для последующего доступа, так как другой пользователь может изменить файл между доступами. В качестве защиты от этого драйвер читает буферы перед доступом к каждой записи. Чтобы отключить перечитывание, включите QUICKSCAN.

Record Filter (фильтр записей) Наберите выражение для ограничения отчета только теми записями, которые совпадают с выражением фильтра. Когда фильтр записи Record filter используется вместе с пределом диапазона Range Limit, рассматриваются только те записи, которые находятся внутри определенного диапазона.

#### Approx Record Count

(приблизительное число записей) Введите примерное число записей, которое, как вы ожидаете, отчет должен обработать. Это число будет использовать индикатор генерации отчета

### Закладка “Пределы диапазона” (Range Limit)

Данная закладка доступна только тогда, если вы назначите ключ для файла в схеме файлов.



### Range Limit Field

(Поле,ограниченное диапазоном) В соответствии с Типом ограничения диапазона определяет запись или группу записей для включения в отчет. Выберите поле, нажмите эллиптическую (...) кнопку. Предел диапазона зависит от ключа; генерированная исходная программа использует оператор SET для отыскания первой подходящей записи.

### Range Limit Type

(тип предела диапазона) Назначает применяемый тип ограничения диапазона. Выберите один из следующих типов из ниспадающего списка.

### Current Value

(текущее значение) Ограничивает ключ текущим значением Поля, ограниченного диапазоном.

### Single Value

(простое значение) Дает вам возможность ограничить ключ одиночным значением. Назначьте переменную, содержащую это значение, в появляющемся поле Range Limit Value (величина допустимого диапазона)

### Range of Values

(диапазон величин) Дает вам возможность назначить верхний и нижний пределы этого диапазона, Назначьте переменную, содержащую диапазон, в полях Low Limit (нижний предел) и High Limit (верхний предел)

### File Relationship

(отношение файлов) Дает вам возможность выбрать ограничивающий диапазон файл из отношения 1:Many (Один-к-многим). Это ограничивает окно просмотра только показом тех дочерних записей, которые соответствуют текущей записи в родительском файле. Например, если ваше поле просмотра было списком заказов, вы могли ограничиться показом только заказов данного покупателя (в файле Покупатель).

## **Закладка “Горячие поля” (Hot Fields)**

Когда вы выбираете закладку Hot Fields (горячие поля), вы можете выбрать поле (или поля) для добавления их к виртуальному файлу VIEW. При просмотрении файла генерированная исходная программа читает данные для этих полей из VIEW, а не с диска. Это оптимизирует исполнение. Элементы первичного ключа Primary Key и текущего ключа всегда включаются во VIEW, так что их не нужно вставлять в список Hot Field.

Кроме того, вы можете связать (BIND) поля через это диалоговое окно. Вы должны связать все поля, используемые в фильтре.

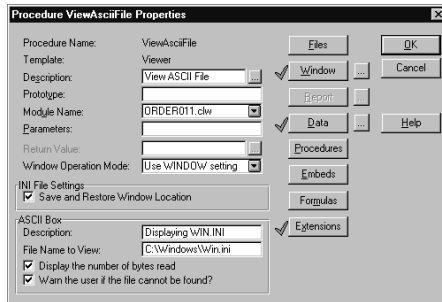
## **Закладка “Рабочие фильтры” (Detail Filters)**

Выберите эту закладку для установки фильтров печати для данных в области Detail . Это позволит вам подавить печать в Detail, если выражение фильтра не удовлетворяется.

## Шаблон Viewer

Шаблон просмотра Viewer обеспечивает заранее установленное окно с полем списка, кнопку ASCII поиска, кнопку ASCII печати и кнопку закрытия Close.

Если вы хотите использовать шаблон для того, чтобы всегда просматривать один и тот же файл ASCII, вы можете использовать его таким, как он есть. Чтобы разрешить просмотр любого ASCII файла, выбранного из стандартного файлового диалога, вам нужно добавить поле ввода для того, чтобы принять имя файла, а также шаблон поиска файла DOS.



В дополнение к обычным командным кнопкам, диалоговое окно Procedure Properties (свойства процедуры) для шаблона просмотра содержит следующие приглашения:

**Description (описание)** Назначает описание для текстового файла, выбранного для просмотра.

**File Name to View**  
(имя файла для просмотра) Определяет имя текстового файла, предназначенного к просмотру. Используйте имя переменной, предваряемое восклицательным знаком (!), в том случае, когда вы хотите дать возможность конечному пользователю выбрать файл для просмотра.

**Display the number of bytes read**  
(показ числа байтов прочитанного текста) Отметьте это поле для того, чтобы был показан размер файла.

**Warn the user if the file cannot be found?**  
(предупреждать ли пользователя, если данный файл не может быть найден?) Отметьте это поле для того, чтобы показывать при работе сообщение в случае, если определенный файл не может быть найден.

### **Просмотр файла, выбранного пользователем**

Чтобы быстро добавить поле для ввода имени файла, предназначенного для просмотра, выберите Populate > Field или используйте инструмент Populate (заселять) в инструментальной панели в Форматере окна. Затем в диалоговом окне Select Field (выбор поля) выберите ASCIIFileName переменную из раздела глобальных данных. Эта переменная добавляется шаблоном процедуры.

Чтобы добавить эллиптическую кнопку(...), которая дает возможность конечному пользователю выбрать файл из стандартного файлового диалога, выберите инструмент Control Template ( шаблон элемента управления) на инструментальной панели элементов управления. Затем выберите шаблон управления поиск файла DOS и щелкните мышью в конструируемом окне. Для получения более полной информации по данному вопросу смотрите следующую главу.

### **Просмотр того же самого файла**

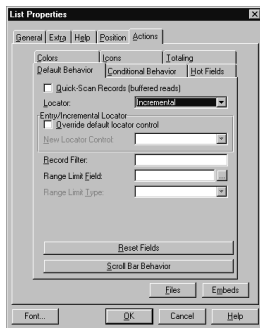
Чтобы настроить окно просмотра таким образом, чтобы конечный пользователь не имел возможности выбора файла, не добавляйте никаких дополнительных элементов управления или шаблонов управления в окно, а вместо этого назначьте нужное имя файла глобальной, добавленной шаблоном процедуры, переменной ASCIIFileName.



## Глава 7 Шаблоны элементов управления, программы и расширений



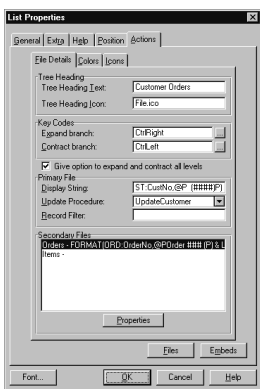
В данной главе вы научитесь использовать шаблоны элементов управления, программы и расширений. Шаблоны элементов управления генерируют заранее определенные элементы управления окна плюс программы для создания, манипуляции и объединения их друг с другом...



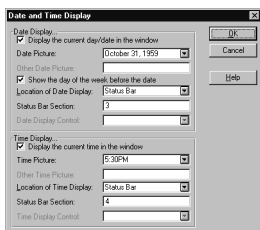
Шаблон управления полем просмотра данных даст вам возможность быстро добавить к окну списковые поля. Вы легко можете добавить дополнительные элементы управления поля просмотра и синхронизировать их.



Шаблон управления “SaveButton” (кнопка сохранения) управляет обновлением файла. Это дает вам возможность установить поведение процедуры обновления и выбрать сообщения, получаемые пользователем.



Шаблон управления “Дерево отношений” (Relation Tree) предоставляет доступ ко всем уровням отношений файлов. Один элемент управления Relation Tree может заменить несколько пар Browse-Form.



Шаблон расширения “Показ даты и времени дня” (Date Time Display) дает возможность показывать время и/или дату в строке статуса или в элементе управления..

**Добавление шаблонов элементов управления****Шаблоны управления****Поле просмотра (BrowseBox)****Шаблоны управления обновлением поля просмотра (BrowseUpdateButtons)****Шаблоны управления выбором в поле просмотра (BrowseSelectButton)****Шаблон управления сохранением данных. Кнопка ОК. (SaveButton)****Шаблон управления Кнопки Cancel (прервать)****Кнопка Close (закрыть)****Просмотр текстовых файлов ( ASCIIBox)****Кнопка управления печатью ASCII файлов (ASCII Print Button)****Кнопка поиска в ASCII - файлах (ASCII Search )****Доступ к стандартному диалогу выбора файлов Windows (DOSFileLookup)****Шаблон управления FileDrop****Шаблон управления FileDropCombo****Шаблон Дерево отношений (Relation Tree)****Шаблон Кнопки обновления дерева отношений (RelationTreeUpdateButtons)****Кодовые шаблоны****Инициировать процесс (thread)****Вызов процедуры доступа к справочнику (CollProcedureAsLOOKUP )****Шаблон проверки достоверности данных (ControlValueValidation)****Поиск записи в несвязанном файле (LookupNonRelatedRecord)****Закрыть текущее окно (CloseCurrentWindow)****Шаблоны расширения****Шаблон показа даты, времени (DateTimeDisplay)****Шаблон проверки правильности записи (RecordValidation)**

Шаблоны элементов управления увеличивают функциональные возможности шаблонов процедур. Шаблоны элементов управления состоят из заранее определенных элементов управления окна и программ создания, обслуживания и объединения их друг с другом.

Например, кроме поля списка, поддерживающего поля просмотра данных (browse), упомянутых в предыдущей главе, другие шаблоны элементов управления контролируют файл ввода/вывода. Сгенерированная исходная программа может автоматически выдавать пользователю предупреждающие сообщения о том, что над файлом выполнены изменения, на тот случай, если конечный пользователь попытается закрыть окно без запоминания этих изменений на диске.

Данная глава описывает шаблоны элементов управления, включенные в Clarion for Windows, и описывает приглашения, генерируемые этими шаблонами. В этой же главе также кратко описано включение шаблонов программ и расширений.

## **Шаблоны элементов управления**

### **Добавление шаблонов элементов управления**

---

Чтобы добавить шаблон элемента управления, когда вы начинаете работу над новой процедурой, выполните следующее:

- ☐ Определите окно, нажав для этого кнопку Window в диалоговом окне Procedure Properties (свойства процедуры).
- ☐ В форматере окна добавьте к окну элемент управления. Для этого щелкните мышью на соответствующей пиктограмме на панели инструментов элементов управления.
- ☐ Выберите шаблон элемента управления из диалогового окна Select control template (выберите шаблон элемента управления), затем поместите элемент управления на окно или отчет, щелкнув для этого в нужном вам месте.
- ☐ Элемент управления (тип элемента зависит от шаблона управления) появляется в окне.
- ☐ Щелкните правой клавишей мыши на элементе управления, затем выберите Actions (действия) из выпадающего меню для доступа к специфическим для данного элемента управления приглашениям; это позволяет определить его (элемента управления) функциональные возможности.
- ☐ Выберите другие закладки (tabs) из диалогового окна Properties (свойства) для модификации свойств элемента управления; это определяет, как будет выглядеть элемент управления, его местоположение и функциональные возможности.

Как только шаблон элемента управления добавлен к процедуре, рядом с кнопкой Extensions (расширения) в диалоге Procedure Properties (свойства процедуры) появляется флажок. Таким образом Вы получаете доступ к приглашениям, добавленным шаблоном элемента управления.

Вы можете редактировать функциональные возможности и вид элементов управления либо через появляющееся после щелчка правой клавишей всплывающее меню в Window Formatter (форматер окна), либо через кнопки Extensions (расширения) в диалоге Procedure Properties.

Шаблоны элементов управления могут также быть размещены в отчете, в зависимости от того, могут ли их функциональные возможности быть логически расширены до “бумаги”.

### **Поле просмотра (BrowseBox)**

---

Этот шаблон элемента управления размещает в окне поле просмотра (списка). Всплывающее меню элемента управления LIST приводит вас в диалоговое окно List Box Formatter (форматер поля списка). В этом диалоге вы можете определить, какие поля или переменные заполняют этот список и увидеть, в каком виде они в нем появятся (включая



активизацию окрашивания и показ пиктограммы). Закладка (tab) Actions (действия) на List Properties (список свойств) обеспечивает приглашения шаблонов, что дает вам возможность определить его функциональные возможности, включая фильтры записей, ограничения диапазонов записей, суммирование, поведение линейки прокручивания и локаторы.

Вы можете разместить поле просмотра в окне, используя панель инструментов. На панели инструментов щелчком мыши выберите пиктограмму шаблонов элементов управления и затем, в появившемся диалоговом окне Select control template (выбор шаблона элемента управления), выберите строку Browse Box - Browse List Box (поле просмотра - поле списка просмотра). После этого укажите щелчком мыши положение в окне будущего реального спискового поля.

### **Свойства**

После размещения спискового поля щелкните правой клавишей на элементе управления LIST и выберите Properties (свойства) из всплывающего меню для вызова диалогового окна List Properties (свойства списка). Для получения подробной информации о имеющихся в этом диалоговом окне опциях смотрите главу Элементы управления и их свойства. Данный раздел описывает только те опции, на которые непосредственно влияет шаблон управления полем просмотра.

Шаблон автоматически определяет для поля просмотра атрибут FROM (откуда) для элемента управления LIST, который указывает источник (QUEUE) (очередь) для данных в списке. Стандартные шаблоны называют QUEUE как Queue:Browse. Шаблон содержит подпрограмму (template routine), которая проверяет, применили вы пределы интервала или используете список как Lookup. В соответствии с этим он устанавливает место необходимой записи. Затем шаблон загружает столько записей в QUEUE, сколько уместится в списке. QUEUE заполняется из VIEW, который получает величины из полей в файлах данных на диске.

Щелкните правой клавишей мыши над элементом управления LIST и выберите List Box Formatter (форматер поля списка) из всплывающего меню для получения доступа к одноименному диалоговому окну, чтобы затем выбрать поля и переменные для заполнения поля списка и определите их вид, используя для этого диалоговое окно.

Кнопка Populate( заселить) позволит вам добавить к полю списка поле файла или переменную, по одному полю или переменной за один раз. Диалог Select Field (выбор поля) представляет схему файлов. В схеме появляется каждый элемент управления просмотром с деревом управления, помеченным "To Do" под ним. Чтобы добавить поле из файла данных, определенного в словаре:

- Выберите позицию “To Do”.
- Нажмите кнопку Insert (вставка).
- Выберите файл из списка, предлагаемого в появившемся диалоговом окне Insert File(Вставить файл). Позиция управления поля просмотра показывает имя файла.
- Если вы хотите использовать ключ, нажмите кнопку Key (ключ) для выбора ключа из диалогового окна Key Access (доступ по ключу). Если вы не выбираете ключ, список показывается в порядке записей, что также отключает возможность установки пределов диапазона.
- Выберите поле из списка, Fields(поля), который появляется с правой стороны диалога Select Field(выбор поля).

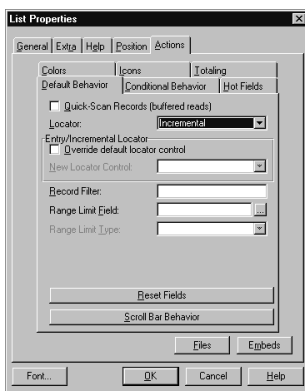
Повторите все это для каждого поля, которое вы хотите добавить к полю списка.

Чтобы добавить переменную к полю списка, выберите Global Data (глобальные данные) или Local Data (локальные данные) из диалога Select Field(выбор поля), выберите необходимую переменную из списка Fields(поля), затем нажмите кнопку Select(выбрать).

После того как будут полностью выбраны файл, ключ и поле (или переменная), появится окно List Field Properties (свойства поля списка). Это диалоговое окно дает вам возможность точно определить поля в списке. В главе Форматер поля списка полностью описаны опции, доступные в этом диалоге.

### **Закладка “Действия “ (Actions )**

Закладка (tab) Actions (действия) диалогового окна List Properties (свойства списка) показывает приглашения шаблонов, позволяющие вам установить многочисленные опции шаблона, а также добавить свой собственный исходный текст для стандартных событий поля списка, таких как перемещение конечным пользователем строки выбора. Диалоговое окно содержит следующие опции:



### **Закладка “Поведение по умолчанию” (Default Behavior)**

Эта закладка содержит приглашения, которые управляют поведением поля просмотра по умолчанию.

#### **Quick-Scan Records**

(Быстрый доступ к записям) Устанавливает режим буферизации доступа к файлам ODBC, ASCII, DOS или BASIC. Эти драйверы файлов читают буфер (не запись), обеспечивая быстрый доступ. В среде многих пользователей эти буферы надежны не на все 100% для последующего доступа, так как другой пользователь может изменить файл между доступами. В качестве защиты драйвер перечитывает буферы перед доступом к каждой записи. Чтобы не допустить перечитывание включите QUICKSCAN.

Locator (Локатор) Локатор дает пользователю возможность искать особые записи в поле списка без необходимости прокручивать вручную весь список. Локатор доступен только при просмотре файла в Порядке ключей (т.е. если установлен КЛЮЧ в Схеме файлов). Сделайте свой выбор типа локатора в ниспадающем списке:

Step (шаг) Назначает локатор требующий для своей активации всего лишь одного нажатия клавиши, без необходимости ввода всего локаторного поля. Когда поле просмотра получает фокус и пользователь вводит с клавиатуры некоторый символ, поле списка перемещается к первому появлению ключевого поля, начинающегося с этого символа (или следующего старшего символа, если ни одно значение ключевого поля не начинается с данного символа локатора). Повторный набор того же самого символа перемещает список к месту следующего появления поля ключа, начинающегося с этого символа.

Entry (Ввод) Назначает локатор связанный с полем ввода в которое вводится искомое значение первичного ключа. Локатор типа Entry активизируется, когда поле ввода принято. Атрибут USE поля ввода является первым свободным компонентом первичного ключа доступа к файлу. Свободная компонента - это компонента, которая не ограничена по диапазону до одиночной величины.

Когда конечный пользователь помещает в поле ввода один или более символов, а затем нажимает TAB или выбирает другой элемент управления на экране, поле списка перемещается к ближайшей подходящей записи.

#### **Incremental**

(инкрементальный) Назначает локатор связанный с полем ввода в которое вводится искомое значение первичного ключа. Локатор типа Incremental активизируется всякий раз когда поле ввода изменяется. Атрибут USE поля ввода является первым свободным компонентом первичного ключа доступа к файлу. Свободная компонента - это компонента, которая не ограничена по диапазону до одиночной величины.

Когда конечный пользователь помещает в поле ввода один или более

символов, поле списка перемещается к ближайшей совпадающей записи. То есть сразу после ввода каждого следующего символа ключевого поля.

None (отсутствует) Локатор не назначается.

Override default locator control

(замена элемента управления Локатор, назначенного по умолчанию) Отметьте это поле чтобы вручную установить элемент управления для использования в качестве локатора с помощью ниспадающего списка New Locator Control (Новый элемент управления Локатор). Это оказывается полезным тогда, когда на одном и том же окне поля просмотра имеется более одного поля ввода.

New Locator Control

(Новый элемент управления Локатор) Выберите элемент управления для использования в качестве локатора из ниспадающего списка.

Record Filter

(фильтр записей) Наберите выражение для ограничения списка в поле просмотра только теми записями, которые соответствуют выражению фильтра. Вы должны связать (BIND) любое поле файла, которое используется в выражении фильтра. Закладка “Горячие поля” (Hot Fields) дает вам возможность связать поля.

Range Limit Field

(Поле ограниченное диапазоном) Вместе с полем Range Limit Type (тип предела диапазона) устанавливает запись или группу записей для включения в поле списка. Наберите имя поля путем нажатия эллиптической (...) кнопки. Предел диапазона зависит от ключа; генерируемая исходная программа будет использовать оператор SET, чтобы найти правильную запись. Это будет возможно только тогда, когда вы назначите ключ KEY для файла в схеме файлов.

Range Limit Type

(тип предела диапазона) Назначает применяемый тип предела диапазона. Выберите нужный из следующего выпадающего списка.

Current Value

(текущее значение) Ограничивает ключ текущим значением Range Limit Field (поле, ограниченное диапазоном).

Single Value

(единственное значение) Дает вам возможность ограничить ключ до единственного значения. Назначьте переменную, содержащую эту величину, в появляющемся поле Range Limit Value (значение предела диапазона)

Range of Values

(диапазон величин) Дает вам возможность установить верхний и нижний пределы. Назначьте переменные, содержащие пределы, в поля Low Limit (нижний предел) и High Limit (верхний предел).

File Relationship

(отношение файлов) Дает вам возможность выбрать ограничивающий диапазон файл из отношения 1:Many (Один-к-многим). Это ограничивает окно просмотра только показом тех дочерних записей, которые соответствуют текущей записи в родительском файле. Например, если ваше поле просмотра было бы списком заказов, вы могли бы ограничиться показом только заказов данного покупателя (в файле Покупатель).

#### Reset Fields button

(кнопка переустановки полей) Нажатие этой кнопки вызывает показ списочного поля, которое позволяет добавить Reset Fields. Если величина какого-либо поля в Reset Fields списке меняется, поле просмотра обновляется.

#### Scroll Bar Behavior button

(кнопка поведения полосы прокрутки) Нажатие этой кнопки вызывает показ диалогового окна, в котором вы можете определить, каким образом работает полоса прокрутки.

#### Scroll Bar Behavior

(поведение полосы прокрутки) Определяет характер работы полосы прокрутки. Выберите Fixed Thumb (фиксированный указатель) и Movable Thumb (подвижный указатель) из выпадающего списка.

#### Key Distribution

(распределение ключа) Эта позиция определяет распределение точек полосы прокручивания. Выберите одно из двух заранее назначенных распределений (алфавитное (Alpha) или по фамилиям (Last Names)) или свое собственное (Custom) или рабочее (Runtime) из ниспадающего списка.

#### Alpha

(алфавитный порядок) Определяет 100 равномерно распределенных в алфавитном порядке точек.

#### Last Names (Фамилии)

Определяет 100 точек, распределенных как обычные фамилии, типичные для Соединенных Штатов. Если ключ доступа отсортирован по цифровым данным, вам следует использовать распределения Custom и Runtime.

#### Custom

Установка пользователя Дает вам возможность назначить свои собственные точки.

#### Run-time

Определение во время исполнения Читает первую и последнюю записи и вычисляет значения для 100 равномерно распределенных в этом промежутке точек.

#### Custom Key Distribution

(собственное распределение ключей) Дает вам возможность установить контрольные точки для распределения вдоль линейки прокрутки (полезно тогда, когда у ваших данных асимметричное распределение). Вставьте величины для каждой точки в список. Строчные константы должны быть заключены в одинарные кавычки (' ').

## Runtime Distribution Parameters

(параметры рабочего распределения) Дает вам возможность назначить тип символов, учитываемых при определении точек распределения. Это годится только тогда, когда Свободным Ключевым Элементом является STRING или CSTRING. Отметьте в полях типы символов, которые вы хотите включить в рассмотрение.

### Закладка “Условное поведение” (Conditional Behavior)

Эта закладка содержит списочное поле, которое дает вам возможность определить характерное поведение, основанное на условиях. Добавьте условия к списку путем нажатия кнопки Insert. Появляется диалоговое окно, в котором вы определяете условие и желаемое поведение, когда это условие выполнено.

Во время работы эти условия оцениваются и используется реакция на условие, соответствующая первому выполняющемуся условию в списке.

В этом диалоговом окне вы можете установить:  
Condition (условие) Любое допустимое выражение.  
Key to Use

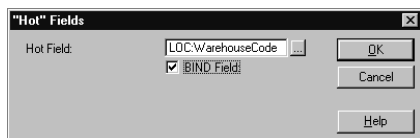
(используемый ключ) Если необходимо, используйте ключ для того, чтобы определить порядок сортировки просмотрového поля, когда это условие выполняется.

Остальные поля и кнопки такие же самые, как и в закладке поведения по умолчанию Default behavior tab.

### Закладка “Горячие поля” (Hot Fields).

Когда вы выбираете закладку горячего поля Hot Fields tab, вы можете выбрать поле (или поля) для сохранения их “живым” в очереди QUEUE. При прокручивании через файл генерированная исходная программа читает данные для этих полей из QUEUE, а не с диска. Это ускоряет обновление спискового поля.

Выбор “горячих” полей дает вам возможность также поместить элементы управления поля файла вне поля просмотра, которое обновляется каждый раз, когда другая запись выбирается в поле списка. Элементы первичного ключа и текущий ключ всегда включены в QUEUE, так что их не нужно вставлять в список горячего поля.



Это диалоговое окно дает вам возможность связывать (BIND) поле. Вы должны связать любое поле файла, которое используется в выражении фильгра или как поле для суммы.

### **Закладка “Суммирование” (Totaling)**

Эта закладка содержит поле списка, которое позволяет вам определить суммарные поля для поля просмотра. Нажмите кнопку INSERT для добавления суммарных полей.

Появится диалоговое окно, в котором вы можете определить суммарные поля для BrowseBox (поле просмотра).

Total Target Field

(итоговое объектное

поле) Переменная для хранения итога. Это может быть локальная, глобальная или модульная переменная. Вы можете также использовать файловое поле; однако, вам нужно писать программу для обновления файла данных.

Total Type (Тип итога) Выберите Count, Sum или Average из выпадающего списка. Count подсчитывает число записей. Sum добавляет величины к полю итога. Average определяет арифметическое среднее поля для итога.

Field to Total

(поле для итога) Поле, которое должно быть суммировано или усреднено. Этот поле недоступно, когда итоговое поле - типа Count.

Total Based On

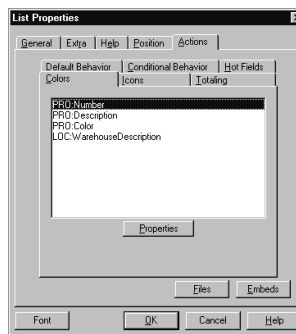
(Итог основан на) Выберите Each Record Read (читать каждую запись) или Specified Condition (оговоренное условие) из выпадающего списка. Это устанавливает, рассматривать ли каждую запись или только те, которые отвечают некоторым критериям фильtringования.

Total Condition

(Условие суммирования) Требуемое условие при использовании итога, основанного на специфическом условии. Вы можете использовать любое допустимое выражение.

### **Закладка “Цвета” (Color)**

Эта закладка доступна только если вы отметите поле флажков Color Cells (ячейки цвета) в формате поля списка. Оно показывает список полей, которые были установлены для окрашивания.



Чтобы назначить цвета, выделите желаемое поле и нажмите кнопку Properties (свойства).

## **Настройка цветов**

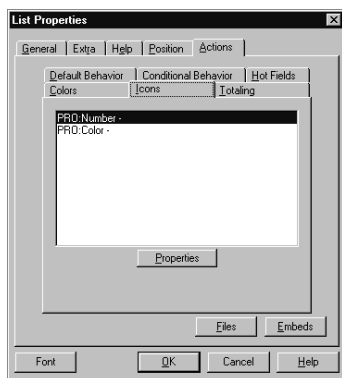
Это диалоговое окно позволяет вам установить цвета переднего плана и для нормальных; строк поля списка и отдельно для выбранных строк (Определение цвета осуществляется отдельно для каждой колонки ).

Ниже раздела цветов по умолчанию имеется список условного назначения цветов. Чтобы добавить условие и выбрать специальные цвета для показа поля при выполнении условия, нажмите кнопку INSERT.

При работе эти условия оцениваются и используются цвета для первого выполненного условия в списке.

## **Закладка “Пиктограммы” (Icons)**

Эта закладка доступна только если вы отметите поле флажков Icons (пиктограммы) в форматере поля списка. Оно показывает список полей, которые были установлены для показа Пиктограммы.



Чтобы выбрать пиктограмму, выделите желаемое поле и нажмите кнопку Properties (свойства).

## **Настройка Пиктограмм поля просмотра**

Это диалоговое окно дает вам возможность выбрать для поля пиктограмму по умолчанию.

Default Icon

(пиктограмма

по умолчанию) Пиктограмма по умолчанию. Вы можете выбрать стандартную пиктограмму или файл пиктограмм (.ICO) на диске.

Conditional Icon Usage

(условное использование

пиктограммы) Ниже раздела пиктограммы по умолчанию имеется список условного использования пиктограмм. Чтобы добавить условие и выбрать специальные пиктограммы для показа при выполнении условия, нажмите



кнопку INSERT. Во время работы эти условия оцениваются и используется пиктограмма для первого выполненного в списке условия.

## **Шаблоны управления обновлением поля просмотра (BrowseUpdateButtons)**

Этот шаблон управления обеспечивают быстрый путь к добавлению стандартных функциональных возможностей для управления записями в поле просмотра данных.

Шаблон управления BrowseUpdateButtons (Кнопки обновления поля просмотра) добавляет три кнопочных элемента управления для манипуляций с записями в поле просмотра. При нажатии кнопок обновления поля ищется соответствующая запись и вызывается процедура, назначенная в качестве процедуры обновления. Нажатие кнопок Change (изменить), Insert (вставить) или Delete (удалить) устанавливает переменную GlobalRequest (глобальный запрос) на 'ChangeRecord' (заменить запись), 'InsertRecord' или 'DeleteRecord' соответственно.

Диалог Properties (свойства) для каждого кнопочного элемента управления идентичен нормальному кнопочному диалогу свойств. Смотрите полное описание в главе Установка свойств управления.

Закладка (tab) Action позволяет вам задать имя процедуры обновления и установить специальные ключи для выполнения действий кнопок.

### **Update Procedure**

(процедура обновления) Введите имя процедуры или выберите из выпадающего списка. Генератор приложений автоматически добавит эту процедуру обновления к дереву процедур.

### **Allow Edit via Popup**

(Разрешение редактирования через всплывающее меню) Отметьте это поле, чтобы включить всплывающее меню, когда пользователь щелкает правой клавишей мыши на поле списка. Всплывающее меню вызывает процедуру обновления, вставки, изменения или удаления записи.

### **When Pressed**

(при нажатии) Стандартный набор приглашений для кнопок (смотрите Установка Свойств кнопки (Button Properties)). Обычно, при использовании шаблона управления, эти приглашения не используются.

## **Шаблон управления выбором в поле просмотра (BrowseSelectButton)**

Этот шаблон управления обеспечивают быстрый путь для выбора записи из списочного поля, когда процедура вызвана для отбора записи.

Сгенерированная программа получает текущую выбранную запись и закрывает поле

просмотра. Для конечного пользователя нажатие на кнопку SELECT эквивалентно двойному щелчку мышью на строке, которую необходимо выбрать в поле просмотра для обработки. Диалоговое окно Properties (свойства) для этой кнопки идентично нормальному диалогу Button Properties (свойства кнопок). Более полное описание вы найдете в главе Элементы управления и их свойства.

Закладка (tab) Action содержит следующее:

Hide the Select button

when not applicable

(скрыть кнопку выбора когда неприменимо)      Определяет, что элемент управления должен быть невидим, когда процедура не вызвана для отбора записи.

Allow Select via Popup

(разрешить выбор из всплывающего меню)

Отметьте это поле, чтобы включить всплывающее меню тогда, когда пользователь щелкает правой клавишей мыши на поле списка. Всплывающее меню вызывает процедуру выбора записи.

When Pressed

(при нажатии) Стандартный набор приглашений для кнопок (смотрите Установка Свойств кнопки). Обычно при использовании шаблона элемента управления, эти приглашения не используются.

## Шаблон кнопки публикации процедуры просмотра

Данный шаблон элемента управления предоставляет кнопку Publish (публикация) для генерации кода на языке разметки гипертекста (Hypertext Markup Language) (HTML) для демонстрации записей из очереди поля просмотра. Иными словами, воспользуйтесь этим шаблоном для публикации вашей информации из поля просмотра на страницу Web в Internet!

Результирующая Web страница показывает строку заголовка, которую вы назначаете, а также заголовки из вашего поля списка. Web страница форматирует данные поля списка с помощью форматов представления значений, установленных в поле списка.

Диалоговое окно Свойства для кнопки публикации процедуры просмотра - это обычное диалоговое окно Свойства кнопки (Button Properties). Более полное описание вы найдете в главе Элементы управления и их свойства. Закладка Actions (Действия) содержит следующие дополнительные опции:

Use variable for HTML name

(используйте переменную для HTML имени)      Отметьте это поле для назначения переменной для имени HTML файла. Это позволяет в поле Variable HTML filename (Переменная имени HTML файла) назвать переменную, и

одновременно деактивирует поле Default HTML Name (HTML имя по умолчанию).

Default HTML Name

(HTML имя по умолчанию)

Назначает имя файла по умолчанию для HTML кода или переменной, которая содержит имя HTML файла.

Нажмите эллиптическую (...) кнопку для выбора файла из стандартного диалогового окна Open File (открыть файл) или для выбора или определения поля словаря данных или переменной памяти из диалогового окна Select Field (Выбор поля).

Если вы не определите полный путь, ваша процедура записывает файл в текущий каталог. Во время работы пользователь может назначить иной файл и иное название пути.

HTML Title

(HTML заголовок)

Назначает заголовок для вашего HTML документа. Заголовок появляется в заголовке Web браузера, когда он показывает документ.

Table Heading

(заголовок таблицы)

Назначает строку заголовка, которая показывается наверху страницы Web.

Background Graphic (фоновая графика)

Назначает графическое изображение, которое показывается “позади” позиций очереди. Обратите внимание на то, что большинство Web браузеров поддерживают графические изображения, однако некоторые старые версии этого делать не могут.

Use Grid Lines

(использование линий сетки) Отметьте это поле для того, чтобы показать позиции очереди в прямоугольной сетке. Обратите внимание на то, что большинство Web браузеров поддерживают линии сетки, однако некоторые старые версии этого делать не могут

Grid Line Width

(ширина линий сетки)

Определяет толщину рамки, окружающей прямоугольную сетку.

### **Рабочие опции**

Во время работы пользователь может назначить имя файла для HTML. Кроме того, пользователь располагает опцией для публикации всех позиций в очереди поля просмотра или только позиций, показываемых в этот момент на экране.

### **Шаблон управления сохранением данных. Кнопка OK. (SaveButton)**

Этот шаблон обеспечивает кнопку ОК для вашего окна плюс способность показывать конечному пользователю сообщение о совершаемом действии. SaveButton также обслуживает большинство действий ввода-вывода файла для данной процедуры.

Диалоговое окно Properties (свойства) для кнопки сохранения - это обычное диалоговое окно Button Properties (свойства кнопки). Полное его описание вы найдете в главе Элементы управления и их свойства.

Закладка (tab) Actions (действия) содержит следующие опции:

Allow (разрешается) Отметьте любую комбинацию трех полей для назначения разрешенных операций I/O файла (ввод-вывод):

Inserts (вставки) Генерирует программу для управления вставкой записи (ADD).

Changes (изменения) Генерирует программу для управления изменением записи (PUT).

Deletes (удаления) Генерирует программу для управления удалением записи (DELETE).

When called for Delete

(при вызове для удаления) Позволяет выбрать, что показывается, когда эта процедура вызвана для удаления записи.

Standard Warning

(стандартное предупреждение) Показывает сообщение, приглашающее подтвердить удаление.

Show Form

(показывать форму) Показывает форму.

Automatic Delete

(автоматическое удаление) Позволяет записи быть удаленной без ее показа или без приглашения для подтверждения удаления.

After successful insert

(после успешной вставки) Выбирает режим вставки по-одной-за-раз или режим с повторением.

Return to caller

(возврат в вызвавшую процедуру) Генерирует RETURN для возврата в вызвавшую процедуру вслед за успешной вставкой. В результате получается режим вставки по-одной-за-раз.

Insert another record

(вставка следующей записи) Не генерирует RETURN для возврата в вызвавшую процедуру после успешной вставки. В результате это приводит к режиму с повторением вставки.

Ask the user before adding another record

(запрашивать пользователя перед добавлением следующей записи) Не генерирует автоматически RETURN для возврата в вызвавшую процедуру вслед за успешной вставкой, а запрашивает пользователя, добавлять ли следующую запись.

Field Priming on insert

(снабжение поля информацией по умолчанию при вставке) Эта процедура дает вам возможность снабдить значениями по умолчанию поля в новой записи. Это значение перекрывает любую первоначальную величину, назначенную в словаре данных. Вы можете выбрать поле и поставить начальное значение

в диалоговом окне Field Priming (заполнение поля информацией).

## Messages and Titles

(сообщения и заголовки) Войдите в диалоговое окно, чтобы выбрать сообщения и их местоположение для показа, нажав для этого кнопку Messages and Titles.



### Insert Message

(сообщение при добавлении) Определяет текст сообщения, когда процедура вызывается для добавления записи. Текст по умолчанию - "Record will be added" (запись будет добавлена).

### Change Message

(сообщение при изменении) Определяет текст сообщения, когда процедура вызывается для изменения записи. Текст по умолчанию - "Record will be changed" (запись будет изменена).

### Delete Message

(сообщение при удалении) Определяет текст сообщения, когда процедура вызывается для удаления записи. Текст по умолчанию - "Record will be deleted" (запись будет удалена).

### On Aborted add/change

(при прерванных добавлениях/изменениях) Определяет действие, которое надо предпринять, когда пользователь нажимает кнопку Cancel во время добавления или модификации записи. Вариантами выбора из ниспадающего списка являются: Offer to save changes (предложение сохранить изменения), Confirm Cancel (подтвердите отмену обновления) или Cancel with out confirming (прервать без подтверждения).

### Location of Message

(местоположение сообщения) Устанавливает, где показывать сообщение. Выберите None/Window Control (нигде/ элемент управления окна) для показа сообщения в элементе управления. Выберите Title Bar (строка заголовка) или Status Bar (строка статуса). Если необходимо, выберите место строки статуса в поле Status Bar Section (выбор места на строке статуса).

### Display Record Identifier on the Title Bar

(показать идентификатор записи в строке заголовка) Дает вам возможность добавить строку символов в конец заголовка на строке заголовков.

## Record Identifier

(идентификатор записи) Устанавливает строку символов для добавления ее в конец строки заголовка, которую вы можете использовать для идентификации записи. Наберите строку символов в поле Record Identifier. Для использования переменного имени предварите его знаком восклицания (!).

## Шаблон управления Кнопки Cancel (прервать)

---

Этот шаблон обеспечивает удобное средство управления окном, которое позволяет пользователю закрыть окно, а разработчику позволяет добавить код “undo” (уничтожать сделанное) при закрытии процедуры.

Сгенерированный исходный код посылает сообщение о закрытии окна. В отличие от приведенного ниже шаблона управления “Close Button” (кнопка закрыть) ( см. дальше), переменная “Local Response” (локально: ответ) принимает значение “ RequestCancelled” (запрос: отказ).

Вы можете вставить программный код “почистить” в точке вставки.

Закладка (tab) Actions (действия) содержит следующее:

### When pressed

(при нажатии) Стандартный набор приглашений для кнопок (смотрите Элементы управления и их свойства ). В обычных условиях, при использовании шаблона управления, эти приглашения не используются

## Кнопка Close (закрыть)

---

Этот шаблон элемента управления добавляет управление одиночной кнопкой, обозначенной как Close(закрыть). При нажатии на эту кнопку сгенерированная программа закрывает текущее окно. С помощью кнопки Action (действие) вы можете определить конкретно, что должно будет происходить, когда конечный пользователь нажмет на эту кнопку.

Диалоговое окно, определяющее свойства для этой кнопки, подобно обычному диалогу определения свойств кнопки Button Properties. Полное описание этого окна вы можете найти в главе Определение свойств элементов управления.

Закладка (tab) Action вызывает диалоговое окно, содержащее следующее:

### When Pressed

(при нажатии) Стандартный набор приглашений для кнопок (смотрите Элементы управления и их свойства ). В обычных условиях, при использовании шаблона элемента управления, эти приглашения не используются.

## Просмотр текстовых файлов ( ASCIIBox )

---

Этот шаблон элемента управления добавляет поле списка, в котором вы можете показать ASCII (текстовый) файл. Если вы хотите просматривать всегда один и тот же ASCII файл, вы можете уточнить имя файла в диалоговом окне Prompts (подсказки).

Закладка (tab) Action вызывает диалоговое окно содержащее следующее:

Description (описание)      Дает вам возможность добавить описание и показать его в окне индикатора выполнения, которое показывается при открывании файла.

File Name to View

(имя файла для View) Устанавливает путь и имя файла для просмотра или переменную, предваряемую знаком восклицания (!).

Display Number of Bytes Read

(показ числа прочитанных байт)      Отметьте это поле, если вы хотите показать размер файла в диалоговом окне индикатора выполнения .

Warn the user if the file cannot be found?

(предупредить пользователя если файл не может быть найден?)      Отметьте это поле, если вы хотите показать сообщение во время работы, когда выбранный файл не может быть найден.

## Кнопка управления печатью ASCII файлов (ASCII Print Button)

---

Этот шаблон добавляет кнопку Print и связанную с ней программу печати ASCII файла (текстового). Используйте этот шаблон совместно с шаблоном поля просмотра ASCII файла.

Редактируйте Actions только в том случае, если вы желаете добавить другое, отдельное действие, которое должно иметь место после печатания. Все программы, необходимые для управления собственно печатанием, исполняются автоматически.

Закладка (tab) Action вызывает диалоговое окно, содержащее следующее:

When pressed

(при нажатии кнопки)      Стандартный набор приглашений для кнопок (смотрите Элементы управления и их свойства ). В нормальных условиях, при использовании шаблона управления, эти приглашения не используются.

## Кнопка поиска в ASCII - файлах (ASCII Search )

---

Этот шаблон добавляет две кнопки: Find и Find Next (найти и найти следующий) и связанную с ними программу, необходимую для функционирования модального диалога поиска, позволяющего конечному пользователю найти заданный текст в ASCII (текстовом) файле. Используйте этот шаблон элемента управления совместно с шаблоном поля

просмотра ASCII файла.

Редактируйте Actions только в том случае, если вы желаете добавить другое, отдельное действие, которое должно иметь место после поиска. Все программы, необходимые для управления поиском, исполняются автоматически.

Закладка (tab) Action вызывает диалоговое окно, содержащее следующее:

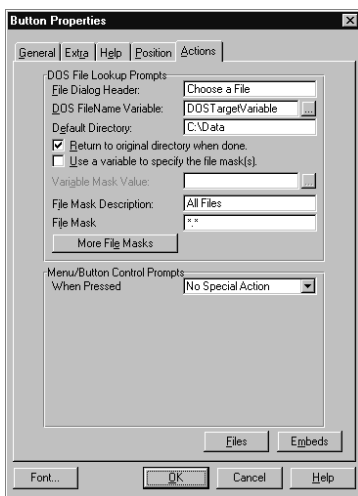
When pressed

(при нажатии)Стандартный набор приглашений для кнопок (смотрите Элементы управления и их свойства). В нормальных условиях, при использовании шаблона элемента управления, эти приглашения не используются.

## **Доступ к стандартному диалогу выбора файлов (DOSFileLookup)**

Этот шаблон элемента управления добавляет эллиптическую кнопку (...), которая вызывает стандартное диалоговое окно Open File (открыть файл). Вы можете установить маску файла и указать переменную для хранения имени выбранного конечным пользователем файла.

Диалог Properties(свойства) для DOSFileLookup идентичен обычному диалогу Button Properties. Полное описание вы можете найти в главе Элементы управления и их свойства.



Закладка (tab) Actions вызывает диалоговое окно, содержащее следующее:

File Dialog Header

(заголовок диалога файла) Наберите текст для заголовка диалогового окна Open File.



### DOS FileName Variable

(переменная имени DOS файла) Нажмите эллиптическую кнопку для просмотра диалогового окна File Schematic (схема файла) и выберите имя файловой переменной. Вы можете набрать имя переменной непосредственно в поле ввода.

### Default Directory

(Оглавление по умолчанию) Позволяет вам установить имя оглавления, из которого запускается диалог Open File.

### Return to original directory when done

(возврат к первоначальному каталогу после выполнения) Отметьте это поле для того, чтобы переустановить рабочий каталог на его значение до момента доступа к стандартному диалогу выбора файлов.

### Use a variable to specify the file mask(s)

(использование переменной для назначения маски файла(ов)) Отметьте это поле для того, чтобы определить переменную маски файла. Это позволяет в поле Variable Mask Value (переменная для маски) поименовать переменную, а также деактивирует поля Mask Description (описание маски), File Mask (маска файла) и More File Masks (дополнительные маски файла).

### Variable Mask Value

(переменная маски файла) Называет переменную, которая содержит маску данного файла.

### Mask Description

(описание маски) Наберите описание типа файла. В диалоговом окне Open File в выпадающем списке появляется набор типов файлов. Вы можете добавить дополнительные маски путем нажатия кнопки More File Masks (дополнительные маски файла).

### File Mask

(маска файла) Наберите спецификацию файла, такую как "\*.TXT." или используйте мультишаблоны для этой маски, отделяя каждый точкой с запятой как в "\*.BMP;\*.GIF".

### More File Masks

(дополнительные маски файла) Если необходимо, нажмите кнопку для добавления масок файла. Эти становятся доступными для пользователя из выпадающего списка в диалоговом окне File Open (открыть файл).

### When Pressed

(при нажатии) Стандартный набор приглашений для кнопок (смотрите Установка свойств управления). В нормальных условиях, при использовании шаблона элемента управления, эти приглашения не используются.

## OOPASCIIViewer

---

Этот шаблон элемента управления по своим функциям эквивалентен шаблону элемента управления ASCIIBox. Однако он обретает свои функциональные возможности через объектно-ориентированный код и является учебным примером, равно как и ценным средством просмотра текста.

Он добавляет поле списка, в котором вы можете показать ASCII (текстовый) файл. Если вы хотите просматривать тот же самый ASCII файл все время, вы можете назначить имя файла в диалоговом окне Prompts (приглашения).

Закладка Actions (действия) содержит следующие приглашения:  
File Name to View

(имя файла для просмотра) Назначает путь и имя файла для просмотра или переменную, предваряемую восклицательным знаком (!).

Warn the user if the file cannot be found?

(предупреждать пользователя, если файл не может быть найден?) Отметьте это поле, если вы хотите показать сообщение во время работы в случае, когда назначенный файл не может быть найден.

Display Loading Window

(показать окно загрузки) Отметьте это поле для того, чтобы вывести окно прогресса, которое показывает процент прочитанного ASCII файла.

Description (описание) Дает вам возможность добавить описание для показа его в окне прогресса; оно появляется при открывании файла.

Display Number of Bytes Read

(показ числа прочитанных байтов) Отметьте это поле, если вы хотите показать размер файла в диалоге прогресса.

## Шаблон элемента управления FileDrop

---

Этот шаблон элемента управления просматривает файл данных и присваивает величину выбранного поля переменной ?Use элемента управления списком. Это не позволяет вводить данные. Если вы хотите располагать возможностью вводить данные и выбирать данные, воспользуйтесь шаблоном элемента управления FileDropCombo.

Имеются два различных сценария, для которых вы можете использовать этот шаблон:

- ◆ Показ, затем сохранение показанных данных.
- ◆ Показ данных, затем сохранение связанного кода.

### Хранение и показ одних и тех же данных

В этом сценарии вы можете выбрать величину из справочного файла и перенести ее в первичный файл. Например, файл "Продукция" с полем, содержащим цвета деталей, с

файлом - справочным файлом цветов.

В этом случае заполните приглашения следующим образом:

**Свойства:**

?Use Поле, которому присваивается величина из поля в справочном файле.

Field to Fill From

(поле, из которого заполнять) Поле из справочного файла. Эта величина присваивается объектному полю.

Remove Duplicates

(удалить дубликаты) Отметьте это поле для удаления дубликатов из показываемого списка.

Target Field

(объектное поле) Поле, которому величина присваивается из поля в справочном файле. В этом случае это то же самое, что и переменная ?USE.

Record Filter

(фильтр записей) Если необходимо, наберите выражение для ограничения содержимого выпадающего списка до тех только записей, которые соответствуют выражению фильтра.

Default to First entry if Use Variable empty

(по умолчанию к первому вводу если переменная Use пустая) Автоматически присваивает величину первого поля в списке переменной ?USE. Поля в списке сортируются в алфавитном порядке (если вы не установите Sort Fields (сортировать поля)).

**Показ текстовых данных и сохранение кода**

В этом сценарии вы можете выбрать величину из текстового поля в справочном файле и перенести ассоциированный с этим текстом код в первичный файл. Например, файл “Продукции” с полем, хранящим код “Размещения”, с файлом - справочником “Размещения”. Вы хотите, чтобы конечный пользователь выбрал “Размещение” из списка разных размещений, но хранил бы номер размещения в файле продукции.

В этом случае заполняйте приглашения следующим образом:

?UseСоздать локальную переменную, которая соответствует текстовому полю.

С помощью List Box Formatter (форматер поля списка) заполните список текстовым полем из файла подстановки. Он автоматически присваивается переменной ?Use.

Field to Fill From

(поле, из которого заполнять) Поле из файла подстановки. Эта величина присваивается объектному полю.

Remove Duplicates

(удалить дубликаты) Отметьте это поле для удаления дубликатов из показанного

списка.

### Target Field

(объектное поле) Поле, которому присваивается величина из поля в файле подстановки.

### Record Filter

(фильтр записей) Если необходимо, наберите выражение для ограничения содержимого ниспадающего списка до тех только записей, которые соответствуют выражению фильтра.

### Default to First entry if Use Variable empty

(по умолчанию к первому вводу если переменная Use пустая) Автоматически присваивает величину первого поля в списке переменной ?USE. Поля в списке сортируются в алфавитном порядке (если вы не установите Sort Fields (сортировать поля)).

### Свойства:

Список свойств List Properties для этого элемента управления тот же самый; однако поле ввода From требует некоторых объяснений.

From: При помещении элемента управления File Drop это поле заполняется с помощью Queue:FileDrop. Это вы не должны изменять.

Внимание: Вы можете воспользоваться List Box Formatter (форматер списочного поля) для заполнения данного элемента управления, однако только первое заполненное поле допустимо для присвоения.

### Закладка “Поля сортировки” (Sort Fields)

Эта закладка позволяет добавлять поля, по которым сортируется список. Порядок сортировки независим от ключей. Нажмите кнопку Insert чтобы добавить поле к списку. Эта процедура сортирует список динамически во время работы.

### Закладка “Пределы диапазона” (Range Limit)

Эта закладка появляется только после того, как вы назначите ключ для файла, ассоциированного с этим элементом управления.

### Range Limit Field

(Поле ограниченное диапазоном) Вместе с полем Range Limit Type (тип предела диапазона) устанавливает запись или группу записей для включения в поле списка. Выберите поле путем нажатия эллиптической (...) кнопки. Предел диапазона зависит от ключа. Сгенерированная программа будет использовать оператор SET, чтобы найти первую правильную запись.

### Range Limit Type

(тип предела диапазона) Назначает применяемый тип ограничения диапазона.

Выберите один из следующих типов из ниспадающего списка.

Current Value

(текущее значение) Ограничивает ключ текущим значением Поля, ограниченного диапазоном.

Single Value

(простое значение) Дает вам возможность ограничить ключ одиночным значением. Назначьте переменную, содержащую это значение, в появляющемся поле Range Limit Value (значение допустимого диапазона)

Range of Values

(диапазон величин) Дает вам возможность назначить верхний и нижний пределы этого диапазона, Назначьте переменные, содержащие диапазон, в полях Low Limit (нижний предел) и High Limit (верхний предел)

File Relationship

(отношение файлов) Дает вам возможность выбрать ограничивающий диапазон файл из отношения 1:Many (Один-к-многим). Это ограничивает окно просмотра только показом тех дочерних записей, которые соответствуют текущей записи в родительском файле. Например, если ваше поле просмотра было списком заказов, вы могли ограничиться показом только заказов данного покупателя (в файле Покупатель).

### **Закладка “Цвета” (Color)**

Эта закладка доступна только если вы отметите поле флажков Color Cells (ячейки цвета) в форматере поля списка. Оно показывает список полей, которые были установлены для окрашивания.

Чтобы назначить цвета, выделите желаемое поле и нажмите кнопку Properties (свойства).

### **Настройка цветов**

Это диалоговое окно дает вам возможность установить цвета по умолчанию переднего плана и заднего плана для нормальных строк поля списка и отдельно для выбранных строк (Определение цвета осуществляется отдельно для каждой колонки ).

Ниже раздела цветов по умолчанию имеется список условного назначения цветов. Чтобы добавить условие и выбрать специальные цвета для показа поле при выполнении условия, нажмите кнопку INSERT.

При работе эти условия оцениваются и используются цвета для первого выполненного условия в списке.

### **Закладка “Пиктограммы” (Icons)**

Эта закладка доступна только если вы отметите поле флажков Icons (пиктограммы) в форматере поля списка. Оно показывает список полей, которые были установлены для

показа Пиктограмм. Чтобы выбрать пиктограммы, выделите желаемое поле и нажмите кнопку Properties (свойства).

### **Настройка Пиктограмм поля просмотра**

Это диалоговое окно дает вам возможность выбрать для поля пиктограмму по умолчанию.

#### **Default Icon**

(пиктограмма по умолчанию) Пиктограмма по умолчанию. Вы можете выбрать стандартную пиктограмму или файл пиктограмм (.ICO) на диске.

#### **Conditional Icon Usage**

(Условное использование пиктограммы) Ниже раздела пиктограммы по умолчанию имеется список условного использования пиктограмм. Чтобы добавить условие и выбрать специальные пиктограммы для показа при выполнении условия, нажмите кнопку INSERT. Во время работы эти условия оцениваются и используется пиктограмма для первого выполненного в списке условия.

## **Шаблон элемента управления FileDropCombo**

Этот шаблон просматривает файл данных и приписывает величину выбранного поля переменной ?Use. Он также позволяет добавлять записи путем набора новой величины в вводную часть комбинированного поля.

Имеются два различных сценария, для которых вы можете использовать этот шаблон:

- ◆ Хранение и показ собственно данных из справочника
- ◆ Показ текстовых данных и хранение ассоциированного с текстом кода.

### **Хранение и показ одних и тех же данных**

В этом сценарии вы можете выбрать величину из справочного файла и хранить ее в первичном файле. Например, файл “Продукция” с полем, хранящим цвет детали в сочетании с файлом, содержащим цвета.

В этом случае заполните приглашения следующим образом:

Свойства:

?UseПоле, которому присваивается величина из поля в справочном файле .

#### **Field to Fill From**

(поле, из которого заполнять) Поле из справочного файла . Эта величина присваивается объектному полю.

#### **Remove Duplicates**

(удалить дубликаты) Отметьте это поле для удаления дубликатов из показываемого списка.

**Target Field**

(объектное поле) Поле, которому величина присваивается из поля в файле подстановки.

В этом случае это то же самое, что и переменная ?USE.

**Record Filter**

(фильтр записей) Если необходимо, наберите выражение для ограничения содержимого выпадающего списка до тех только записей, которые соответствуют выражению фильтра.

**Default to First entry if Use**

Variable empty (по умолчанию к первому вводу, если переменная Use пустая)

Автоматически присваивает величину первого поля в списке переменной ?USE. Поля в списке сортируются в алфавитном порядке (если вы не установите Sort Fields (сортировать поля)).

**Проведение обновления**

В этом сценарии форма Form HE нужна для обновления файла подстановок. Проверка поля Allow Updates (разрешить обновление) дает возможность обновить поле прямо из этого элемента управления.

**Показ текстовых данных и хранение программы**

В этом сценарии вы можете выбрать величину из текстового поля в справочном файле и перенести ассоциированный с этим текстом код в первичный файл. Например, файл Продукции с полем, хранящим код размещения с файлом - справочником Размещения. Вы хотите, чтобы конечный пользователь выбрал Размещение из списка различных размещений, но хранил номер размещения в файле Продукции.

В этом случае заполняйте приглашения следующим образом:

?UseСоздать локальную переменную, которая соответствует текстовому полю.

С помощью List Box Formatter (форматер поля списка) заполните список текстовым полем из файла подстановки. Он автоматически присваивается переменной ?Use.

**Field to Fill From**

(поле, из которого заполнять) Поле из файла подстановки. Эта величина присваивается объектному полю.

**Remove Duplicates**

(удалить дубликаты) Отметьте это поле для удаления дубликатов из показанного списка.

**Target Field (объектное поле)** Поле, которому величина присваивается из поля в файле подстановки.

**Record Filter (фильтр записи)** Если необходимо, наберите выражение для ограничения содержимого выпадающего списка до тех только записей, которые соответствуют выражению фильтра.

**Default to First entry if Use Variable empty**

(по умолчанию к первому вводу если переменная Use пустая) Автоматически присваивает величину первого поля в списке переменной ?USE. Поля в

списке сортируются в алфавитном порядке (если вы не установите Sort Fields (сортировать поля)).

### **Поведение обновления**

В данном сценарии форма (Form) нужна для обновления файла подстановки; если вы хотите разрешить обновления, определите соответствующую процедуру типа “Форма”.

#### **Свойства:**

Список свойств List Properties для этого элемента управления тот же самый; однако поле ввода From требует некоторых объяснений.

From: При помещении элемента управления File Drop Combo это поле заполняется с помощью Queue:FileDropCombo. Вы не должны изменять это поле.

Внимание: Вы можете воспользоваться List Box Formatter (форматер списочного поля) для заполнения данного элемента управления, однако только первое заполненное поле допустимо для вводной части элемента управления

### **Закладка “Пределы диапазона” (Range Limit)**

Эта закладка (tab) становится доступной только после того, как вы назначите ключ для файла, ассоциированного с этим элементом управления.

#### **Range Limit Field**

(Поле, ограниченное диапазоном) Вместе с полем Range Limit Type (тип предела диапазона) устанавливает запись или группу записей для включения в поле списка. Выберите поле путем нажатия эллиптической (...) кнопки. Предел диапазона зависит от ключа. Сгенерированная программа будет использовать оператор SET, чтобы найти первую правильную запись.

#### **Range Limit Type**

(тип предела диапазона) Назначает применяемый тип ограничения диапазона. Выберите один из следующих типов из ниспадающего списка.

#### **Current Value**

(текущее значение) Ограничивает ключ текущим значением Поля, ограниченного диапазоном.

#### **Single Value**

(единственное значение) Дает вам возможность ограничить ключ одиночным значением. Назначьте переменную, содержащую это значение, в появляющемся поле Range Limit Value (величина допустимого диапазона)

#### **Range of Values**

(диапазон величин) Дает вам возможность назначить верхний и нижний пределы этого диапазона, Назначьте переменные, содержащие диапазон, в полях Low Limit (нижний предел) и High Limit (верхний предел)



## File Relationship

(отношение файлов)

Дает вам возможность выбрать ограничивающий файл диапазон из отношения 1:Many (Один-к-многим). Это ограничивает окно просмотра только показом тех дочерних записей, которые соответствуют текущей записи в родительском файле. Например, если ваше поле просмотра было списком заказов, вы могли ограничиться показом только заказов данного покупателя (в файле Покупатель).

### **Закладка “Цвета” (Color)**

Эта закладка доступна только если вы отметите поле флажков Color Cells (ячейки цвета) в формате поля списка. Оно показывает список полей, которые были установлены для окрашивания.

Чтобы назначить цвета, выделите желаемое поле и нажмите кнопку Properties (свойства).

### **Настройка цветов**

Это диалоговое окно позволяет вам установить цвета переднего и заднего плана для нормальных строк поля списка и отдельно для выбранных строк (Определение цвета осуществляется отдельно для каждой колонки).

Ниже раздела цветов по умолчанию имеется список условного назначения цветов. Чтобы добавить условие и выбрать специальные цвета для показа поля при выполнении условия, нажмите кнопку INSERT.

При работе эти условия оцениваются и используются цвета для первого выполненного условия из списка.

### **Закладка “Пиктограммы” (Icons)**

Эта закладка доступна только, если вы отметите поле флажков Icons (пиктограммы) в формате поля списка. Оно показывает список полей, которые были установлены для показа Пиктограммы. Чтобы выбрать пиктограмму, выделите желаемое поле и нажмите кнопку Properties (свойства).

### **Настройка Пиктограмм поля просмотра**

Это диалоговое окно позволяет вам выбрать для поля пиктограмму по умолчанию.

#### Default Icon

(пиктограмма по умолчанию)

Пиктограмма по умолчанию. Вы можете выбрать стандартную пиктограмму или файл пиктограмм (.ICO) на диске.

#### Conditional Icon Usage

(Условное использование пиктограммы) Ниже раздела пиктограммы по умолчанию имеется

список условного использования пиктограммы. Чтобы добавить условия и выбрать специальные пиктограммы для показа по выполнению условия, нажмите кнопку INSERT. Во время работы эти условия оцениваются и используется пиктограмма для первого выполненного в списке условия.

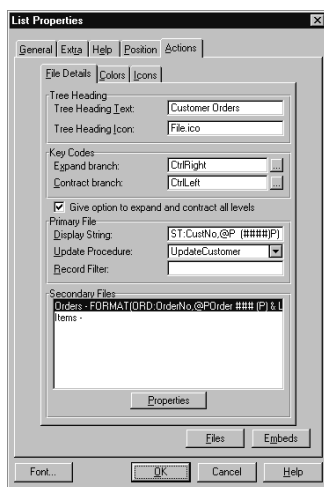
## Шаблон Дерево отношений (Relation Tree)

Элемент управления “Дерево отношений” - это поле списка, отформатированное для показа в виде дерева. Данный шаблон обеспечивает альтернативу для парадигмы Просмотр-Форма. Один единственный элемент управления Дерево Отношений может заменить несколько пар Просмотр-Форма.

С помощью шаблона Relation Tree вы можете установить множество связанных файлов

для показа на множестве уровней иерархического списка. Элемент управления “Дерево отношений” может показывать неограниченное число файлов с отношениями - с ассоциированной процедурой обновления для каждого уровня. Связанные файлы заявляются в Схеме файлов - Первичный (родительский) файл и одиночная цепь связанных вторичных дочерних файлов (Родитель-Дети-Внуки).

Шаблон Дерево отношений использует полностью загруженную очередь QUEUE для корневого уровня. Дочерние уровни загружаются по требованию, когда ветвь расширяется. Этот шаблон не подходит для баз данных с очень большим первичным файлом. Вместо него вам следует в этом случае использовать шаблон BrowseBox (поле просмотра), который загружается страницами.



Чтобы создать дерево с помощью шаблона Дерево отношений:

1. Поместите шаблон элемента управления Relation Tree в окно. Появляется диалоговое окно List Box Formatter (форматер поля списка). Не пользуйтесь форматером для заселения вашего дерева.
2. Если вы хотите включить окрашивание или показ пиктограммы в своем элементе управления Дерево, нажмите кнопку Properties (свойства) на List Box Formatter и отметьте соответствующие поля.
3. Нажмите кнопку OK на List Box Formatter.
4. Щелкните правой клавишей на шаблоне Дерево отношений и выберите Actions (действия) из всплывающего меню.

5. Нажмите кнопку Files (файлы) для назначения схемы файлов для данного элемента управления.

6. Установите детали для файлов:

Tree Heading Text

(текст, возглавляющий дерево)      Необязательный текст, возглавляющий вершину дерева. Текст во главе дерева требуется для того, чтобы дать возможность пользователю добавить запись на корневом уровне.

Tree heading Icon

(пиктограмма, возглавляющая дерево)      Необязательная пиктограмма в вершине дерева. Для того чтобы использовать пиктограммы, нужно чтобы они были разрешены для включения в формате поля списка.

Expand Branch

(расширить ветвь)      Назначает клавишу нажатие которой вызывает расширение списка для выбранной позиции - то есть показ ее дочерних записей. Нажмите эллиптическую кнопку (...) для того, чтобы выбрать специальные клавиши, такие как ESC, TAB или ENTER. Более полную информацию об этом диалоговом окне вы можете найти в разделе Элементы управления - Обычные атрибуты элементов управления - Установка атрибута КЛАВИША.

Contract Branch (сжать ветвь)

Назначает клавишу нажатие которой вызывает сжатие отображенного списка позиций - чтобы спрятать дочерние. Нажмите эллиптическую кнопку (...) для того, чтобы выбрать специальные клавиши, такие как ESC, TAB или ENTER. Более полную информацию об этом диалоговом окне вы можете найти в разделе Элементы управления - Обычные атрибуты элементов управления - Установка атрибута КЛАВИША

Give option to expand and contract all levels

(дать опцию расширения и сжатия всех уровней)      Назначает всплывающее меню, вызываемое щелчком правой клавишей мыши и предназначенное для дерева отношений, оно содержит команды Сжать все и Расширить все.

### **Установки первичного файла**

Display String

(показ строки символов)      Предназначено для показа имени файла или текста для уровня первичного файла. Это может быть любое правильное выражение Clarion, например: CLIP(CUST :LastName)&' '&CUST:FirstName

Update Procedure

(процедура обновления)      Процедура обновления, вызываемая для первичного файла. Доступ к процедуре может быть получен с помощью вызываемого правой

клавишей мыши всплывающего меню, автоматически сопутствующего дереву отношений, или же к ней можно получить доступ с помощью Кнопок Обновления Деревя Отношения - смотрите ниже.

#### Record Filter

(фильтр записей)

По желанию, наберите выражение для ограничения содержания списка только теми записями первичного файла, которые соответствуют выражению фильтра.

### **Установки вторичного файла**

По желанию, установите показываемые строки символов, процедуры обновления и фильтры записей для любых вторичных файлов. Выделите вторичный файл и нажмите кнопку Properties (свойства) под полем списка Secondary Files (вторичные файлы).

#### Display String

(показ строки символов)

Предназначенные для показа имя файла или текст для уровня вторичного файла. Это может быть любое правильное выражение Clarion, например: `ORD:OrderNo & LEFT(FORMAT(ORD:OrderDate,@D1))`

#### Update Procedure

(процедура обновления)

Процедура обновления, вызываемая для вторичного файла. Доступ к процедуре может быть получен с помощью вызываемого правой клавишей мыши всплывающего меню, автоматически сопутствующего дереву отношений, или же к ней можно получить доступ с помощью Кнопок Обновления Деревя Отношения - смотрите ниже.

#### Record Filter

(фильтр записи)

По желанию, наберите выражение для ограничения содержания списка только теми записями вторичного файла, которые соответствуют выражению фильтра.

### **Вызов процедур обновления**

Одна из наиболее мощных характеристик шаблона элемента управления “Дерево отношений” - это способность вызвать процедуру обновления для выбранного уровня дерева.

Процедура обновления вызывается тогда, когда пользователь дважды щелкнет клавишей мыши на позиции в иерархическом списке.

Другой способ заключается в том, что щелчок правой клавишей вызывает всплывающее меню, которое, в свою очередь, вызывает процедуру обновления для вставки, изменения или удаления записей. Меню показывает текст, видимый на ассоциированных кнопках RelationTreeUpdateButtons (обновление дерева отношений).

Третий метод вызова процедуры обновления - поместить шаблон процедуры

RelationTreeUpdate Buttons на окно.

### **Шаблон Кнопок обновления дерева отношений (RelationTreeUpdateButtons)**

---

Этот шаблон добавляет три кнопки (Insert, Change и Delete), что позволяет пользователю вызвать ассоциированную процедуру обновления для выбранного уровня Деревя отношений (если процедура обновления была назначена). Для этого элемента управления нет приглашений. Процедура обновления устанавливается для каждого уровня шаблона элемента управления деревом отношений.

Кнопки Change и Delete соответствуют текущей выделенной записи. Кнопка Insert добавляет дочернюю запись (следующий уровень вниз по структуре дерева).

## Кодовые шаблоны

Кодовые шаблоны генерируют исполняемый код. Назначением этих шаблонов является выполнение окончательной настройки генерируемого программного текста - добавление вставок программного текста, которые заставляют работать программу именно так, как вам это необходимо. Каждый кодовый шаблон имеет одну хорошо определенную задачу. Например, кодовый шаблон Инициализации процесса(thread) просто запускает новый процесс, и более ничего. Обыкновенно кодовый шаблон обеспечивает диалоговое поле с опциями и инструкциями.

Добавляйте кодовые шаблоны к своим процедурам с помощью диалогового окна Embedded Source (вставка программного текста). Смотрите об этом в разделе Генератор приложений - Встроенная исходная программа.

Clarion for Windows содержит следующие кодовые шаблоны:

### **Инициировать процесс (thread)**

---

При открытии MDI окна из головной процедуры приложения вы должны запустить процесс(thread). Данный шаблон программы обеспечивает удобный способ инициализации процесса.

В диалоговом окне Prompts for Initiate Thread (подсказки для инициализации процесса) вы просто должны назвать имя процедуры, которая открывает MDI окно.

По желанию вы можете добавить строку кода для исполнения в том случае, если приложение было не в состоянии открыть процесс. Введите этот код в поле редактирования, помеченном Error Handling (работа с ошибками).

Например

```
BEEP ; MESSAGE('Couldn't start thread!', 'Error', Icon:Hand)
```

вызовет звук “биип” и покажет поле сообщений с пиктограммой СТОП (изображение поднятой руки), если процесс не стартовал.

Вы можете добавить имя процедуры, чтобы организовать подачу звукового сигнала об ошибке, набирая имя соответствующей процедуры в поле Error Handling. Затем вы должны будете добавить процедуру к Application Tree с помощью команды Insert Procedure.

### **Вызов процедуры доступа к справочнику (CollProcedureAsLOOKUP )**

---

Этот кодовый шаблон помогает вызвать процедуру для выбора записи из справочника. Он устанавливает значение переменной LocalResponse в зависимости от того, был ли выбор успешно завершен.

Lookup Procedure

(процедура Lookup) Назначает вызываемую процедуру.

**Code before (код прежде)** Наберите исполняемую программу для выполнения перед lookup. Мультиоператоры можно использовать, если разделить их точкой с запятой.

**Code After, Completed**  
(код после, выполнено) Наберите исполняемую программу для выполнения после завершения lookup. Мультиоператоры можно использовать, если разделить их точкой с запятой

**Code After, Canceled**  
(код после, отменено) Наберите исполняемую программу для выполнения, если lookup отменено. Мультиоператоры можно использовать, если разделить их точкой с запятой

## **Шаблон проверки достоверности данных (ControlValueValidation)**

---

Этот кодовый шаблон получает величину от элемента управления и сравнивает ее с величиной в ключе. Вы можете добавить этот кодовый шаблон к событию “ENTRY” (“Нажатие клавиши ENTRY”), в точке вставки Accepted (принято) или Selected (выбрано). Код, генерируемый этим кодовым шаблоном, получает величину, введенную в поле и затем сравнивает ее с заданной величиной.

Этот шаблон может также вызвать процедуру поиска, чтобы дать возможность конечному пользователю выбрать величину из списка. Вы можете установить, каков результат поиска, прерыв значение переменной LocalResponse.

## **Поиск записи в несвязанном файле (LookupNonRelatedRecord)**

---

Этот кодовый шаблон используется для выполнения процедуры поиска величины, основанной на отношении, независимо от того, определена она или нет в словаре данных (незапланированное отношение). Вы можете добавить этот кодовый шаблон к точке встраивания Look Up Related Records (поиск связанных записей).

**Lookup Key**

(ключ поиска) Наберите имя ключа или нажмите эллиптическую (...) кнопку для выбора ключа из схемы файлов.

Ключ поиска используется для выполнения поиска в файле поиска. Он должен быть уникальным ключом. Если ключ - многокомпонентный ключ, другие элементы ключа должны быть проинициализированы перед исполнением этого кодового шаблона.

**Lookup Field**

(поле поиска) Наберите имя поля или нажмите эллиптическую (...) кнопку для выбора поля из списка компонентов.

Поле поиска Lookup Field должно быть компонентом ключа

просмотра. Это должна быть уникальная величина в пределах файла поиска.

### Related Field

(поле отношения) Наберите имя поля отношения или нажмите эллиптическую (...) кнопку для выбора его из схемы файла.

Поле отношения обеспечивает уникальную величину, используемую для выполнения поиска.

Этот кодовый шаблон генерирует следующий код:

LookUpField = RelatedField	! Move value for lookup
GET(LookUpFile,LookUpKey)	! Get value from file
IF ERRORCODE( )	! IF record not found
CLEAR(LookupfileRecord)	! Clear the record buffer
END	! END (IF record not found)

### **Закреть текущее окно (CloseCurrentWindow)**

---

Этот кодовый шаблон просто посылает событие EVENT:CloseWindow (СОБЫТИЕ:ЗакретьОкно), которое закрывает действующее в данное время окно. Здесь нет подсказок для заполнения.



## Шаблоны расширений

Шаблоны расширений добавляют функциональные возможности к процедурам, но они не привязаны к какому-либо элементу управления или точке вставки. Каждый шаблон расширения имеет одну хорошо определенную задачу. Например, Date Time Display дает вам возможность показывать дату и работающие часы.

Из диалогового окна Procedure Properties (свойства процедур) добавьте шаблон расширения, нажав для этого кнопку Extensions (расширения).

Внимание: Только шаблоны расширений могут быть добавлены и удалены с помощью кнопки Extensions (расширения). Шаблоны элементов управления могут быть здесь модифицированы, но не могут быть добавлены или удалены. Для добавления или удаления шаблонов элементов управления пользуйтесь Форматером окна.

Clarion for Windows включает следующие шаблоны расширений:

### **Шаблон показа даты, времени (DateTimeDisplay)**

---

Этот шаблон расширения добавляет к шаблону процедуры возможность показывать время и/или дату в строке статуса или в элементе управления.

Опции, появляющиеся в диалоговом окне Показа даты и времени (Date Time Display) разделены на две группы полей - Date Display и Time Display -

Display in Window

(показ в окне)      Отметьте поле или поля для добавления к вашему окну.

Picture

(Шаблон изображения)      Выберите изображение для показа даты и/или времени из выпадающего списка. Список показывает примеры, такие как “October 31, 1959,” и “5:30P.M.”

Other Picture

(другое изображение)      Наберите шаблон изображения по вашему выбору, если тип изображения, который вам нужен, не появляется в списке. Смотрите также: Шаблоны изображения даты или Шаблоны изображения времени в Справочнике Языка.

Day of Week

(день недели) (Только для даты).      Если необходимо, показывает дни недели.

Location

(размещение)      Выберите между показом даты и/или времени на строке статуса или в элементе управления.

Status Bar Section

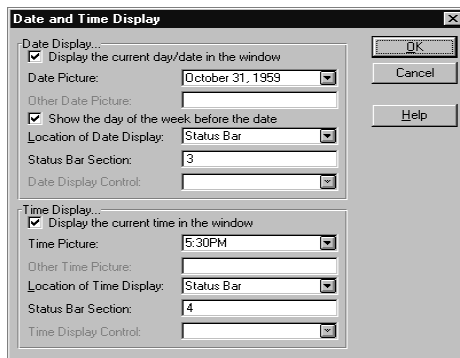
(участок строки статуса)      Когда на строке статуса должны появиться дата или время,

уточните номер сектора строки статуса.

## Display Control

(управление показом)

Когда в элементе управления должны появиться дата или время, выберите элемент управления из выпадающего списка.



## Шаблон проверки правильности записи (RecordValidation)

Этот шаблон расширения добавляет функциональные возможности к шаблону процедуры путем усиления проверки, контролируемой и определенной словарем данных. Он также позволяет вам назначить элементы управления, которые нужно исключить из проверки правильности.

### Validate when the control is Accepted

(проверять правильность когда управление принято)

Устанавливает, что проверка правильности происходит, когда элемент управления генерирует событие EVENT:Accepted, что случается тогда, когда конечный пользователь завершает ввод или перемещает фокус из поля.

### Validate during NonStop Select

(проверять правильность во время безостановочного отбора)

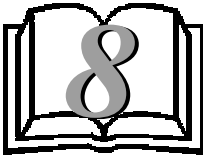
Устанавливает, что проверка правильности происходит когда какая-либо величина, связанная с элементом управления меняется в то время когда окно находится в режиме AcceptAll (Non-Stop) и имеет фокус.

### Do Not Validate

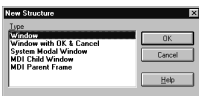
(не проверять правильность)

Открывает диалоговое окно Do Not Validate (не проверять правильность), которое позволяет выбрать поля из выпадающего списка. Поля, которые вы выбираете, будут исключены из проверок правильности.

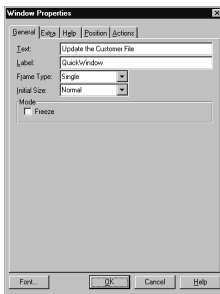
## Глава 8 Форматер окна



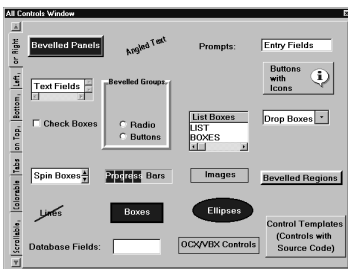
Используя Форматер окна, вы визуальнo проектируете окна и диалоговые поля вашего приложения. Форматер окна автоматически генерирует операторы Языка Clarion из окна, которое вы спроектировали на экране; кроме того, если вы будете вручную редактировать генерированную исходную программу, изменения появятся на экране, когда вы снова загрузите Форматер окна.



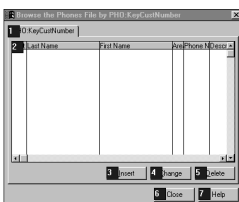
Когда вы вызываете Форматер окна из генератора приложений или из текстового редактора, вы сначала выбираете тип окна, которое предстоит создать.



Диалоговое окно “Свойства окна” позволит вам установить важные опции для вашего окна, такие, как, например, будет ли оно иметь системное меню.



Чтобы разместить элемент управления в окне, выберите тип элемента из поля инструментов, затем щелкните мышью в окне.



Предварительно просмотрите ваше окно, выровняйте элементы управления, установите порядок закладок, сетки и привязки.

**Обзор: создание окон вашего приложения**

**Выбор типа окна**

**Обзор: Окна**

**Структуры окна по умолчанию**

**Системное модальное окно System Modal Window**

**Настройка окон вашего приложения**

**Инструменты форматера окна**

**Процедуры форматера окна**

**Использование диалога Свойства окна (Window Properties)**

**Размещение элементов управления в окне**

**Панель инструментов элементов управления**

**Меню Форматера окна**

**Использование всплывающего меню (POPUP)**

**Использование меню редактирования (EDIT)**

**Использование меню элементов управления (CONTROL)**

**Использование меню выравнивания (ALIGNMENT)**

**Использование меню (MENU)**

**Использование панели инструментов меню (TOOLBAR)**

**Использование меню заселения (POPULATE)**

**Использование меню Параметры (OPTIONS)**

**Использование Предварительного просмотра! (PREVIEW!)**

Используйте Window Formatter (Форматер окна) для визуального проектирования элементов окна - окон и элементов управления. Форматер окна автоматически генерирует исходную программу на языке Clarion, которая описывает эти элементы, а генератор приложений помещает сгенерированный исходный текст в соответствующей точке в вашем приложении.

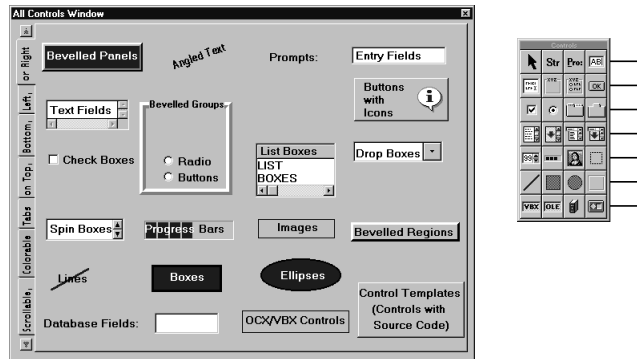


Рисунок: Форматер окна демонстрирует пример окна, показывая элементы управления, которые вы помещаете в него. Вы можете изменить размеры любого элемента управления или изменить его положение, используя для этого специальные рукоятки управления размером (для изменения размера достаточно потянуть мышью за рукоятку)

В этой главе:

- Рассказано, как вызвать Форматер окна для создания новой структуры или редактирования существующей.
- Дан список различных типов свойств окна и описано, как их установить для окна.
- Объяснен режим Preview (предварительный просмотр), который позволит вам видеть создаваемые вами окна точно в том виде, как они будут появляться перед пользователем.
- Описаны меню и команды, доступные в форматере окна.

## **Обзор: создание окон вашего приложения**

Наиболее вероятно, что ваше приложение будет использовать большое число окон: окна для вывода инструкций пользователю, ввода и просмотра данных или для работы с другой информацией пользователя. Вообще говоря вот то, что вы должны сделать, чтобы поместить такое окно на экран:

1. Выберите или создайте процедуру, которая покажет окно.

Более полную информацию вы найдете в главе Генератор приложений.

2. Из диалогового окна Application Tree (дерево приложения) вызовите диалоговое окно Procedure Properties (свойства процедуры) для выбранной вами процедуры.

Дважды щелкните клавишей мыши на имени процедуры или щелкните правой клавишей мыши на имени процедуры и выберите Properties из всплывающего меню или же выделите имя процедуры и нажмите кнопку Properties.

3. Если необходимо, установите данные, файлы или и то и другое, что будет использовать процедура..

Нажмите кнопки Data(данные) и Files(файлы) и выберите что вам нужно в появляющихся диалоговых окнах. Возвратитесь к окну Procedure Properties (свойства процедуры). Более полную информацию вы найдете в главе Генератор приложений.

4. Нажмите кнопку Window (окно) для конструирования вашего окна. Или, из диалогового окна Application Tree (дерево приложения) щелкните правой клавишей мыши и выберите из всплывающего меню Window.

Если по умолчанию никакое окно не определено, появится диалоговое окно New Structure (новая структура) и вы должны будете решить, какой тип окна использовать. Смотрите ниже Выбор типа окна. Если окно по умолчанию уже определено, появится Window Formatter (форматер окна).

**Совет: Вы можете также получить доступ к форматуру окна из текстового редактора! Для создания нового окна из текстового редактора поместите курсор на пустую строку, затем выберите Edit>Format Structure или нажмите CTRL+F. Для редактирования существующего окна поместите курсор куда-нибудь в пределах структуры исходной программы, определяющей окно, затем выберите Edit Edit>Format Structure или нажмите CTRL+F. См. ОКНО и ПРИЛОЖЕНИЕ в Справочнике языка, там вы найдете более полную информацию.**

5. Из диалогового окна New Structure выберите тип окна, который должна показывать данная процедура.

Определите свой выбор двойным щелчком мыши или выделением варианта выбора в списке типов окон Type (тип), затем нажмите кнопку ОК. Появится форматер окна.

6. Настройте в соответствии с вашими требованиями размер окна и определите его свойства. Данная глава объясняет, как сделать это.

См. ниже Определение окон вашего приложения.

7. Если необходимо, поместите меню в ваше окно, используя для этого редактор меню Menu Editor. В следующей главе будет рассказано, как сделать это.

8. Поместите в окно элементы управления - это могут быть поля ввода для редактирования полей из базы данных, командные кнопки для инициализации или прекращения действия, текст, строки или приглашения, содержащие инструкции для пользователя, и другие элементы управления, предназначенные для улучшения вида окна и удобства пользования им. Как это сделать - объясняется в главе Определение свойств элементов управления и ниже в Размещение элементов управления в окне.

9. Возвратитесь к диалогу Procedure Properties.

## **Выбор типа окна**

В большинстве программ Windows имеются три типа используемых экранных окон: окна приложения, окна документов и диалоговые поля. Окно приложения является первым окном, открытым в программе Windows. Обычно оно содержит главное меню как вводную точку ко всей остальной программе. Все другие окна в программе являются документными окнами или диалоговыми полями.

### **Окна приложения - SDI или MDI**

---

Наряду с этими тремя типами экранных окон имеются два стандартных типа интерфейса пользователя, используемых в программах Windows: Интерфейс одиночного документа (SDI) и Интерфейс многих документов (MDI).

Программа SDI содержит только линейную логику, которая позволяет пользователю предпринять только один процесс исполнения (thread) за раз; она не открывает отдельные процессы, между которыми мог бы перемещаться пользователь. Это тот же тип программной логики, который используется в большинстве программ DOS. Структура Clarion WINDOW (без атрибута MDI) использована для определения окна приложения программы SDI и последующих документных окон или диалоговых полей, открытых поверх него.

Программа MDI позволяет пользователю выбирать несколько процессов (threads) исполнения и переходить от одного к другому в любое время. Этот интерфейс пользователя на основе программы Windows очень привычный и распространенный, так как он предоставляет пользователю максимальную свободу работать по своему выбору. Структура APPLICATION (приложение) Clarion назначает MDI-окно приложения. MDI-окно приложения действует как родительское окно для всех MDI дочерних окон (окон документов и диалоговых полей) в том смысле, что дочерние окна прикреплены к его рамке и автоматически двигаются, когда перемещается рамка приложения. Они могут быть также спрятаны все сразу при минимизации родительского окна. В программе Clarion Windows в любой момент может быть открытым только одно ПРИЛОЖЕНИЕ.

Более полную информацию о преимуществах MDI-окон по сравнению с не-MDI окнами вы можете найти в Приложении Вопросы конструирования окон.

### **Окна документов и диалоговые поля**

---

Документные окна и диалоговые поля очень похожи друг на друга в том отношении, что они все определены как структуры Clarion WINDOW. Они отличаются по условному контексту, в котором они обычно используются, и по условиям, касающимся вида и атрибутов. Во многих случаях разница не столь различима и не имеет значения. Родовым термином как для документных окон, так и для диалоговых полей, является “окно” и это тот термин, который используется всюду в тексте.



Документные окна обычно показывают данные. Принято, что они подвижны и изменяют размеры. Обычно у них есть заголовок, системное меню и кнопка максимизации. Например, в среде Windows групповое окно “главной” программы, которое появляется после того, как вы дважды щелкните мышью на пиктограмме в Диспетчере программ, является документным окном.

Диалоговые поля обычно запрашивают информацию у пользователя или побуждают пользователя выбрать некое условие, обычно перед тем как выполнить некоторое действие, запрашиваемое пользователем. Они могут быть, а могут и не быть перемещаемыми, поэтому могут иметь, а могут и не иметь системное меню и заголовок. По соглашению они не меняют свой размер, хотя они могут иметь кнопку максимизации, которая придает окну два альтернативных размера. Диалоговое поле может быть системно модальным (пользователь должен отвечать прежде чем сделать что-либо еще в Windows), модальным относительно приложения (пользователь должен отвечать прежде чем делать что-либо в приложении) или немодальным. Например, в среде Clarion окно, которое появляется из открытого выбора меню файла, является модальным относительно диалоговым полем, которое запрашивает имя открываемого файла.

## Структуры окна по умолчанию

Некоторые типы окон, которые вы можете создать, появляются в диалоговом окне New Structure (новая структура). Позиции в диалоге New Structure представляют собой структуры Clarion. Вы можете видеть структуры окна или структуры отчета или и то и другое в зависимости от того, как вы получаете доступ к диалогу. Структура окна - это группа операторов на языке Clarion, которая определяет все атрибуты окна. Вы можете хотеть думать о структуре окна как об определении окна.

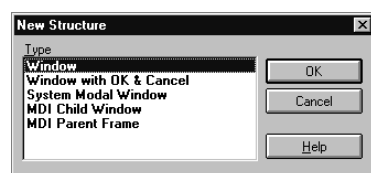
В данном разделе обсуждаются только структуры окна по умолчанию, поставляемые с данной версией; однако, выбрав структуру окна по умолчанию, вы можете ее модифицировать. Вы можете даже добавить свои собственные структуры окна по умолчанию путем редактирования файла C:\CW 20\LIBSRC\DEFAULTS.CLW. Если вы редактируете файл DEFAULTS.CLW, не забудьте предварить каждую новую структуру следующей строчкой:

```
!!>title
```

где “title” - это имя структуры, которая должна появиться в диалоговом окне Structure (структура).

Диалоговое окно New Structure перечисляет типы структур окон и отчетов, которые вы можете создать.

Далее приводится описание типов структур окна, поставляемых с данной версией.



## Окно

Для создания окна общего назначения или диалогового поля, выберите Window из диалогового окна New Structure (новая структура). Форматер окна генерирует не-MDI WINDOW-структуру без элементов управления. То есть пустое окно.

Это окно примет любые элементы управления (поля списка, поля ввода, кнопки и т.д.), которые вы захотите добавить. См. ниже Размещение элементов управления в окне. См. также главу Установка свойств элементов управления.

## Окно с кнопками ОК и Cancel (отказ)

Для создания диалогового окна общего назначения с кнопками ОК и Cancel выберите окно типа Window with OK & Cancel (окно с кнопками ОК и Cancel) из диалогового окна New Structure. Как и раньше, форматер окна генерирует не-MDI структуру WINDOW, однако это окно уже содержит кнопки ОК, Cancel. Окно примет любые другие элементы управления (поля списка, поля ввода, кнопки и т.д.), которые вы захотите поместить в данное окно. Вы можете также уничтожить кнопки ОК и Cancel, если захотите.

На уровне исходной программы единственное различие между этим типом окна и предыдущем заключается в присутствии двух дополнительных операторов, создающих кнопки ОК и Cancel, - различий же в самом типе окна нет и нет вызовов процедуры, связанных с двумя кнопками. Clarion обеспечивает этот дополнительный тип окна просто для удобства - потому, что эти две кнопки являются весьма популярными элементами окна и вашему приложению скорее всего они тоже потребуются.

**Совет: Для большинства окон ввода данных вы можете использовать один из этих двух выборов (диалоговые поля). Окно может появиться с рамкой в виде одиночной линии, полями ввода, в которых пользователь должен ввести данные для новой записи, плюс командные кнопки. Когда пользователь нажимает кнопку ОК, приложение может добавить запись; если пользователь нажмет кнопку Cancel, операция будет отменена.**

## Системное модальное окно System Modal Window

Чтобы создать Системное модальное окно, из диалогового окна New Structure выберите тип окна System Modal Window (системное модальное окно). Системно-модальный означает, что пользователь не в состоянии выполнять какую либо еще работу или даже переключиться к другому приложению до тех пор, пока это модальное окно не будет закрыто. Форматер окна будет генерировать структуру WINDOW с атрибутом MODAL

**Совет: Чтобы просигнализировать о критической ошибке, используйте системное модальное окно. Windows - это кооперативная многоцелевая среда, в которой пользователь должен управлять множеством процессов. Системное модальное окно временно ограничивает свободу пользователя в выполнении других приложений. Если у вашего приложения нет достаточно веской причины остановить работу всей системы - например, серьезная ошибка файла, которая может привести в результате к потере данных, если не будет исправлена тотчас - сведите до минимума использование вашим приложением этого типа окна.**



### Дочернее окно MDI

Чтобы создать документное окно, которое будет появляться внутри рамки приложения, выберите тип окна MDI child window (дочернее окно MDI). Форматер окна будет генерировать структуру WINDOW с атрибутом MDI.

В общем случае MDI - Интерфейс многих документов дает метод для представления и редактирования более чем одного документа или более чем одного вида документа в одном приложении. То есть, одно и то же MDI окно может быть открыто много раз из того же самого приложения. Каждый документ появляется в дочернем окне внутри главного окна приложения.

Дочернее окно обычно выглядит как обычное окно, с рамкой, системным меню, кнопками максимизации и минимизации и пиктограммой. Пользователь должен иметь возможность манипулировать им, как любым другим окном, - за исключением того, что дочернее окно не может передвигаться за пределы главного окна приложения. Примером типичного использования MDI-окна может быть “одновременное” представление другого, второго, или третьего видов просмотра базы данных вашего приложения.

Все MDI-окна должны размещаться в отдельных процедурах и процессах относительно головного окна APPLICATION (приложение) (смотрите ниже Родительская рамка MDI). Это означает, что генератор приложений должен использовать функцию START для того, чтобы начать новый процесс для MDI окна, вызываемого из рамки APPLICATION. Вы можете исполнять в процессе более одного окна MDI.

**Совет: Любые меню и панели инструментов, которые вы создаете для MDI-окна, будут автоматически сливаться с меню и панелью инструментов приложения, когда дочернее MDI окно является активным окном!**

## **Родительская рамка MDI**

Чтобы создать рамку APPLICATION или главное окно MDI-приложения, выберите MDI Parent Frame (Родительская рамка MDI). Это обеспечивает “внешнюю” рамку, в которой будут появляться дочерние MDI-окно.

**Совет: Типичное окно APPLICATION должно иметь рамку с изменяемыми размерами, системное меню, кнопки максимизации, минимизации и меню. Обычно меню File должно обеспечивать команду для открытия или создания дочернего MDI окна, а меню Window должно обеспечивать команды для управления отдельными дочерними окнами.**

Окно APPLICATION не может иметь элементы управления внутри окна - фактически Clarion просто не позволит вам поместить их таким образом. Дочерние MDI-окна содержат все элементы управления MDI-приложения. Окно APPLICATION должно содержать только дочерние окна и, если необходимо, панель инструментов (которая может содержать элементы управления).

Поместите любые команды глобального меню или элементы панели инструментов в окно APPLICATION. Каждое дочернее окно наследует эти команды и панели инструментов и, если это необходимо, может добавить дополнительные позиции или сделать доступными или недоступными глобальные команды и элементы панели инструментов.

Окно APPLICATION и его дочерние MDI окна не должны находиться в одной и той же процедуре. Это требует использования функции START, так что MDI дочернее окно выполняется как отдельный процесс. Несколько MDI-окон могут составлять один процесс, но обязательно отличный от процесса головного окна APPLICATION.

**Совет: Когда исполняется оператор окна APPLICATION, Clarion делает это окно невидимым до тех пор, пока он не встретится первый оператор DISPLAY или цикл ACCEPT. Это позволяет вашему приложению делать любые косметические изменения головного окна APPLICATION прежде, чем его увидит пользователь.**

## Определение окон вашего приложения

Как только вы определили тип создаваемого окна, появляется Window Formatter (форматер окна). Он имеет богатый набор инструментов и меню, которые предназначены для того, чтобы помочь вам создать и отредактировать ваше окно.

### Процедуры Форматера окна

---

Форматер окна дает вам возможность непосредственно манипулировать вашим окном и находящимися внутри него элементами управления. Исходный образец окна, например, снабжен “рукоятками” - маленькими прямоугольными полями, расположенными по углам и по сторонам окна. Выбрав рукоятку и протаскив ее мышью, вы можете изменить размер окна. Окно, которое пользователь видит во время работы вашего приложения, имеет тот же самый размер, который вы создали указанной процедурой протаскивания за рукоятки.

Когда Форматер окна генерирует исходную программу для данного окна, он помещает данные, определяющие размер и положение окна (установленные вами при протаскивании мышью) в атрибут AT оператора, определяющего окно.

Аналогичным образом Форматер окна ~~подставляет~~ добавляет другие атрибуты, предоставляя вам для этого опции, поля флажков и поля, в которых вы назначаете предпочтительные для вас варианты выбора.

Вот как выглядит типичный процесс настройки нового окна с помощью Форматера окна:

1. Установите размер окна, передвинув рукоятки окна-образца до того размера, который вам нужен.

2. Установите другие атрибуты окна с помощью диалогового окна Window Properties (свойства окна).

Щелкните правой клавишей мыши в окне и выберите из всплывающего меню Properties (свойства) или выберите окно и затем выберите Edit Properties.

К числу других атрибутов относятся заголовок окна, возможность изменения размера окна, атрибут прокрутки окна, относящиеся к окну пиктограммы, сообщения, файлы помощи, а также типы курсоров, связанные с данным окном, и многие другие. Смотрите ниже раздел Диалог Свойства окна.

3. Закройте диалоговое окно Window Properties (свойства окна).

4. Поместите в окно элементы управления.

Смотрите дальше раздел Размещение элементов управления в окне. Также смотрите

главу Установка свойств элементов управления.

5. Посмотрите, каким получилось ваше окно, выбрав команду предварительного просмотра Preview! на линейке действий. Нажмите клавишу ESC для того, чтобы вернуться в Форматер окна и выполнить дополнительные регулировки.

6. Выберите команду Exit! на линейке действий для того, чтобы вернуться к Генератору приложений или Текстовому редактору.

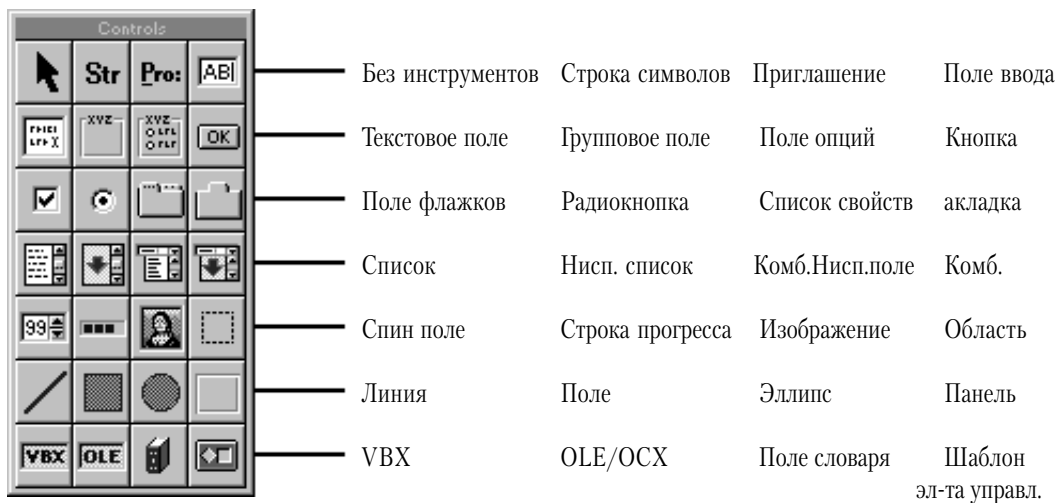
## Инструменты форматера окна

### Образец окна

Window Formatter является визуальным инструментом конструирования. Вы всегда видите образец окна когда работаете над ним. Кроме того, вы можете видеть окно точно таким как оно появится перед конечным пользователем, если выберите Preview! (предварительный просмотр) на панели действий

### Панель элементов управления

Window Formatter содержит плавающую панель инструментов Controls (элементы управления), подобную той, которую вы находите во многих программах черчения и рисования. Просто выберите элемент управления с панели (щелкнув над ним), затем щелкните над образцом окна чтобы поместить данный элемент управления в это окно.



Все элементы управления в панели инструментов доступны также из меню Controls (элементы управления); исключение составляют Шаблон элемента управления и Поле словаря, доступ к которым открывает меню Populate (заселить).

**Совет: Поместите курсор над каким-либо инструментом и подождите пол-секунды. Появится подсказка об инструменте, объясняющая вам тип элемента управления, который создается этим инструментом.**



Отпускает текущий выбранный элемент управления



String (строка символов)

Помещает в создаваемом окне элемент управления STRING (строка символов). См. Установка свойств элемента управления Строка.



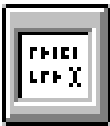
Prompt (приглашение)

Помещает в создаваемом окне элемент управления PROMPT (приглашение). См. Установка свойств элемента управления Приглашение.



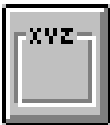
Entry Box (поле ввода)

Помещает в создаваемом окне элемент управления ENTRY (поле ввода). См. Установка свойств элемента управления Поле ввода.



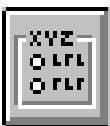
Text Box (текстовое поле)

Помещает в создаваемом окне элемент управления TEXT (текст). См. Свойства элемента управления Текстовое поле.



Group Box (групповое поле)

Помещает в создаваемом окне элемент управления GROUP (групповое поле). См. Свойства элемента управления Групповое поле..



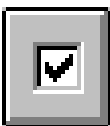
Option Box (поле опций)

Помещает в создаваемом окне элемент управления OPTION (структура OPTION, которая появляется как групповое поле с радиокнопками). См. Свойства элемента управления Поле опций.



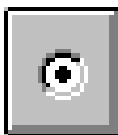
Button (кнопка)

Помещает в создаваемом окне элемент управления BUTTON (кнопка). См. Свойства элемента управления Кнопка нажатия.



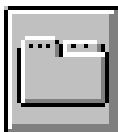
Check Box (поле флажков)

Помещает в создаваемом окне элемент управления CHECK (флажок). См. Свойства элемента управления Поле флажков.

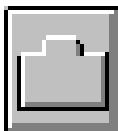


### Radio Button (радиокнопка)

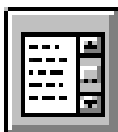
Помещает в создаваемом окне элемент управления RADIO. См. Свойства элемента управления Радиокнопка.



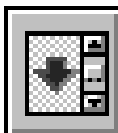
Помещает в создаваемом окне элемент управления SHEET (лист). Элемент управления Sheet содержит элементы управления Tab. См. Свойства элемента управления лист.



Помещает элемент управления TAB (закладка) в создаваемом окне. Элементы управления TAB могут содержать любые другие типы элементов управления. См. Свойства элемента управления TAB.



Помещает в создаваемом окне элемент управления LIST (списочное поле или ниспадающее поле списка). См. Создание полей списка в главе Элементы управления и их свойства.



Помещает в создаваемом окне элемент управления Drop LIST (ниспадающий список). См. Создание поля списка в главе Элементы управления и их свойства.



### Combo Box (комбинированное поле)

Помещает в создаваемом окне элемент управления COMBO (комбинированное поле или выпадающее комбинированное поле). См. Свойства элемента управления Комбинированное поле.

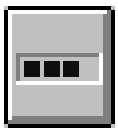


Помещает в создаваемое окно элемент управления DROP COMBO (ниспадающее комбинированное поле). См. Свойства элемента управления Ниспадающее комбинированное поле.



### Spin Box (спин поле)

Помещает в создаваемом окне элемент управления SPIN (прокрутка). См. Свойства элемента управления Поле прокрутки..



### Progress Bar (строка прогресса)

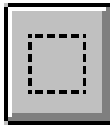
Помещает в создаваемом окне элемент управления PROGRESS (прогресс). См. Свойства элемента управления Строка прогресса.



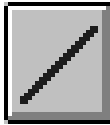
### Image (изображения)

Помещает в создаваемом окне элемент управления IMAGE (изображение). См. Свойства элемента управления Изображение.

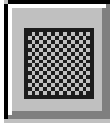


**Region (область)**

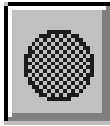
Помещает в создаваемое окно элемент управления REGION (область). См. Свойства элемента управления Область.

**Line (линия)**

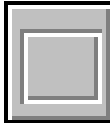
Помещает на создаваемое окно элемент управления LINE (линия). См. Свойства элемента управления Линия.

**Box (поле)**

Помещает в создаваемом окне элемент управления BOX (поле). См. Свойства элемента управления Поле.

**Ellipse (эллипс)**

Помещает на создаваемое окно элемент управления ELLIPSE (эллипс). См. Свойства элемента управления Эллипс.

**Panel (панель)**

Помещает на создаваемое окно элемент управления PANEL (панель). См. Свойства элемента управления панель.

**VBX (Visual Basic)**

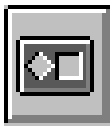
Помещает на создаваемое окно элемент управления VBX (Visual Basic). См. Свойства элемента управления VBX.

**OLE/OCX Container**

Помещает на создаваемое окно элемент управления OLE/OCX Container. См. Свойства элемента управления OLE.

**Dictionary Field (поле словаря)**

Дает вам возможность выбрать поле, определенное в словаре данных, и помещает это поле плюс ассоциированное с ним приглашение PROMPT в создаваемое окно.

**Control Template (шаблон элемента управления)**

Помещает в создаваемом окне элемент управления CONTROL TEMPLATE. См. главу Шаблоны элементов управления, программы и расширений.



Шаблоны элементов управления создают элементы управления и исходные программы для поддержания этих элементов. Например, шаблон элемента управления Поле просмотра не только генерирует исходную программу для показа поля списка, он также генерирует программу загрузки данных из файла в очередь QUEUE, затем показывает данные в списочном поле с полной прокруткой и возможностью выбора с помощью щелчка мыши.

**Совет: В общем случае для вас будет полезнее использовать шаблон элемента управления, а не один пустой элемент управления.**

Показать или спрятать панель инструментов Элементы управления (Controls) можно путем выбора Options > Toolbox. Все элементы управления на панели доступны также из меню Controls. См. ниже Помещение элементов управления в окно. См. также главу Установка свойств элементов управления.

### Панель инструментов Заселение Полей

Window Formatter (форматер окна) содержит плавающую панель инструментов Populate Field (заселение поля). Эта панель инструментов дает вам возможность быстро “заселить” окно элементами управления для полей в файлах данных. Во-первых, выберите файл из выпадающего списка, затем дважды щелкните клавишей мыши над полем, которое вам нужно видеть на своем окне, чтобы поместить приглашение и элемент управления вводом для выбранного поля. Тип элемента управления (поле ввода, поле флажков, радиокнопка и т.д.) определяется установками для данного конкретного поля в словаре данных. Поле автоматически выравнивается.



Вы можете показать или спрятать панель инструментов Populate Field, обращаясь к OptionsIIIFieldbox. Размер панели инструментов Populate Field измените, помещая курсор на край поля. Когда вид курсора изменится и станет стрелкой с двумя наконечниками, щелкните и тащите.

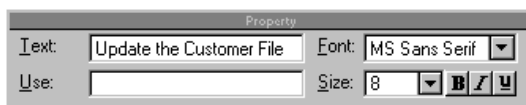
Вы можете также заселить окно с помощью элементов управления вводом для полей в ваших файлах данных с помощью меню Populate (заселить) или с помощью инструмента Полей словаря на панели инструментов Controls (элементы управления).

### Панель инструментов Свойства

Панель Window Formatter's Property позволяет быстро установить вид и содержание текста в каждом элементе управления в окне и в строке заголовка окна. Управляйте

шрифтом, размером, стилем и содержанием всех ваших текстов с помощью стандартных кнопок системы подготовки текстов и выпадающих списков.

Покажите или спрячьте панель инструментов Property выбирая Options\Propertybox. Размер панели инструментов Property измените помещая курсор на край поля. Когда вид курсора изменится и станет стрелкой с двумя наконечниками, щелкните и тащите.



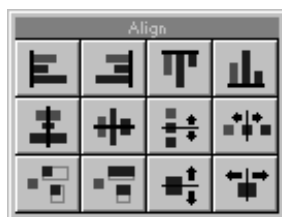
Кнопки форматирования текста - Жирный, Курсив и Подчеркивание.

### Панель инструментов Выравнивание

Панель Window Formatter's Align дает вам возможность быстро, профессионально и точно выровнять элементы управления в вашем окне. Выберите элементы управления для выравнивания (CTRL+ЩЕЛЧОК позволяет выбрать многие элементы управления, или же вы можете выбрать сразу несколько элементов управления "оконтуривая" с помощью CTRL+ТАЩИТЬ), затем щелкните клавишей на соответствующем инструменте выравнивания. Все действия выравнивания доступны также из меню Align (выровнять).

Покажите или спрячьте панель инструментов Align выбирая Options\Alignbox. Размер панели инструментов Align измените помещая курсор на край поля. Когда вид курсора изменится и станет стрелкой с двумя наконечниками, щелкните и тащите.

**Совет: Для большинства функций настройки первый выбранный элемент(ы) управления (синие квадратики - рукоятки) выравнивается с последним выбранным элементом управления (красные рукоятки). Это значит, что последний выбранный элемент управления является элементом по которому выравниваются все остальные. Он не движется, другие - движутся.**



- |   |                 |                |                      |                     |
|---|-----------------|----------------|----------------------|---------------------|
| — | Левое выравн.   | Правое выравн. | Выравн. верха        | Выравн. дна         |
| — | Вертик. выравн. | Гориз. выравн. | Размах по вертик.    | Размах по гориз.    |
| — | Тот же размер   | Та же высота   | Центровка по вертик. | Центровка по гориз. |



Выравнивает левые границы выбранных элементов управления с левой границей последнего выбранного элемента управления (красные квадратики - рукоятки ).



Align Right (выровнять справа)

Выравнивает правые границы выбранных элементов управления с правой границей последнего выбранного элемента управления (красные квадратики - рукоятки ).



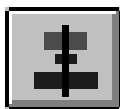
Align Top (выровнять верх)

Выравнивает верхние границы выбранных элементов управления с верхней границей последнего выбранного элемента управления (красные квадратики - рукоятки ).



Align Bottom (выровнять низ)

Выравнивает нижние границы выбранных элементов управления с нижней границей последнего выбранного элемента управления (красные квадратики - рукоятки ).



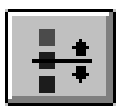
Align Vertical (выровнять по вертикали)

Вдоль вертикальной оси выравнивает центры выбранных элементов управления с центром последнего выбранного элемента управления (красные квадратики - рукоятки ).



Align Horizontal (выровнять по горизонтали)

Вдоль горизонтальной оси выравнивает центры выбранных элементов управления с центром последнего выбранного элемента управления (красные квадратики - рукоятки ).



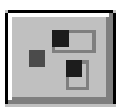
Spread Vertical (Упорядочить по вертикали)

Уравнивает вертикальные промежутки между выбранными элементами управления.



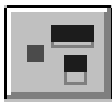
Spread Horizontal (Упорядочить по горизонтали)

Уравнивает горизонтальные промежутки между выбранными элементами управления.

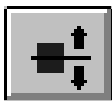


Same Size (тот же размер)

Делает все выбранные элементы управления одинаковыми по высоте и ширине с последним выбранным элементом управления (красные квадратики - рукоятки ).

**Same Height (та же высота)**

Делает все выбранные элементы управления одинаковыми по высоте с последним выбранным элементом управления (красные квадратики - рукоятки).

**Center Vertical (центровка по вертикали)**

Как группу (относительные позиции выбранных элементов управления не изменяются) центрирует выбранные элементы управления вертикально внутри окна.

**Center Horizontal (центровка по горизонтали)**

Как группу (относительные позиции выбранных элементов управления не изменяются) центрирует выбранные элементы управления горизонтально внутри окна.

**Совет: Поместите курсор над каким-нибудь инструментом и подождите полсекунды. Появится подсказка об инструменте, объясняющая вам выравнивания, который будет выполнен этим инструментом.**

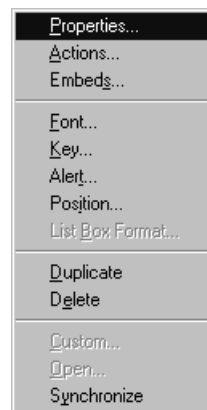
**Ограниченное протаскивание**

Нажмите и удерживайте клавишу SHIFT во время протаскивания элемента управления для того, чтобы ограничить перемещения элемента управления движением по одной оси. То есть, комбинация SHIFT+ПРОТАСКИВАНИЕ перемещает элемент управления либо по вертикали, либо по горизонтали, но не по обоим осям одновременно.

**Меню Формatera окна****Всплывающее меню (POPUP MENU)**

Получите доступ к всплывающему меню, для чего выберите мышью окно или какой-либо элемент управления. Всплывающее меню Формatera окна дает вам возможность манипулировать и настраивать окно, а также элементы управления в окне, в зависимости от того, что выбрано - окно или элемент управления.

- ☐ Чтобы выбрать окно, поместите курсор в строку заголовка образца окна и ЩЕЛКНИТЕ правой клавишей мыши.
- ☐ Чтобы выбрать элемент управления, поместите курсор на этот элемент управления и ЩЕЛКНИТЕ правой клавишей.
- ☐ Чтобы выбрать элемент управления "Лист" (SHEET), поместите курсор куда-нибудь на списке, но не на другие элементы



управления, и не на закладку, затем ЩЕЛКНИТЕ правой клавишей.

□ Чтобы выбрать элемент управления “Закладка” (ТАВ) , поместите курсор на соответствующую закладку и ЩЕЛКНИТЕ правой клавишей.

**Совет: Все команды всплывающего меню доступны также на меню редактирования (Edit menu) формatera окна.**

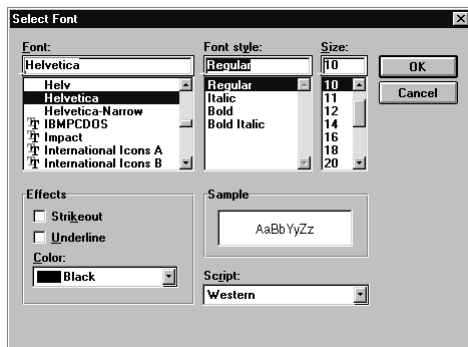
Ниже приводятся описания выборов всплывающего меню.

**Properties (свойства)** Чтобы отредактировать свойства элемента управления или окна, выберите элемент управления или окно и выберите команду меню Properties (свойства) Смотрите выше Использование диалога свойств окна или главу Установка свойств элементов управления, где дается дополнительная информация. Вы можете также щелкнуть правой кнопкой мыши в области элемента управления или окна и выбрать команду Properties в появившемся меню.

**Actions (действия)** Чтобы назначить действия, связанные с элементом управления или окном, отберите его и выберите команду Actions (действия). Дополнительная информация - в главе Установка свойств элементов управления.

**Embeds (Вставка)** Чтобы добавить или отредактировать вставляемый исходный текст, связанный с элементом управления или окном, выберите его команду Embeds (вставки. Смотрите раздел Определение вставной исходной программы и главу Использование генератора приложений.

**Font (шрифт)** Для управления видом текста, показываемого в элементе управления или окне, отберите элемент управления или окно и выберите команду Font (шрифт). Установите шрифт, размер, стиль, сценарий и цвет из выпадающих списочных полей. Переключайте Strikeout (перечеркивание) и Underline (подчеркивание) с помощью полей флажков. Диалоговое окно Select Font (выбор шрифта) покажет образец конструкции выбранного вами текста.

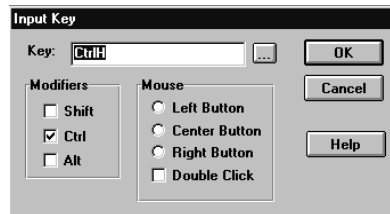


Установка атрибутов текста с помощью диалога Выбор шрифта

Key (клавиша)

Чтобы установить “горячую” клавишу для элемента управления, отберите элемент управления и выберите команду Key (атрибут KEY применим как к окнам, так и к некоторым другим элементам управления). Используйте диалоговое окно Input Key (клавиша ввода) для добавления атрибута KEY к вашему элементу управления. Атрибут KEY определяет “горячую” клавишу, или комбинацию клавиш, которая, будучи нажата пользователем, перенесет немедленно фокус на данный элемент управления или, для элемента управления действия, такого как кнопка, будет инициировать действие.

Из диалогового окна Input Key отберите горячую клавишу или комбинацию клавиш, нажимая для этого или требуемую клавишу или комбинацию клавиш. Нажатые вами клавиши появятся в поле Key и будут подставлены как параметры к атрибуту KEY для этого элемента управления.

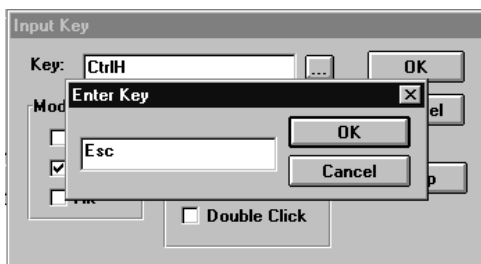


### Добавление “горячей” клавиши для CTRL+N

Щелчки клавишами мыши могут быть использованы в качестве горячих клавиш; однако, щелчки не могут быть установлены с помощью щелканья клавишей мыши. Для щелчков мыши отметьте соответствующее поле(я) флажков. Например, чтобы дать фокус для элемента управления когда пользователь щелкает два раза левой клавишей, отметьте поле Left Button (левая клавиша) и поле Double

Click (двойной щелчок).

Если необходимо, добавьте модификатор или модификаторы для создания многоклавишной последовательности (например, CTRL+N, или ALT+ ЩЕЛЧОК ПРАВОЙ КЛАВИШЕЙ) отметьте Ctrl, Alt или Shift или какие-либо комбинации этих трех.



Добавление ESC к горячей последовательности ключей.

Клавиши ESC, ENTER и TAB не могут быть назначены их нажатием. Для этих клавиш нажмите эллиптическую кнопку (...) и наберите “esc”, “enter” или “tab”.

Следующие элементы управления получают фокус от атрибута KEY:

- Combo Box - комбинированное поле
- Entry Field - поле ввода
- Group Box - групповое поле
- List Box - списочное поле
- Option Box - опционное поле
- Prompt - приглашение, подсказка
- Property Sheet - список свойств
- Spin Box - спин поле
- Tab - закладка
- Text Field - текстовое поле

Следующие элементы управления получают от атрибута KEY фокус и немедленно исполняются:

- Button - кнопка
- Check Box - поле флажков
- Radio Button - радиокнопка
- OLE - связывание
- VBX - Visual Basic Control

Атрибуты KEY неприменимы к следующим элементам управления:

- String - строка символов
- Progress Bar - строка прогресса
- Image - изображение
- Region - область
- Line - линия
- Box - поле
- Ellipse - эллипс
- Panel - панель



**Alert (Установка клавиши, порождающей событие)** Чтобы назначить порождающую событие клавишу для окна или элемента управления, отберите окно или элемент управления и выберите команду Alert. Используйте диалоговое окно Alert Keys (Установка клавиши, порождающей событие) и диалоговое окно Input Key (клавиша ввода) чтобы добавить атрибут ALRT для вашего окна или элемента управления. Когда установлен атрибут ALRT, окно генерирует EVENT:AlertKey если пользователь нажимает выбранные вами в этом диалоге клавиши. Вы можете назначить более одной клавиши Alert для окна или элемента управления.

Смотрите выше Клавиша, где обсуждается, как назначить клавиши с помощью диалога Input Key.

**List Box Format (формат списочного поля)** Чтобы установить вид и функциональные возможности элемента управления полем списка, отберите списочное поле и выберите команду List Box Format (формат списочного поля. Смотрите дополнительную информацию в главе Использование форматера списочного поля.

**Duplicate (поместить копию)** Чтобы поместить в то же самое окно копию элемента управления, отберите оригинал и выберите команду Duplicate. Копия появится рядом с оригиналом.

**Совет: Вы можете дублировать (изготавливать копии) сразу несколько элементов управления путем выбора многих элементов управления перед выполнением команды Duplicate. Используйте комбинацию CTRL+ЩЕЛЧОК для выбора многих элементов управления, или же захватите много элементов управления с помощью CTRL+ЩЕЛЧОК+ТАЩИТЬ.**

**Delete (удалить)** Чтобы удалить элемент управления или окно, отберите объект для удаления и выберите команду Delete, или же отберите объект и нажмите клавишу DELETE.

**Custom** Чтобы открыть Лист свойств OCX, ассоциированного с элементом управления OLE, отберите элемент управления и выберите команду Custom.

**Open (открыть)** Чтобы открыть Сервер OLE, ассоциированный с элементом управления OLE, отберите данный элемент управления и выберите команду Open.

**Synchronize**

**(синхронизировать)** Придает атрибуты элемента управления, назначенные в словаре данных, выбранному элементу управления, а если выбрано окно - всем элементам управления в окне. Атрибуты придаются в соответствии с назначением в закладке Synchronization диалогового окна Application Options (опции приложения). См. Генератор приложений - Установка опций приложения.

## Меню редактирования (EDIT)

Меню Edit (редактирование) в форматере окна Window Formatter дает вам возможность манипулировать и настраивать окно, а также элементы управления в окне, в зависимости от того, что отображено - окно или элемент управления.

□ Чтобы отобразить окно, поместите курсор в строку заголовка окна-образца и ЩЕЛКНИТЕ мышью.

□ Чтобы отобразить элемент управления, поместите курсор на этот элемент управления и ЩЕЛКНИТЕ мышью.

□ Чтобы отобразить элемент управления Property Sheet (лист), поместите курсор где-нибудь на листе, но не на другие элементы управления и не на закладку (tab), затем ЩЕЛКНИТЕ мышью.

□ Чтобы отобразить элемент управления Tab (закладка), поместите курсор на соответствующую закладку и ЩЕЛКНИТЕ мышью.

**Совет: Многие команды меню редактирования Edit доступны также из всплывающего меню, к которому вы попадаете с помощью правого щелчка над данным элементом управления или окном.**

Ниже приводится описание позиций меню редактирования Edit, не описанных в разделе Использование всплывающего меню:

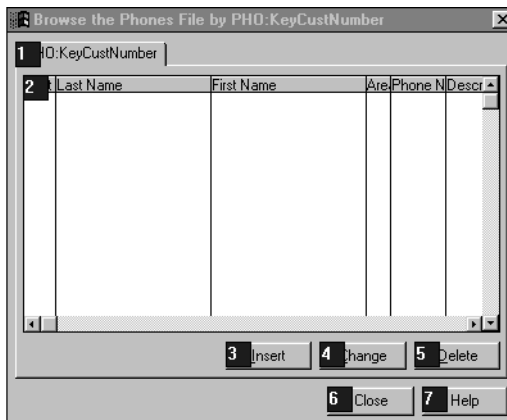
Undo (откат)      Чтобы отменить последнее действие редактирования выберите команду Undo (уничтожить сделанное, вернуть). Все действия форматера окна Window Formatter могут быть отменены, за исключением удаления элемента управления.

Redo (делать вновь)      Чтобы сделать вновь отмененное действие, выберите команду Redo (делать вновь) . Не все действия могут быть повторены вновь.

Set Tab Order (установить порядок по клавише tab)      Чтобы визуальным образом установить порядок для выбранных элементов управления при нажатии клавиши tab, отберите окно, групповое поле или поле параметров, затем выберите команду Set Tab Order . Откроется диалоговое окно, которое даст вам возможность определить, как установлен порядок по клавише tab : Автоматически или Вручную, Горизонтально или Вертикально. Альтернативный метод установки порядка дается в Установить порядок элементов управления.

Из диалога Ordering Type (тип упорядочивания) выберите радиокнопку Manual (ручной), затем нажмите кнопку ОК для назначения порядка ручным щелканьем на элементах управления. На каждом элементе управления появляется цифра, указывающая порядковый номер. Щелкните

на элементах управления для изменения порядка на желаемый.

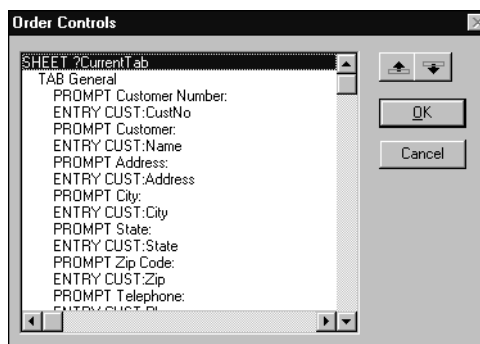


Другой способ. Из диалогового окна Ordering Type выберите радиокнопку Automatic (автоматический), затем выберите радиокнопку либо Horizontally (горизонтально) либо Vertically (вертикально). Нажмите кнопку ОК для автоматической установки порядка на основе позиций элементов управления. Horizontally сначала ставит самые верхние элементы управления. Vertically сначала ставит самые левые элементы.

Перенаберите команду Set Tab Order или щелкните над строкой заголовка окна-образца для того, чтобы вернуться к нормальному режиму редактирования.

Set Control Order (установить порядок элементов управления) Чтобы установить порядок элементов управления среди перекрывающихся закладок, выберите команду Set Control Order. Открывается диалоговое окно Order Controls (упорядочивание элементов управления), который показывает все элементы управления на окне в иерархическом порядке. Перераспределите элементы управления и порядок их, отбирая для этого требуемый элемент управления и нажимая кнопки со стрелками для перемещения данного элемента управления вверх или вниз в пределах списка.

Установка порядка  
табуляции по числам.



## Synchronize Window

(синхронизировать окно) Чтобы придать атрибуты элементов управления, назначенные в словаре данных, всем элементам управления в окне, выберите команду Synchronize Window. Атрибуты придаются в соответствии с назначением в закладке Synchronization диалогового окна Application Options (опции приложения). См. Генератор приложений - Установка опций приложения.

## Control Templates

(шаблоны управления) Чтобы добавить шаблоны расширений к процедуре или чтобы отредактировать приглашения шаблонов элементов управления и шаблона расширения для данной процедуры, выберите команду Control Templates (шаблоны элементов управления). Открывается диалоговое окно Extensions and Control Templates (шаблоны расширений и элементов управления), в котором вы можете добавить шаблоны Расширений или можете отредактировать Шаблоны элементов управления или Шаблоны расширений. Дополнительную информацию вы можете найти в главе Шаблоны элементов управления, программы и расширений.

## **Меню элементов управления (CONTROL)**

Меню Control (элементы управления) дает список всех элементов управления, которые появляются в инструментальной панели элементов управления за исключением Полей словаря и Шаблона элемента управления (См. Меню заселения).

Исполнение команды из меню Control идентично щелканью мышью на соответствующей пиктограмме панели инструментов. Меню служит для удобства. Список элементов управления панели инструментов приведен выше в разделе Инструменты формatera окна. Смотрите также раздел Элементы управления и их свойства.

## **Меню выравнивания (ALIGNMENT)**

Меню Alignment (выравнивание) перечисляет те же инструменты выравнивания, которые появляются на панели инструментов выравнивания. Исполнение команд из меню Alignment идентично щелканью мышью на соответствующей пиктограмме панели инструментов. Меню служит для удобства.

Список инструментов выравнивания приведен выше в разделе Инструменты формatera окна.

## **Меню (MENU)**

Меню Menu позволяет вам добавлять, изменять или удалять меню на вашем окне или на элементе управления OLE.

Когда вы устанавливаете меню для окна вашего приложения или для дочерних MDI-окон, Clarion автоматически сливает меню приложения с меню дочернего окна-MDI, когда

последнее имеет фокус. Это избавляет вас от проблем по включению, выключению, вставке и замещению разных меню в зависимости от того, на котором из окон сейчас сосредоточен фокус.

Обращайтесь к главе Меню и панели инструментов за указаниями по созданию меню и панелей инструментов для вашего приложения.

### **Меню Панели инструментов (TOOLBAR)**

Меню Toolbar (панель инструментов) позволяет вам добавлять, изменять или удалять панели инструментов из вашего окна.

Определите панель инструментов для окна вашего приложения или для дочерних MDI окон. Clariion автоматически сливает панель инструментов окна приложения с панелью дочернего окна MDI, когда последнее имеет фокус. Это избавляет вас от беспокойства по выключению, включению, вставке и замещению разных инструментов в зависимости от того, на котором окно сейчас имеет фокус.

Обращайтесь к главе Создание меню и панелей инструментов за указаниями по созданию меню и панелей инструментов для вашего приложения.

### **Меню заселения (POPULATE)**

Populate Menu (меню заселения) появляется в форматере окна только тогда, когда активен генератор приложений. Он помещает переменную поля или памяти в окне, вместе с соответствующим элементом управления. Для полей тип элементов управления зависит от того, как поле определено в словаре данных.

Когда генератор приложений активен, две новых пиктограммы инструментов появляются внизу панели инструментов; они соответствуют следующим командам:

**Field (поле)** Дает вам возможность поместить поле ввода, связанное с полем файла или переменной. Когда вы щелкаете мышью в окне, появляется диалоговое окно File Schematic Definition (Определение схемы файлов). Выберите поле или переменную, затем ЩЕЛКНИТЕ мышью в окне. Это эквивалентно инструменту Полей словаря в панели инструментов Элементы управления. Если вы заранее определили формат поля, на закладке (tab) Window диалога Field Properties (свойства поля) (например, назначая спин-управление), появится назначенный вами элемент управления, а не поле ввода.

**Multiple Fields (мультиполя)** Дает вам возможность поместить поле ввода, привязанное к полю или переменной. Когда вы щелкаете мышью в окне, появляется диалоговое окно File Schematic Definition (Определение схемы файлов). Выберите поле или переменную, затем ЩЕЛКНИТЕ в окне.

Если вы заранее определили формат поля, на закладке (tab) Window

диалога Field Properties (свойства поля) (например, назначая спин-управление), появится назначенный вами элемент управления, а не поле ввода.

После размещения первого поля диалоговое окно File Schematic Definition появится снова, готовое к тому, чтобы вы разместили другое поле. Когда все поля размещены, нажмите кнопку Cancel для возврата к нормальному редактированию.

### Control Template (шаблон

элемента управления)

Дает вам возможность добавить шаблон элемента управления к создаваемому окну. Выберите шаблон из диалогового окна Select a Control Template (выбор шаблона элемента управления).

Шаблон элемента управления добавляет элемент управления или элементы управления к окну, а также программу для их ведения. Например, шаблон элемента управления поле просмотра помещает в окно списочное поле, дает вам возможность выбрать поля для списка и добавляет все исполняемые программы для ведения списочного поля (загрузки его, чтения и т.д.).

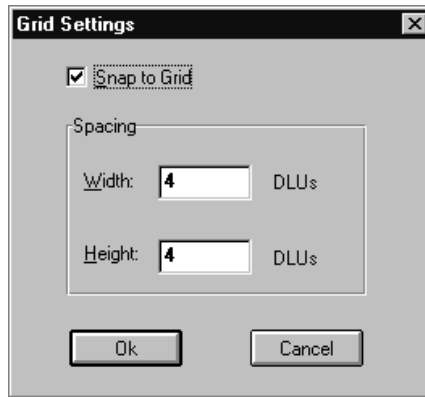
Поместив шаблон элемента управления, вы можете назначить его свойства и действия ПРАВЫМ ЩЕЛЧКОМ и выбором Properties (свойства) или Actions (действия) из всплывающего меню. Дополнительную информацию вы найдете в главе Шаблоны элементов управления, кода и расширения.

## **Меню Параметры (OPTIONS)**

Меню Options (Параметры) дают вам возможность показать или спрятать инструменты сетки и панели инструментов форматора окна

**Snap to Grid (выравнивание по сетке)** Чтобы переключить выравнивание по сетке в положение включено или выключено, выберите команду Snap to Grid (выравнивание по сетке).

**Grid Size (размер сетки)** Чтобы включить или выключить выравнивание по сетке, а также чтобы установить величины размера сетки, выберите команду Grid Size (размер сетки). Выравнивание по сетке вынуждает верхний левый угол новых элементов управления настраиваться на совпадение с узлом сетки в окне. Конечный пользователь не видит сетки во время работы; это только конструкторский инструмент. Для того чтобы включить выравнивание по сетке, отметьте поле Snap to Grid.



**Совет: Выравнивание по сетке элементов управления в ваших диалоговых полях придаст вашему приложению более профессиональный вид. Конкретные предложения по выравниванию различных типов элементов управления вы можете найти в Вопросах конструирования окон.**

Чтобы установить ширину и высоту ячейки сетки между ее точками, введите величины в поля Width (ширина) и Height (высота) в диалоговом окне Grid Settings (установки сетки). См определение диалоговых единиц в Глоссарии.

Show Toolbox (показать панель инструментов) Чтобы иметь возможность включить или выключить показ инструментальной панели элементов управления, выберите команду Show Toolbox (показ инструментальной панели).

При конструировании больших окон может оказаться полезным убрать инструментальную панель с экрана, выиграв тем самым дополнительное пространство для окна. Тем не менее, вы имеете доступ ко всем инструментам управления, выбирая их из меню Control и Populate.

Show Alignbox (показать панель выравнивания) Чтобы переключать показ панели выравнивания из положения включено в выключено и обратно, выберите команду Show Alignbox (показ панели выравнивания).

Это вопрос индивидуального предпочтения. Вы можете также получить доступ ко всем командам управления, выбирая их из меню Alignment (выравнивание).

Show Propertybox (показать панель свойств) Чтобы переключать показ поля свойств, выберите команду Show Propertybox.

Show Fieldbox (показать панель полей) Чтобы переключать показ Панели заселения из положения включено в выключено и обратно, выберите команду Show

Fieldbox (показ блока полей). Это вопрос предпочтения. Вы можете еще заселить поля с помощью панели инструментов Controls (элементы управления) или меню Populate (заселение).

### VBX Custom Control Registry

(регистратура заказных VBX элементов управления) Чтобы зарегистрировать заказную библиотеку элементов управления, список выберите команду VBX Custom Control Registry (регистратура заказных элементов управления). Нажмите кнопку Add (добавить) в диалоге VBX Custom Control Registry. Затем щелкните дважды мышью на имени файла .VBX.

Это дает возможность формaterу окна поместить элементы управления из библиотеки VBX в ваше окно (смотрите Элементы управления и их свойства). Чтобы удалить элемент управления из регистра, нажмите кнопку Remove.

### **Предварительный просмотр! (PREVIEW!)**

Чтобы показать активное окно, подобное тому, которое будет видеть ваш пользователь, выберите команду Preview (предпросмотр). Единственное отличие от реальности будет в том, что не будет натуральных данных в полях и командные кнопки не будут исполнять команды. Чтобы выйти из режим Preview!, нажмите ESC.

**Совет: Вы должны всегда испытывать ваши окна и диалоговые поля. Хотя формater окна визуальный, он не показывает вам, как трехмерная тень будет влиять на “вид” вашего окна, не будет он также “прятать” скрытые элементы управления. Кроме того, вы можете испытать порядок табуляции, находясь в режиме Испытания, чтобы проверить, что текущий порядок имеет смысл.**

### **Диалог Свойства окна (Window Properties)**

Используйте диалоговое окно Window Properties для установки всех свойств, или атрибутов, окна. В число свойств входят шапка окна, возможность изменения размера окна, связанные с окном полосы прокрутки, пиктограммы, сообщения, файлы помощи, типы курсора, а также многое другое. Короче говоря, все свойства, связанные с окном в отличие от свойств, связанных с процедурами, элементами управления, полями и т.д.

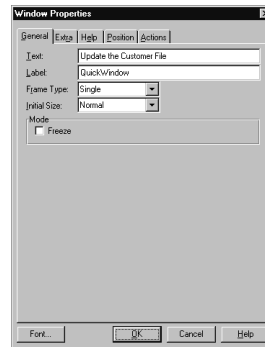
- ☐ Для показа диалога Window Properties (Свойства окна) из формatera окна вы можете:
- ☐ Щелкнуть правой кнопкой мыши на окне-образце и выбрать Properties из появляющегося при этом меню.
- ☐ Выбрать окно-образец и нажать кнопку ENTER.
- ☐ Отобразить окно-образец, а потом выбрать Edit III Properties из меню.



Кроме того, выбор окна из диалогового окна New Structure (новая структура) ведет к диалоговому окну Window Properties.

### Закладка “Общая” (General)

**Title (заголовок)** Чтобы установить текст заголовка окна, наберите константу в виде строки символов в поле Title (заголовок).



**Совет: Вы можете динамически изменить текст заголовка во время выполнения программы. Например вы можете поместить переменную имени файла в строку заголовка: ' Notepad - FileName.TXT'. Чтобы выполнить это, вставьте следующий код после открытия окна и прежде оператора ACCEPT:**

```
MyWindow {PROP:text} = 'My Caption -' & FileNameVariable
```

Если вы создаете системное модальное окно, оставьте линейку заголовка пустой. Нормальный стиль Windows для этого типа окна - показ окна без заголовка.

**Label (метка)** Чтобы определить метку окна, наберите ее в поле Label (метка). Эта метка используется для ссылки на WINDOW в исходной программе. В следующем примере “CustEntry” - метка для CustEntry окна:

```
CustEntry WINDOW ... ! defines CustEntry window
END
CODE
OPEN (CustEntry)      ! opens CustEntry window
```

Метка может содержать прописные и строчные буквы, цифры, символ подчеркивания или запятую. Пробелы запрещены. Первым символом должна быть буква или символ подчеркивания. Зарезервированные слова Clarion'a не могут служить метками.

**Frame Type (тип рамки)** Чтобы выбрать для вашего окна рамку, сделайте это в выпадающем списке **Frame Type (тип рамки)**, содержащем поддерживаемые системой типы рамок. Рамка определяет границы окна. Возможные варианты таковы:

**Single (одинарная)** Одинарное изображение рамки. Размеры одинарной рамки пользователь изменить не может. Одинарная рамка - наиболее подходящий вид рамки для полей диалогов.

**Double (двойная)** Толстая рамка, размеры которой пользователь не может изменить. Используйте рамку этого типа для системного модального окна (без линейки заголовка) или для поля модального диалога (с линейкой заголовка).

**Resizable (с изменяемыми размерами)** Толстая рамка, размеры которой пользователь может изменить. Выберите этот тип рамки для окон приложения и дочерних окон MDI.

**None (никакой)** Одинарное изображение рамки в Windows 95 и никакой рамки для Windows 3.1. Размеры одинарной рамки пользователь изменить не может. Одинарная рамка - наиболее подходящий вид рамки для полей диалогов.

**Initial Size (исходный размер)** Чтобы установить исходный размер и исходное состояние вашего окна, выберите опцию из выпадающего списка **Initial Size (исходный размер)**. Имеются следующие варианты:

**Normal (нормальный)** Показ окна с размером, принятым по умолчанию. Если вы не установите размер по умолчанию, библиотека поддержки Clarion установит его за вас.

**Maximized (увеличенный до предела)** Окно этого размера заполняет весь экран или всю рамку приложения, это зависит от того, является ли окно окном приложения, либо MDI дочерним окном.

**Iconized (свернутый до пиктограммы)** В Windows 3.1 это окно появляется в виде пиктограммы - как окно с изображением 32 на 32 внизу экрана (окно приложения) или внутри в пределах рамки приложения (для дочернего MDI окна).

В Windows 95 окно появляется в пиктографическом состоянии в строке задачи.

**Совет: Если вы выбираете свернутое до пиктограммы окно, не забудьте указать в поле Icon (пиктограмма) имя файла, содержащего данную пиктограмму. Если вы не сделаете этого,**

**ваше окно не сможет получить команду Restore (восстановление) в своем системном меню и, следовательно, навсегда останется свернутым до уровня пиктограммы. Указание имени файла также добавит окну кнопку minimize (уменьшать до предела) к элементам управления окном, давая возможность пользователю снова свернуть окно до пиктограммы после его восстановления к нормальному виду.**

Freeze

(замораживание) Чтобы “заморозить” все элементы управления на окне, так чтобы к ним не применялись последующие изменения словаря данных, отметьте данное поле. Вы можете отменить атрибут #Freeze для всех элементов управления или же только для отдельных элементов управления. См. Генератор приложений - Установка опций приложения.

### Закладка “Дополнительные свойства” (Extra Properties)

Colors (цвета) Введите правильное значение цвета в любое из следующих полей для того, чтобы добавить атрибут COLOR к объявлению вашего окна WINDOW. См. Справочник языка, где имеется дополнительная информация о ЦВЕТЕ и ОКНЕ.

См. ..\CW20\LIBSRC\EQUATES.CLW для получения списка приравнивания правильных цветов.

Смотрите: Приложение Вопросы конструирования окон, где обсуждается проблема использования цвета для улучшения вашего приложения.

Background (фон) Чтобы применить определенный цвет ко всему окну, за исключением линейки заголовка, выбранного текста и элементов управления окна, наберите правильное значение приравнивания цвета в этом поле или же нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна Background Color (цвет фона).

Selected Text

(выбранный текст) Чтобы применить определенный цвет для выбранного текста окна, наберите правильное значение приравнивания цвета в этом поле или же нажмите эллиптическую (...) кнопку для того, чтобы выбрать нужный цвет в диалоговом окне Selected Color (выбранный цвет).

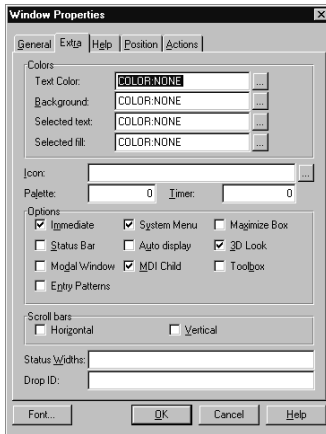
Selected fill

(выбранное заполнение) Чтобы применить определенный цвет к фону выбранного текста в окне, наберите правильное значение приравнивания цвета в этом поле или же нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна Selected Background Color (выбранный цвет фона).

Icon (пиктограмма) Чтобы связать с окном пиктограмму, определите пиктограмму в поле Icon. Вы можете набрать имя файла или нажать эллиптическую кнопку

(...), затем выбрать имя файла пиктограммы, используя для этого стандартное диалоговое окно Open File(открытие файла).

Вы всегда должны назначать пиктограмму для окна приложения и для дочерних MDI-окон. Объявление имени пиктограммы автоматически размещает в строке заголовка вашего приложения или дочернего MDI окна кнопку minimize .



Диалоговое окно о Свойства Окна показывает атрибуты, которые вы добавляете к структуре Window.

**Palette (палитра)** Чтобы установить максимальное количество оттенков цвета, заполните поле Palette (палитра). Это правда не означает, что оборудование вашего конечного пользователя будет в состоянии поддерживать такую палитру. Наберите полное число цветов, которое вы хотите поддержать. Например, 24-битовый цвет даст палитру в 16777215 цветов. Оставьте это поле пустым для установки палитры, принятой в системе конечного пользователя по умолчанию.

**Timer (таймер)** Чтобы определить окно для получения сообщений о событиях таймера из Windows , заполните поле Timer. Установите интервал таймера в сотых долях секунды.

Например, если вы установите в поле цифру 100, окно будет автоматически получать событие EVENT:Timer один раз в секунду (100/100 секунд). Это может быть удобным для добавления часов к строке состояния. См. также Вопросы конструирования окон - Обработка фонового процесса и раздел Таймер в Справочнике языка.

**Совет:** Хотя Windows налагает ограничения на количество активных таймеров, вы можете поместить столько таймеров и во столько же окнах в вашем приложении Clarion for Windows, сколько

**хотите. Во время прогона ваше приложение использует один и только один таймер.**

#### Options

(возможности) Чтобы включить или выключить различные возможности, отметьте или очистите соответствующие флажки.

#### Immediate

(немедленный) Чтобы генерировать событие каждый раз, когда конечный пользователь передвигает окно или изменяет его размеры, сделайте отметку в поле Immediate(немедленный). Вы отвечаете за программу, которая выполняется после извещения о событии.

#### Status Bar

(строка статуса) Чтобы обеспечить строку для сообщений в нижней части вашего окна сделайте отметку в поле Status Bar. Смотрите ниже Ширины строки статуса относительно информации о сегментации или зонировании вашей строки статуса.

**Совет: строка сообщений в окне приложения - это прекрасный способ обеспечить обратную связь с пользователем в вашем приложении. Clarion делает простой процедуру отправки сообщения в строку состояния, сообщая пользователю, что именно делает приложение. Усиление обратной связи с пользователем сказывается в том, что пользователь чувствует себя свободнее в управлении. Это придает пользователю больше уверенности в правильности своих действий и повышает эффективность использования вашего приложения.**

#### Modal Window

(модальное окно) Чтобы сделать ваше окно системным модальным окном, сделайте отметку в поле Modal Window . Это поле появится с уже проставленной отметкой, если вы выберете System Modal Window (системное модальное окно) из диалога New Structure (новая структура).

Системное модальное окно берет на себя управление всей системой и предупреждает любые другие задачи - даже в других приложениях - с исполнения до момента, пока окно не будет закрыто.

#### Entry Patterns

(Заполнение полей ввода) Чтобы добавить атрибут MASK к вашему окну, установите флажок в поле Entry Patterns . Это заставит Clarion шаблоны изображения для всех полей в это окне. Например, вы могли установить фон ввода @P###-##-####P для номера поля социального страхования.

Шаблоны изображения полей ввода или символьные шаблоны поля ввода устанавливаются на закладке General Tab в диалоговом окне Entry

Properties. См. главу Установка свойств элементов управления (Установка свойств поля ввода. См. более полную информацию об атрибуте MASK в Справочнике языка.

### System Menu

(системное меню)

Чтобы разместить в вашем окне системное меню, сделайте отметку в поле System Menu(системное меню). System Menu всегда активируется кнопкой, полем или пиктограммой в верхнем левом углу окна. Стандартные позиции System Menu включают Restore (восстановить), Minimize (минимизировать), Maximize (максимизировать) и Close (закреть).

Каждая рамка приложения должна иметь системное меню. Для пользователей систем, не имеющих мыши, системное меню обеспечивает единственный способ изменения размера окна и его свертки.



Стандартное системное меню

**Совет:** Даже если вы планируете, что окно вашего готового приложения НЕ должно иметь системного меню, хорошей практикой является размещение системного меню в вашем приложении, пока вы занимаетесь его разработкой. С помощью двойного щелчка мышью на системном меню или выбором Close вы можете закрыть свое приложение, даже если ваша нормальная процедура выхода не работает.

**Совет:** Даже если оно не является необходимым, системное меню служит для создания дополнительных удобств в поле диалогов. Многие пользователи автоматически делают операцию двойного щелчка мышью над полем системного меню для закрытия диалога (так как оно всегда сохраняется на стандартном месте, его иногда проще найти, чем найти кнопку Cancel).

### Auto Display

(автоматический показ)

Чтобы добавить атрибут AUTO к вашему окну, сделайте отметку в поле Auto Display (автоматический показ). Это автоматически обновляет содержание всех элементов управления на экране при каждом проходе цикла ACCEPT.

## MDI Child

(дочернее MDI окно) Чтобы установить что ваше окно - дочернее MDI окно, сделайте отметку в поле MDI Child. Дочернее окно не может перемещаться за пределы главного окна приложения. Типичным применением окна MDI может быть представление различных расположений данных в базе данных вашего приложения.

## Maximize Box

(поле максимизации) Для размещения кнопки maximize (увеличение до предела) в вашем окне сделайте отметку в поле Maximize Box. В общем случае вы должны помещать кнопку maximize в главных окнах приложения и дочерних MDI- окнах, но не в диалоговых окнах.

## 3D Look

(трехмерный вид) Чтобы придать серому окну трехмерный выпуклый вид в вашем приложении, сделайте отметку в поле 3D Look. Это конечно вопрос стиля, но это большой шаг на пути придания вашему приложению профессионального вида.

Серый фон невидим на стадии проектирования вашего окна с помощью формatera окна. Однако, он становится видимым в режиме Preview! (предварительный просмотр) и во время исполнения вашего приложения.

## Toolbox

(поле инструментов) Чтобы добавить атрибут TOOLBOX к вашему окну, отметьте поле Toolbox. Атрибут TOOLBOX заставляет ваше окно всегда располагаться “поверх” других окон.

Scroll Bars (строки прокрутки) Чтобы переключить следующие опции строки прокрутки между положениями вкл/выкл, отметьте или очистите соответствующие поля.

## Horizontal

(горизонтальная) Чтобы добавить к вашему приложению горизонтальную строку прокрутки, отметьте поле Horizontal.

## Vertical

(вертикальная) Чтобы добавить к вашему приложению вертикальную строку прокрутки, отметьте поле Vertical.

## Status Widths

(ширина строки статуса) Чтобы установить ширину зоны (зон) строки статуса, наберите необходимую величину или список величин в поле Status Widths (ширина статуса). Вы должны также установить флажок в поле Status Bar (линейка статуса). Смотрите выше. Величины, которые вы вводите в это поле, обеспечивают параметры атрибута STATUS() для вашего окна. Более полную информацию об атрибуте STATUS() вы найдете в Справочнике языка.

Если ваше приложение не имеет строки статуса или имеет только одну зону на строке статуса, вы можете пропустить это поле.

Зоны - это области внутри строки статуса, отделенные друг от друга трехмерными заштрихованными полями. Первая зона слева по умолчанию показывает текст атрибута MSG от управления с входным фокусом. Это полезно для вывода коротких инструкций помощи или другой информации для пользователя. Если ваше приложение имеет только одну зону в строке статуса, вы можете пренебречь этим полем.

Вводимые вами величины представляют ширину каждой зоны в диалоговых единицах. Диалоговая единица равна 1/4 ширины среднего символа в наборе символов по умолчанию. Так величина 40 создает зону примерно в 10 символов длиной. Величина 400 создает зону примерно в 100 символов длиной.

Вы можете установить растягиваемую зону путем набора отрицательного числа. Это создает зону с минимальной шириной, но расширяемой насколько позволяют размеры окна.

Используйте стандартный синтаксис назначения свойства для размещения текста в любой зоне. Для размещения строки символов во второй зоне, например:

```
MyWindowLabel {PROP:Status Text , 2} = 'Some String'
```

**Совет: Многозонная строка статуса может придать вашему приложению профессиональный вид. Например, вы можете показать текст помощи в зоне один , а номер текущей записи при редактировании - в зоне два.**

## Drop ID

(Идентификатор мишени

операции “Drag and Drop”) Чтобы добавить к вашему окну атрибут DROPID, наберите до 16 разделенных запятыми сигнатур в поле Drop ID (Идентификатор мишени операции “Drag and Drop”). Атрибут DROPID указывает, что это окно является правильным объектом для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop правильны для этого окна.

Drag and drop означает, что конечный пользователь может выбрать объект в одном окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

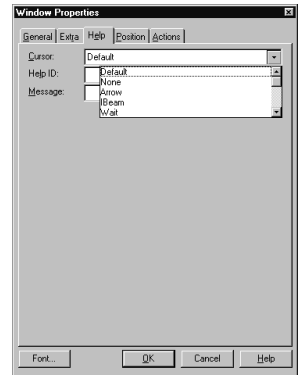
Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID цели, окна или элемента управления, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную



программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочнике языка. Смотрите также главу Форматер списочного поля- Добавление способности drag and drop к списочному полю.

### Закладка “Помощь” (Help)

**Cursor (курсор)** Чтобы определить вид курсора, используемого в окне, выберите тип курсора из выпадающего списка или наберите имя .CUR файла. Когда пользователь перемещает курсор над вашим окном, курсор принимает ту форму, которая определена в файле .CUR. Элементы управления внутри окна автоматически наследуют тот же самый тип курсора, если не приняли решение изменить его.



**Совет: Смотрите главу Конструирование окон, в которой даны советы, как использовать каждый курсор.**

### Help ID

**(идентификатор помощи)** Чтобы связать Help ID(идентификатор помощи) с окном, заполните поле Help ID строкой символов, которая является или ключевым словом или контекстом (перед контекстной строкой поставьте символ тильды ~) в поле . Это формирует атрибут HLP в структуре Window.

**Совет: Вы должны подготовить свой файл помощи, используя для этого систему подготовки текстов, которая поддерживает выход к файлам .RTF (такой как, например, Microsoft Word for Windows™). С помощью Windows Help Compiler фирмы Microsoft Вы должны компилировать файл помощи.**

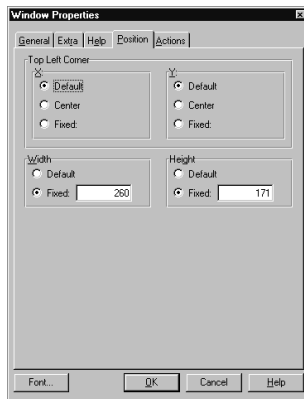
Когда пользователь вызывает Windows Help при активном в данный момент окне, подсказка открывается на теме, которая ассоциирована с данным активным окном. Если вы установите Help ID для отдельных элементов управления в окне, при получении фокуса эти элементы будут отменять Help ID окна и связываться с собственными темами помощи.

При генерации программы генератор приложений вызывает контекстную строку символов или ключевое слово в файле .HLP, определенном вами, в диалоговое окно Application Properties (свойства приложения). Для получения более полной информации смотрите главу Использование генератора приложений.

**Message (сообщение)** Чтобы показать строку символов в первой зоне строки статуса, когда окно активно, наберите строку в поле Message (сообщение). Это обеспечит атрибут MSG для окна. Сообщения могут быть также установлены для элементов управления в окне. Когда элемент управления имеет фокус, связанное с этим элементом сообщение будет показано вместо сообщения окна. Смотрите выше разделы Строка статуса и Ширины статуса, где имеется более полная информация о строке статуса. Об атрибуте MSG смотрите Справочник языка.

### **Закладка “Позиция” (Position Properties )**

У вас имеется возможность отрегулировать размер своего окна для этого можно использовать рукоятки окна. Рукоятками являются маленькие цветные квадратики, которые появляются в углах и по сторонам выбранной детали. Однако вы можете установить размер окна и его позицию из Position Properties Tab (закладка Позиция).



Диалоговое окно Position управляет положением и размерами окна. При выборе положения по умолчанию Windows помещает ваше окно в точку, независимую от того, где было открыто последнее окно (даже от другого приложения).

#### **Top Left Corner**

(верхний левый угол) Чтобы установить исходное положение (верхний левый угол) вашего окна, выберите желаемые координаты X и Y. Вариантами являются: Default (по умолчанию) Инструктирует Windows установить координаты X и/или Y позиции верхнего левого угла окна, равные величине принятой по умолчанию, которая будет зависеть от системы пользователя и от числа других активных приложений.

**Совет: Для придания вашему приложению “стандартного” вида, свойственного другим приложениям Windows, там, где это возможно, определите установку Default.**

#### **Center (центрирование)**

Помещает ваше окно в центр экрана. Вы можете выбрать горизонтальное или вертикальное центрирование или то и другое одновременно. Добавляет атрибут CENTER к WINDOW. Смотрите Справочник языка.

**Fixed (фиксированный)** Для задания некоторого конкретного положения и размера, сначала установите флажок в поле Fixed для координат X и Y. Это позиционирует верхний левый угол окна. Для рамки APPLICATION эта позиция берется по отношению к экрану, для дочернего MDI окна эта позиция берется относительно рамки APPLICATION для области пользователя.

Единицы измерения для этих полей - это диалоговые единицы. Диалоговые единицы являются относительными единицами, основанными на наборе символов по умолчанию. Диалоговая единица равна 1/4 ширины среднего символа и 1/8 высоты среднего символа. Таким образом, Windows автоматически изменяет положения окна при различном разрешении экрана.

**Width (ширина)** Чтобы установить ширину вашего окна, выберите желаемую величину, Вариантами являются:

**Default (по умолчанию)** Инструктирует Windows установить ширину окна по величине умолчания, которая зависит от системы пользователя и от числа других активных окон и приложений.

**Fixed (фиксированное)** Чтобы установить особую величину, отметьте Fixed choice (фиксированный выбор) и установите величину. Это даст ширину окна в диалоговых единицах.

**Height (высота)** Чтобы установить высоту вашего окна, выберите желаемую величину. Варианты выбора те же, что и для ширины.

### **Закладка действий**

Нет действий по умолчанию, предназначенных специально для окна. Однако данная закладка обеспечивает альтернативный доступ к Схеме файлов и точкам вставки кода для данной процедуры. Просто нажмите соответствующие кнопки.

## **Размещение элементов управления в окне**

---

В данном разделе объясняется, как разместить элемент управления в окне. В главе Элементы управления и их свойства объясняется, как осуществить настройку элементов управления, которые вы размещаете в вашем окне.

Чтобы разместить элемент управления:

1. Щелкните мышью на пиктограмме нужного вам элемента управления в инструментальной панели Controls (элементы управления) или выберите элемент управления из меню Controls (элементы управления) или Populate (заселение).

2. После того как вы отобрали нужный элемент управления или шаблон элемента управления, проведите курсором поперек образца окна.

При этом курсор приобретает форму креста. Поместите крест там, где вы хотите, чтобы появился элемент управления.

3. Щелкните мышью.

Верхний левый угол данного элемента управления помещен на перекрестии курсора после этого ЩЕЛЧКА клавиши мыши.

4. Если вы выбрали LIST, COMBO или BUTTON и отметили поле Translate controls to control templates when populating (транслируйте элементы управления шаблонам управления при заселении) в диалоговом окне Application Options (опции приложения), сделайте выбор из списка элементов управления и шаблонов управления.

Шаблоны элементов управления создают элементы управления и исходную программу для их поддержания. Например, шаблон элемента управления Поле просмотра не только генерирует исходную программу для показа элемента управления списочного поля, он также генерирует код, предназначенный для того, чтобы загрузить данные из файла в очередь QUEUE, затем показать данные в списочном поле с возможностью полного прокручивания и выбора с помощью щелчка мыши.

**Совет: В общем случае вам выгоднее использовать шаблон элемента управления нежели просто голый элемент управления.**

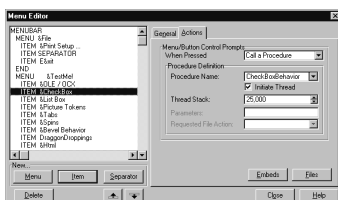
**Совет: При использовании Формatera окна для размещения Поля словаря автоматически открывается диалоговое окно Select Field (выбор поля), так что вы можете отобразить или определить поле словаря данных или переменную памяти для связи с данным элементом управления. Выполнив размещение, вы можете получить доступ к диалоговому окну свойств элемента управления из меню редактора или из всплывающего меню, открывающегося правым щелчком мыши..**

5. При необходимости ЩЕЛКНИТЕ и потащите за ручки элемент управления, чтобы изменить его размер. ЩЕЛКНИТЕ на внутренней части элемента управления и протащите мышью для его перемещения .

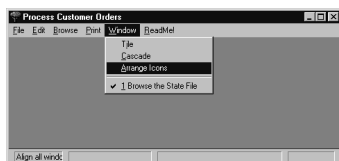
## Глава 9 Меню и панели инструментов



Создавайте меню и панели инструментов с помощью Редактора меню, к которому вы получите доступ через Форматер окна.



Диалог Редактор меню представляет структуру MENUBAR в виде дерева. Добавьте позиции меню, набирая их в соответствующих полях.



STD ID Clarion for Windows (Стандартные функции Clarion for Windows) обеспечивает автоматическое выполнение обычных команд Windows, таких как Copy (копировать) и Paste (вклеить). Этот файл обеспечивает даже автоматический список окон для MDI приложений.



Clarion for Windows обеспечивает стандартную панель пиктограмм инструментов для общепринятых функций, таких как copy и paste.



Удобное размещения любого типа элементов управления в панели инструментов.

Меню интерфейса многих документов (MDI)

Сливающиеся меню

Планирование и выполнение меню

Вызов редактора меню

Создание меню вашего приложения

Другие функции редактора меню

Реализация стандартного поведения Windows

Установка положений меню и поведения слияния

Добавление горячей клавиши

Другие поведения меню - блокирование и переключение

Управление вашим меню

Панели инструментов

Слияние панелей инструментов

Кнопки

Кнопки с фиксацией

Взаимно исключающие кнопки

Предварительный просмотр меню и панелей инструментов

Меню - это просто список различных действий, которые может выполнять ваше приложение. В Clarion этот список действий (меню) показывается с помощью структуры MENUBAR (линейка меню) , структур MENU (меню) и ITEMS (позиций). В этой главе слово меню в родовом смысле относится к списку действий, которые может выполнить ваше приложение. Слова MENUBAR, MENU и ITEM используются для ссылки на операторы языка Clarion, которые определяют меню вашего приложения.

Данная глава:

- ◆ Обсудит вопросы динамического сопровождения меню и панелей инструментов для приложений типа Интерфейс многих документов (MDI).
- ◆ Покажет вам, как вызвать редактор меню Menu Editor и как создать меню.
- ◆ Объяснит, как автоматически выполнить стандартное поведение Windows (SWB) для таких команд, как Edit > Copy путем привязывания Clarion Standard ID (STD атрибут) к позиции (ITEM) или меню (MENU).
- ◆ Покажет вам, как создать панель инструментов. Clarion даже обеспечивает доступ к пиктограммам для стандартных действий, таких как File > Open так что ваше приложение будет выглядеть более профессиональным.

## ***Меню интерфейса многих документов (MDI)***

Приложения Интерфейса многих документов (MDI) предъявляют особые требования к программе. Часто программа может поддерживать разнообразные документные окна, каждое из которых имеет слегка отличный набор команд, из которого пользователь может сделать выбор. Смотрите дополнительную информацию об этом в приложении Вопросы конструирования окон.

### **Сливающиеся меню**

---

Обычно в MDI приложении разработчик пишет программы для слежения за тем, какое окно активно, и для изменения меню и панелей инструментов чтобы отразить текущие опции, доступные для пользователя. Clarion делает это автоматически путем слияния меню и панелей инструментов в соответствии с предпочтениями, которые вы назначаете с помощью Menu Editor (редактор меню). Однако точное описание требует некоторого понимания и планирования со стороны разработчика приложения.

#### **Глобальные позиции**

В рамке APPLICATION (приложение) MENUBAR определяет глобальные позиции меню для программы. Эти глобальные позиции меню обычно доступны на всех дочерних MDI окнах. Однако если на MENUBAR приложения присутствует атрибут NOMERGE (неслияние), тогда нет глобального меню и меню приложения является локальным меню, показываемым только тогда, когда не открыто ни одно дочернее MDI окно.

#### **Локальные позиции**

На MDI-дочернем окне MENUBAR определяет позиции локального меню, которые автоматически слиты с позициями глобального меню, определенного на MENUBAR приложения. Как позиции глобального, так и позиции локального меню доступны тогда, когда MDI-дочернее окно имеет входной фокус. Как только окно теряет фокус, позиции локального меню убираются из позиций глобального меню. Если атрибут NOMERGE определен на MENUBAR дочернего MDI окна, локальное меню переписывает и заменяет глобальное меню.

#### **Не-MDI окна**

На не-MDI окне позиции локального меню никогда не сливаются с позициями глобального меню. MENUBAR на не-MDI окне всегда появляется в окне и не на рамке приложения, которое могло быть ранее открыто.

#### **Порядок слияния**

Нормально, когда меню MDI окна (локальные позиции) сливается в меню приложения (глобальные позиции), позиции глобального меню появляются “первыми”, за ними следуют



позиции локального меню. Первые означает либо ближе к левой стороне либо к верху, в зависимости от того, показан ли слитый выбор на строке действий (горизонтальный список) или в меню (вертикальный список).

Процесс слияния также учитывает, совпадают ли какие-нибудь локальные меню с глобальными меню. MENUs, которые имеют то же самое имя и тот же самый уровень MENUBAR, являются совпадающими. Когда совпадений нет, меню сливаются в нормальном порядке. Однако когда меню совпадают, одиночное меню (вертикальный список) суммируется с глобальными выборами, появляющимися над локальными позициями. Это новое меню имеет все атрибуты глобального меню, такие как MSG, FIRST и т.д. Внутри этого слившегося меню подменю также слиты в одиночное меню. Обратите внимание, что позиции ITEMS не слиты даже тогда, когда они совпадают.

Нормальный порядок слияния может быть модифицирован при использовании выпадающего списка Menu Editor's Position (положение редактора меню) (см. ниже Установка положений меню и поведение слияния) чтобы добавить атрибуты FIRST (первый) или LAST (последний) к индивидуальным меню и позициям. Приоритет положению слияния:

1. Глобальные позиции с атрибутом FIRST
2. Локальные позиции с атрибутом FIRST
3. Глобальные позиции без атрибутов FIRST или LAST
4. Локальные позиции без атрибутов FIRST или LAST
5. Глобальные позиции с атрибутом LAST
6. Локальные позиции с атрибутом LAST

Дополнительная информация об этих атрибутах имеется в Справочнике языка.

## Планирование и выполнение меню

---

Чтобы создать меню для MDI приложений:

1. Создайте основное меню для рамочного окна приложения.

Вероятнее всего оно будет включать меню Файл и меню Помощь, так как они содержат функции, которые доступны даже тогда, когда не открыт ни один документ.

**Совет: Шаблон процедуры Рамка приложения Clarion поступает с заранее определенным меню с многими из самых известных уже поставленных вам функций.**

Для создания своих меню вы будете пользоваться Window Formatter's Menu Editor (редактор меню форматера окна) . Будьте внимательны и выберите атрибут FIRST для Меню Файл и атрибут LAST для Меню Помощь из выпадающего списка Position (положение). Это обеспечит вам, что когда Clarion сольет это глобальное меню с локальными меню, Файл и Помощь сохранят свои правильные положения.

**2. Спланируйте дополнительные меню для дочерних окон.**

Могут ли они все иметь одинаковые заглавия меню? Многие ли команды у них общие? В идеале, большинство меню и позиций меню могут быть активными во всех дочерних окнах. Если имеется всего несколько команд, специфичных для некоторых окон, планируйте выключать те меню и позиции меню в окнах, которые не поддерживают их, и включать те, которые поддерживают.

**3. Создайте меню для первого дочернего окна.**

Вы опять будете использовать Window Formatter's Menu Editor для создания данного меню. Добавьте какие-нибудь позиции, специфичные для окна, к первому дочернему окну. То есть, специфичные для окна меню, которых нет в рамке приложения - такие как Edit, Insert и т.д.

Если необходимо, добавьте к первому дочернему окну меню Файл. Это нужно только тогда, когда данное дочернее окно нуждается в Позиции меню Файл, которая еще не включена в меню Файл приложения. Например, может быть полезным дополнение команды Close (закрыть). Если это так, добавьте меню Файл к первому дочернему окну. Добавьте элемент Close к меню Файл.

Добавьте меню Window к первому дочернему окну. Меню Window стандартны для большинства программ окон. Типичное меню Window включает следующие элементы: Arrange Icons (расположить пиктограммы), Tile (черепица), Cascade (каскад) плюс список документов (окон), который показывает все открытые дочерние окна и позволяет пользователю переключаться между ними. Во многих случаях полное меню, включая список документов, может быть реализовано со стандартными ID's (идентификаторами) (StID's). Смотрите ниже Создание меню вашего приложения.

**4. Выйдите из Menu Editor (редактор меню) и сохраните меню.**

**5. Проверьте взаимодействие этих двух меню.**

Сливаются ли они так, как вы планировали? Правильные ли имеются позиции для окна, которое находится в фокусе? Любые регулировки выполните с помощью Menu Editor.

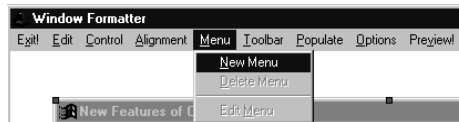
6. Повторите шаги с 3 по 5 для других дочерних окон.

## Вызов редактора меню

Чтобы создать меню для вашего приложения, воспользуйтесь редактором меню Menu Editor. Доступ к редактору меню вы получите через формater окна Window Formatter.

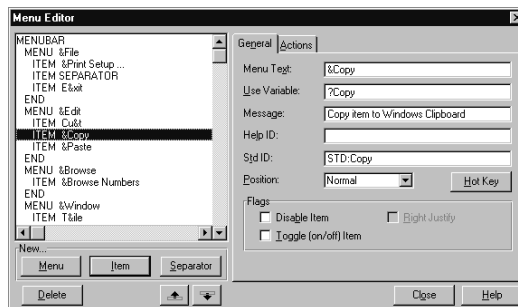
**Совет: Вы можете также создать панель инструментов для вашего приложения с помощью формatera окна. Вы можете поместить любой тип элементов управления на свою панель инструментов, хотя вероятнее всего вы больше будете пользоваться командными кнопками.**

В этом разделе приводятся подробные примеры использования редактора меню для создания меню. Из формatera окна выберите Menu > New Menu для создания нового меню или выберите Menu > Menu Editor для редактирования существующего.



Диалоговое окно редактора меню визуально представляет структуру данных Clarion MENUBAR.

Дерево меню (в левой стороне диалогового окна) выглядит как упрощенный синтаксис языка Clarion, содержащий следующие ключевые слова Clarion:



Редактор меню показывает создаваемое меню как иерархический список.

- Ключевое слово MENUBAR наверху.
- Оператор или операторы MENU, за которыми следуют имя меню и соответствующий оператор END.
- Оператор или операторы ITEM, за которыми следует имя команды.

Командные кнопки редактора меню позволяют вам добавить и удалить меню и элементы меню. Вы можете также перемещать меню и элементы внутри структуры MENUBAR (линейка меню) с помощью кнопок ⏴ и ⏵.

Правая сторона диалогового окна позволяет вам установить текст ваших меню и позиций меню, метки эквивалентности, используемые для ссылки на меню и позиции меню в исполняемой программе, а также действия, которые происходят, когда пользователь выбирает позицию меню.

**Совет: При использовании генератора приложений каждую Позицию, которую вы помещаете на меню или линейку меню, автоматически добавляет точку вставки к дереву реагирования на контрольное событие в диалоге Embedded Source (вставленная исходная программа). Это дает вам возможность легко добавить функциональные возможности вашим элементам меню.**

Следующий раздел главы описывает пошаговую процедуру создания меню. Далее за ним следуют разделы, детально рассматривающие Команды и Опции редактора меню и обсуждение соображений, которые надо принимать во внимание при создании меню MDI приложения.

## Создание меню вашего приложения

Ниже приводится пошаговая процедура создания меню, начинающаяся с пустого окна в формате окна.

1. Выберите команду Menu III New Menu.

Появится диалоговое окно Menu Editor (редактор меню). В данный момент в нем присутствует только оператор MENUBAR.

2. В групповом поле New (новый) нажмите кнопку Menu (новое меню).

В результате будет добавлен первый оператор MENU, готовый для редактирования, его имя и соответствующий ему оператор END.

Знак амперсанда & в имени меню означает, что символ, следующий за &, будет служить клавишей быстрого доступа. Т.е., символ подчеркнут (например: Menu1 ) и когда пользователь нажимает ALT+ быстрая клавиша, появляется меню.

3. В поле Menu Text (текст меню) наберите текст для этого меню.

Например, наберите &File, чтобы конечный пользователь увидел File.

4. В поле Use Variable (используемая переменная ) наберите Метку соответствия поля.

Метка соответствия поля начинается с вопросительного знака (?) и вам следует сделать ее описательной. Например, ?Файл показывает, что это меню для манипуляции файла. Вы можете сослаться на данное меню в исполняемой программе с помощью его метки соответствия поля.

5. В групповом поле New нажмите кнопку Item (позиция).

В результате между оператором MENU и его оператором END будет вставлен оператор ITEM . Обратите внимание, что позиции меню используются для исполнения команд или процедур, в то время как меню используются для показа выбора других меню или элементов.

6. В поле Menu Text наберите текст для этой позиции меню.

Например, наберите &Open, чтобы конечный пользователь увидел Open. Знак & в имени элемента означает, что символ, следующий за &, является быстрой клавишей. То есть, символ подчеркнут, и когда пользователь нажимает быструю клавишу, выполняется действие, связанное с позицией.

**Совет: быстрая клавиша меню требует для своего действия клавишу ALT, в то время как быстрая клавиша позиции не требует клавиши ALT, но требует, чтобы позиция меню была видна. Другой**

**метод доступа к меню и элементам описан ниже в разделе  
Добавление горячей клавиши.**

7. В поле Use Variable (используемая переменная ) наберите метку соответствия поля Field Equate Label.

Метка соответствия поля имеет впереди вопросительный знак (?) и вам следует сделать ее описательной. Например, ?File Open показывает при беглом взгляде желаемую цель данной позиции меню открыть файл.

На элемент меню в исполняемой программе вы ссылаетесь с помощью метки соответствия. Например, в цикле ACCEPT сгенерированной исходной программы структура CASE FIELD() будет ссылаться на метку соответствия поля ?FileOpen тогда, когда будет проверять, не выбрал ли пользователь этот элемент меню.

8. Наберите содержание атрибута MSG в поле Message (сообщение).

Этот текст выводится в строке статуса (если разрешено) когда пользователь отмечает это меню или позицию меню.

9. В поле Help ID наберите либо ключевое слово помощи, либо контекстную строку, присутствующую в файле .HLP.

Если вы заполнили Help ID для меню или позиции, когда пользователь выделяет меню или позицию и нажимает F1, открывается файл помощи на нужном вопросе. Если ключевому слову соответствует более одной темы, появляется диалоговое окно помощи.

Поле Help ID (атрибут HLP) принимает строчную константу, назначающую ключ для доступа к конкретному вопросу в файле Windows Help. Это может быть либо ключевое слово помощи, либо контекстная строка. Ключевое слово помощи - это слово или фраза, индексированные таким образом, что пользователь может искать его в диалоговом окне помощи Search (поиск).

**Совет: При разработке файла помощи Windows вы указываете ключевое слово при помощи подстрочного символа 'K'. Контекстная строка помощи является произвольной строкой, которая уникально идентифицирует каждую страницу оглавления файла помощи для компилятора помощи Windows. При создании файла помощи подстрочный знак # помечает контекстную строку. Эта задача всегда выполняется для вас продавцами инструмента помощи.**

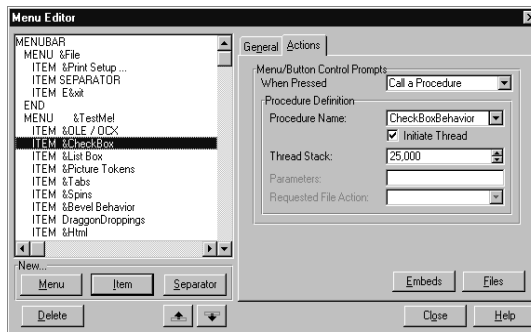
При ссылке на контекстную строку в поле Help ID вы должны идентифицировать ее предваряющим знаком тильды (~).

10. Из закладки Actions (закладка действий) выберите Call a Procedure (вызов

процедуры) из выпадающего списка When Pressed (когда нажата).

Назначенная вами процедура выполняется тогда, когда пользователь выбирает эту позицию. Вы можете установить параметры для выполнения и стандартные файловые действия (вставить, изменить, удалить или выбрать) если это нужно, вы также можете инициировать новый процесс. Процедура появляется как позиция помеченная как “ToDo” в вашем дереве приложения (если вы не вызвали процедуру, которая уже существует).

Использование закладки Действия (Actions) для назначения вызываемой процедуры при выборе позиции меню.



Это один путь придания функциональных возможностей вашей позиции меню. Вы можете также придать функциональные возможности, сделав выбор Run a Program (выполнить программу) из выпадающего списка, путем вставки исходной программы или набирая STD ID в поле STD ID. STD ID придает вашему приложению стандартное поведение Windows (SWB) для таких обычных действий, как File/Open и Edit/Cut, Copy а также Paste. Смотрите ниже Осуществление стандартного поведения Windows.

Выполнив по очереди эти шаги, вы получаете одиночное меню, называемое Файл с одиночной позицией, называемой Открыть (Open). Чтобы добавить к меню другие позиции, повторите шаги с 4 по 11. Чтобы добавить второе меню, выберите оператор END и нажмите кнопку Menu. Чтобы добавить подменю, выберите оператор MENU или ITEM и нажмите кнопку Menu.

11. Чтобы завершить меню и вернуться к формату окна, нажмите кнопку Close (закрыть).

## Другие функции редактора меню

Дополнительные кнопки и поля флажков позволяют вам устанавливать исполнение стандартных действий окон, размещение разделителей меню, назначение горячих клавиш, состояний меню по умолчанию и предпочтений при слиянии меню.

### Реализация стандартного поведения Windows

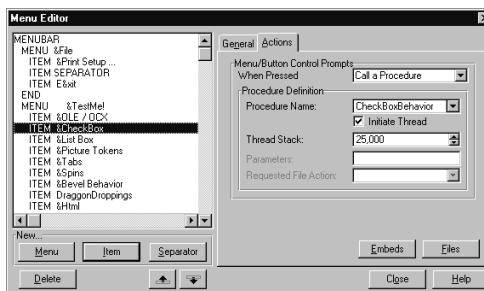
Имеются некоторые меню и команды, которые вы встречаете почти во всякой Windows-программе. Например, Cut (вырезать), Copy (копировать), Paste (вставить из буфера). Clarion предоставляет простой метод осуществления этих стандартных действий в ваших меню приложения - с помощью поля Std ID (стандартные команды) на диалоговом окне Menu Editor (редактор меню).

Чтобы установить стандартное действие для вашей позиции меню, введите одно из приравниваний (список их приведен ниже) в поле StdID. Clarion будет автоматически выполнять эти команду с помощью стандартного поведения окон и вам не потребуется никакой другой поддержки ее в вашей программе. Стандартные метки соответствия и ассоциированные с ними действия содержатся также в файле C:\CW\LIBSRC\EQUATES.CLW.

STD:PrintSetup	Диалог опций принтера.
STD:Close	Закрывает активное окно.
STD:Undo	Отменяет последнее действие редактирования
STD:Cut	Удаляет выделенный текст и копирует его в буфер.
STD:Copy	Копирует выделенное в буфер.
STD:Paste	Вставляет содержимое буфера в указанную точку.
STD:Clear	Удаляет выделенное.
STD:TileWindow	Располагает дочерние окна черепицей.
STD:TileHorizontal	Располагает дочерние окна черепицей.
STD:TileVertical	Располагает дочерние окна черепицей.
STD:CascadeWindow	Располагает дочерние окна так, что все строки заголовков видны.
STD:ArrangeIcons	Выравнивает пиктограммы дочерних окон.
STD:WindowList	Добавляет имена дочерних окон в меню.
STD:Help	Открывает файл .HLP на странице содержания.
STD:HelpIndex	Открывает файл .HLP на индексе.
STD:HelpOnHelp	Открывает файл Microsoft .HLP для системы помощи Windows.
STD:HelpSearch	Открывает утилиту Microsoft Help Search для файла .HLP.

Осуществление стандартного поведения окна с помощью STD ID.





## Положения меню и поведение слияния

Параметры размещения Positions позволяют установить компоновку меню приоритетный порядок позиций меню когда Clarion сливает меню. Имеющиеся варианты:

**Normal (нормальная позиция)** Для нормального упорядочивания меню при их слиянии выберите Normal из выпадающего списка Position. При нормальном слиянии глобальные позиции предшествуют локальным позициям. Смотрите выше Слияние меню.

**First (первая позиция)** Чтобы вытеснить выбранное меню или позицию меню на верхнюю позицию при слиянии меню, выберите First из выпадающего списка Position. Это добавляет атрибут FIRST к MENU или к оператору ITEM. Смотрите Справочник языка а также выше раздел Слияние меню.

**Last (последняя позиция)** Чтобы вытеснить меню или позицию меню на нижнюю позицию при слиянии меню, выберите Last. Это добавляет атрибут LAST к MENU или к оператору ITEM. Смотрите Справочник языка а также выше раздел Слияние меню.

Следующие два флажка Flags позволят вам установить, должны или нет быть ваше меню слито, а также правильное выравнивание позиций, показанных на строке меню:

**Do not Merge (не сливать)** Чтобы приказать Clarion никогда не сливать это поле меню с другими полями меню, отметьте поле Do Not Merge. Это доступно только для поля меню(MENUBAR). Дополнительная информация об атрибуте NOMERGE имеется в Справочнике языка.

**Right Justify**

**(выровнять справа)** Чтобы выровнять справа выбранное меню, отметьте поле Right Justify. Это возможно только для меню на действующем поле. Вложенные меню (подменю) не могут быть выровнены справа. Отметка в этом поле показывает выбранные меню, а также все меню после выбранного меню на правой стороне действующего поля.

## Добавление горячих клавиш

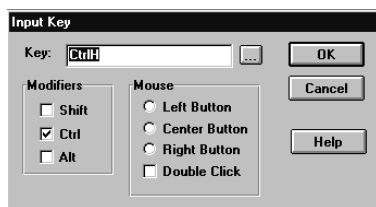
Горячая клавиша очень похожа на быструю клавишу (accelerator key). Горячая клавиша или клавишная комбинация позволяет конечному пользователю немедленно показать меню или исполнить действие, связанное с позицией меню, без щелканья клавишей мыши и без показа меню, содержащего данную позицию меню. Горячая клавиша, как правило, имеет вид CTRL+ символ или CTRL + SHIFT + символ. Чтобы добавить горячую клавишу,

1. Нажмите кнопку Hot Key (горячая клавиша) .

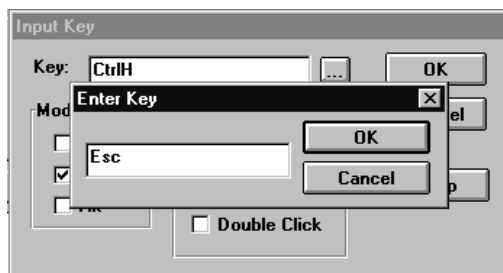
Появится диалоговое окно Input Key (клавиша ввода). С помощью этого диалогового окна добавьте атрибут KEY к вашему меню или позиции. Атрибут KEY устанавливает “горячую” клавишу или клавишную комбинацию.

2. Из диалогового окна Input Key установите горячую клавишу или клавишную комбинацию, нажимая для этого необходимую клавишу или клавишную комбинацию.

Нажатые вами клавиши появятся в поле Key (клавиша) и будут поставлены как параметры атрибута KEY для этой позиции меню.



Щелчки клавишами мыши могут быть использованы в качестве горячих клавиш; однако щелчки клавиши мыши не могут быть установлены с помощью щелчков. Для щелчков мыши отметьте соответствующее поле(я) флажков. Например, чтобы исполнить команду Открыть когда пользователь щелкнет дважды, отметьте поле Left Button (левая клавиша) и поле Double Click (двойной щелчок).



Клавиши ESC, ENTER и TAB могут быть использованы в качестве горячих клавиш, но они не могут быть установлены путем их нажатия. Для этих клавиш нажмите эллиптическую (...) кнопку и наберите “esc”, “enter”, или “tab”.

3. Для возврата к редактору меню нажмите кнопку ОК.

**Совет:** Вам может потребоваться добавить горячую клавишную комбинацию к тексту меню для того, чтобы сигнализировать его доступность для пользователя. См. приложение Вопросы конструирования окон, где имеется список обычных горячих клавиш, связанных со стандартными командами окном.

## **Другие поведения меню - блокирование и переключение**

---

Следующие два флажка (Flags) позволяют вам заблокировать позицию и установить позицию меню как переключатель.

### Disable Item

(сделать позицию недоступной) Чтобы сделать меню или позицию меню недоступной (закрасить серым и сделать ее недоступной для пользователя), отметьте поле Disable Item. Это добавит атрибут DISABLE к оператору MENU или ITEM.

**Совет:** Поле недоступности удобно когда вы добавляете в программу модальность - когда один тип дочернего окна не поддерживает команды, которые поддерживает другой тип. Для того окна, которое не поддерживает команду, не обязательно удалять эту команду из меню, можно просто сделать ее недоступной. Это позволит избежать замешательства пользователя при исчезновении команд меню, появляющихся вновь в зависимости от того, какое окно активировано.

### Toggle (on/off) Item

(переключение (вкл/выкл)

позиции

Чтобы создать переключатель вкл/выкл для выбранной позиции меню, отметьте поле Toggle (on/off) Item. Позиция должна иметь численную переменную в поле Use Variable (используемая переменная). Переменная должна быть объявлена с помощью одного из диалогов данных или во вставке. См. выше Создание меню вашего приложения. Редактор меню добавляет к этой позиции атрибут CHECK.

С атрибутом CHECK, когда пользователь выбирает позицию меню в первый раз, ITEM (позиция) “включена”, величина переменной использования равна единице (1) и знак отметки появляется около ITEM (позиции). Когда пользователь выбирает данную позицию меню второй раз, ITEM (позиция) “выключена”, переменная использования имеет величину нуль (0) и никакая отметка не показывается. Вам нужно добавить исходную программу (вставку) для управления поведением приложения в зависимости от состояния переменной использования.

## Управление вашими меню

Separator (разделитель) Чтобы добавить разделительную полосу между текущими выделенными меню или позициями, нажмите Separator button.

**Совет: разделительные полосы могут предоставить пользователю визуальный ключ того, что группа позиций меню на меню выполняет связанные функции.**



Всплывающее меню форматера окна использует разделитель для группирования связанных команд.

## **Панели инструментов**

Вы можете добавить панель инструментов к любому окну с помощью простой команды в форматере окна. Вы можете поместить на панель инструментов любой элемент управления, хотя, вероятнее всего, вы поместите на панели командные кнопки, поля флажков, радиокнопки и выпадающие поля списков. Как и в случае меню, Clarion будет в некоторых ситуациях автоматически сливать панели инструментов.

### **Слияние панелей инструментов**

---

#### **Глобальные и локальные панели инструментов**

Структура TOOLBAR (панель инструментов) объявляет инструменты, показываемые для приложения или окна. На приложении панель инструментов определяет глобальные инструменты для данной программы. Если атрибут NOMERGE (неслияние) установлен на панели инструментов приложения, инструменты локальны и показываются только тогда, когда не открыто ни одно дочернее окно; нет глобальных инструментов. Глобальные инструменты активны и доступны на всех MDI-окнах если структура TOOLBAR дочернего MDI-окна не имеет атрибут NOMERGE.

#### **MDI-окна**

На MDI-окнах панель инструментов определяет инструменты, которые автоматически слиты с глобальной панелью инструментов. Как глобальные инструменты, так и инструменты окна в этом случае активны когда MDI “дочернее” окно имеет фокус. Как только окно теряет фокус, его специфические инструменты удаляются из глобальной панели инструментов. Если на панели инструментов MDI окна определен атрибут NOMERGE, инструменты переписывают и замещают глобальную панель инструментов.

#### **Не-MDI окна**

На не- MDI-окне панель инструментов никогда не слита с глобальным меню. Панель инструментов на не-MDI окне всегда появляется в окне, ни на каком приложении, которое могло бы быть ранее открыто.

#### **Порядок слияния**

Когда панель инструментов MDI-окна объединяется с панелью инструментов приложения, глобальные инструменты появляются первыми, вслед за ними - локальные инструменты. Панели инструментов слиты так, что поля в панели инструментов окна начинаются сразу от положения, определенного величиной параметра ширины атрибута AT панели инструментов приложения. Высота показанной панели инструментов имеет максимальную высоту самого “высокого” из инструментов, глобального или локального. Если какая-либо часть элемента управления падает ниже дна, высота соответственно увеличивается.

**Внимание: Чтобы слить панели инструментов, ширина панели инструментов приложения АТ должна быть меньше чем ширина рамки приложения.**

## Кнопки

Ниже описывается, как добавить панель инструментов к окну. Стартовой точкой является Форматер окна, открытый на “свежем” окне:

1. Из меню Toolbar (панель инструментов) выберите New Toolbar (новая панель).

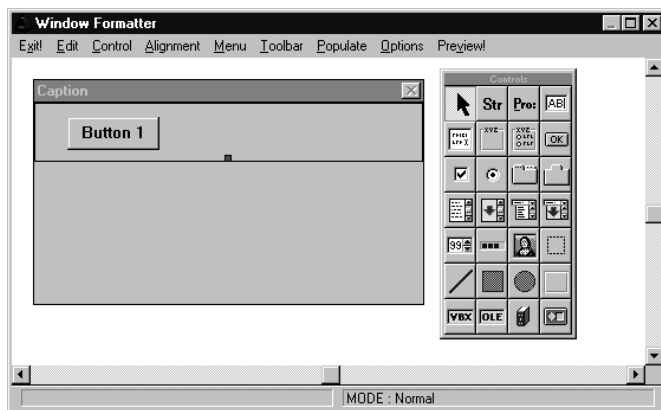
Наверху окна появляется прямоугольная область. Это панель инструментов. Во время выполнения программы она будет выглядеть темно-серой.

2. Выберите Options > Grid Settings, затем отметьте поле Snap to Grid (защелкнуть на сетке).

Это облегчает установление размеров и размещение элементов управления.

3. Нажмите пиктограмму Button (кнопка) (OK) на панели инструментов элементов управления, затем щелкните внутри новой панели управления в окне-образце.

Появляется кнопочный элемент управления.



4. Нажмите правую клавишу мыши и выберите Properties (свойства) из всплывающего меню или выберите Edit > Properties.

Появится диалоговое окно Button Properties (свойства кнопок) для новой кнопки.

5. Удалите текст, принятый по умолчанию в поле Parameter (параметр).

Это дает вам возможность создать кнопку с изображением без текста на ней.

6. Из закладки Extra выберите пиктограмму из выпадающего списка пиктограмм или выберите свой собственный файл пиктограмм (\*.ICO).

Список содержит большое число пиктограмм, принятых по умолчанию для таких стандартных действий, как File/Open (файл/открыть), или Cut (вырезать), Copy (копировать) и Paste (вставить из буфера).

7. На закладке General (общий) введите в поле Use метку соответствия.

Для кнопки File/Open, вы можете, например, набрать ?OpenButton. Метка соответствия появится в диалоговом окне Embedded Source (вставная исходная программа). Содержательное имя облегчит его идентификацию, когда вы захотите расширить функциональные возможности данной кнопки.

8. Нажмите кнопку ОК для закрытия диалога Button Properties.

9. Измените размер кнопки до нужного вам размера, потянув для этого ее рукоятки

Рукоятки - это маленькие цветные квадратики, которые появляются по углам и сторонам позиции.

**Совет: Для кнопок панели инструментов Clarion for Windows использует файлы .ICO, которые имеют формат 32x32 элемента изображения для кнопок вашей панели инструментов. Большинство кнопок вашей панели инструментов будут меньше - например, 16x18 элементов. Используя эти файлы, мы можем создавать пиктограммы для изображения “недоступного” состояния кнопки из того же самого файла и не требовать отдельного файла. При создании заказного .ICO файла для кнопки панели инструментов поместите изображение, которое вы хотите иметь для кнопки, в центр файла пиктограмм. Clarion автоматически обрезает изображение пиктограммы до размеров кнопки.**

## Кнопки с фиксацией

---

Кнопка с фиксацией “остается нажатой” будучи щелкнутой, затем возвращается к ее первоначальному состоянию будучи щелкнутой второй раз. Чтобы разместить кнопку с фиксацией:

1. Выберите пиктограмму Check Box (поле флажков) в панели инструментов элементов управления, затем щелкните мышью внутри новой панели инструментов в окне-образце.

Появится диалоговое окно Select Field (выбрать поле).

2. Выделите локальные данные, затем нажмите кнопку New.

Появится диалоговое окно New Field Properties (свойства нового поля).

3. В поле Field Name ( имя поля) введите имя поля, затем выберите BYTE из выпадающего списка типа данных.

4. Нажмите кнопку ОК чтобы вернуться в Форматер окна.

5. Щелкните правой клавишей мыши на данном поле флажков и выберите Properties из всплывающего меню.

Появится диалоговое окно Check Box Properties (свойства поля флажков).

6. Из закладки Extra выберите пиктограмму из выпадающего списка Icon (пиктограмма) или наберите имя своего собственного файла пиктограмм (\*.ICO).

Добавление пиктограммы к полю флажков “преобразует” поле флажков в кнопку. Кнопка, созданная из элемента управления поле флажков имеет два режима работы: включено или выключено. Когда поле флажков “включено”, кнопка выглядит “нажатой”, а значение ее переменной USE равно единице. Когда “выключена” - нулю.

7. Нажмите кнопку ОК.

Создание кнопки на этом завершается; вам нужно только отрегулировать ее размер и положение на панели инструментов, потянув для этого за рукоятки.

## **Взаимоисключающие кнопки**

---

Группа кнопок предоставляет пользователю возможность выбора из нескольких взаимоисключающих альтернатив. Например, в группе из трех кнопок только одна может быть “нажатой”, пусть это будет кнопка номер один. Если по ходу дела будет нажата кнопка номер два, кнопка номер один при этом выскочит (то есть, автоматически отождествится). Например, группа кнопок может обеспечить такие элементы управления, например, для выравнивания текста слева, справа и центрального выравнивания; только один из вариантов может быть активным в данный момент времени.

Чтобы создать группу кнопок:

1. Щелкните клавишей мыши на пиктограмме Option Box в панели инструментов элементов управления, затем щелкните внутри панели инструментов.

Форматер окна помещает Поле опций на панель управления. Вы можете изменить ее размеры, потягивая ее за рукоятки. Поле опций - структура OPTION- должно всегда окружать набор радиокнопок. Однако это поле опций не появится на панели инструментов, так как вы скроете его.

2. Щелкните правой клавишей мыши на поле опций и выберите Properties (свойства) из всплывающего меню.

Появится диалоговое окно Option Properties (свойства поля опций).

3. Наберите “JUSTIFICATION” в поле Use.



Поле Use (атрибут USE) принимает метку переменной. Вы должны объявить переменную с помощью диалогового окна Global Data (глобальные данные), диалоговое окно Local Data (локальные данные) или каким-либо методом. Эта переменная получит величину, указывающую на то, какую кнопку выбрал пользователь. Переменная может быть переменной в виде строки символов или цифровой переменной. Если это строка символов, она получит текстовую величину, либо текст из выбранной кнопки или переменная текстовая величина, которую вы установите. Если это цифровая, она получит целую величину, соответствующую выбранной кнопке, то есть кнопку 1, 2 или 3.

4. На закладке Extra очистите поле Boxed (в рамке).

Это прячет рамку поля опций от пользователя. Она появляется в диалоге форматера окна, но не появится при работе.

5. Нажмите кнопку ОК.

6. Щелкните мышью на пиктограмме радиокнопки в панели инструментов элементов управления, затем щелкните внутри поля опций.

Генератор приложений размещает радиокнопку там, где вы щелкнули в поле опций.

7. Щелкните правой клавишей мыши на радиокнопке и выберите Properties из всплывающего меню.

Появится диалог Radio Button Properties.

8. Очистите поле параметров Parameter.

Очищая это поле, вы передвинете текст из кнопки, так что мы можем добавить вместо этого пиктограмму.

9. В поле Value (величина) наберите “Left” (левый).

Когда пользователь нажимает эту кнопку, строка символов “Left” приписывается к переменной USE, которую мы установили выше.

10. Из закладки Extra выберите пиктограмму из выпадающего списка Icon (пиктограмма) или наберите имя своего собственного файла пиктограмм (\*.ICO).

11. Нажмите кнопку ОК.

Создание первой кнопки на этом завершается; вам нужно только отрегулировать ее позицию на панели инструментов, потягивая для этого ее центр.

12. Повторите шаги с 6 по 11 для “центральной” и “правой” кнопок.

## **Предварительный просмотр меню и панелей инструментов**

---

Для того чтобы проверить рабочий вид ваших панелей инструментов и меню:

1. Выберите Preview! (предварительный просмотр) из меню форматера окна.

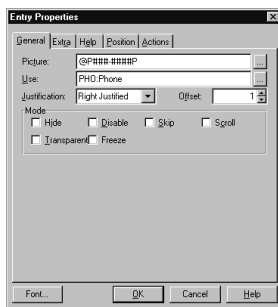
Это покажет окно, включая панель инструментов и меню, так, как его увидит пользователь. Путем нажатия кнопок проверьте действие кнопок с фиксацией и радиокнопок. Нажмите ESC после просмотра вашего окна.

2. Выберите Exit! (выход) из меню форматера окна для сохранения вашего окна.

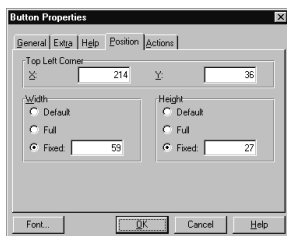
## Глава 10 Элементы управления и их свойства



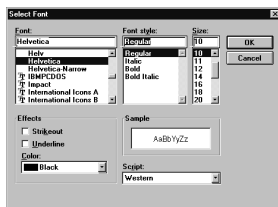
Когда вы размещаете элементы управления с помощью формatera окна, вы должны заполнить различные диалоговые окна свойств, чтобы придать элементам управления необходимые функциональные возможности.



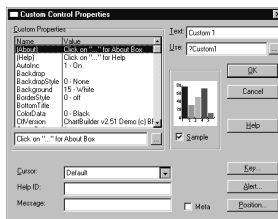
Диалоговое окно Entry Properties (Свойства поля ввода) управляет видом и функциональными возможностями обычных полей ввода, которые принимают данные, вводимые пользователем.



Для того, чтобы установить местоположение любого элемента управления, вы можете передвигать его с помощью мыши в форматере окна. Для точного управления используйте диалоговое окно Position (Позиция).



Установите шрифт для любого типа элемента управления.



Используйте в вашем приложении элементы управления OLE, .OCX и .VBX.

## Обзор

Общие атрибуты элементов управления

Интерактивные элементы управления

Свойства кнопки (Button)

Свойства радиокнопки (Radio Button)

Свойства поля флажков (Check Box)

Свойства поля списка (List Properties)

Свойства комбинированного поля списка (ComboBox)

Свойства spin-поля (Spin Box)

Свойства поля ввода (Entry Box)

Свойства текстового поля (Text)

Свойства листа (Sheet)

Свойства закладки (Tab)

Неинтерактивные элементы управления

Свойства строки символов (String)

Свойства элемента управления приглашение (PROMPT)

Свойства элемента управления Групповое поле (GROUP BOX)

Свойства строки индикатора выполнения (Progress Bar)

Свойства элемента управления изображение (Image)

Свойства элемента управления Область (Region)

Свойства элемента управления Линия (Line)

Свойства элемента управления Прямоугольная область (Box)

Свойства элемента управления Эллипс (Ellipse)

Свойства элемента управления Панель (Panel)

Элементы управления OLE

## Обзор

Свойства элемента управления OLE

Размещение таблицы/графика/документа/ в вашем приложении

Размещение таблицы/графика/документа/ в каждой записи

Элементы управления OLE с OCX

Использование OCX с глобальными функциями CallBack

Использование OCX с локальными функциями CallBack

Заказные элементы управления

Свойства элементов управления VBX

Функционирование VBX

## Обзор



Свойства элемента управления могут быть установлены для одиночного элемента управления с помощью Форматера окна. Однако лучше, если свойства элемента управления будут установлены для каждого элемента управления, ассоциированного с конкретным полем базы данных или переменной памяти путем нажатия кнопки Свойства (Properties) на закладках Window (окно) и Report (отчет) в диалоговом окне Field Properties (свойства поля). Когда свойства элемента управления устанавливаются таким способом, они оказываются применимыми каждый раз, когда вы помещаете данное поле на окно или отчет и каждый раз, когда вы синхронизируете ваше приложение и словарь данных.

В этой главе описывается, как устанавливать свойства элементов управления. Предполагается, что вы понимаете, как использовать Форматер окна для того, чтобы выбрать элемент управления, разместить его на экране или в отчете и установить его размеры (смотрите главу Форматер окна).

Данная глава дает обзор типов элементов управления в их отношении к вводу данных; сначала в ней обсуждаются свойства, применимые ко всем элементам управления, затем рассматривается каждый тип элементов управления отдельно. Она также показывает вам, как ассоциировать содержание переменной с управлением вводом данных или их показом.

Некоторые особые вопросы, рассматриваемые в этой главе, включают:

- ◆ Как нанести на кнопку текст, картинку или и то и другое.
- ◆ Как установить свойства радиокнопки и поля флажков, включая их настройку на трехмерное представление, подходящее для панелей инструментов.
- ◆ Как создать поле для ввода данных и как ассоциировать его с переменной, которая затем будет хранить эти, вводимые пользователем данные.
- ◆ Как установить такие свойства поля прокрутки, как величину приращения, когда пользователь нажимает кнопки увеличения или уменьшения.
- ◆ Как установить свойства элементов управления группа, лист и закладка, которые визуальнo организуют отдельные элементы управления в окне.
- ◆ Как форматировать элементы управления поля списка, включая процедуру создания многоколонных полей списков и иерархические списки.
- ◆ Как включить в окно графические элементы управления, такие как растровые и векторные рисунки, линии, прямоугольники и эллипсы.
- ◆ Как установить свойства для заказных OLE, .OCX и .VBX элементов управления.

## Типы элементов управления

---

Для целей нашего обсуждения мы разделим элементы управления на три категории, Interactive Controls (элементы управления интерактивного взаимодействия), Non- Inter-

active Controls (неинтерактивные элементы) и Custom Controls (покупные элементы управления).

### **Элементы управления интерактивного взаимодействия**

Эти элементы управления набираются пользователем или же пользователь щелкает по ним клавишей мыши.

- ◆ Элементы управления действиями - **BUTTON** (кнопка) - приводят к мгновенному результату. Сюда могут относиться закрывание диалогового окна и завершение операций, содержащихся в нем. Clarion поддерживает также непрерывное взаимодействие с кнопкой. Для пользователя это означает, что нажатие кнопки и удержание ее в нажатом состоянии эквивалентно повторному нажатию на нее.

- ◆ Элементы управления выбором пользователя - **CHECK**, **RADIO**, **COMBO**, **SPIN** и **LIST** - позволяют пользователю вводить данные посредством выбора из группы возможных альтернатив. Ввод с клавиатуры необязателен. Они создают потоковый пользовательский ввод, так как обычно оказывается более быстрым выбрать нужную позицию из списка, чем набирать имя позиции с клавиатуры, которое вы можете и не помнить. Используйте элементы управления выбора для того, чтобы вынудить пользователя выбрать что-то единственное из набора взаимоисключающих выборов - создайте “западающие” кнопки, которые остаются нажатыми до тех пор, пока их не нажмут повторно, или группы радиокнопок, где только одна из кнопок может быть выбрана за раз. Смотрите в этой книге главу Создание Меню и Панелей инструментов.

- ◆ Элементы управления вводом - **ENTRY** и **TEXT** - делают возможным ввод данных с клавиатуры. Clarion for Windows обеспечивает обширные возможности для автоматической проверки правильности вводимых пользователем данных.

- ◆ Гибридные элементы управления - **SHEETs** (листы), **TABs** (закладки) и **REGIONs** (области) - взаимодействуют с пользователем, но единственное действие, которое они выполняют - это направить пользователя к другим элементам управления.

### **Элементы управления не-интерактивного взаимодействия**

Эти элементы управления обеспечивают визуальные подсказки, которые помогают пользователю понять и обслуживать интерактивные элементы управления.

- ◆ Статические элементы управления - такие как **PANEL**, **PROMPT** и **GROUP BOX**, **ELLIPSE**, **LINE** и **IMAGE** - не выполняют действий, а вместо этого направляют пользователя к другим элементам управления или иным образом способствуют наведению визуальной красоты. Они могут принимать форму группового поля, закладки, линии или графического изображения, которые визуальным образом организуют и подчеркивают другие элементы управления. Элементы управления **GROUP** кроме того помогают вам разрабатывать и поддерживать ваше приложение, предоставляя вам возможность применения обычных атрибутов, таких как позиционирование, активация или упрятывание для нескольких элементов управления сразу.

### **Покупные элементы управления**

Покупные элементы определены вне интегральной Среды разработки Clarion и могут быть как интерактивными, так и не-интерактивными.

- ♦ Элементы управления OLE - они являются “контейнерами” для связанных или встроенных объектов из других программ, таких как Excel, Word или PowerPoint. Программы должны быть зарегистрированными OLE серверами. Эти объекты могут быть документами, созданными серверными программами, такими как электронные таблицы, документы текстовых процессоров или демонстрации слайдов. Или же эти объекты могут быть “директорами”, которые позволяют серверным программам появляться и работать в вашей программе Clarion.

- ♦ Элементы управления OCX - являются также “контейнерами” для элементов управления OCX. OCX - это дополнительные элементы управления, купленные у третьей стороны. Вы можете поместить эти элементы управления с помощью Формatera окна как только вы зарегистрировали OCX в Windows.

- ♦ Элементы управления .VBX - являются “добавляемыми” от сторонних продавцов элементами управления в формате Visual Basic Extension. Clarion поддерживает VBX элементы управления, если они совместимы с Microsoft Visual Basic 1.0. Вы можете поместить эти элементы управления с помощью Формatera окна после того как вы зарегистрируете библиотеки .VBX в среде разработки Clarion for Windows.

## Общие атрибуты элементов управления

Атрибуты, которые вы добавляете к элементу управления, определяют, как этот элемент будет выглядеть и действовать. Различные элементы управления поддерживают различные функции и поэтому требуют различных атрибутов. Для всех элементов управления Clarion позволяет устанавливать два общепринятых атрибута: USE и AT. Кроме того, большинство элементов управления допускают атрибуты TEXT, COLOR, KEY, ALRT, FONT, SKIP, HIDE, DISABLE, SCROLL, CURSOR, HLP, MSG и TIP. В этом разделе объясняется, как устанавливать эти наиболее общие свойства элементов управления. Подробно каждый атрибут обсуждается в Справочнике языка.

### Установка атрибута USE

---

Поле USE принимает либо метку соответствия или метку переменной. Если вы поместили шаблон элемента управления, вы можете просто принять метку по умолчанию, или же вы можете назначить по желанию свою собственную метку.

#### Метки соответствия

Используйте метку соответствия, когда вам не нужно присваивать значение из элемента управления для переменной. Метка соответствия является допустимой меткой Clarion, которой предшествует вопросительный знак (?). Эта метка сопоставляется с данным элементом управления в пределах вашей исходной программы (См. более полную информацию в Справочнике языка).

Например:

HIDE( ?MyVar )                      ! прячет элемент управления с USE атрибутом ?MyVar

#### Метки переменной

Используйте метку переменной метку когда вам действительно нужно приписать значение из данного элемента управления переменной.

**Совет: Некоторые элементы управления, такие как Приглашения (PROMPTS) и Линии (LINES), не имеют значений для присваивания и не могут иметь метку переменной в качестве своего USE атрибута.**

Переменная должна быть объявлена в вашей программе, модуле или процедуре. Нажмите эллиптическую (...) кнопку в диалоговом окне свойств данного элемента управления для



того, чтобы объявить переменную или чтобы отобразить переменную из уже объявленных ранее.

Метка переменной автоматически служит также меткой соответствия для данного элемента управления! Например:

```
MESSAGE( 'My Variable = '& MyVar)      ! показывает переменную MyVar  
HIDE( ?MyVar )! прячет элемент управления ввода ?MyVar
```

### Дубликатные метки соответствия

Два или более элементов управления ввода могут обновлять одну и ту же переменную. Однако они не могут иметь одну и ту же метку соответствия. В этих обстоятельствах Форматер окна автоматически создает уникальные метки соответствия путем добавления последовательного номера к меткам соответствия, которые в ином случае были бы продублированы. Уникальная метка соответствия устанавливается третьим параметром атрибута USE. Смотрите более полную информацию в Справочнике языка. Например, если вы поместите три элемента управления на одно окно, и они все обновляют одну и ту же переменную, Форматер окна генерирует код следующего типа:

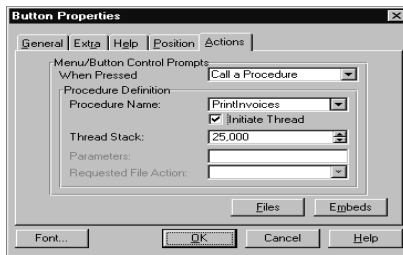
```
window WINDOW( 'Caption' ), AT (.,183,119), GRAY  
    SPIN(@s20), AT (66,27), USE(MyVar)  
    COMBO(@s20), AT (66,50), USE( MyVar „?MyVar:2)  
    ENTRY(@s20) , AT (67,7) ,USE( MyVar „?MyVar:3)  
END
```

### Чтобы установить атрибут USE

1. Щелкните клавишей мыши на элементе управления и выберите Properties (свойства) из всплывающего меню.

Появится закладка (tab) General (общие) диалогового окна свойств соответствующего элемента управления, которая даст вам возможность установить атрибут USE для вашего элемента управления.

Установка атрибута USE для вашего элемента управления, так что вы можете сослаться на элемент управления по имени в вашей исходной программе.



2. Если данный элемент управления является элементом управления ввода или имеет ассоциированную переменную (CHECK, COMBO, CUSTOM, ENTRY, LIST, OPTION, PROGRESS, SHEET, SPIN, STRING или TEXT), нажмите эллиптическую (...) кнопку поля USE для выбора или назначения поля словаря данных или переменной памяти из диалогового окна Select Field (выбор поля).

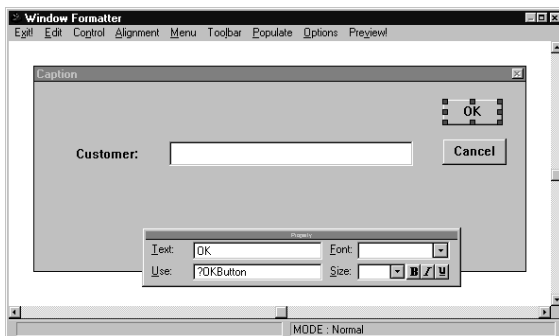
Если данный элемент управления не является элементом управления ввода (BOX, BUTTON, ELLIPSE, GROUP, IMAGE, LINE, PANEL, PROMPT, RADIO, REGION или TAB), наберите допустимую метку Clarion, предваряемую знаком вопроса (?). Обратите внимание, что для этих элементов управления в поле USE нет эллиптической (...) кнопки.

Помните, что вы всегда можете воспользоваться меткой по умолчанию, предлагаемой Генератором приложений.

3. Нажмите кнопку ОК.

**Совет: Метки соответствия и синтаксис свойств Clarion дают вам возможность модифицировать данный элемент управления во время работы. Например, вы можете использовать оператор DISABLE (блокировать) для затенения элементов управления в ситуациях, когда они должны быть недоступны для пользователя: DISABLE ( ?MyList)**

Ниже приводится альтернативный метод для установки атрибута USE. Этот метод работает лучше для не-интерактивных элементов управления, так как при этом не требуется отбора или определения переменной.



1. Из меню формatera окна выберите Options > Show Propertybox.

Появится панель инструментов Property, которая дает вам возможность установить для ваших элементов управления атрибут USE.

В качестве альтернативного метода используйте для установки атрибута USE панель инструментов Свойства (Property).

2. Щелкните клавишей мыши над элементом управления, который вы хотите изменить.

3. В поле USE панели инструментов Property наберите метку соответствия или метку переменной для использования вместе с элементом управления.

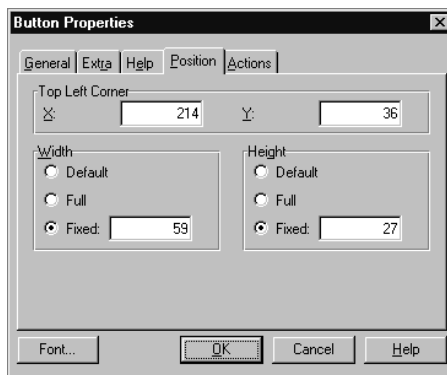
## Установка атрибута АТ

Атрибут АТ назначает положение и размер элемента управления. Форматер окна дает вам возможность визуально установить атрибут АТ каждого элемента управления простым перетаскиванием его туда, куда вам нужно. Вы можете также определить положение и размер вручную, набирая для этого координаты в диалоговом поле свойств или с помощью инструментов выравнивания. Чтобы установить атрибут АТ, который определяет положение и размер элемента управления:

1. Щелкните правой клавишей мыши на данном элементе управления и выберите Position во всплывающем меню.

Появится закладка Position (положение) диалога свойств элемента управления.

Поля ширина и высота могут изменять размеры элемента управления вплоть до того, что он займет все окно, или могут установить для элемента точный размер.



2. Установите координаты верхнего левого угла элемента управления.

Наберите координаты в 'X' (по горизонтали) и 'Y' (по вертикали). Они определяют положение верхнего левого угла элемента управления по отношению к верхнему левому углу окна. Единицы измерения для координат - это диалоговые единицы, которые обеспечивает относительную величину экрана, основанную на логических измерениях на экране. Смотрите Глоссарий, где дается определение диалоговых единиц. Они обеспечивают относительную меру, основанную на размере набора широко употребляемых символов.

3. Установите параметры Width (ширина) и Height (высота).

Выберите Default (по умолчанию) и Clarion будет автоматически выбирать размер элемента управления, основанный на шаблоне изображения элемента управления вводом. Или выберите Fixed и наберите с клавиатуры значения ширины и высоты для данного элемента управления.

**Совет: для элементов управления изображением (IMAGE) по умолчанию показывается картинка с размером, который был создан.**

Вы можете также установить, чтобы элемент целиком заполнял диалоговое поле или

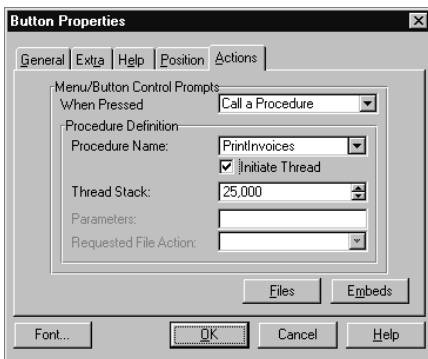
окно. Выберите для этого опцию Full (полный). Это добавит элементу управления атрибут FULL. Смотрите Справочник языка.

**Совет: Вы можете обеспечить ваших пользователей полнооконным текстовым редактором для полей MEMO. Создайте окно и поместите в него элемент управления TEXT. Если необходимо, поменяйте курсор на I - луч и установите регулировки Width и Height в положение Full.**

## Установка атрибута TEXT

Многие элементы управления, такие как BUTTONs, TABs, CHECKs, GROUPs и т.д., несут на себе надпись. Это наиболее прямолинейный и распространенный метод информирования пользователя о том, как и когда применять данный элемент управления.

Чтобы установить атрибут текст:



1. Щелкните правой клавишей мыши на данном элементе управления и выберите Properties (свойства) из всплывающего меню.

Появится закладка General (общая) соответствующего диалогового окна свойств элемента управления, которая даст вам возможность установить атрибут текст для вашего элемента управления.

2. В поле Text (текст) просто наберите текст, предназначенный для показа.

Знак (&) внутри текста означает, что следующий символ - это мнемонический символ для данного элемента управления. При показе символ появляется подчеркнутым и когда пользователь нажимает клавишу ALT+ соответствующую клавишу, инициируется действие данного элемента управления.

Например, наберите &Print для показа Print и выполнения по ALT+P инициации действия данного элемента управления.

**Совет: Избегайте использовать клавиатурные комбинации быстрого доступа для кнопок 'OK' и 'Cancel'.**

3. Нажмите кнопку OK.

**Совет: Метки соответствия позволяют вам использовать исполняемые операторы и синтаксис свойств Clarion для модификации текста данного элемента управления во время работы. Например, вы можете на лету изменить текст на: ?MyButton{PROP:Text}='Send'**

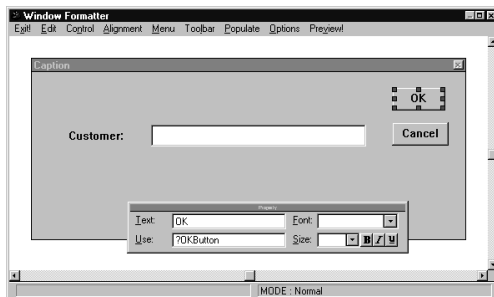
Ниже приводится другой метод установки атрибута текста.

1. Из меню Формatera окна выберите Options ➤ Show Propertybox.

В результате появится панель инструментов Property, которая даст вам возможность назначить атрибут текст для ваших элементов управления.

2. Щелкните мышью на элементе управления, который вы хотите изменить.

3. В поле Text (текст) панели инструментов Property наберите текст, предназначенный для показа.

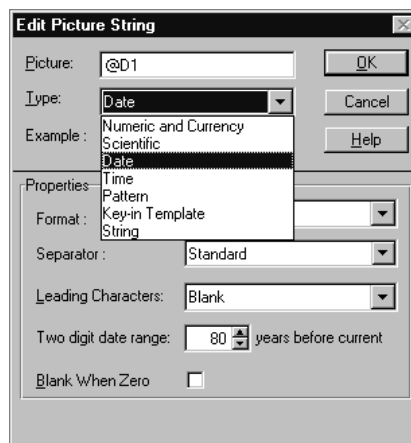


## Редактор шаблона изображения

Некоторые элементы управления (ENTRY, STRING, COMBO, и SPIN) дают вам возможность установить шаблона изображения, который обеспечивает специальный вид для показа и редактирования переменных. Более подробную информацию об этом можно найти в Справочнике языка.

Имеется богатый и разнообразный синтаксис шаблона изображения, который зависит от типа данных, которые вы форматируете: строки символов, числа, валюта, научные данные, даты, время и т.п.

Диалоговое окно Edit Picture (редактирование шаблона изображения) дает вам возможность быстро и легко построить шаблона изображения без необходимости запоминания синтаксиса шаблона. Вызывается это с помощью диалогового окна путем нажатия эллиптической (...) кнопки рядом с приглашением Picture (шаблон изображения) в диалоге свойств данного элемента управления.



- Example (пример)** Пример формата показа, установленный в данный момент в диалоговом окне. То, что вы видите - это то, что вы получите.
- Picture (шаблон)** Текущая установка шаблона изображения. Этот шаблона изображения дает показанный выше пример изображения.
- Picture Type (тип шаблона)** Выберите тип данных для форматирования из выпадающего списка. Вы берите из следующих: Строка символов, Число и Валюта, Научные изображения, Дата, Время, Образец и Шаблон ввода с клавиатуры.

### **Строка символов**

Шаблон изображения Строка символов назначает длину без какого-либо иного форматирования. См. Шаблоны изображения для строк в Справочнике языка.

- Length (длина)** Устанавливает длину строки символов. Эта длина также определяет ширину данного элемента управления, если ширина не установлена уже атрибутом AT данного элемента управления.

### **Число и валюта**

Шаблоны изображения числа и валюты устанавливают длину плюс дополнительное форматирование для положительных и отрицательных значений, различных видов валют и т.д. См. Шаблоны изображения числа и валюты в Справочнике языка.

- Size (размер)** Общее число значащих цифр плюс любые символы форматирования. Например, \$22.25 - это 4 значащих цифры + 3 форматирующих символа дают размер 7.
- Decimal Digits (десятичные цифры)** Число цифр справа от десятичной запятой.
- Currency (валюта)** Вы берите из Никакой, Головной и Хвостовой. Никакой означает отсутствие символа валюты. Головной ставит символ валюты слева от числа, а Хвостовой помещает символ валюты справа от числа.
- Symbol (символ)** Показываемый символ валюты: либо знак доллара (\$), либо постоянная строка символов.
- Negative Sign (знак минус)** Устанавливает, как форматируются отрицательные величины.
- Bracket (скобка)** Отрицательные величины взяты в круглые скобки.
- Leading (головной)** Отрицательные значения получают впереди знак минус.
- Trailing (хвостовой)** Отрицательные значения получают хвостовой знак минус.
- None (никакой)** Никакой знак не показывается.
- Decimal Separator (десятичный разделитель)** Устанавливает символ, вставляемый между целой и дробной частью величины.
- Period (точка)** Точка является разделителем.
- Comma (запятая)** Запятая является разделителем.
- None (никакой)** Не показывается никакой разделитель.
- Grouping (группирование)** Устанавливает разделитель, вставляемый после каждой третьей цифры для удобства чтения.
- Comma (запятая)** Запятая является разделителем.

Period (точка)	Точка является разделителем.
Space (пробел)	Пробел является разделителем.
Hyphen (знак переноса)	Знак переноса является разделителем.
Leading Character (ведущий символ)	Устанавливает символ, представляющий ведущие нули.
Clip (вырезать)	Удаляет ведущие нули, так что ведущие символы формата предшествуют цифрам.
Zero( нуль)	Головные нули показываются как нули (0).
Space (пробел)	Головные нули показываются как пробелы ( ).
Asterisk (звездочка)	Головные нули показываются как звездочки (*).
Blank When Zero (пусто, когда нуль)	Отметьте это поле, чтобы ничего не показывать, когда величина равна нулю.

### **Научная запись**

Шаблоны изображения научной записи чисел дают вам возможность показать очень большие или очень маленькие числа с десятичным форматом в виде степени десяти. Показ имеет вид -9.99e+999. См. Шаблоны научной записи величин в Справочнике языка.

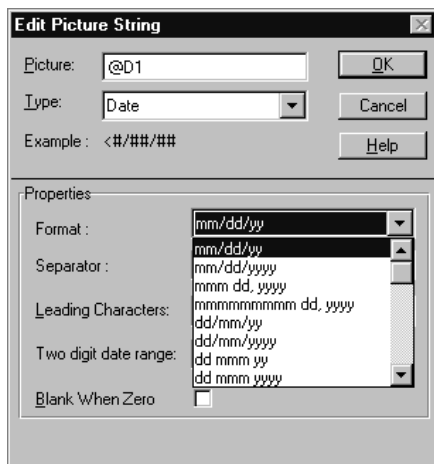
Number of Characters (число символов)	Общее число символов, включая 7 символов формата. Например, -1.96e+007 требует 10 символов.
Leading Digits (ведущие цифры)	Число цифр слева от десятичной точки (обычно 1).
Decimal Separator (десятичный разделитель)	Устанавливает символ, вставляемый после каждой третьей цифры для удобства чтения.
Point (точка)	Точка является разделителем.
Comma (запятая)	Запятая является разделителем.
Space (пробел)	Пробел является разделителем.
Separator (разделитель)	Устанавливает символ, вставляемый между целой и дробной частями величины.
Point (точка)	Точка является разделителем.
Comma (запятая)	Запятая является разделителем.
Blank When Zero (пусто, когда нуль)	Отметьте это поле, чтобы ничего не показывать, когда величина равна нулю.

### **Дата**

Шаблоны изображения даты дают вам возможность показывать даты в большом многообразии разных форматов. Выберите формат, который вам нужен, из выпадающего списка Format (формат). См. Шаблоны изображения даты в Справочнике языка.

Более того, шаблоны изображения даты в полях ввода автоматически вызывают рабочие функции синтаксического анализа Clarion, так что вы можете ввести '21', а Clarion расширит это до 21-го числа текущего месяца и года. Или вы можете ввести 'ДЕК' и Clarion расширит это до 1-го дня декабря текущего года. Дата затем форматируется в соответствии с шаблоном изображения.

**Совет: Атрибут MASK (поле флажков шаблон ввода ) на окне выгружает функции синтаксического анализа даты.**



- Format (формат)** Выберите формат, который вам нужен, из выпадающего списка. Вы получите то, что вы видите, за исключением форматов Windows короткое и Windows длинное. Кроме того, символ разделителя и головные нули могут быть установлены независимо от выбранного формата.
- Windows Short (короткое)** Использует короткий формат данных, установленный в панели управления Windows или панели управления региональных установок Windows 95.
- Windows Long (длинное)** Использует длинный формат данных, установленный в панели управления Windows или панели управления региональных установок Windows 95.
- Separator (разделитель)** Выберите из следующих: Стандартный (/), Точка (.), Дефис (-), Пробел ( ) и Запятая (,).
- Leading Characters (головные символы)** Устанавливает символы, которые должны представлять головные нули.
- Zero( нуль)** Головные нули показываются как нули (0).
- Blank (пусто)** Убирает головные нули.
- Asterisk (звездочка)** Головные нули показываются как звездочки (\*).
- Two digit date range (диапазон данных в две цифры)** Изменяет вековую интерпретацию по умолчанию для данных, введенных с двумя цифрами года. По умолчанию Clarion полагает, что любая дата, введенная с двумя цифрами года (т.е. век опущен) падает на период с сегодняшней даты -80 и сегодняшней +19 лет.

Например, если сегодня 1 июня 1996 года, а введена дата 9/2/59, Clarion



полагает, что 59 означает 1959 год, так как 1959 попадает в интервал между сегодняшний год-80 (1 июня 1916 года) и сегодняшний год+19 лет (1 июня 2015 года). Чтобы вынудить иную интерпретацию, поставьте Two digit date range на 30. Теперь Clarion будет предполагать, что 59 означает 2059, так как 2059 попадает между сегодняшний год-30 лет (1 июня 1966 года) и сегодняшний год + 69 лет (1 июня 2065 года).

Blank When Zero (пусто, когда нуль)      Отметьте это поле, чтобы ничего не показывать, когда величина равна нулю.

### **Время**

Шаблоны изображения показа времени дает вам возможность показывать время в большом многообразии разных форматов. Выберите формат, который вам нужен, из выпадающего списка Format (формат). См. Шаблоны изображения в Справочнике языка.

Format (формат)      Выберите формат, который вам нужен, из выпадающего списка. Вы получите то, что вы видите, за исключением форматов Windows короткое и Windows длинное. Кроме того, символ разделителя и головные нули могут быть установлены независимо от выбранного формата.

Windows Short (короткое)      Использует короткий формат данных, установленный в панели управления Windows.

Windows Long (длинное)      Использует длинный формат данных, установленный в панели управления Windows.

Separator (разделитель)      Выберите из следующих: Стандартный (:), Точка (.), Дефис (-), Пробел ( ) и Запятая (,).

Leading Characters (головные символы)      Устанавливает символы, которые должны представлять головные нули.

Zero( нуль)      Головные нули показываются как нули (0).

Blank (пусто)      Убирает головные нули.

Blank When Zero (пусто, когда нуль)      Отметьте это поле, чтобы ничего не показывать, когда величина равна нулю.

### **Образец**

Шаблон изображения “образец” дает вам возможность построить свои собственные форматы показа различных чисел: телефонных номеров, чисел социального обеспечения, номеров комнат, дат, времени, замеров и т.п. См. Шаблоны в Справочнике языка.

Picture (шаблон изображения)      Наберите шаблон изображения между двумя ‘Р’ в соответствии с легендой, о которой сказано ниже. Ваш шаблон изображения может включать любые показываемые символы, в том числе все стандартные символы клавиатуры.

Во время работы константы в шаблоне изображения показываются точно так, как они появляются в шаблоне. Знак меньше (<) и знак фунта (#) показывают отдельные цифры из переменной для показа.

Legend (легенда) < целое, пусто если нуль  
 # целое константа (любой показываемый символ, за исключением < и #)  
 Blank When Zero (пусто, когда нуль) Отметьте это поле, чтобы ничего не показывать, когда величина равна нулю.

**Совет: Чтобы использовать строчную p в вашем шаблоне, используйте P прописную в начале и конце вашего шаблона изображения. Чтобы использовать прописную P в вашем шаблоне, используйте в шаблоне изображения строчную p в начале и в конце.**

### Шаблон ввода с клавиатуры

Шаблоны изображения дает вам возможность построить обычные форматы редактирования для строк символов STRING, CSTRING и PSTRING, содержащих смешанные буквенно-числовые символы. Хотя шаблоны ввода с клавиатуры действуют на процедуру вывода так же, как и на процедуру ввода, их главной целью является обеспечение обычного редактирования поля и оценка правильности ввода. См. Шаблоны ввода с клавиатуры в Справочнике языка.

Picture (шаблон) Наберите шаблон между двумя 'К' в соответствии с легендой, о которой сказано ниже. Ваш шаблон изображения может включать любые символы (показываемые или нет), в том числе все стандартные символы клавиатуры.

Legend (легенда) < принять целое, пусто если нуль  
 # принять целое  
 ? принять любой символ (даже не-показываемый)  
 ^ принять прописной символ  
 \_ принять строчный символ  
 | ввод можно остановить здесь  
 константа (любой показываемый символ, за исключением < и #)  
 \ показывать следующий символ (дает вам возможность показать <#? ^ \_ | или \)  
 Only alphabetic characters (только алфавитные символы) Отметьте это поле для того, чтобы принять только алфавитные символы.  
 Blank When Zero (пусто, когда нуль) Отметьте это поле, чтобы ничего не показывать, когда величина равна нулю.

**Совет: Чтобы использовать строчную k в вашем шаблоне, используйте K прописную в начале и конце вашего шаблона изображения. Чтобы использовать прописную K в вашем шаблоне, используйте в шаблоне строчную k в начале и в конце.**

## Установка атрибута цвета

**Colors (цвета)** Введите допустимую величину цвета в любое из следующих поле для того, чтобы атрибут COLOR (цвет) к определению вашего элемента управления.

См. `..\\CW20\\LIBSRC\\EQUATES.CLW` для получения списка правильных установок цвета.

См. Приложение Вопросы конструирования окон, где обсуждается использование цвета для улучшения вида приложения.

**Text Color (цвет текста)** Чтобы применить определенный цвет к тексту элемента управления, наберите допустимый код цвета в этом поле, или нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна Text Color (цвет текста).

**Background (фон)** Чтобы применить определенный цвет ко всему элементу управления, за исключением выбранного текста, наберите допустимый код цвета в этом поле или нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна Background Color (цвет фона).

**Selected text (выбранный текст)** Чтобы применить определенный цвет к выбранному тексту элемента управления, наберите допустимый код цвета в этом поле или нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна Selected Color (выбранный цвет).

**Selected fill (выбранное заполнение)** Чтобы применить определенный цвет к фону выбранного текста элемента управления, наберите допустимый код цвета в этом поле или нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна Selected Background Color (выбранный цвет фона).

**Grid Color (LIST and COMBO only)** Цвет сетки (СПИСОК и КОМБИНИРОВАННЫЙ СПИСОК только)

Чтобы применить определенный цвет к линиям сетки СПИСКА, наберите допустимый код цвета в этом поле или нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна Grid Color (цвет сетки).

**Border Color (BOX, ELLIPSE and REGION only) (цвет кромки (только поле, эллипс и область))** Чтобы применить определенный цвет к кромке данного элемента управления, допустимый код цвета в этом поле или нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна Border Color (цвет кромки).

**Fill Color (BOX, ELLIPSE, REGION, and PANEL only) (цвет заполнения (только поле, эллипс, область и панель))** Чтобы применить определенный цвет к внутренней части данного элемента управления, наберите допустимый код цвета в этом поле или нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна Fill Color (цвет заполнения).

## Диалоговое окно Цвет

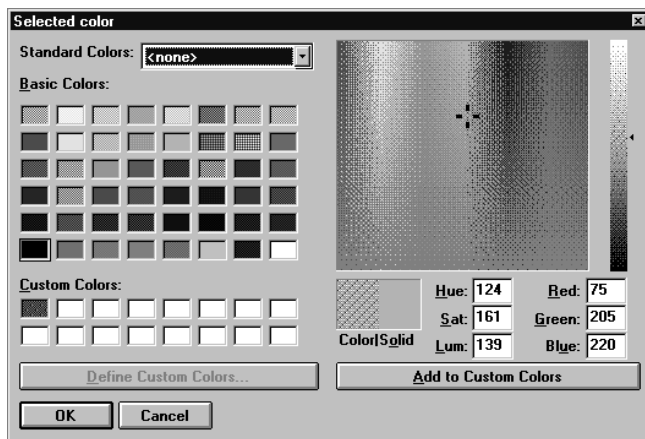
Диалоговое окно Color (цвет) вы можете вызвать из нескольких мест в пределах Среды Clarion for Windows, но в первую очередь вы вызываете его, когда устанавливаете цвет окна или элемента управления окна. Диалоговое окно Color - это продукт фирмы Microsoft, однако оно описано здесь для вашего удобства.

**Standard Colors (стандартные цвета)** Выберите из ниспадающего списка стандартных цветов, которые Clarion for Windows использует для создания своего стандартного интерфейса Windows приложения. Эти цвета соответствуют цветам, используемым по умолчанию процедурам Clarion, генерированным шаблонами.

**Basic Colors (базовые цвета)** Выберите из 48 заранее установленных образцов цвета. Щелкните на картинке того цвета, который вам нужен, а затем нажмите кнопку ОК.

**Custom Colors (пользовательские цвета)** Выберите из 16 пользовательских образцов, которые вы назначаете сами. Щелкните клавишей мыши на нужном вам цвете, а затем нажмите кнопку ОК.

**Define Custom Colors (назначить пользовательские цвета)** Чтобы назначить пользовательский цвет, щелкните на одном из 16 полей образцов пользовательского цвета, а затем нажмите кнопку Define Custom Colors (назначить пользовательский цвет).



**Color Continuum Pad (планшет цветового континуума)** Показывает континуум цветовых выборов. Щелкните на этом планшете клавишей мыши для того, чтобы грубо установить ваш цвет. Тонкая настройка вашего выбора цвета с помощью элементов управления описана ниже.

**Luminance Continuum Slider (ползунок континуума яркости цвета)** Показывает континуум яркости. Щелкните мышью и потяните внутри удлиненного прямоугольника для того, чтобы отрегулировать яркость цвета в интервале от самого темного (черный) до самого яркого (белый).

Color|Solid (цвет|фиксированный) Показывает образец только что назначенного (смешанного) цвета и его ближайшего эквивалента фиксированного цвета. Для преобразования только что назначенного цвета в его ближайший фиксированный эквивалент наберите комбинацию Alt+O. Преобразование автоматически настроит величины шести перечисленных ниже компонент до получения соответствующего эквивалента фиксированного цвета.

Hue (оттенок)	Целое число от 0 до 240, представляющее оттенок.
Sat (насыщенность)	Целое число от 0 до 240, представляющее насыщенность цвета..
Lum (яркость)	Целое число от 0 до 240, представляющее яркость.
Red (красный)	Целое число от 0 до 255, представляющее красный цвет
Green (зеленый)	Целое число от 0 до 255, представляющее зеленый цвет..
Blue (синий)	Целое число от 0 до 255, представляющее синий цвет..

Add to Custom Colors (добавить к пользовательским цветам) Когда вы удовлетворены выполненным назначением пользовательского цвета, нажмите эту кнопку.

## Установка атрибута KEY (клавиша)

Атрибут KEY применяется к любому элементу управления, который может оказаться в фокусе (комбинированное поле, поле ввода, групповое поле, списочное поле, поле опций, таблица, спин поле, закладки, текстовое поле, кнопка, поле флажков, покупные элементы управления и радиокнопка). Этот атрибут определяет горячую клавишу, дающую немедленное фокусирование на данном элементе. Для элемента управления типа “действие”, такого, как командная кнопка, горячая клавиша будет инициировать действие. Смотрите дополнительную информацию в Справочнике языка.

Чтобы установить атрибут KEY:

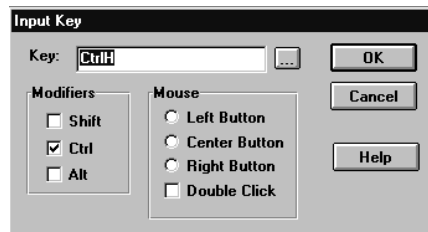
1. Щелкните правой клавишей мыши на данном элементе управления и выберите Key во всплывающем меню.

Появится диалоговое окно Input Key (ввод клавиши), которое позволяет установить для вашего элемента управления атрибут KEY.

Этот элемент получает фокус, когда пользователь нажимает CTRL+H

2. Нажмите нужную клавишу или комбинацию клавиш.

Нажатая вами клавиша или клавишная комбинация появится в поле Key и используется как параметр для атрибута KEY для этого элемента управления. Можно поступить иначе, нажмите символ или функциональную (F1, F2 и т.д.) клавишу и отметьте комбинацию полей Ctrl, Shift или Alt для установки горячей клавишной комбинации.



Щелчки мыши могут быть использованы как горячие клавиши; однако щелчки мыши не могут быть установлены щелчком клавиши мыши. Для щелчков мыши отметьте соответствующее поле (поля). Например, чтобы дать фокус элементу управления когда пользователь применяет ALT+ двойной щелчок, отметьте поле Alt, поле Left Button (левая клавиша) и поле Double Click (двойной щелчок).



Клавиши ESC, ENTER и TAB не могут быть установлены простым их нажатием, так как эти клавиши являются стандартными клавишами управления Windows. Для этих клавиш нажмите эллиптическую (...) кнопку и наберите “esc”, “enter” или “tab”.

3. Нажмите кнопку OK.

**Совет: Избегайте использовать в качестве клавиатурных комбинаций быстрого доступа комбинации клавиши ALT с буквами. Такие комбинации обычно резервируются для мнемонического доступа к меню.**

## Установка атрибута ALRT

Атрибут ALRT может быть применен к любому элементу управления, который может оказаться в фокусе. Этот атрибут активизирует клавишу Alert для элементов управления, находящихся в фокусе. Когда пользователь нажимает активизированную клавишу, генерируется событие EVENT:AlertKey. Это дает вам возможность выполнять какие либо действия, пока пользователь еще находится в поле ввода. Например, вы можете установить ALRT, чтобы показать пользователю дополнительную информацию при нажатии функциональной клавиши. Смотрите дополнительную информацию в Справочнике языка.

Для установки атрибута ALRT:

1. Щелкните правой клавишей мыши над данным элементом управления и выберите Alert во всплывающем меню.

Появится диалоговое окно Alert Keys (клавиши, порождающие события), которое дает вам возможность установить атрибут ALRT для вашего элемента управления. Для одного элемента управления вы можете установить столько Alert клавиш, сколько вам нужно.

2. Нажмите кнопку Add (добавить).

Появится диалоговое окно Insert Key (вставить клавишу), которое позволит вам установить атрибут ALRT для вашего элемента управления. Это тот же самый диалог,

который был использован для установки атрибута KEY. Смотрите выше Клавиша, чтобы получить информацию о пользовании этого диалога.

3. Нажмите кнопку ОК.

## Установка атрибута FONT

Вы можете установить вид текста, показываемого на элементе управления. Выберите начертание шрифта, размер, цвет, стиль и сценарий из стандартного выпадающего списка. Выберите также эффекты выделения и подчеркивания. Более полная информация имеется в Справочнике языка.

Чтобы установить атрибут FONT (шрифт):

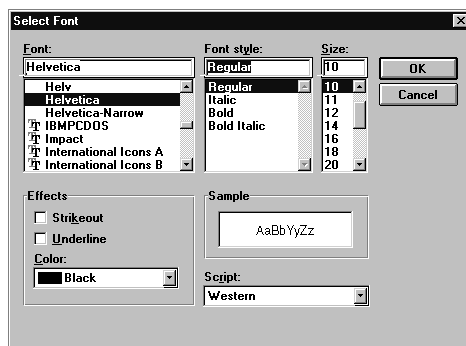
1. Щелкните правой клавишей на элементе управления и выберите Font из всплывающего меню. Появится диалоговое окно Select Font выбрать шрифт)

2. Выберите начертание шрифта, размер, цвет, стиль и сценарий из стандартного выпадающего списка

Появляется в диалоговом окне образец текста, сконструированный по вашему выбору.

3. Отметьте поля Strikeout (перечеркивание) или Underline (подчеркивание).

4. Нажмите кнопку ОК.



**Совет: Убедитесь, что шрифт, который вы отобрали, присутствует в системе пользователя. Если это не так, Windows постарается заменить его эквивалентным шрифтом; однако, так как вы не контролируете процесс замены, вы не можете быть уверенным, что результат будет выглядеть так, как вы хотели.**

Имеется альтернативный метод установки атрибута FONT:

1. Из меню форматера окна выберите Options > Show Propertybox.

Появляется панель инструментов Property (свойства), что дает вам возможность установить атрибут FONT для ваших элементов управления.



Установка шрифта элемента управления с помощью панели инструментов Property.

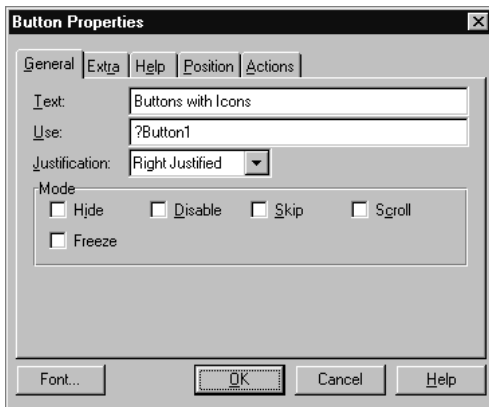
2. Щелкните клавишей мыши над нужным вам элементом управления.

3. В панели инструментов Property выберите начертание шрифта и его размер из стандартных выпадающих списков.

4. В панели инструментов Property выберите стиль шрифта с помощью стандартных кнопок **bold** (жирный), *italic* (курсив) и underline (подчеркивание).

## Установка режимов элементов управления

Закладка General (общие) различных диалогов свойств элементов управления позволяет вам установить атрибуты, которые управляют “режимом” (вид, исчезновение и доступность) элементов управления вашего окна. Чтобы установить режим элемента управления:



1. Щелкните правой клавишей мыши над элементом управления и выберите Properties во всплывающем меню.

Появляется диалоговое окно Properties для выбранного элемента управления.

2. Выберите закладку General.

Появится закладка General, которая содержит поля флажков Mode (режим).

Установка режимов элемента управления.

3. Отметьте любую комбинацию полей Mode.

Вариантами выбора и их эффектами являются:

**SKIP (пропускать)** Дает инструкцию формaterу окна исключить элемент управления из последовательности элементов, обходимых по клавише Tab (порядок, в котором элементы управления получают фокус когда пользователь нажимает клавишу TAB). Когда пользователь переходит от поля к полю по клавише Tab, Windows будет предоставлять фокус данному полю. Это полезно для редко используемых элементов управления, так как пользователь может еще получить доступ к элементу управления щелкая над ним клавишей мыши. Форматер окна добавит к описанию поля атрибут SKIP. (смотрите Справочник языка).

**Disable (вывести из действия)**

Выводит из действия или “затемняет” элемент



управления, когда ваша программа впервые показывает его, так что он недоступен для пользователя. Вы можете использовать оператор ENABLE, чтобы позволить пользователю получить доступ к управлению. Форматер окна добавит к описанию поля атрибут DISABLE (смотрите Справочник языка).

**Hide (спрятать)** Делает элемент управления невидимым, когда ваша программа впервые пытается его показать. Windows в действительности создает элемент управления - просто он не показывает его на экране. Форматер окна добавит к описанию поля атрибут HIDE. Для того, чтобы сделать элемент управления видимым на экране используйте атрибут UNHIDE (смотрите Справочник языка)..

**Scroll (прокручивать)** Устанавливает, должен ли элемент управления оставаться в окне, когда пользователь прокручивает окно. По умолчанию, (нет отметки в поле), элемент управления остается в окне. Отметьте поле Scroll чтобы создать элемент, который может быть “перемещен из окна”. Форматер окна добавит к описанию поля атрибут SCROLL (смотрите Справочник языка)..

**Transparent (прозрачный)** Устанавливает, нарисован ли данный элемент управления на окне, или там имеется только его текст или пиктограмма. По умолчанию (не отмечено) элемент управления нарисован. Отметьте поле Transparent (прозрачный) для создания невидимого, если не считать текст или пиктограмму, элемента управления. Форматер окна помещает атрибут TRN на данный элемент управления (см. Справочник языка).

**Freeze (замораживать)** “Замораживает” данный элемент управления, а с ним и все его дочерние элементы, так что последующие изменения словаря данных к нему оказываются неприменимыми. Вы можете переустановить атрибут #Freeze для всех элементов управления или только для отдельных элементов. См. Генератор приложений - Установка опций приложения.

4. Нажмите кнопку ОК.

## **Установка атрибутов помощи**

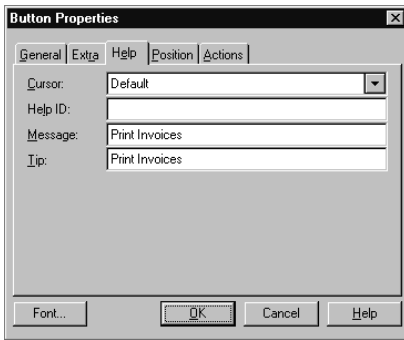
---

Закладка HELP различных диалогов свойств элементов управления дает вам возможность установить четыре атрибута, которые предоставляют пользователю информацию о работе данного элемента управления.

1. Щелкните правой клавишей мыши над элементом управления и выберите Properties во всплывающем меню.

Появляется диалоговое окно Properties для выбранного элемента управления.

2. Выберите закладку Help.



Появится закладка Help, которая содержит поля ввода курсора, помощи и сообщения.

Установка атрибутов курсора, помощи и сообщения.

3. Заполните по своему выбору любое из четырех полей.

Полями являются:

**Cursor (курсор)** Дает вам возможность установить альтернативную форму для курсора, когда пользователь проводит им по элементу управления. Выпадающий список Cursor обеспечивает стандартные варианты курсора, такие как I-Beam (луч) и Crosshair (перекрестие). Чтобы выбрать внешний файл курсора (чье расширение должно быть .CUR), выберите Select File..(выбор файла) из выпадающего списка, затем выберите файл с помощью стандартного файлового диалога. Форматер окна помещает атрибут CURSOR на элемент управления (смотрите Справочник языка).

**Совет: Луч, сигнализирующий ввод текста, является превосходным выбором активного курсора для элемента управления поля ввода или текста.**

**Help ID (идентификатор помощи)** Устанавливает атрибут HLP для элемента управления (смотрите Справочник языка). Когда на элементе управления фокус и пользователь нажимает F1, открывается файл помощи Windows на теме, к которой вас отнес атрибут HLP. В поле Help ID наберите либо ключевое слово помощи, либо контекстную строку символов помощи, находящуюся в файле .HLP.

Ключевое слово помощи - это слово или фраза, индексированные таким образом, чтобы пользователь мог искать ее в диалоге поиска помощи. Если более чем одно слово совпадает с ключевым словом, появляется диалоговое окно поиска.

Контекстная строка символов помощи - это произвольная строка символов, которая уникальным образом идентифицирует каждую страницу темы для компилятора помощи Windows. Контекстная строка помощи должна быть предварена тильдой (~).

**Совет: При разработке файла помощи Windows укажите ключевое слово с подстрочным 'К'. Контекстная строка помощи - это произвольная строка символов, которая уникально**

**идентифицирует каждую страницу темы для компилятора помощи Windows. При создании файла помощи подстрочный знак ‘#’ отмечает контекстную строку. Все эти проблемы решены для вас поставщиком инструментов помощи.**

- Message (сообщение) Устанавливает для элемента управления атрибут MSG (смотрите Справочник языка). Атрибут MSG устанавливает текст для показа в первой зоне строки статуса, когда данный элемент управления имеет фокус. В поле Message (сообщение) наберите текст для показа в строке статуса.
- Tip (совет) Устанавливает атрибут TIP для элемента управления (смотрите Справочник языка). TIP показывает текст в небольшом поле около курсора когда курсор неподвижен на данном элементе управления в течение установленного периода в полсекунды. Эта техника известна также как “Ballon Help” (помощь в кружочке). В поле Tip наберите текст для показа в поле совет.

## Интерактивные элементы управления

### Свойства кнопки (Button)

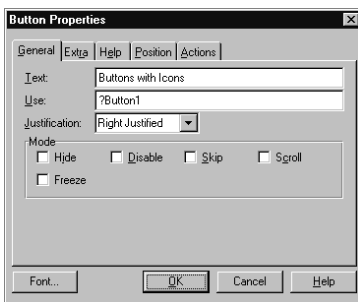
Элемент управления Кнопка - это элемент управления, который выполняет действие тогда, когда пользователь нажимает ее. В дополнение к обычным атрибутам элементов управления, описанным выше, форматор окна дает вам возможность установить следующие свойства кнопки:

- ◆ Надпись на кнопке (Text) .
- ◆ Пиктограмма на кнопке.
- ◆ Действие, выполняемое после нажатия.
- ◆ STD ID, устанавливающий стандартное действие окон для кнопки.
- ◆ Должна или не должна данная кнопка быть кнопкой по умолчанию.
- ◆ Drop ID, устанавливающий операции “тащить и бросать”, для которых кнопка является допустимой целью.

Принято, что кнопка - это прямоугольная область, содержащая текст, картинку или то и другое. Когда пользователь нажимает (щелкает на) кнопку, она исполняет команду, описанную текстом или картинкой.

Чтобы установить свойства кнопки, щелкните правой клавишей мыши кнопочный элемент управления и выберите из всплывающего меню Properties. Появится диалоговое окно Button Properties (свойства кнопки). Это диалоговое окно поможет вам установить атрибуты для оператора BUTTON.

#### Закладка “Общая” (General)



1. В поле Text (текст) наберите текст, который вы хотите видеть на кнопке.

Текст в поле Text - это строка символов. Символ & внутри текста означает, что следующий символ является мнемонической, быстрой клавишей для данной кнопки. Символ подчеркнут и когда пользователь нажимает ALT+соответствующая клавиша, инициируется действие кнопки. Текст кнопки может быть также установлен в поле Text (заголовок) панели инструментов Property.

**Совет: Microsoft рекомендует вам не помещать быструю клавишу на кнопках, на которых обозначено ‘OK’, ‘Cancel’.**

2. В поле Use наберите метку соответствия.

Эта метка соответствия является правильной меткой Clarion, предваряемой знаком вопроса (?).

Используйте метку соответствия для ссылки на кнопку в операторах программы. Смотрите выше Установка атрибута USE.

3. Ниспадающий список Justification (выравнивание) используется вместе с кнопочными пиктограммами.

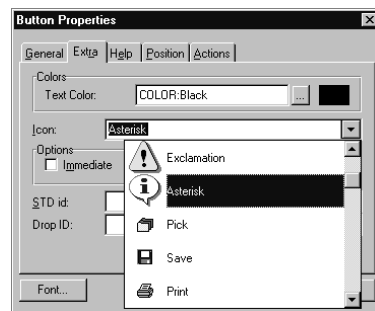
Default (по умолчанию) помещает пиктограмму над любым текстом. Right Justified (правое выравнивание) помещает пиктограмму справа от любого текста. Left Justified (левое выравнивание) помещает пиктограмму слева от любого текста.

4. Выбор режима (Mode): смотрите Общие свойства элементов управления - Установка режимов элементов управления.

### Закладка “Дополнительные возможности” (Extra)

5. В поле Icon (пиктограмма), если необходимо, выберите из выпадающего списка стандартную пиктограмму.

Это приведет к появлению в дополнение к тексту маленькой картинки на лицевой грани кнопки (с необходимой центровкой и обрезанием). Для выбора стандартной пиктограммы отберите одну из предлагаемых в выпадающем списке позиций. Для выбора файла пиктограмм (расширением файла должно быть .ICO) в выпадающем списке выберите строку Select File..., затем отберите файл с использованием стандартного файлового диалога.



**Совет: Пиктограмма и текст вместе дают кнопку. Текст появляется ниже картинки пиктограммы по умолчанию, однако приглашение Выравнивание обеспечивает альтернативное позиционирование.**

6. Отметьте соответствующие поля флажков Options.

Имеются три флажка кнопки, которые вы можете независимо друг от друга устанавливать либо убирать

**Immediate (немедленный)** (атрибут IMM) дает возможность создать кнопочное управление, которое повторяет исполняемые действия непрерывно, так долго, как долго пользователь держит кнопку нажатой. Нормально, кнопка генерирует событие только после того, как пользователь нажмет и отпустит кнопку мыши.

**Required (требуется)** (атрибут REQ) устанавливает, что когда кнопка нажата, ваша программа автоматически проверяет, чтобы ни одно поле ввода ENTRY с атрибутом REQ не было ни пустым, ни нулевым. Кнопка с этим атрибутом является кнопкой “требующей проверки полей”. Установите данный тип кнопки, если окно содержит поле ввода ENTRY или TEXT с атрибутом REQ (или вместо этого используйте функцию INCOMPLETE() для испытания элементов управления ENTRY). Когда пользователь нажимает кнопку с атрибутом REQ, а поле ENTRY пустое или нулевое, в фокус попадает первое требуемое поле ввода, которое пусто или нулевое.

**Default Button (кнопка по умолчанию)** (атрибут DEFAULT) “нажимает” кнопку, когда пользователь нажимает клавишу ввода ENTER. Вокруг кнопки появится резкая граница, сигнализирующая пользователю, что это кнопка по умолчанию. В общем случае атрибут DEFAULT нужно поместить на кнопку, если она представляет наиболее вероятное действие, из тех, которые пользователь захочет выполнить. В окне размещайте только одну кнопку по умолчанию.

7. Если необходимо, в поле STD ID вы можете выбрать стандартные действия окон из выпадающего списка.

Это один из способов сообщения вашей кнопке о необходимых действиях. Имеются некоторый набор стандартных действий, которые вы видите почти в каждой Windows - программе. Например, Cut (вырезать), Copy (копировать) и Paste (вставить из буфера). Clarion дает простой метод выполнения этих стандартных действий в вашем приложении - с помощью поля STD ID.

Clarion будет автоматически выполнять любые стандартные действия, представленные в выпадающем списке. Метки соответствия STD ID и их ассоциированные действия имеются в файле ..\CW20\CW\LIBSRC\EQUATES.CLW.

**Внимание: Не комбинируйте вызов процедуры или программы с каким-либо STD ID, так как элемент управления с STD ID не генерирует событие АССЕРТ когда пользователь активизирует элемент управления..**

8. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет

атрибут DROPID к вашей кнопке. Атрибут DROPID указывает, что эта кнопка является правильным объектом для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этой кнопки.

Drag and drop означает, что конечный прользователь может выбрать объект в одном окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “брошенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID цели и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление возможности drag and drop к списочному полю.

### **Закладка “Помощь” (Help)**

Смотрите Общие атрибуты элементов управления - Установка атрибутов помощи.

### **Закладка “Позиция” (Position Properties )**

Смотрите Общие атрибуты элементов управления - Установка атрибута AT.

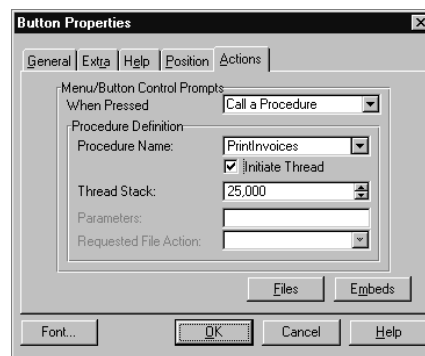
### **Закладка “Действия “ (Actions )**

Приглашения закладки Actions приходят из шаблонов, иными словами, приглашения, которые вы видите здесь, меняются вместе с шаблонами, используемыми для создания данного элемента управления. Ниже приводятся стандартные приглашения действий для всех кнопочных элементов управления. Дополнительную информацию об этих и других приглашениях вы найдете в главе Шаблоны элементов управления, программы и расширений.

Добавление функциональных возможностей к вашей кнопке.

9. Из ниспадающего списка When Pressed (когда нажата) выберите Call a Procedure, Run a Program или No Special Action.

Устанавливаемая вами процедура или программа выполняется, когда пользователь нажимает кнопку. Выборам являются:



**Call a Procedure (вызов процедуры)** Вы должны установить Procedure Name (имя процедуры) и будет ли процедура Initiates a Thread (инициировать процесс).  
**Procedure Name (имя процедуры)** Из выпадающего списка Procedure Name выберите существующее имя процедуры или наберите новое имя процедуры. Новая процедура появится как позиция “ToDo” в вашем дереве приложения.  
**Initiate a Thread (инициировать процесс)** По желанию отметьте поле Initiate a Thread. Если процедура инициирует процесс, установите размер Thread Stack (стек процесса). Clarion использует для инициации нового процесса функцию START. Если процедура инициирует процесс, вы не можете установить Parameters (параметры) или Requested File Action (желаемое файловое действие). Если процедура не запускает процесса, вы можете установить и Parameters и Requested File Action.

**Совет: Кнопка на панели инструментов рамки приложения, вызывающая процедуру дочернего MDI-окна, не должна инициировать процесс.**

**Thread Stack (стек процесса)** Оставьте без изменения установку принятую по умолчанию в спин поле Thread Stack, если у вас нет необычайных требований к программе. Чтобы изменить величину, наберите новую величину или щелкните клавишей мыши на стрелках спин поля.  
**Parameters (параметры)** В поле Parameters (параметры) наберите по желанию через запятую список переменных для передачи процедуре.  
**Requested File Action (запрашиваемое файловое действие)** Из выпадающего списка Requested file Action по желанию выберите None (никакое), Insert (вставить), Change (изменить), Delete (удалить) или Select (выбрать). Выбор по умолчанию None. Глобальная переменная GlobalRequest получает выбранную величину. Вызванная процедура может затем проверяет значение этой переменной и выполняет запрашиваемое файловое действие.  
**Run a Program (Выполнить программу)** Вы должны установить Program Name (имя программы) и любые параметры по желанию.  
**Program Name (имя программы)** В поле Program Name (имя программы) наберите имя программы. Программа должна находиться в .DLL или .LIB, определенные в файле проекта вашего приложения (.PRJ).  
**Parameters (параметры)** В поле Parameters (параметры) по желанию наберите список величин, которые должны быть переданы программе.  
**No Special Action (никакого специального действия)** Выберите это, если вы обеспечиваете функциональные возможности вашей кнопки другим методом, таким как вставная исходная программа или STD ID (смотрите выше Закладка Extra).

**Совет: Вы можете комбинировать вызов процедуры или программы с вставной исходной программой, но не с STD ID.**



10. По желанию нажмите кнопку Files (файлы) для получения доступа к схеме файлов для этой процедуры.

11. По желанию нажмите кнопку Embeds (вставки) для получения вставок исходного кода в точках, окружающих событие только для этой кнопки.

12. Нажмите кнопку ОК для возврата в форматер окна.

## **Свойства радиокнопки (Radio Button)**

---

Радиокнопка, называемая также кнопкой с зависимой фиксацией или опционной кнопкой, обеспечивает пользователю набор взаимоисключающих альтернатив. По умолчанию заполненный кружок представляет текущий выбор.

### **Взаимосвязь между RADIO и OPTION**

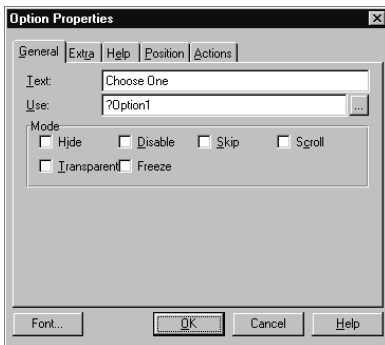
Поле опций - структура OPTION - должно всегда окружать все виды выбора радиокнопки. Форматер окна автоматически подсказывает вам о необходимости создания поля опций, если вы пытаетесь поместить радиокнопку вне опционного поля. Опционное поле появляется во время работы в виде прямоугольника с заголовком в верхней границе и с радиокнопкой внутри. Когда вы устанавливаете свойства радиокнопки, вы должны также установить свойства для данного поля опций.

Когда пользователь выбирает радиокнопку, переменная OPTION's USE получает величину, указывающую, которая из кнопок была выбрана: текст параметра выбранной кнопки, номер кнопки или другую величину, которую вы устанавливаете. Ваша программа может затем предпринять соответствующее действие, основанное на величине переменной OPTION's USE.

Чтобы поместить радиокнопку и связанное с ней поле опций, активируйте инструмент радиокнопка и щелкните клавишей мыши в образце окна. Форматер окна автоматически пригласит вас создать опционное поле. Щелкните клавишей на YES. Появятся опционное поле и одна радиокнопка.

Чтобы установить свойства поля, щелкните правой клавишей мыши на поле опций и выберите Properties (свойства) из всплывающего меню; появится диалоговое окно Option Properties (свойства поля опций).

## Закладка “Общая” (General)



1. В поле Text (текст) наберите метку поля опций.

Поле Text требует строковой константы, содержащей приглашение для группы элементов управления. Эта строка символов появляется во время работы на верхней границе поля опций. Знак (&) внутри текста означает, что следующий символ является быстрой клавишей для данного элемента управления. Символ подчеркнут и когда пользователь нажимает ALT+ соответствующая клавиша, радиокнопка получает фокус. Этот текст может быть также установлен в поле Text (текст) панели инструментов Property (свойство).

**Совет: Хотя структура OPTION должна присутствовать, она не обязан появляться видимой на экране. Вы можете скрыть ее от пользователя очисткой поля 'Boxed' на закладке 'Extra' этого диалогового окна.**

2. В поле USE наберите метку переменной.

Поле USE (атрибут USE) принимает метку переменной. Когда пользователь выбирает радиокнопку, переменная OPTION's USE получает величину, указывающая, которая кнопка была выбрана. Когда переменная USE является переменной типа строки данных, она получает либо текст выбранной кнопки, либо другую строковую переменную, которую вы устанавливаете (смотрите ниже Закладка общее - Радиокнопка). Когда переменная USE является числовой переменной, она получает номер кнопки.

3. Отметьте любую комбинацию полей Mode (режим).

Смотрите Общие свойства элементов управления - Установка режимов элементов управления.

## Закладка “Дополнительные возможности (Extra) - Поле опций”

4. По желанию, выберите цвет текста и фона.

Смотрите Общие свойства элементов управления - Установка атрибута ЦВЕТ.

5. По желанию, установите атрибут BEVEL (фаска) для поля опций.

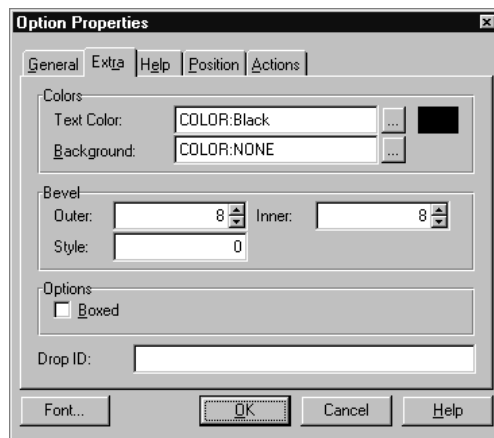
Атрибут Фаска придает трехмерный вид полю опций. Поле выглядит приподнятым, заглубленным или и таким и таким.

Outer (внешний) Положительная величина делает групповое поле кажущимся

приподнятым над плоскостью окна. Чем больше эта величина, тем более приподнятым будет казаться поле. Отрицательная величина делает групповое поле кажущимся заглубленным ниже уровня плоскости окна. Этот эффект фаски начинает проявляться с внешней кромки поля.

**Inner (внутренний)** Положительная величина делает групповое поле кажущимся приподнятым над плоскостью окна. Чем больше эта величина, тем более приподнятым будет казаться поле. Отрицательная величина делает групповое поле кажущимся заглубленным ниже уровня плоскости окна. Этот эффект фаски начинает проявляться с внутренней кромки поля.

**Style (стиль)** USHORT переменная, чьи шестнадцать бит определяют стиль (но не размер) каждого из четырех краев Опции (OPTION). Атрибут STYLE (стиль) предоставляет вам очень тонкий инструмент управления внешним видом фаски. Смотрите Справочник языка, где объясняется значение каждого бита.



6. По желанию очистите поле **Boxed** (в рамке) чтобы спрятать поле опций, но не радиокнопки, от пользователя во время его работы.

Это вызывает слегка отличный эффект чем атрибут **HIDE**. Атрибут **HIDE** прячет как поле опций, так и элементы управления внутри поля.

7. В поле **Drop ID** при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции "Drag and Drop"). Форматер окна добавляет атрибут **DROPID** к вашей кнопке. Атрибут **DROPID** указывает, что эта кнопка является правильной мишенью для операций "Drag and Drop" (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этой кнопки.

Drag and drop означает, что конечный прользователь может выбрать объект в одном окне или элемент управления и удерживая нажатой левую клавишу мыши, "перетащить" этот объект в другое окно, где освободив клавишу мыши, "бросить" его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать "сброшенный" на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел

атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление возможности drag and drop к списочному полю.

### **Закладка “Помощь” (Help) (Поле опций)**

Смотрите Общие атрибуты элементов управления - Установка атрибута ПОМОЩЬ.

### **Закладка “Позиция” (Position Properties )**

Смотрите Общие атрибуты элементов управления - Установка атрибута AT.

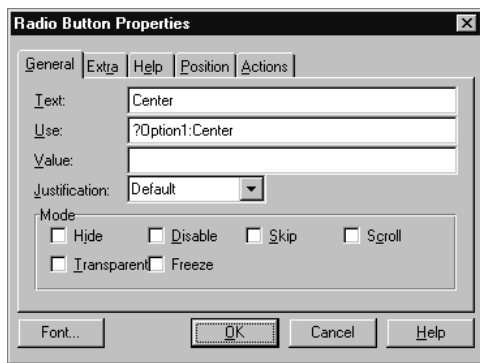
8. Нажмите кнопку ОК чтобы закончить установку атрибутов для поля OPTION.

Поле OPTION появляется в окне. Если вы очистили поле Boxed, рамка будет видна в режиме компоновки, но станет невидимой в режиме Preview! (предварительный просмотр).

Теперь, когда вы закончили определение свойств поля опций, вы должны установить свойства радиокнопок; щелкните правой клавишей мыши на радиокнопке и выберите Properties (свойства) из всплывающего меню; появится диалоговое окно Radio Button Properties (свойства радиокнопок).

### **Закладка “Общая” (General)**

Радиокнопка, чей текст говорит “Центр”



9 В поле Text (текст) наберите текст радиокнопки.

Поле Текст требует строковой константы, содержащей приглашение для радиокнопки. Знак (&) внутри текста означает, что следующий символ является быстрой клавишей для данного элемента управления. Символ подчеркнут и когда пользователь нажимает ALT+ соответствующая клавиша, радиокнопка получает фокус. Этот текст может быть также установлен в поле Text (текст) панели инструментов Property (свойство).

10. В поле USE наберите метку соответствия.

Метка соответствия является меткой Clagion, предваряемой знаком вопроса (?). Используйте метку соответствия для ссылки на радиокнопку в операторах программы. Смотрите Общие свойства элементов управления - Установка атрибута USE.

11. Наберите величину в поле Value (величина).

Когда пользователь выбирает радиокнопку, переменная OPTION's USE получает величину, которую вы устанавливаете здесь. Величина, которую вы вводите, должна совпадать с типом данных переменной OPTION's USE.

Если вы оставите поле Величина пустым, переменная OPTION's USE получит либо строку символов, обнаруженную в поле Text, либо номер кнопки, в зависимости от типа данных переменной OPTION's USE.

Номер кнопки соответствует положению кнопки в поле опций. Из форматера окна выберите Edit > Order Control dialog чтобы увидеть порядок кнопок по клавише Tab в поле опций.

12. Из выпадающего списка Justification (выравнивание) выберите Left Justification (левое выравнивание), Right Justification (правое выравнивание) или Default (по умолчанию).

Левое выравнивание располагает кнопку (или пиктограмму) налево от текста. Правое выравнивание располагает кнопку (или пиктограмму) направо от текста. Позиция по умолчанию располагает кнопку в соответствии с любой применимой установкой в словаре данных.

13. Отметьте любую комбинацию полей Mode (режим).

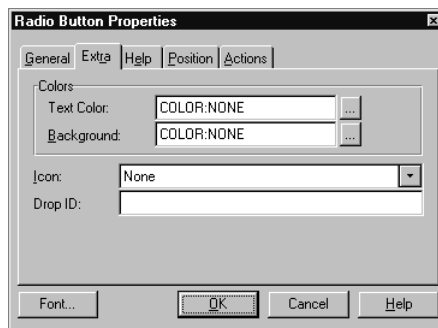
Смотрите Общие свойства элементов управления - Установка РЕЖИМОВ элементов управления.

**Закладка “Дополнительные возможности” (Extra) - (Радиокнопка)**

14. По желанию выберите цвет Background (фон) для радиокнопки.

Смотрите Общие свойства элементов управления - Установка атрибута ЦВЕТ.

15. Из выпадающего списка Icon (пиктограмма), отберите стандартную пиктограмму или выберите файл покупных пиктограмм.



Добавление пиктограммы к радиокнопке делает радиокнопку выглядящей подобно командной кнопке.

**Совет: Когда вам нужен набор кнопок для панели инструментов, из которых только одна может быть активна в данный момент, используйте радиокнопки с пиктографическим атрибутом. Смотрите подробности в главе Создание меню и панелей инструментов.**

Для выбора стандартной пиктограммы выберите одну из позиций в выпадающем списке. Для выбора файла пиктограмм (чьим расширением должно быть .ICO) выберите Select File из выпадающего списка и затем отберите файл, используя для этого стандартный файловый диалог.

**Совет: Если вы добавляете пиктограмму и текст, вы получите радиокнопку с тем и другим! Сделайте результирующую кнопку достаточно большой чтобы показать все.**

16. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет атрибут DROPID к вашей кнопке. Атрибут DROPID указывает, что эта кнопка является правильной мишенью для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этой кнопки.

Drag and drop означает, что конечный пользователь может выбрать объект в одном окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID цели, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

### **Закладка “Помощь” (Help) - Радиокнопка**

Смотрите Общие свойства элементов управления - Установка атрибута ПОМОЩЬ.

### Закладка “Позиция” (Position Tab)- (Радиокнопка)

Смотрите Общие свойства элементов управления - Установка атрибута AT.

17. По желанию добавьте дополнительные радиокнопки внутрь структуры OPTION.

**Совет: Чтобы создать профессионально выглядящие группы радиокнопок, включите управление сеткой Grid и используйте инструменты выравнивания. На линейке инструментов форматера окна появится поле флажков Grid. Это облегчит вам выравнивание радиокнопок.**

### Свойства поля флажков (Check Box)

Поле флажков управляет переменной, которую пользователь может включать или выключать. Активируйте инструмент создания поля флажков или выберите Check Box (поле флажков) из меню Control, затем щелкните в разрабатываемом окне. Форматер окна автоматически открывает диалоговое окно Select Field (выбор поля), так что вы можете выбрать или создать поле словаря данных или переменную памяти для ассоциации с полем флажков. После того как поле помещено, щелкните на нем правой клавишей мыши и выберите из всплывающего меню Properties (свойства); появится диалоговое окно Check Box Properties (свойства поля флажков).

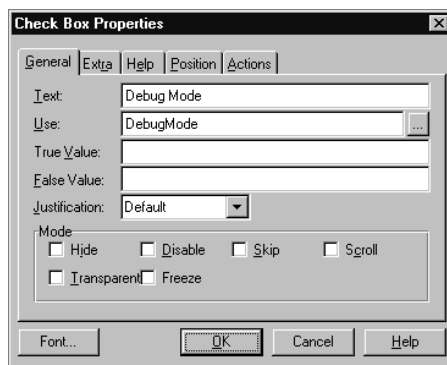
**Совет: Для получения наилучших результатов используйте переменную типа BYTE с вашими полями флажков. Кроме того, так как значение переменной должно быть ограничено в точности до двух величин, вы должны только обновлять переменную с помощью поля флажков или с помощью элемента управления, который ограничивает ввод двумя допустимыми величинами.**

### Закладка “Общая” (General)

Диалоговое окно Свойства поля флажков.

1. В поле Text (текст) наберите текст, который вы хотите видеть на поле флажков.

Текст в поле Text является строковой константой. Знак (&) внутри текста означает, что следующий символ является быстрой клавишей для данного поля флажков. Символ подчеркнут и когда пользователь



нажимает ALT+ соответствующая клавиша, поле флажков получает фокус, а поле переключается в состояние отмеченного, если оно было чисто, или в состояние очищенного, если оно было отмечено. Этот текст может быть также установлен в поле Text (текст) панели инструментов Property (свойства).

По умолчанию, текст появляется справа рядом с полем флажков. Для изменения этой установки по умолчанию используйте Justification (выравнивание).

## 2. Поле USE уже должно содержать метку переменной.

Если это не так, наберите метку переменной или нажмите эллиптическую кнопку, чтобы выбрать или создать поле словаря данных или переменную памяти с помощью диалогового окна Select Field (выбор поля).

**Совет: Присвойте переменной подходящую первоначальную величину, в особенности если вы используете True Value (истинная величина) и False Value (ложная величина). Смотрите Редактор словаря - Добавление или модифицирование полей.**

3. По желанию, в поле True Value (истинная величина), наберите величину для присваивания в том случае, когда поле отмечено. В поле False Value (ложная величина), наберите величину для присваивания в том случае, когда данное поле очищено. Эти величины также используются для определения начального состояния поля флажков (отмеченного или очищенного) при его первоначальном показе.

True Value (истинная величина) и False Value (ложная величина) дают вам возможность простого управления правомочностью данных с помощью поля флажков, это также дает вам возможность использовать символьные величины, такие как “T” и “F” или “Yes” и “No”, как вам подходит. Например, если ваше поле законности содержит “True” (истинно) и “False” или “Y” и “N”, а не 1 и 0, тогда True Value (истинная величина) и False Value (ложная величина) могут модифицировать установленное по умолчанию поведение поля флажков так, чтобы находиться в согласии с правомочностью данных. Если вы оставляете оба поля пустыми, вы получите величины и поведение, установленные по умолчанию, то есть 1 для отмеченного и 0 для очищенного.

**Совет: True Value (истинная величина) и False Value (ложная величина) чувствительны к регистру, так что “True” - это не то же самое, что “TRUE”, а “T” - не то же самое, что “t”.**

## Поля флажков - Более, чем вам когда-либо нужно было знать

Состояние поля флажков (отмечено оно или очищено) влияет на величину ассоциированной переменной и, наоборот, величина ассоциированной переменной влияет на состояние поля флажков, хотя влияния эти не совсем одинаковы.



### Поведение по умолчанию (без True Value (истинная величина) и False Value (ложная величина))

#### **Numeric Fields (цифровые поля)**

Assigning the Variable (присваивание переменной) Отметка в этом поле всегда присваивает величину 1 для ассоциированной переменной. Очистка этого поля всегда присваивает величину 0 для ассоциированной переменной.

Drawing the Check Box (рисование поля флажков) Любая ненулевая величина создает отметку в поле флажков, в то время как только нулевая величина создает очищенное поле.

#### **Character Fields (символьные поля)**

Assigning the Variable (присваивание переменной) Отметка в этом поле всегда присваивает величину 1 для ассоциированной переменной. Очистка этого поля всегда присваивает величину пусто для ассоциированной переменной.

Drawing the Check Box (рисование поля флажков) Любая ненулевая величина создает отметку в поле флажков, в то время как только величина пусто создает очищенное поле.

### Наведенное, не установленное по умолчанию (без True Value (истинная величина) и False Value (ложная величина))

#### **Numeric Fields (цифровые поля)**

Assigning the Variable (присваивание переменной) Отметка в этом поле присваивает величину, установленную в поле True Value (истинная величина), если величина цифровая. Если величина не цифровая, присваивания не происходит. Очистка этого поля присваивает величину False Value (ложная величина) если величина цифровая. Если величина не является цифровой, никакого присваивания не происходит.

Drawing the Check Box (рисование поля флажков) Любая величина кроме нуля или ложной величины создает отмеченное поле флажков, и наоборот, нулевая величина либо ложная величина создают очищенное поле.

#### **Character Fields (символьные поля)**

Assigning the Variable (присваивание переменной) Отметка в этом поле всегда присваивает величину, установленную в поле True Value (истинная величина). Очистка этого поля всегда присваивает величину, установленную в поле False Value (ложная величина).

Drawing the Check Box (рисование поля флажков) Любая величина, но не пусто или ложная величина, создает заполненное поле флажков, и наоборот, пустая величина или ложная величина создают очищенное поле.

4. Из выпадающего списка Justification (выравнивание) выберите Left Justified (левое выравнивание), Right Justified (правое выравнивание) или Default (по умолчанию).

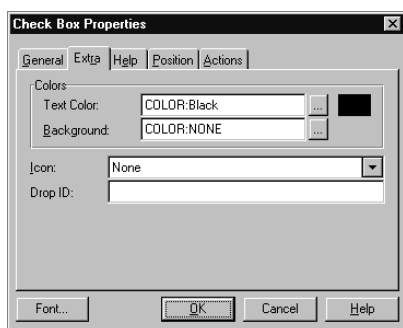
Left Justified (левое выравнивание)      Располагает поле флажков (или пиктограмму) налево от текста.

Right Justified (правое выравнивание)      Располагает поле флажков (или пиктограмму) направо от текста.

Default (по умолчанию)      Позиция по умолчанию располагает поле флажков в соответствии с любой применимой установкой в словаре данных.

5. Выбор Mode (режим): Смотрите Общие свойства элементов управления - Установка режимов элементов управления.

### Закладка “Дополнительные возможности” (Extra)



6. По желанию выберите цвет Background (фон) для радиокнопки.

Смотрите Общие свойства элементов управления - Установка атрибута ЦВЕТ.

7. В поле Icon (пиктограмма) по желанию отберите стандартную пиктограмму из выпадающего списка.

Добавление пиктограммы к полю флажков делает поле флажков выглядящим подобно командной кнопке. Пиктограмма появляется в виде маленькой картинки на грани кнопки в дополнение к любому тексту поля флажков.

Для выбора файла пользовательской пиктограммы (чьим расширением должно быть .ICO) выберите Select File... из выпадающего списка и затем отберите файл, используя для этого стандартный файловый диалог.

**Совет: Если вы добавляете пиктограмму и текст, вы получите поле флажков с тем и другим! Сделайте результирующую кнопку достаточно большой чтобы показать все.**

8. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”).

Форматер окна добавляет атрибут DROPID к вашему полю флажков. Атрибут DROPID указывает, что данное поле флажков является правильным объектом для операций “Drag

and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этой кнопки.

Drag and drop означает, что конечный прользователь может выбрать объект в одном окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

### **Закладка “Помощь” (Help)**

Смотрите Общие свойства элементов управления - Установка атрибутов помощи

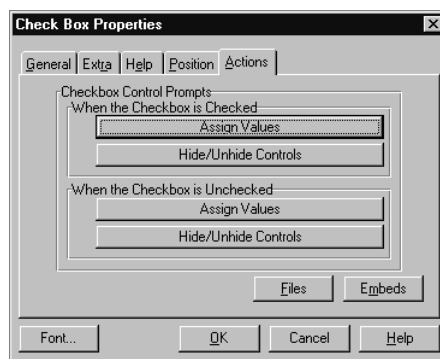
### **Закладка “Позиция” (Position Properties )**

Смотрите Общие свойства элементов управления - Установка атрибута AT.

### **Закладка “Действия “ (Actions )**

Приглашения закладки “Действия” поступают из шаблонов, иными словами, приглашения, которые вы видите здесь, меняются вместе с шаблонами, использованными для создания данного элемента управления. Ниже приводятся приглашения стандартных действий для всех элементов управления поля флажков. Смотрите более подробную информацию в главе Шаблоны элементов управления, программы и расширений.

Закладка Действия ведет к диалоговым окнам, позволяющим вам назвать переменные (иные чем переменная USE) и изменить их величины когда конечный пользователь отмечает или очищает поле флажков. Кроме того, вы можете спрятать или открыть другие элементы управления в окне.



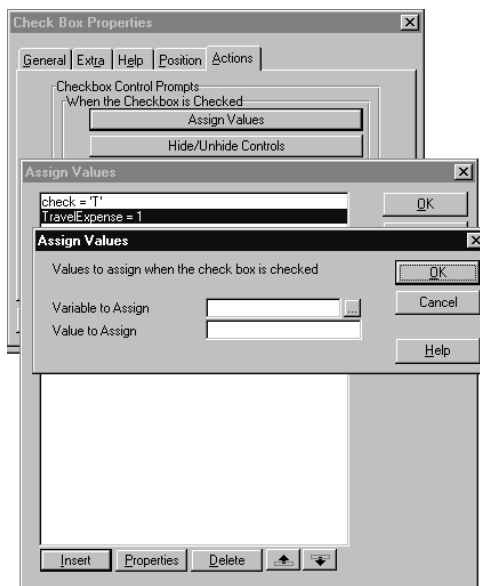
На закладке Действия появляются два групповых поля с двумя парами кнопок. Эти кнопки устанавливают поведение для When the Check Box is Checked (когда поле флажков

отмечено) и When the Check Box is Unchecked (когда поле флажков очищено).

9. Нажмите кнопку Assign Values (присвоить величину), после чего откроется диалоговое окно Assign Values.

Вы можете установить присвоения для многих величин. Нажмите кнопку Insert (вставить) чтобы добавить новое присвоение. В поле ввода Variable to Assign (переменная для присвоения) наберите имя переменной или нажмите эллиптическую (...) кнопку чтобы выбрать или создать поле словаря данных или переменную памяти с помощью диалогового окна Select Field (выбор поля).

В поле ввода Value to Assign (величина для присвоения) наберите величину, присваиваемую переменной. Нажмите кнопку ОК для окончания диалогов.



10. Нажмите кнопку Hide/Unhide Controls (спрятать/открыть элементы управления) и откройте диалоговое окно Hide/Unhide Controls.

Вы можете установить многие элементы управления для скрывания/открывания. Чтобы добавить новое действие Hide/Unhide (спрятать/открыть) к списку, нажмите кнопку Insert. В поле ввода Control to Hide/Unhide (элемент управления, который нужно спрятать/открыть) наберите метку соответствия элемента управления или нажмите эллиптическую (...) кнопку для выбора из списка меток соответствия.

В поле ввода Hide or Unhide control (спрятать или открыть элемент управления) выберите Hide (спрятать) или Unhide (открыть). Чтобы завершить диалоги, нажмите кнопку ОК.

11. Нажмите по желанию кнопку Files (файлы) чтобы получить доступ к диалоговому окну File Schematic Definition (Определение схемы файлов) для данной процедуры.
12. По желанию нажмите кнопку Embeds (вставки) для вставки исходной программы в точках, окружающих события только для этого поля флажков.
13. Нажмите кнопку ОК для возврата в форматер окна.

## Создание полей списков (Creating List Boxes)

Элемент управления список наиболее полезен для представления пользователю выбора из большого числа вариантов. Оно может показывать большое количество данных на небольшой площади, что определяет его использование, как многоцелевого элемента управления данными. С помощью Clarion for Windows вы можете создавать поля списков, которые выглядят, как таблицы, можете выполнять задачи перемещения элементов этих таблиц при помощи мыши (drag and drop) и другие.

Читая этот раздел, обратите, пожалуйста, внимание, что вся информация, которая применяется для элементов управления LIST (список), применима также к элементам управления COMBO.



Шаблоны элементов управления создают элементы управления и исходную программу для управления ими. Например, шаблон элемента управления Поле просмотра (BrowseBox) не только генерирует исходную программу для описания списочного поля, он также генерирует исходную программу для загрузки данных из файла в очередь, и для показа затем данных очереди в списочном поле с полной прокруткой и способностью отбора с помощью щелчка мыши. Смотрите главу Шаблоны элементов управления, программы и расширений, где имеется дополнительная информация о шаблонах, которые обслуживают списочные поля.

При создании поля списка из шаблона элемента управления или в виде голого элемента управления вы определяете его источник данных, его формат и его функциональные возможности. Среда разработки делит эти определения свойств между тремя диалогами:

- ♦ Диалоговое окно List Properties (Свойства списка) устанавливает файл или очередь, которые поставляют данные для поля списка, общую способность прокручивания и тот факт, будет ли это ниспадающий список или обыкновенный список. Другими словами, диалоговое окно Свойства списка устанавливает все свойства поля списка, которые не имеют вида столбцов. Этот диалог рассматривается в данной главе.
- ♦ Диалоговое окно List Box Formatter (форматер поля списка) позволяет вам добавить, удалить, изменить порядок или размер для специфических полей или столбцов, которые показаны в списочном поле. Этот инструмент обсуждается в следующей главе.
- ♦ Диалоговое окно List Field Properties (свойства полей списка) является частью

диалогового окна List Box Formatter (форматер поля списка) и определяет вид и поведение отдельных столбцов поля списка. Например, вы можете определить заголовки столбцов, ширины и индивидуальное прокручивание столбцов. Это диалоговое окно также определяет внешний вид и поведение групп столбцов внутри данного списочного поля. Этот диалог обсуждается в следующей главе.

## Свойства списка (List Properties)

1. Находясь в Форматере окна, выберите Control > List Box из меню (или выберите Populate > Control Template (шаблон элемента управления), после этого выберите поле просмотра BrowseBox), затем щелкните клавишей мыши в окне.

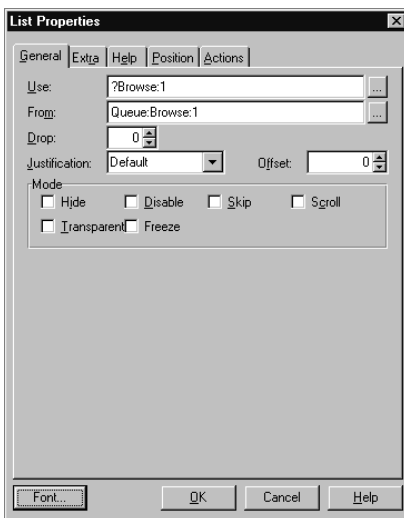
Появится диалоговое окно List Box Formatter (форматер поля списка). Это диалог управляет столбцами или полями в вашем списочном поле и обсуждается в следующей главе.

2. Нажмите кнопку ОК чтобы вернуться к Форматеру окна.

3. Щелкните правой клавишей мыши поле списка и выберите Properties (свойства) из всплывающего меню.

Появится диалоговое окно List Properties (свойства списка).

### Закладка “Общая” (General)



4. Поле USE принимает либо метку соответствия, либо метку переменной для того, чтобы получить величину, которую пользователь выбирает из списка.

Если вы поместили шаблон элемента управления, вы можете просто принять метку соответствия, назначенную по умолчанию.

Используйте метку соответствия тогда, когда не нужно приписывать переменной выбор, сделанный пользователем. Метка соответствия - это допустимая метка Clarion, которой предшествует знак вопроса (?). Эта метка ссылается на данный элемент управления в вашей исходной программе, например: `HIDE( ?MyList )`

Используйте метку переменной тогда, когда нужно приписывать переменной выбор, сделанный пользователем.

Эта переменная должна быть объявлена в вашей программе, модуле или в процедуре.

Нажмите эллиптическую (...) кнопку для того, чтобы объявить поле словаря данных или переменную памяти. Метка переменной служит также меткой соответствия для списочного поля! Например:

```
...  
MESSAGE( 'You Selected '& MyList)      !показывает переменную MyList  
      ! которая содержит выбранную позицию  
HIDE( ?MyList) !прячет элемент управления      ?MyList
```

5. В поле From (откуда) укажите источник данных.

Установите атрибут FROM для списка LIST. Смотрите подробности в Справочнике языка. В общем случае это метка структуры QUEUE, но это может быть также и поле в QUEUE или строковая постоянная.

Если вы используете шаблон управления или Мастера для построения своего списка, метка QUEUE вам поставляется, также как и код, необходимый для определения и загрузки QUEUE.

6. В поле Drop установите количество “выпадающих” строк для выпадающего списка.

Поместите ноль в поле Drop для обычного списочного поля. Чтобы создать выпадающий список, наберите количество “выпадающих” строк, которые вы хотите показать.

7. Из выпадающего списка Justification (выравнивание) выберите выравнивание Left Justified (левое выравнивание), Centered (центрирование), Right Justified (правое выравнивание), Decimal (десятичное) или Default(по умолчанию).

Добавляет атрибут LEFT, CENTER, RIGHT или DECIMAL к LIST. Смотрите подробности в Справочнике языка. Позиции Left Justified (левое выравнивание), Centered (центрирование) или Right Justified (правое выравнивание) помещают список данных заранее с левым, центральным или правым выравниванием в списочном поле. Позиция по умолчанию помещает данные в соответствии с установками в словаре данных. Выравнивание Десятичное выравнивает величины по их десятичным точкам. Каждое выравнивание может быть смещено на установленное вами расстояние.

Эти атрибуты сменяются атрибутом FORMAT, который добавляется шаблоном Browse Clarion. То есть, если вы используете формater поля списка для заселения вашего списочного поля, размещение данных будет определено генерированным Clarion строкой символов FORMAT, а не этими атрибутами выравнивания.

8. В поле Offset (смещение) установите смещение выравнивания в диалоговых единицах.

Смотрите глоссарий, где дано определение диалоговых единиц. Устанавливает величину

смещения для атрибутов LEFT, RIGHT, CENTER и DECIMAL. Смотрите выше. Для выравнивания ЦЕНТРИРОВАНИЕ отрицательная величина смещает влево от центра, положительная - вправо от центра.

Для выравнивания ДЕСЯТИЧНОЕ отрицательная величина смещает налево от десятичной точки, а положительная - вправо.

**Совет: Для десятичного выравнивания используйте смещение, равное 4 \* (десятичные места+ 1)**

9. Отметьте любую комбинацию полей Mode (режим).

Смотрите Общие свойства элементов управления - Установка режимов элементов управления..

### **Закладка “Дополнительные возможности” (Extra Tab)**

10. В поле Mark (маркер) наберите имя поля QUEUE ,если вы хотите позволить пользователю выбрать более одной позиции из списка.

Используйте поле Mark (маркер) с шаблоном элемента управления BrowseBox (поле просмотра), чей список LIST не прокручивается вертикально (так как BrowseBox загружено страницами, отмеченные позиции, которые прокрутились из данной страницы, теряют свою маркировку). Шаблон элемента управления поля просмотра BrowseBox определяет поле QUEUE отмеченным как BRW#::MARK, где # - порядковый номер данного шаблона элемента управления.

**Совет: Просмотрите генерированную исходную программу для того, чтобы найти имя поля QUEUE, генерированное данным шаблоном управления для хранения маркеров.**

Или же используйте поле Mark тогда, когда вы вручную кодируете ваш список LIST. Поле QUEUE может быть в той же самой очереди QUEUE , которая показывается в СПИСКЕ, или оно может быть в другой ОЧЕРЕДИ; однако от вас зависит обеспечить, чтобы две ОЧЕРЕДИ имели одно и то же количество позиций.

Флажками поля QUEUE отметьте выбранные позиции. Выбранные позиции получают '1', невыбранные позиции - '0'. Поле может быть полем цифровых или символьных данных.

11. Отметьте поле флажков VCR для обеспечения кнопок прокручивания VCR для вашего списочного поля.

Эти специальные кнопки прокручивания включают следующие: Top of List (Начало списка), Page Up (Страницу вверх), Entry Up (Одна позиция вверх), Locate (найти), Entry



Down (одна позиция вниз), Page Down (Страницу вниз) и Bottom of List (Конец списка).



Обратите внимание на то, что кнопка Locate не всегда значащая во всех реализациях СПИСКА. Например, для того, чтобы использовать Locate с шаблоном элемента управления поля просмотра BrowseBox, вы должны просмотреть файл в ключевом порядке и использовать ENTRY локатор, а не пошаговый или инкрементальный. Смотрите более полную информацию в Шаблонах элементов управления, программы и расширений.

12. В поле ввода направо от поля флажков VCR (поле локатора VCR) по желанию наберите метку соответствия элемента управления поля ввода, чья переменная USE является ключом доступа к просматриваемому файлу.

**Совет: Используйте это поле только с локатором голого LIST.**

Шаблон элемента управления BrowseBox не требует этого поля.

Элемент управления поля ввода ENTRY, идентифицируемый в поле локатора VCR, должен быть объявлен прежде ассоциированного с ним элемента управления LIST. Выберите Edit > Set Control Order для того, чтобы передвинуть ENTRY в положение перед LIST.

Во время работы пользователь набирает величину в поле ввода локатора, а затем нажимает кнопку Locate для прокручивания списка до позиции, которая является ближайшей величиной к полю ввода локатора

13. По желанию выберите цвета для данного списочного поля.

Смотрите Общие свойства элементов управления - Установка атрибута ЦВЕТ.

14. По желанию отметьте поле Immediate (быстрое) для размещения атрибута IMM на LIST.

Атрибут IMM генерирует событие каждый раз, когда пользователь нажимает клавишу (такую, как стрелка вниз или клавишу page up или даже алфавитную клавишу) в то время, когда списочное поле находится в фокусе. Эта характеристика позволяет показать связанную информацию по мере того, как поле прокручивается или выделяются новые строки поля списка.

15. По желанию отметьте поле Select Columns (выбор столбцов) для того, чтобы сделать возможным выбор отдельного столбца в много-столбцовом списочном поле.

Позволяет пользователю выделить много-столбцовый список поле за полем, а не по одной строке за раз. Это обеспечивает стиль движения полосы выделения как в сетке таблицы.

16. По желанию отметьте поле Hide Selection (спрятать выбор) для размещения атрибута NOBAR.

Атрибут NOBAR устанавливает, что текущий выбранный элемент в списке LIST выделяется только тогда, когда элемент управления LIST имеет фокус.

17. По желанию отметьте поля Horizontal и Vertical для добавления к вашему списочному полю линеек прокручивания.

Отметьте компоненты линеек прокручивания по желанию. Линейки прокручивания манипулируют всем списком. Вы можете добавить горизонтальные линейки прокручивания для отдельных столбцов с помощью Формatera поля списка, который описывается в следующей главе. Вертикальная линейка прокручивания появляется только тогда, когда число позиций превышает имеющееся количество мест в списочном поле.

**Совет: Шаблон элемента управления BrowseBox требует атрибута IMM для того, чтобы заставить работать вертикальное прокручивание.**

18. В поле Drag ID если необходимо наберите до шестнадцати (16) разделенных запятой сигнатур. (Идентификатор мишени операции “Drag and Drop”).

Форматер окна добавляет атрибут DRAGID к вашему полю списка. Атрибут DRAGID указывает, что данный список является правильным объектом для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для вашего списка.

Drag and drop означает, что конечный пользователь может выбрать объект в одном окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID цели, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

19. В поле Drop ID если необходимо наберите до шестнадцати (16) разделенных запятой сигнатур.

Форматер окна добавляет атрибут DROPID к вашему полю списка. Атрибут DROPID указывает, что данный список является правильным объектом для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для вашего списка.

Смотрите подробности и примеры в Справочнике языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

### **Закладка “Помощь” (Help)**

Смотрите Общие свойства элементов управления - Установка атрибута ПОМОЩЬ.

### **Закладка “Позиция” (Position Properties)**

Смотрите Общие свойства элементов управления - Установка атрибута АТ.

### **Закладка “Действия” (Actions)**

Закладка “Действия” будет пустой, если вы не используете шаблон элемента управления для построения своего списочного поля. Смотрите главу Шаблоны элементов управления, программы и расширений относительно инструкций о приглашениях Действий для каждого шаблона элемента управления.

## **Свойства комбинированного поля списка (ComboBox)**

Поле Combo является комбинацией поля списка и поля ввода. Оно полезно тогда, когда вы ожидаете данные в виде строк символов, которые обычно должны быть членами списка, но которые могут и не быть ими. Форматер окна дает вам возможность создать либо нормальное поле Combo, либо поле Combo выпадающего типа.

Свойства комбинированного поля устанавливаются точно так же, как и свойства списочного поля, за исключением следующих четырех дополнительных свойств.

### **Закладка “Общая” (General)**

1. В поле Picture установите шаблон изображения для данного элемента управления.

Нажатие эллиптической кнопки даст вам возможность выбрать шаблон изображения из диалогового окна Edit Picture String (редактировать шаблон изображения). Смотрите

Общие свойства элементов управления - Редактор шалбонов.

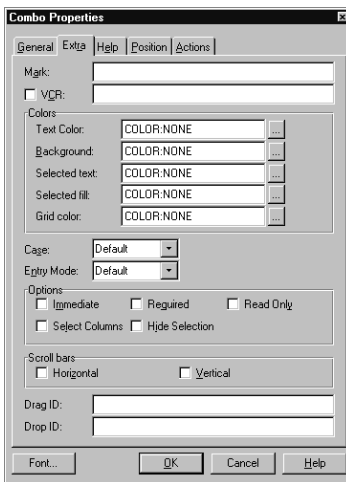
Знак шаблона изображения вынуждает вводимые данные иметь специфический формат. Например, знак шаблона @P##/##/##P требует типичного формата даты.

Вы можете проверить ввод, выполненный пользователем, относительно шаблона в двух отношениях: как пользователь набивает данные, или же тогда, когда пользователь перемещает фокус на другой элемент управления (например, переходя с помощью клавиши TAB в другое поле).

Проверка данных по мере их ввода пользователем несколько замедляет работу. Отметьте поле Entry Patterns (образец ввода) в диалоговом окне Window Properties (свойства окна). Это добавит атрибут MASK к WINDOW. Когда пользователь набирает данные, программа пытается преобразовать их до совпадения с образцом. Если программа не может преобразовать данные до соответствия образцу, она подает звуковой сигнал и возвращает фокус к данному элементу управления для того, чтобы пользователь смог сделать новую попытку.

Если атрибут MASK отсутствует, проверка ввода происходит при переносе пользователем фокуса на другой элемент управления.

### Закладка “Дополнительные возможности” (Extra)



#### 2. Установите атрибут регистра для поля ввода.

Поле ввода может автоматически преобразовать символ из одного регистра в другой во время набора пользователем. Uppercase (верхний регистр) (атрибут UPR) автоматически преобразует все в символы верхнего регистра. Capitals (заглавные) (атрибут CAP) эквивалентен “Собственному имени” (первая буква каждого слова появляется в виде заглавной). Default (по умолчанию) принимает ввод в том регистре, в котором набирает пользователь.

#### 3. В выпадающем списке Entry Mode (режим ввода) выберите Default, Insert или Overwrite.

Устанавливает режим ввода для поля ввода комбинированного поля. Insert (вставить) заставляет каждый удар клавиши вставлять новый символ и сдвигать уже имеющиеся символы вправо. Overwrite (переписать) заставляет каждый удар клавиши набивать новый символ поверх существующего символа. Default заставляет удар клавиши вести себя в соответствии с текущими установками системы.

Режим ввода Entry Mode относится только к окнам с атрибутом MASK. Смотрите более полное описание в Справочнике языка.

4. По желанию отметьте поле Read Only (только для чтения) для предотвращения ввода данных в этот элемент управления.

Добавляет атрибут READONLY к комбинированному полю (смотрите Справочник языка).

## Свойства spin-поля

Spin-поля - это специализированные поля ввода, которые принимают величины только из заранее определенных диапазонов. Spin-поля снабжены также кнопками “увеличить” и “уменьшить”, которые многим нравятся, так как можно использовать мышь для изменения значения поля. Пользователь может также ввести данные в Spin-поле с клавиатуры как в обычное поле ввода.

1. Из форматера окна выберите инструмент спиновое поле или выберите Spin Box (спиновое поле) из меню Control (элемент управления), затем щелкните в окне клавишей мыши.

2. Щелкните правой клавишей спинового поля и выберите Properties (свойства) из всплывающего меню.

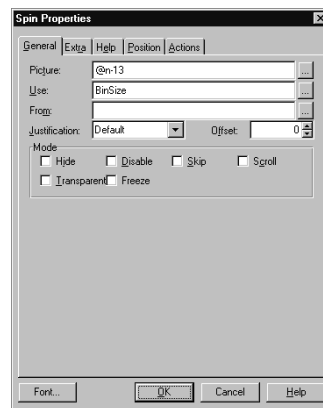
Появляется диалоговое окно Spin Properties(Свойства спин поля).

### Закладка “Общая” (General)

3. В поле Picture (Шаблон изображения) назначьте шаблон изображения для данного элемента управления.

Нажатие эллиптической (...) кнопки дает вам возможность отобразить шаблон изображения из диалогового окна Edit Picture String (редактирование строки символов шаблона). Смотрите Общие свойства элементов управления - Редактор шаблонов изображения.

Вы можете проверить ввод, выполненный пользователем, относительно шаблона в двух отношениях: как пользователь набирает данные, или же тогда, когда пользователь перемещает фокус на другой элемент управления (например, переходя с помощью клавиши TAB в другое поле).



Проверка данных по мере их ввода пользователем несколько замедляет работу. Отметьте поле Entry Patterns (образец ввода) в диалоговом окне Window Properties (свойства окна).

Это добавит атрибут MASK к WINDOW. Когда пользователь набирает данные, программа пытается преобразовать их до совпадения с образцом. Если программа не может преобразовать данные до соответствия образцу, она подает звуковой сигнал и возвращает фокус к данному элементу управления для того, чтобы пользователь смог сделать новую попытку.

Если атрибут MASK отсутствует, проверка ввода происходит при переносе пользователем фокуса на другой элемент управления.

#### 4. Наберите переменную метку в поле Use.

Переменная получит величину, выбранную пользователем из списка. Та же самая метка, предваряемая знаком вопроса (?), является меткой соответствия поля, которая ссылается на спин- поле в операторах исходной программы..

#### 5. Представьте данные источника для спин поля в поле From (откуда).

Устанавливает атрибут FROM для SPIN. Смотрите подробности в Справочнике языка. Это метка структуры QUEUE, поля внутри QUEUE или строковой константы.

Атрибут From полезен для величин, которые изменяются с непостоянным приращением. Вы можете также предоставить пользователю возможность при помощи spin- поля выбирать нечисловые данные, когда варианты выбора представляют собой натуральную прогрессию, такую, как, дни недели или месяцы года, например.

Поле From и поля пределов диапазона Range взаимоисключают друг друга.

#### 6. Из выпадающего списка Justification (выравнивание) выберите выравнивание Left Justified (левое выравнивание), Centered (центрирование), Right Justified (правое выравнивание), Decimal (десятичное) или Default (по умолчанию).

Добавляет атрибут LEFT, CENTER, RIGHT или DECIMAL к SPIN. Смотрите подробности в Справочнике языка. Left Justified (левое выравнивание), Centered (центрирование), Right Justified (правое выравнивание) помещают данные заранее с левым, центральным или правым выравниванием в спиновом поле. Позиция по умолчанию помещает данные в соответствии с любыми применяемыми установками в словаре данных. Десятичное выравнивание выравнивает величины по их десятичным точкам. Каждое выравнивание может быть смещено на установленное вами расстояние.

#### 7. В поле Offset (смещение) установите смещение выравнивания в диалоговых единицах.

Смотрите глоссарий, где дано определение диалоговых единиц. Устанавливает величину смещения для атрибутов LEFT, RIGHT, CENTER и DECIMAL. Для выравнивания

Центрирование отрицательная величина смещает влево от центра, положительная - вправо от центра.

Для десятичного выравнивания отрицательная величина смещает налево от десятичной точки, а положительная - вправо.

**Совет: Для десятичного выравнивания используйте смещение, равное  $4 * (\text{десятичные позиции} + 1)$**

8. Отметьте любую комбинацию полей Mode (режим).

Смотрите Общие свойства элементов управления - Установка режимов элементов управления (выше).

### **Закладка “Дополнительные возможности” (Extra)**

9. По желанию выберите цвета для данного комбинированного поля.

Смотрите выше Общие свойства элементов управления - Установка атрибута ЦВЕТ.

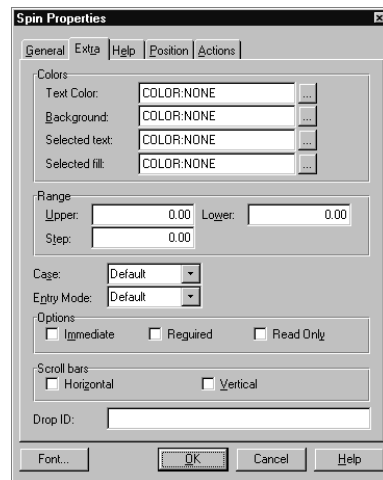
10. Установите верхний и нижний пределы диапазона Range и величину шага Step.

Поместите наибольшее значение переменной, которое Spin-поле может установить в Spin-поле, в поле Range Upper (верхний предел). Величина должна быть отформатирована так, чтобы соответствовать символьному шаблону в поле Picture. Поместите наименьшее значение переменной, которое Spin-поле может установить, в поле Lower (нижний). Поместите величину приращения - величину, на которую каждое нажатие кнопок увеличения или уменьшения изменит величину spin поля - в поле Step.

Поле From и поля пределов диапазона Range взаимоисключают друг друга.

11. Установите атрибут регистра для поля ввода.

Поле ввода может автоматически преобразовать символ из одного регистра в другой во время набора пользователем. Uppercase (верхний регистр) (атрибут UPR) автоматически преобразует все в символы верхнего регистра. Capitals (заглавные) (атрибут CAP) эквивалентен “Собственному имени” (первая буква каждого слова появляется в виде заглавной). Default (по умолчанию) принимает ввод в том регистре, в котором набирает пользователь.



12. В выпадающем списке Entry Mode (режим ввода) выберите Default, Insert или Overwrite.

Устанавливает режим ввода для поля ввода комбинированного поля. Insert (вставить) заставляет каждый удар клавиши вставлять новый символ и сдвигать уже имеющиеся символы вправо. Overwrite (переписать) заставляет каждый удар клавиши набивать новый символ поверх существующего символа. Default заставляет удар клавиши вести себя в соответствии с текущими установками системы.

Режим ввода относится только к окнам с атрибутом MASK. Смотрите более полное описание в Справочнике языка.

13. Отметьте соответствующие поля Options (необязательные параметры).

Имеются три флажка выбора, которые вы можете переключать независимо.

Immediate (немедленный) (атрибут IMM) устанавливает немедленную генерацию события как только пользователь нажимает какую-нибудь клавишу.

Required (требуется) (атрибут REQ) устанавливает, что элемент управления не может быть оставлен пустым или нулевым.

Read Only (только для чтения) (атрибут READONLY) предупреждает ввод данных в этот элемент управления. Используйте это для объявления данных только для показа.

14. Установите внешний вид кнопок элемента управления Spin, отметив для этого комбинацию двух полей Scroll Bar (поле прокручивания).

Если ни одно из этих полей не отмечено или поле Vertical отмечено, это приводит к тому, что кнопки выглядят меньшими и расположенными по вертикали. Отметка обоих полей или поля Horizontal (горизонтальный) дает большие кнопки, размещенные друг около друга.

15. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет атрибут DROPID к вашему SPIN-полю. Атрибут DROPID указывает, что это поле является правильным объектом для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этого поля.

Drag and drop означает, что конечный пользователь может выбрать объект в одном окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный”



на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

### **Закладка “Помощь” (Help)**

Смотрите Общие свойства элементов управления - Установка атрибута ПОМОЩЬ.

### **Закладка “Позиция” (Position Properties )**

Смотрите выше Общие свойства элементов управления - Установка атрибута АТ.

## **Свойства поля ввода (Entry Box)**

Элемент управления Поле Ввода позволяет пользователю вводить данные с клавиатуры. Clarion обеспечивает обширные опции для автоматической проверки вводимых пользователем данных.

- ◆ Вы можете определить для поля ввода символьный шаблон, автоматически форматирующий данные, вводимые пользователем.
- ◆ Вы можете назначить для поля начальную величину.
- ◆ Вы можете оценить достоверность данных, вводимых пользователем, либо в то время, когда он их набирает, либо в момент, когда фокус переносится на другой элемент управления.

Чтобы установить свойства для поля ввода:

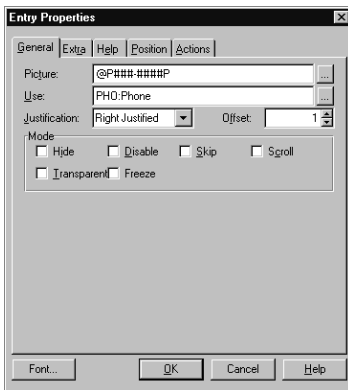
1. Из форматера окна выберите инструмент Поле ввода или выберите Entry Box (Поле ввода) из меню Control (элемент управления), затем щелкните клавишей мыши в окне.

Появится диалоговое окно Select Field (выбор поля), позволяющее вам выбрать или создать поле словаря данных или переменную памяти для данных, введенных в этот элемента управления.

2. Щелкните правой клавишей мыши на Поле ввода и выберите из всплывающего меню Properties (свойства).

Появляется диалоговое окно Entry Properties (свойства поля ввода).

## Закладка “Общая” (General)



1. В поле Picture ( шаблон изображения) назначьте шаблон для данного элемента управления.

Нажатие эллиптической кнопки дает вам возможность выбрать шаблон изображения из диалогового окна Edit Picture String (редактирование строки символов изображения). Смотрите выше Общие свойства элементов управления - Редактор шаблонов.

Шаблон изображения определяет формат, в котором данные будут вводиться в поле ввода. Например, шаблон @P##/##/##P определяет типичный формат даты.

Вы можете проверить ввод, выполненный пользователем, относительно шаблона в двух отношениях: как пользователь набирает данные, или же тогда, когда пользователь перемещает фокус на другой элемент управления (например, переходя с помощью клавиши TAB в другое поле).

Проверка данных по мере их ввода пользователем несколько замедляет работу. Отметьте поле Entry Patterns (образец ввода) в диалоговом окне Window Properties (свойства окна). Это добавит атрибут MASK к WINDOW. Когда пользователь набирает данные, программа пытается преобразовать их до совпадения с образцом. Если программа не может преобразовать данные до соответствия образцу, она подает звуковой сигнал и возвращает фокус к данному элементу управления для того, чтобы пользователь смог сделать новую попытку.

Если атрибут MASK отсутствует, проверка ввода происходит при переносе пользователем фокуса на другой элемент управления.

2. Установите атрибут Use.

Наберите имя переменной, которая примет данные, введенные пользователем в поле ввода; или нажмите эллиптическую (...) кнопку для выбора или назначения переменной.

3. Из выпадающего списка Justification (выравнивание) выберите выравнивание Left Justified (левое выравнивание), Centered (центрирование), Right Justified (правое выравнивание), Decimal (десятичное) или Default (по умолчанию).

Добавляет атрибут LEFT, CENTER, RIGHT или DECIMAL к ENTRY. Смотрите подробности в Справочнике языка. Left Justified (левое выравнивание), Centered (центрирование), Right Justified (правое выравнивание) помещают данные заранее с левым, центральным или правым выравниванием в спиновом поле. Позиция по умолчанию

помещает данные в соответствии с любыми применяемыми установками в словаре данных. Десятичное выравнивание выравнивает величины по их десятичным точкам. Каждое выравнивание может быть смещено на . установленное вами расстояние.

4. В поле Offset (смещение) установите смещение выравнивания в диалоговых единицах.

Смотрите глоссарий, где дано определение диалоговых единиц. Устанавливает величину смещения для атрибутов LEFT, RIGHT, CENTER и DECIMAL. Для выравнивания Центрирование отрицательная величина смещает влево от центра, положительная - вправо от центра.

Для десятичного выравнивания отрицательная величина смещает налево от десятичной точки, а положительная - вправо.

**Совет: Для десятичного выравнивания используйте смещение, равное  $4 * (\text{десятичные позиции} + 1)$**

5. Возможны варианты выбора режима Mode: Смотрите выше Общие свойства элементов управления - Установка режимов элементов управления.

### Закладка “Дополнительные возможности” (Extra)

6. По желанию выберите цвета для данного поля ввода.

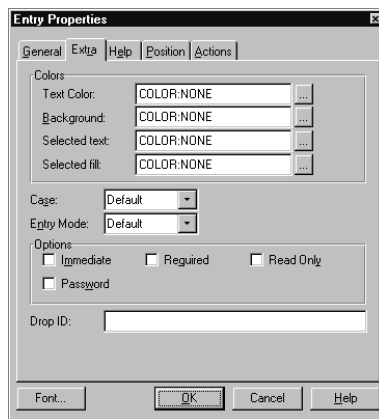
Смотрите выше Общие свойства элементов управления - Установка атрибута ЦВЕТ.

7 Установите атрибуты регистра для поля ввода.

Поле ввода может автоматически преобразовать символ из одного регистра в другой непосредственно во время ввода данных. Uppercase (атрибут UPR) (верхний регистр) автоматически переводит все в буквы верхнего регистра. Capitals (атрибут CAP) эквивалентно “Имени собственному” (первые буквы каждого слова появятся в виде заглавных букв). Default (без атрибута) принимает данные в том регистре, в котором их набирает пользователь.

8. В выпадающем списке Entry Mode (режим ввода) выберите Default (по умолчанию), Insert (вставить, атрибут INS) или Overwrite (переписать, атрибут OVR).

Установите данный режим для поля ввода. Insert заставляет каждый удар клавиши вставлять новый символ и подвигать существующие символы направо. Overwrite заставляет каждый удар клавиши набирать новый символ поверх существующего символа. Default вынуждает каждый удар клавиши вести соответственно текущим системным установкам.



Режим ввода применяется только для окон с установленным атрибутом MASK. Смотрите более полную информацию в Справочнике языка.

#### 9. Установите флажки выбора по желанию.

Имеются следующие флажка ввода, которые вы можете устанавливать или убирать независимо друг от друга:

**Immediate (немедленный)** (атрибут IMM) устанавливает немедленное генерирование события от клавиатуры каждый раз, когда пользователь нажимает любую клавишу.

**Required (требуется)**(атрибут REQ) устанавливает, что поле ввода не может быть оставлено пустым или нулевым.

**Read Only (только для чтения)** (атрибут READONLY) запрещает ввод данных с клавиатуры в это поле. Используйте этот вариант для объявления полей, предназначенных только для вывода на данных.

**Password (пароль)** (атрибут PASSWORD) назначает не показывать данные, введенные в данный элемент управления, то есть все набранные символы показываются как звездочки.

10. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет атрибут DROPID к вашему полю. Атрибут DROPID указывает, что эта поле является правильной мишенью для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этого поля.

Drag and drop означает, что конечный прользователь может выбрать объект в одном окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

### Закладка “Помощь” (Help)

Смотрите Общие свойства элементов управления - Установка атрибута ПОМОЩЬ.

**Совет: Стрелка, которая сигнализирует о текстовом вводе, является стандартным выбором формы курсора для элементов управления текстом или вводом.**

### Закладка “Позиция” (Position )

Смотрите Общие свойства элементов управления - Установка атрибута АТ.

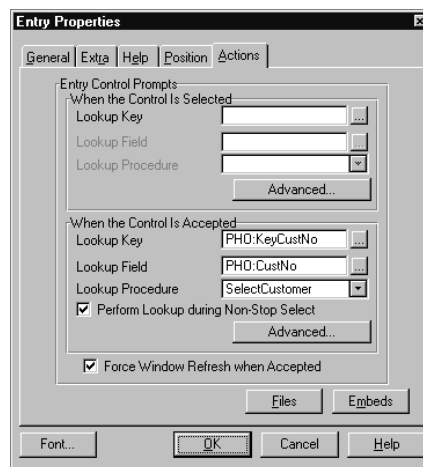
### Закладка “Действия “ (Actions )

Приглашения закладки “Действия” - из шаблонов, иначе говоря, приглашения, которые вы видите, изменяются с изменением шаблона, используемого для создания элемента управления. Ниже приводятся стандартные приглашения действий для всех элементов управления вводом. Дополнительная информация имеется в главе Шаблоны элементов управления, программы и расширений.

Стандартные приглашения Действий предназначены для того, чтобы обеспечить поддержку проверки правильности данных для ваших элементов управления поля ввода. Закладка Действия разделена на две параллельные секции. Секция When the Control is Selected (когда выбран элемент управления) обеспечивает проверку правильности, когда элемент управления получает фокус (когда пользователь использует клавишу TAB для переключения или щелкает на элементе управления клавишей мыши). Секция When the Control is Accepted (когда элемент управления принят) обеспечивает проверку правильности данных, когда элемент управления теряет фокус после того, как данные введены. Элемент управления теряет фокус тогда, когда пользователь по клавише TAB переходит к следующему полю, щелкает клавишей мыши на другом элементе управления или окне или когда он покидает окно. Две эти секции не являются взаимно исключающими, поэтому вы можете обеспечить проверку правильности в обеих точках.

Стандартные приглашения Действий сконструированы со списком возможностей выбора, имеющем в виду постановочную проверку правильности, однако они достаточно гибки чтобы позволить любые самодельные проверки, которые вы хотите обеспечить.

Установка подстановочной процедуры проверки правильности для поля ввода.



10. В поле Lookup Key (клавиша подстановки) наберите метку ключа файла - справочника или нажмите эллиптическую (...) кнопку для выбора ключа из диалогового окна Select Key (выбор ключа).

Файл справочник - это файл, который содержит все правильные величины для поля ввода, и доступ к ним осуществляется через уникальный ключ, который является ключом подстановки, который вы здесь определяете.

Например, файл, содержащий все имена покупателей для вашего приложения может быть файлом - справочником. Метка ключа может быть CUS:KeyCustNumber.

Диалоговое окно Select Key (выбор ключа) позволяет сделать выбор из файлов и ключей, уже определенных в словаре данных, или определить при необходимости новую ключ.

**Внимание: Определение нового ключа изменяет формат файла и может следовательно понадобится, чтобы вы преобразовали некоторые существующие файлы к новому формату.**

**Совет: Эта подстановочная проверка правильности лучше работает с однокомпонентным уникальным ключем.**

11. В поле Lookup Field (поле подстановки) наберите метку компонентного поля ключа подстановки или нажмите эллиптическую (...) кнопку чтобы выбрать поле из диалогового окна Select component from key (выбрать компоненту из ключа).

Это поле внутри ключа, которое содержит ту же самую величину, правильность которой проверяется. Идеально, если это поле является единственной компонентой уникального ключа. Вслед за нашим вышеприведенным примером метка поля могла быть такой: CUS:CustNumber.

12. В комбинированном поле Lookup Procedure (процедура подстановки) наберите имя процедуры или выберите существующую процедуру из выпадающего списка.

Это процедура, которая вызывается, когда пользователь вводит неправильную величину и проверка по справочнику ее отвергает. Обычно цель этой процедуры - позволить пользователю выбрать правильную величину из файла-справочника.

Процедуры выбора (или процедуры просмотра, генерированные Мастером Clarion) являются подходящими для этой цели. Вы можете также написать такую процедуру вручную. Продолжая наш приведенный выше пример, имя процедуры может быть Selectcustomer.

Основываясь на первых трех вводах, Clarion встраивает в ваши процедуры следующий код проверки правильности

```
LookupField = EntryField ! Принять ввод пользователя
GET ( LookupFile,LookupKey) ! Искать эту величину в справочнике
IF ERRORCODE( )! Если не найдена
    GlobalRequest = SelectedRecord ! подготовить вызов процедуры для выбора
    LookupProcedure! вызвать процедуру
    LocalResponse = GlobalResponse ! получить ответ о характере завершения
процедуры
    IF LocalResponse = RequestCmpleted ! если выбор сделан.
        EntryField = LookupField! поместить результат в поле ввода
        DISPLAY(?EntryField)! перерисовать экран
    ELSE! если выбор не сделан
        SELECT(?EntryField)! перейти к полю ввода
        CYCLE ! передать управление пользователю
    END
END
```

13. По желанию нажмите кнопку Advanced (передовой) для ввода (выбранной или принятой) исходного кода программы, обслуживающей событие для этого элемента управления поля ввода.

Нажимая кнопку Advanced, вызовите диалоговое окно Embedded Source (вставная исходная программа). Единственная показанная точка вставки находится после кода, сгенерированного для вызова процедуры подстановки, установленной выше. Чтобы получить другие точки вставки и для дальнейшей настройки, нажмите кнопку Embeds (вставки). Смотрите также Генератор приложений - Назначение встроенной исходной программы.

14. По желанию отметьте поле Perform lookup during non-stop select (выполните подстановку во время безостановочного выбора).

Отметка этого поля сообщает Clarion о необходимости выполнения проверки правильности, когда окно принято, даже если элемент управления вводом никогда не получал фокус. С практической точки зрения отметка этого поля страхует пользователя от пропуска полей ввода (нажата кнопка ОК при пустых вводных полях).

Этот выбор применим только к секции When the Control is Accepted (когда элемент управления принят).

15. По желанию отметьте поле Force Window Refresh when Accepted (заставьте окно обновиться когда принято).

Отметка в этом поле вызывает регенерацию всех полей (включая формулы и другие поля ввода) в окне, когда пользователь покидает по клавише TAB данное вводное поле.

16. По желанию нажмите кнопку Files (файлы) для получения доступа к диалоговому окну File Schematic Definition (Определение схемы файлов) для данной процедуры.

17. По желанию нажмите кнопку Embeds (вставки) для вставки исходной программы в точках, окружающих событие, имеющее дело только с этим полем флажков.

18. Нажмите кнопку ОК для возврата в Форматер окна.

## Свойства текстового поля (Text)

Элемент управления “Текст” обеспечивает многострочное поле ввода. Этот тип поля особенно подходит для обработки длинных символьных строк STRING или MEMO.

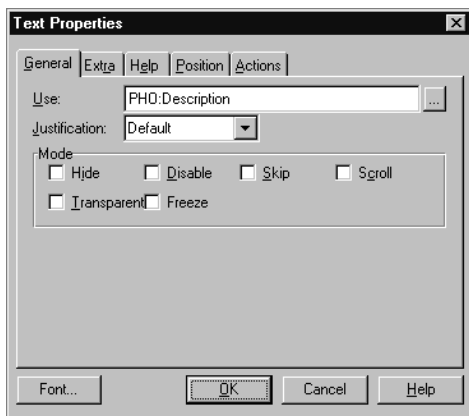
Чтобы установить свойства для текстового поля:

1. Из Форматера окна выберите инструмент Текстовое поле или выберите Text Box (текстовое поле) из меню Control (элемент управления), затем щелкните в окне.

Появится диалоговое окно Select Field (выбор поля), позволяющее выбрать или создать поле словаря данных или переменную памяти для хранения данных, введенных для этого элемента управления.

2. Щелкните правой клавишей мыши на текстовом поле и выберите из всплывающего меню Properties (свойства).

### Закладка “Общая” (General)



3. Определите атрибут Use.

Введите имя переменной, получающей значение, которое пользователь вводит в поле; или же нажмите эллиптическую кнопку (...) для того, чтобы выбрать или назначить переменную. При использовании многострочных полей проверьте, что переменная достаточного размера, чтобы разместить то количество символов, которое, как вы предполагаете, будет вводится пользователем.

4. Из выпадающего списка Justification (выравнивание) выберите выравнивание Left Justi-



fied (левое выравнивание), Centered (центрирование), Right Justified (правое выравнивание) или Default (по умолчанию).

Добавляет атрибут LEFT, CENTER или RIGHT к ТЕКСТУ. Смотрите подробности в Справочнике языка. Left Justified (левое выравнивание), Right Justified (правое выравнивание) и Centered (центрирование) предсказуемо помещают список данных с левым, центральным или правым выравниванием в текстовом поле. Default (по умолчанию) помещает данные в соответствии с любыми применяемыми установками в словаре данных.

5. Варианты выбора режима Mode: смотрите Общие свойства элементов управления  
- Установка режимов элементов управления.

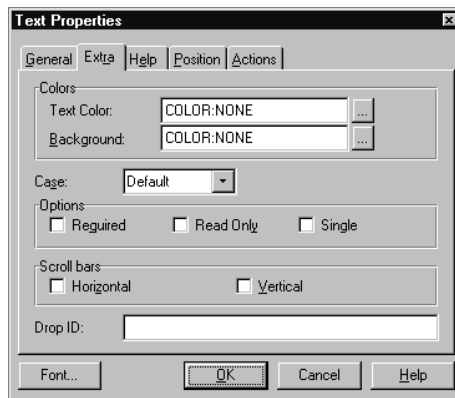
### **Закладка “Дополнительные возможности” (Extra)**

6. По желанию выберите цвета для данного текстового поля.

Смотрите Общие свойства элементов управления  
- Установка атрибута ЦВЕТ.

7. Установите атрибуты регистра для текстового поля.

Поле ввода может автоматически преобразовать символ из одного регистра в другой непосредственно во время ввода данных. Uppercase (UPR) (верхний регистр) автоматически переводит все в символы верхнего регистра. Default (без атрибута) принимает данные в том регистре, в котором их набивает пользователь.



8. Установите флажки выбора по желанию.

Имеются два флажка, которые вы можете устанавливать или убирать независимо друг от друга.

Required (требуется)(атрибут REQ) устанавливает, что поле не может быть оставлено пустым или нулевым.

Read Only (только для чтения) (атрибут READONLY) предотвращает ввод данных в этот элемент управления. Используйте это для объявления данных только для показа.

9. По желанию отметьте поля Horizontal и Vertical для добавления к вашему тексту линеек прокручивания.

10. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет атрибут DROPID к вашему полю. Атрибут DROPID указывает, что это поле является правильной мишенью для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этого поля.

Drag and drop означает, что конечный пользователь может выбрать объект в окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

### Закладка “Помощь” (Help)

Смотрите Общие свойства элементов управления - Установка атрибута ПОМОЩЬ.

**Совет: Стрелка, которая сигнализирует о текстовом вводе, является стандартным выбором формы курсора для элементов управления текстовые поля.**

### Закладка “Позиция” (Position Properties )

Смотрите Общие свойства элементов управления - Установка атрибута AT.

## **Свойства Листа (SHEET)**

---

Элемент управления SHEET (лист) объявляет группу элементов управления типа закладки TAB, которая предлагает пользователю многостраничные элементы управления для одиночного окна. Элементы управления с многими закладками в структуре SHEET назначают страницы, показываемые пользователю. Переменная USE структуры SHEET получает текст элемента управления закладка, выбранного пользователем.

## Закладка Общая

1. Поместите в поле USE метку соответствия.

Метка соответствия ссылается на лист в программных операторах.

2. Из ниспадающего списка Justification (выравнивание) установите места закладок для данного листа SHEET. Сделайте выбор из следующего:

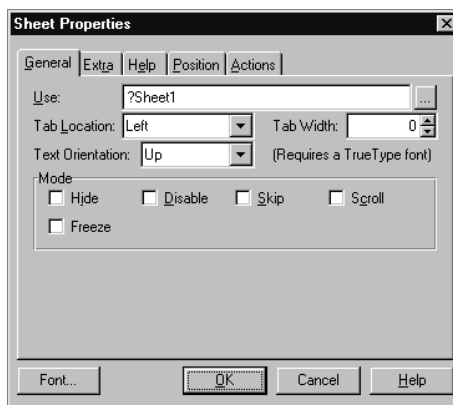
Default (по умолчанию) Закладки расположены сверху вдоль листа.

Left (левое) Закладки спускаются по левой стороне листа. Это добавляет атрибут LEFT (ширина) к SHEET.

Right (правое) Закладки спускаются по правой стороне листа. Это добавляет атрибут RIGHT (ширина) к SHEET.

Above (поверху) Закладки размещены по верху листа. Это добавляет атрибут ABOVE (ширина) к SHEET.

Below (понизу) Закладки размещены по низу листа. Это добавляет атрибут BELOW (ширина) к SHEET.



**Совет: Чтобы уместить много закладок на одном листе, установите выравнивание поверху, ориентацию текста установите на Up (вверх). Или установите выравнивание правое, а установку для ориентации текста на имеющуюся по умолчанию. Кроме того, смотрите закладку Дополнительные возможности для организации прокручивания закладок!**

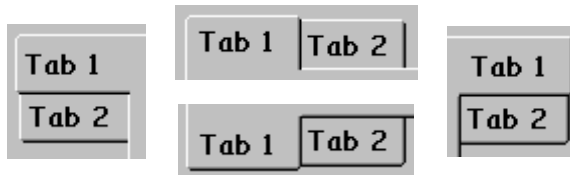
3. В поле Tab Width (ширина закладки) назначьте ширину закладки в диалоговых единицах.

Это установит величину параметра ширины для атрибутов LEFT, RIGHT, ABOVE или BELOW. Путем установки этой ширины вы можете сделать ваши закладки имеющими один размер независимо от того, как меняется длина текста от закладки к закладке.

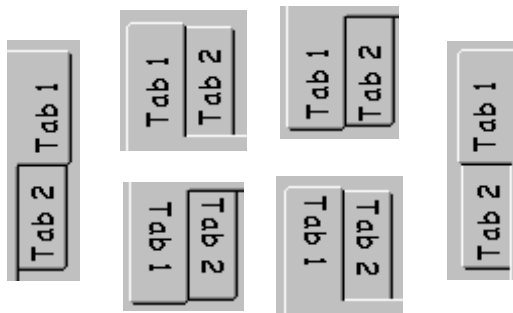
Ширина закладки - это расстояние между краями закладки, которые перпендикулярны к ориентации текста. То есть, ширина определяет, какой промежуток образуется на каждом конце текста вашей закладки, а не то, какой промежуток образуется поверху или внизу вашего текста. Это справедливо независимо от ориентации текста.

4. Воспользуйтесь ниспадающим списком Text Orientation (ориентация текста) для установки ориентации закладок и их текста. Ориентация текста иная, чем Default (по умолчанию) требует True Type шрифта. Смотрите Общие свойства элементов управления - Установка атрибута FONT, где имеется дополнительная информация. Выберите одну из следующих ориентаций:

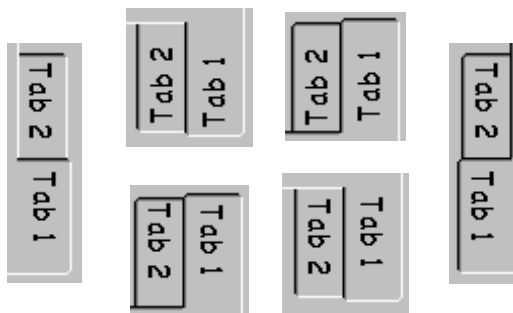
Default (по умолчанию) Текст читается слева направо, а форма закладки - горизонтальный прямоугольник.



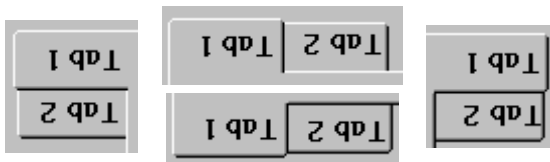
Up (вверх) Текст читается снизу вверх, а форма закладки - вертикальный прямоугольник. Это добавляет атрибут UP к SHEET



Down (вниз) Текст читается сверху вниз, а форма закладки - вертикальный прямоугольник. Это добавляет атрибут DOWN к SHEET.



Inverted (инвертированная) Текст читается перевернутым, а форма закладки - горизонтальный прямоугольник. Это добавляет атрибуты UP и DOWN к SHEET.



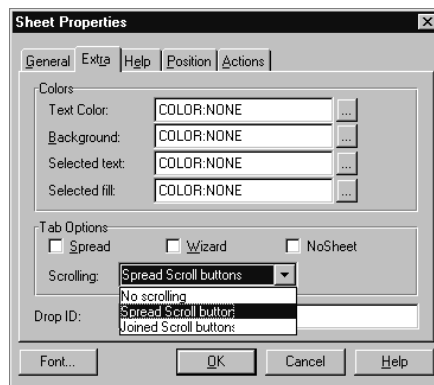
5. Выборы по желанию режима Mode: смотрите выше Общие свойства элементов управления - Установка режимов элементов управления.

### Закладка “Дополнительные возможности” (Extra Tab)

6. По желанию установите цвета для листа SHEET.

Цвета применяются для всех закладок на листе, но могут быть заменены установкой цветов для отдельных закладок. Смотрите Общие свойства элементов управления - Установка атрибута COLOR.

7. Отметьте поле Spread (размах) для изменения размеров “закладок” закладок с целью заполнения всего имеющегося пространства на листе SHEET.



Алгоритм изменения размеров учитывает размер и ориентацию текста, показываемого на каждой закладке, число закладок и доступное пространство на листе свойств. Это добавляет атрибут SPREAD к SHEET. Смотрите более полную информацию в Справочнике языка.

8. Отметьте поле Wizard (мастер) для того, чтобы спрятать часть “закладка” элемента управления TAB.

Спрятывание закладок помогает при создании мастера. Мастер - это окно с “беззакладочным” элементом управления SHEET (лист), содержащее одну или более TAB (закладок). От вас потребуется написать программу для управления “переворачиванием страниц”.

Это добавляет к SHEET атрибут WIZARD. Смотрите более полную информацию в Справочнике языка. В интерактивной помощи смотрите Как создать мастер.

**Совет: Не отмечайте это поле до тех пор, пока вы не закончили конструирование окна!**

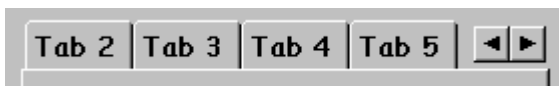
9. Отметьте поле NoSheet (без листа) для того, чтобы удалить границы закладок, так чтобы была видна только выступающая, предназначенная для отбора часть закладки.

Имеется дополнительный сопутствующий эффект чтобы заставляя закладки располагаться поверх листа, идти понизу и быть размещенными ниже подъема листа к верхушке листа. Это добавляет к SHEET атрибут NoSheet. Смотрите более полную информацию в Справочнике языка.

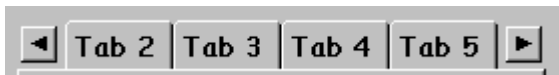
10. Из ниспадающего списка Scrolling (прокручивание) выберите поведение прокручивания закладок путем выбора одного из следующих вариантов:

No Scrolling (без прокручивания) Закладки не прокручиваются. Это установка по умолчанию.

Joined Scroll Buttons (присоединенные кнопки прокручивания) Закладки могут прокручиваться с помощью соседних кнопок прокручивания. Добавляет к SHEET атрибут JOINED. Более полную информацию можно найти в Справочнике языка.



Spread Scroll Buttons (кнопки прокручивания по краям) Закладки могут прокручиваться с помощью кнопок прокручивания, расположенных на противоположных сторонах листа. Добавляет к SHEET атрибут HSCROLL. Более полную информацию можно найти в Справочнике языка.



11. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур . Форматер окна добавляет атрибут DROPID к вашему полю. Атрибут DROPID указывает, что это поле является правильной мишенью для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этого поля.

Drag and drop означает, что конечный прользователь может выбрать объект в одном окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DR

AGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

### Закладка “Помощь” (Help)

Смотрите Общие свойства элементов управления - Установка атрибута HELP.

### Закладка “Позиция” (Position Properties )

Смотрите Общие свойства элементов управления - Установка атрибута AT.

## **Свойства элемента управления “Закладка”**

Структура закладка TAB определяет группу элементов управления. Эта группа - одна из многих страниц, которые могут содержаться в структуре лист SHEET. Структуры закладки внутри структуры листа SHEET определяют страницы, показываемые пользователю. Атрибут USE структуры SHEET получает текст данного элемента управления ЗАКЛАДКА, отобранного пользователем.

Стандартом Windows 95 перехода от закладки к закладке является клавишная комбинация CTRL+TAB. Элементы управления ЗАКЛАДКА Clarion следуют этому стандарту, как в среде разработки, так и в скомпилированном приложении.

**Внимание: Если вы вкладываете ЗАКЛАДКИ, комбинация CTRL+TAB контролирует только верхний уровень.**

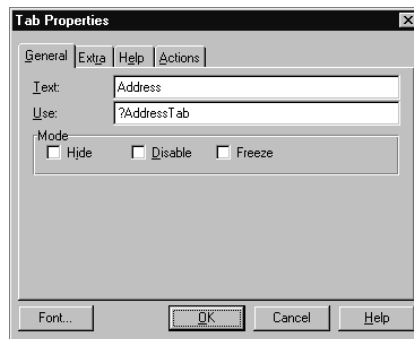
### Закладка общая (General Tab)

1. В поле Text (текст) наберите строку символов.

Если данный элемент управления предназначен для показа переменной, наберите шаблон изображения в этом поле.

2. В поле USE наберите метку соответствия.

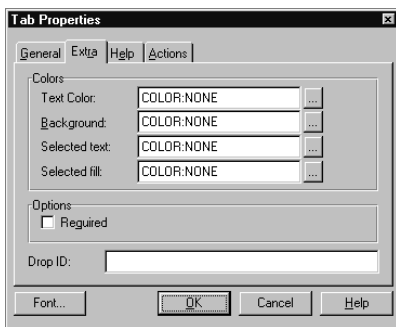
Метка соответствия ссылается на данный элемент управления закладка в программных операторах.



3. Возможные выборы режима Mode: смотрите выше Общие свойства элементов управления -

Установка режимов элементов управления.

### Дополнительные возможности (Extra Tab)



4. По желанию установите цвета для закладки TAB. Эта установка переопределяет установку, определенную для листа.

Смотрите Общие свойства элементов управления - Установка атрибута ЦВЕТ

5. Отметьте поле Required (требуется) для того, чтобы привести в действие ввод для требуемых полей ввода.

Когда это поле отмечено, ваша программа автоматически проверяет, чтобы все элементы управления ввода с атрибутом REQ не были ни пустыми, ни нулевыми.

Назначьте этот тип закладки, когда окно также содержит элементы управления ENTRY, COMBO, SPIN или TEXT с атрибутом REQ (или используйте функцию INCOMPLETE (неполная) для проверки элементов управления ввода). Когда пользователь щелкает клавишей мыши на закладке с атрибутом REQ и поле ввода пусто или нулевое, первый требуемый элемент управления, который пуст или нулевой, получает фокус.

6. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет атрибут DROPID к вашему полю. Атрибут DROPID указывает, что это поле является правильной мишенью для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этого поля.

Drag and drop означает, что конечный пользователь может выбрать объект в окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.



**Закладка “Помощь” (Help)**

Смотрите Общие свойства элементов управления - Установка атрибута HELP.

**Положение (Position)**

Положение элемента управления TAB определяется положением родительского листа SHEET.

## Неинтерактивные элементы управления

Неинтерактивные элементы управления не принимают данных, вместо этого они направляют пользователя к другим элементам управления либо с помощью текста, либо с помощью графики. Например:

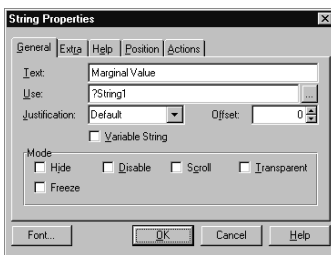
- ◆ Строка символов в диалоговом поле может давать разъяснения для заполнения полей данных.
- ◆ Один из простейших графических элементов - групповое поле - может визуально ассоциировать группу элементов управления, сигнализируя пользователю, что весь ввод относится к одной и той же теме.
- ◆ Изображение или график могут сделать больше, чем просто украсить диалог. Они могут придать значение процессу, что без их применения потребовало бы многих и многих слов.

### Свойств строки символов

Элемент управления типа “строка символов” дает вам возможность разместить на экране строковую константу. Это, если необходимо, позволяет вам показать переменную.

Диалоговое окно String Properties (свойства строки символов) содержит следующие параметры:

#### Закладка “Общая” (General)



1. В поле Text (текст) наберите строковую константу или шаблон изображения.

Константа в виде строки символов выглядит так, как она набита.

Если отметить поле Variable String (переменная строка символов), то это изменяет Text : приглашение на Picture: . Шаблон изображения используется для форматирования переменной строки символов для показа. Нажмите эллиптическую кнопку (...) для того, чтобы назначить соответствующий шаблон изображения. Смотрите Общие свойства элементов управления - Редактор шаблонов изображения.

2. В поле Use наберите метку соответствия или назовите метку переменной для показа.

Метка соответствия употребляется для ссылки на данный элемент управления в исходной программе. Нажмите эллиптическую кнопку (...) для того, чтобы отобразить или назначить для показа переменную.

3. Из выпадающего списка Justification (выравнивание) выберите выравнивание Left Justified (левое выравнивание), Centered (центрирование), Right Justified (правое

выравнивание), Decimal (десятичное) или Default(по умолчанию).

Добавляет атрибут LEFT, CENTER, RIGHT или DECIMAL к STRING. Смотрите подробности в Справочнике языка. Left Justified (левое выравнивание), Centered (центрирование), Right Justified (правое выравнивание) помещают данные заранее с левым, центральным или правым выравниванием в элементе управления. Позиция по умолчанию помещает данные в соответствии с любыми применяемыми установками в словаре данных. Десятичное выравнивание выравнивает величины по их десятичным точкам. Каждое выравнивание может быть смещено на установленное вами расстояние.

4. В поле Offset (смещение) установите смещение выравнивания в диалоговых единицах.

Смотрите глоссарий, где дано определение диалоговых единиц. Устанавливает величину смещения для атрибутов LEFT, RIGHT, CENTER и DECIMAL. Для выравнивания Центрирование отрицательная величина смещает влево от центра, положительная - вправо от центра.

Для десятичного выравнивания отрицательная величина смещает налево от десятичной точки, а положительная - вправо.

**Совет: Для десятичного выравнивания используйте смещение, равное  $4 * (\text{десятичные места} + 1)$**

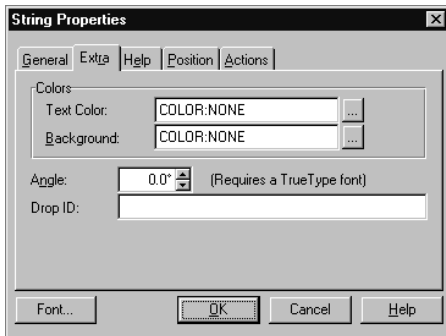
5. Если необходимо, установите отметку в поле Variable String (строка символов переменной).

Этот параметр устанавливает, что вы хотите показать содержание переменной в поле строки символов. Если это так, поместите символьный шаблон в поле параметра, @s24 например.

6. Варианты выбора режима Mode: Смотрите Общие свойства элементов управления - Установка режимов элементов управления..

**Совет: Когда вы помещаете текст поверх ИЗОБРАЖЕНИЯ или цветной графики типа ПОЛЯ, отметь поле Transparent (прозрачный) (атрибут TRN) , так чтобы текст не затемнял графику.**

## Закладка “Дополнительные возможности” (Extra)



7. По желанию выберите цвета для данной строки символов.

Смотрите Общие свойства элементов управления - Установка атрибута COLOR (ЦВЕТ).

8. По желанию установите Angle (угол) текста.

Отличный от нуля угол требует True Type шрифта. Смотрите Общие свойства элементов управления - Установка атрибута FONT, где имеется дополнительная информация. Добавляет к STRING атрибут ANGLE. С помощью него вы можете поворачивать текст от его нормального положения на все 360 градусов.

9. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет атрибут DROPID к вашей кнопке. Атрибут DROPID указывает, что эта кнопка является правильной мишенью для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этой кнопки.

Drag and drop означает, что конечный прользователь может выбрать объект в окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

## Закладка “Помощь” (Help)

Смотрите Общие свойства элементов управления - Установка атрибута Help.

## Закладка “Позиция” (Position Properties)

Смотрите Общие свойства элементов управления - Установка атрибута АТ.

## Свойства элемента управления Приглашение (PROMPT)

Элемент управления приглашение дает вам возможность разместить на экране строку символов, которая автоматически обеспечивает быструю клавишу доступа к следующему активному элементу управления, следующему за приглашением. Он почти идентичен элементу управления STRING (строка символов) за исключением того, что он не обладает способностью изменения и способностью наклона. Смотрите выше Свойства строки символов.

## Свойства элемента управления Групповое поле (GROUP BOX)

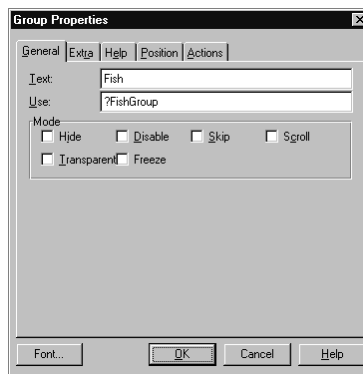
Групповое поле рисует рамку вокруг других элементов управления. Оно визуальное объединяет элементы управления с точки зрения пользователя и дает вам возможность обращаться ко всем элементам управления как к одному целому - облегчая, например, вывод из действия всех членов группы одновременно.

Диалоговое окно Group Properties (свойства группы) содержит следующие параметры.

### Закладка “Общая” (General)

1. В поле Text (текст) наберите константу в виде строки символов

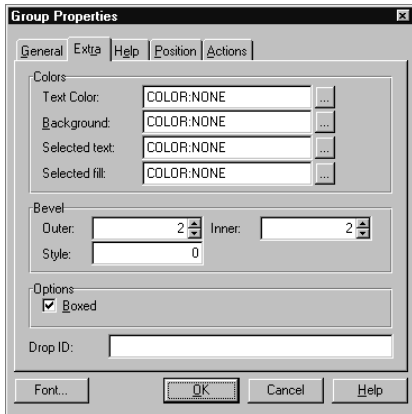
Поле Text (текст) принимает константу в виде строки символов, содержащую приглашение для группы элементов управления. Эта строка символов появляется во время работы на верхней границе группового поля. Знак (&) внутри текста означает, что следующий символ - это быстрая клавиша для группы. В результате этот символ подчеркивается, и когда пользователь нажимает ALT+соответствующая клавиша, первый элемент управления в группе получает фокус. Этот текст может быть также установлен в поле Text (текст) панели инструментов Property (свойство)..



2. В поле Use наберите метку соответствия поля.

3. Выбор режима (Mode) по желанию: смотрите выше Общие свойства элементов управления - Установка режимов элементов управления.

## Закладка “Дополнительные возможности” (Extra)



4. По желанию установите цвета для данного группового поля.

Смотрите Общие свойства элементов управления - Установка атрибута COLOR.

5. По желанию, установите атрибут BEVEL (фаска) для ГРУППЫ.

Атрибут Фаска придает трехмерный вид групповому полю. Поле выглядит приподнятым, заглубленным или и таким и таким.

**Outer (внешний)** Положительная величина делает групповое поле кажущимся приподнятым над плоскостью окна. Чем больше эта величина, тем более приподнятым будет казаться поле. Отрицательная величина делает групповое поле кажущимся заглубленным ниже уровня плоскости окна. Этот эффект фаски начинает проявляться с внешней кромки поля.

**Inner (внутренний)** Положительная величина делает групповое поле кажущимся приподнятым над плоскостью окна. Чем больше эта величина, тем более приподнятым будет казаться поле. Отрицательная величина делает групповое поле кажущимся заглубленным ниже уровня плоскости окна. Этот эффект фаски начинает проявляться с внутренней кромки поля.

**Style (стиль)** USHORT, чьи шестнадцать бит определяют стиль (но не размер) каждого из четырех краев Опции (OPTION). Атрибут STYLE (стиль) предоставляет вам очень тонкий инструмент управления внешним видом фаски. Смотрите Справочник языка, где объясняется значение каждого бита.

6 Если необходимо, сделайте групповое поле невидимым.

Путем очистки поля **Boxed** вы можете сделать групповое поле и его текст невидимым для пользователя. Это удаляет из GROUP атрибут BOXED. Групповое поле является видимым в Форматере окна, но невидимым в режиме предварительного просмотра Preview! и во время работы. Это создает иной эффект нежели спрятавание или дезактивация группы. Спрятавание и дезактивация группы также прячут и дезактивируют все элементы управления в пределах группы.

7. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет атрибут DROPID к вашей кнопке. Атрибут DROPID указывает, что эта кнопка является правильным объектом для операций “Drag and Drop” (тащить и бросать). Сигнатура - это

строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этой кнопки.

Drag and drop означает, что конечный пользователь может выбрать объект в окне или элемент управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

### **Закладка “Помощь” (Help)**

Смотрите Общие свойства элементов управления - Установка атрибута Help.

### **Закладка “Позиция” (Position Properties)**

Смотрите Общие свойства элементов управления - Установка атрибута AT.

## **Свойства строки индикатора выполнения (Progress Bar)**

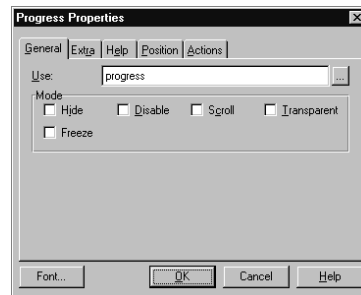
Элемент управления PROGRESS показывает строку индикатора выполнения. Эта строка обычно показывает текущий процент выполнения пакетной обработки путем инкрементного “заполнения” строки по мере развития процесса.

Диалоговое окно Progress Properties (свойства индикатора выполнения) содержит следующие возможные параметры.

### **Закладка “Общая” (General)**

1. В поле USE наберите метку цифровой переменной, которая отслеживает прогресс вашего процесса.

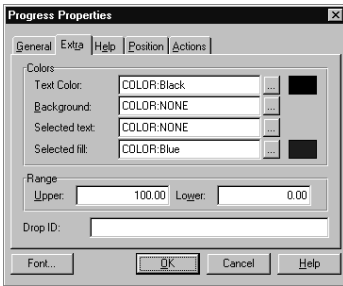
Строка индикатора выполнения автоматически обновляется каждый раз, когда величина в этой переменной изменяется. Если атрибут USE является меткой соответствия, вы должны непосредственно обновить строку прогресса приписыванием величины (в пределах, определенных



атрибутом RANGE (диапазон) свойству элемента управления PROP:progress.

2. Режим Mode по выбору: смотрите Общие свойства элементов управления - Установка режимов элементов управления.

### **Закладка “Дополнительные возможности” (Extra)**



3. По желанию установите цвета для строки прогресса.

Смотрите Общие свойства элементов управления - Установка атрибута COLOR ( ЦВЕТ ).

4. Установите диапазон (Range) величин, показываемых строкой индикатора выполнения.

Если этого не делать, то диапазон по умолчанию будет составлять от нуля (0) до ста (100).

5. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет атрибут DROPID к вашей кнопке. Атрибут DROPID указывает, что эта кнопка является правильной мишенью для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этой кнопки.

Drag and drop означает, что конечный прользователь может выбрать объект в окне или элемент управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

### **Закладка “Помощь” (Help)**

Смотрите Общие свойства элементов управления - Установка атрибута Help.

### **Закладка “Позиция” (Position Properties)**

Смотрите Общие свойства элементов управления - Установка атрибута AT.



## Свойства изображения (Image)

Элементы управления Изображение дают вам возможность размещать в окне растровые и векторные изображения. Поддерживаются растровые форматы: .BMP, .CUR, .PCX, .GIF, .ICO и .JPG и векторный формат .WMF. Clarion for Windows поддерживает палитру в 16.7 миллионов цветов.

**Совет:** Используйте атрибут PALETTE в вашем окне, чтобы обеспечить поддержание цвета для ваших изображений. Атрибут PALETTE устанавливает, сколько цветов вы бы хотели чтобы использовало ваше окно, когда оно является окном переднего плана. Это применимо для аппаратного режима, где используется палитра и имеются лишние цвета. Смотрите подробности в Справочнике языка.

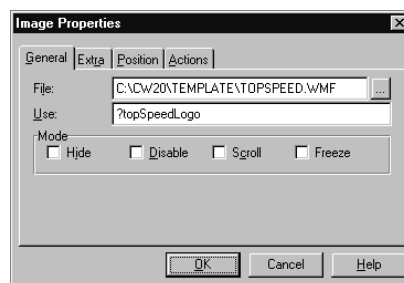
Диалоговое окно Image Properties (свойства изображения) содержит следующие параметры.

### Закладка “Общая” (General)

Нажмите эллиптическую кнопку для выбора файла изображения из диалога Open File.

1. Выберите графический файл.

Наберите имя файла или нажмите эллиптическую (...) кнопку справа от поля File для выбора графического файла, используя стандартный диалог File Open.



**Совет:** Растровые изображения могут занять много памяти. Если ваше приложение использует много растровых картинок, следите за размером свободной памяти.

2. Поместите метку соответствия в поле Use.

Метка соответствия ссылается на изображение в операторах программы.

3. Варианты выбора режима Mode: смотрите выше Общие свойства элементов управления - Установка режимов элементов управления.

### Закладка “Дополнительные возможности” (Extra)

4. Если необходимо, добавьте линейки прокрутки.

Отметьте Horizontal, Vertical или оба поля флажков для того, чтобы добавить линейки прокрутки в том случае, если изображение больше, чем размер данного элемента управления,

### **Закладка “Позиция” (Position Properties)**

Смотрите Общие свойства элементов управления - Установка атрибута AT.

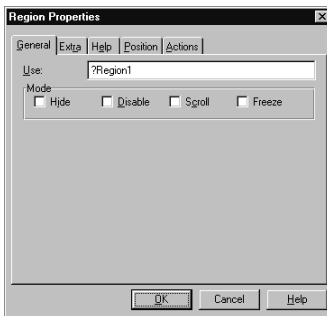
**Совет: Для элементов Изображение по умолчанию показывается изображение в размере, который был создан.**

## **Свойства элемента управления Область (Region)**

Элемент управления Область - это просто прямоугольная площадка экрана. Ее главным назначением является обеспечить ссылку, чтобы проверить, случилось ли данное событие - такое, как событие мыши - внутри этой области.

Вы можете придать области цвет или обеспечить изменение курсора, когда пользователь проводит мышью над ней.

### **Закладка “Общая” (General)**

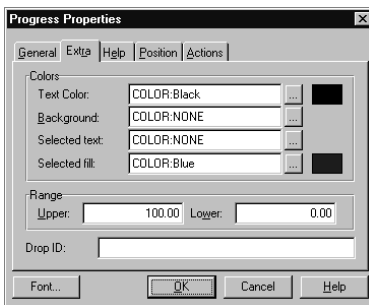


1. Поместите метку соответствия в поле USE.

Метка соответствия ссылается на эту область в операторах программы.

2. Варианты выбора режима работы Mode: смотрите Общие свойства элементов управления - Установка режимов элементов управления.

### **Закладка “Дополнительные возможности” (Extra)**



3. По желанию установите цвета заполнения Fill и цвет рамки Border.

Наберите допустимую величину соответствия цвета или нажмите эллиптическую кнопку для того, чтобы выбрать цвет. Появится стандартное диалоговое окно Color (цвет). Выберите цвет, щелкнув на квадратике выбора цвета, или добавьте заказной цвет. Смотрите Общие свойства элементов управления - Диалоговое окно цвета.

4. По желанию, установите атрибут BEVEL (фаска) для REGION.

Атрибут Фаска придает трехмерный вид области. Область выглядит приподнятой, заглубленной или и такой и такой.

Outer (внешний) Положительная величина делает область кажущейся приподнятой над плоскостью окна. Чем больше эта величина, тем более приподнятой будет казаться область. Отрицательная величина делает область кажущейся заглубленной ниже уровня плоскости окна. Этот эффект фаски начинает проявляться с внешней кромки области.

Inner (внутренний) Положительная величина делает область кажущейся приподнятой над плоскостью окна. Чем больше эта величина, тем более приподнятой будет казаться область. Отрицательная величина делает область кажущейся заглубленной ниже уровня плоскости окна. Этот эффект фаски начинает проявляться с внутренней кромки области.

Style (стиль) USHORT, чьи шестнадцать бит определяют стиль (но не размер) каждого из четырех краев Опции (OPTION). Атрибут STYLE (стиль) предоставляет вам очень тонкий инструмент управления внешним видом фаски. Смотрите Справочник языка, где объясняется значение каждого бита.

5. Вы можете добавить атрибут IMM к элементу управления Область, отмечая поле флажков Immediate (немедленный).

Это генерирует события (EVENT:MouseIn, EVENT:MouseOut, EVENT:MouseMove) (мышь внутри, мышь вне, движение мыши) когда пользователь проходит мышью над областью; однако это требует потерь рабочего времени, поэтому этим следует пользоваться умеренно.

6. В поле Drag ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет атрибут DRAGID к вашей области. Атрибут DRAGID указывает, что эта область является правильным источником для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этой области.

Drag and drop означает, что конечный прользователь может выбрать объект в одном окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом.

Осуществление этой способности требует, чтобы элемент управления - источник имел атрибут DRAGID с сигнатурой, которая совпадает с сигнатурой DROPID мишени, и чтобы процедуры, которые ведут каждое окно, имели подходящую исходную программу для

обработки событий “тащить и бросать”. Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

6. В поле Drop ID при необходимости наберите до шестнадцати (16) разделенных запятой сигнатур (Идентификатор мишени операции “Drag and Drop”). Форматер окна добавляет атрибут DROPID к вашей области. Атрибут DROPID указывает, что эта область является правильной мишенью для операций “Drag and Drop” (тащить и бросать). Сигнатура - это строковая константа, которая идентифицирует, какие типы операций drag and drop допустимы для этой области.

Смотрите более подробно и с примерами в Справочник языка. Смотрите также главу Форматер списочного поля - Добавление способности drag and drop к списочному полю.

### **Закладка “Помощь” (Help)**

Смотрите Общие свойства элементов управления - Установка атрибута Help.

### **Закладка “Позиция” (Position Properties)**

Смотрите Общие свойства элементов управления - Установка атрибута AT.

## **Свойства элемента управления Линия (Line)**

Элемент управление Линия дает вам возможность помещать в ваши окна прямую линию. Вы можете установить цвет линии, толщину и положение. Элемент управления Линия не может получить фокус, не может он и генерировать события.

Диалоговое окно Line Properties(свойства линий) содержит следующие параметры.

### **Закладка “Общая” (General)**

1. Поместите метку соответствия в поле USE.

Метка соответствия ссылается на линию в операторах программы.

2. По желанию установите толщину линии.

Наберите величину во вращающемся элементе управления Line Width (ширина линии). Величина по умолчанию равна 1 пиксел.

3. Варианты выбора режима (Mode): смотрите выше Общие свойства элементов управления - Установка режимов элементов управления.

### Закладка “Дополнительные возможности” (Extra)

#### 3. Установка цвета линии Line Color.

Наберите допустимую величину соответствия цвета или нажмите эллиптическую кнопку для того, чтобы выбрать цвет. Появится стандартный диалог Color. Выберите цвет, щелкая мышью на квадратике образца цвета или добавьте заказной цвет. Смотрите Общие свойства элементов управления - Диалоговое окно цвета

**Совет: Чтобы усилить “резной” вид трехмерного окна со строкой меню, поместите горизонтальную белую линию начиная с точки 0,0 с атрибутом FULL и с высотой 0. Линия оттеняет серую область окна относительно строки меню.**

### Закладка “Позиция” (Position Properties)

Смотрите Общие свойства элементов управления - Установка атрибута AT.

## Свойства элемента управления Прямоугольная область (Box)

Элемент управления Прямоугольная область дает вам возможность разместить в окне квадрат или прямоугольник. Вы можете заполнить поле цветом и установить цвет контура. Вы можете также установить, чтобы прямоугольник получился с закругленными углами. Элемент управления Прямоугольная область не может получить фокус, не может он также и генерировать события.

Диалог Box Properties (свойства поля) содержит следующие параметры.

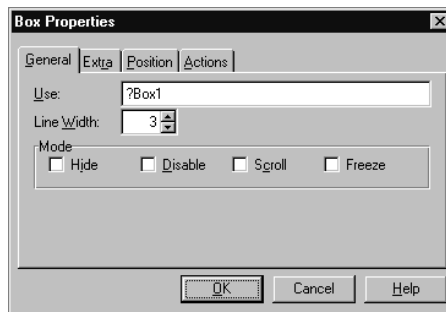
### Закладка “Общая” (General)

#### 1. Поместите метку соответствия в поле Use.

Эта метка соответствия будет ссылается на прямоугольник в операторах программы.

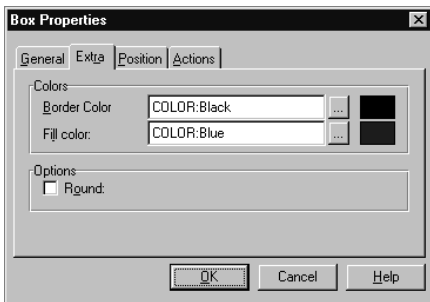
#### 2. По желанию установите толщину границы области.

Наберите величину во вращающемся элементе управления Line Width (ширина линии). Величина по умолчанию равна 1 пиксел.



#### 3. Варианты выбора режима (Mode): смотрите выше Общие свойства элементов управления - Установка режимов элементов управления.

### Закладка “Дополнительные возможности” (Extra)



4. Установите цвета заполнения Fill и контура Border.

Наберите допустимую величину соответствия цвета или нажмите эллиптическую кнопку для того, чтобы выбрать цвет. Появится стандартный диалог Color. Выберите цвет, щелкая мышью на квадратике образца цвета или добавьте заказной цвет. Смотрите Общие свойства элементов управления - Диалоговое окно цвета

5. Если необходимо, установите, что прямоугольная область должна появляться в виде “закругленного” прямоугольника. Отметьте это в поле флажков Rounded.

Углы поля получатся закругленными.

### Закладка “Позиция” (Position Properties)

Смотрите Общие атрибуты элементов управления - Установка атрибута AT.

**Совет: Хотя вы и можете установить размер прямоугольника и других графических элементов управления набором вручную координат, гораздо проще прямо рисовать их в форматере окна.**

### Свойства элемента управления Эллипс (Ellipse)

Элемент управления Эллипс позволяет вам поместить в ваши окна кружок или эллипс. Вы можете заполнить эллипс цветом и установить цвет контура.

Эллипс не может быть в фокусе, он не генерирует событий.

Диалоговое окно Ellipse Properties (свойства эллипса) содержит следующие параметры.

### Закладка “Общая” (General)

1. Поместите метку соответствия в поле Use.

Эта метка соответствия будет ссылается на эллипс в операторах программы.

2. По желанию установите толщину границы эллипса.

Наберите величину во вращающемся элементе управления Line Width (ширина линии). Величина по умолчанию равна 1 пиксел.

3. Варианты выбора режима (Mode): смотрите выше Общие свойства элементов управления - Установка режимов элементов управления.

### Закладка “Дополнительные возможности” (Extra)

4. Установите цвет заполнения Fill и цвет контура Border.

Наберите допустимую величину соответствия цвета или нажмите эллиптическую кнопку для того, чтобы выбрать цвет. Появится стандартный диалог Color. Выберите цвет, щелкая мышью на квадратике образца цвета или добавьте заказной цвет. Смотрите Общие атрибуты элементов управления - Диалоговое окно цвета.

### **Закладка “Позиция” (Position Properties)**

Смотрите Общие атрибуты элементов управления - Диалоговое окно цвета.

**Совет: Хотя вы можете установить размер эллипса и других графических элементов управления вручную, набирая их координаты, намного легче сделать это, рисуя их непосредственно в формате окна.**

### **Свойства панели (Panel Properties)**

---

Элемент управления “Панель” дает вам возможность поместить приподнятую или вдавленную прямоугольную панель в ваших окнах. Вы можете закрасить панель. Элемент управления не может получить фокус, не может он и генерировать события.

Диалоговое окно Panel Properties (свойства панели) содержат следующие возможные выборы.

### Закладка “Общая” (General)

1. Поместите метку соответствия в поле Use.

Эта метка соответствия будет ссылается на панель в операторах программы.

2. Варианты выбора режима (Mode): смотрите выше Общие свойства элементов управления - Установка режимов элементов управления.

### Закладка “Дополнительные возможности” (Extra)

#### 3. Установите цвет заполнения Fill.

Наберите допустимую величину соответствия цвета или нажмите эллиптическую кнопку для того, чтобы выбрать цвет . Появится стандартное диалоговое окно Color. Смотрите Общие атрибуты элементов управления - Диалоговое окно цвета.

#### 4. По желанию, установите атрибут BEVEL (фаска) для ПАНЕЛИ.

Атрибут Фаска придает трехмерный вид области. Область выглядит приподнятой, заглубленной или и такой и такой.

Outer (внешний) Положительная величина делает панель кажущейся приподнятой над плоскостью окна. Чем больше эта величина, тем более приподнятой будет казаться панель. Отрицательная величина делает область кажущейся заглубленной ниже уровня плоскости окна. Этот эффект фаски начинает проявляться с внешней кромки панели.

Inner (внутренний) Положительная величина делает панель кажущейся приподнятой над плоскостью окна. Чем больше эта величина, тем более приподнятой будет казаться панель. Отрицательная величина делает панель кажущейся заглубленной ниже уровня плоскости окна. Этот эффект фаски начинает проявляться с внутренней кромки панели.

Style (стиль) USHORT, чьи шестнадцать бит определяют стиль (но не размер) каждого из четырех краев Опции (OPTION). Атрибут STYLE (стиль) предоставляет вам очень тонкий инструмент управления внешним видом фаски. Смотрите Справочник языка, где объясняется значение каждого бита.

### Закладка “Позиция” (Position Properties)

Смотрите Общие атрибуты элементов управления - Установка атрибута AT.

**Совет: Хотя вы можете установить размер панели и других графических элементов управления вручную, набирая их координаты, намного легче сделать это, рисуя их непосредственно в формате окна.**



## ЭЛЕМЕНТЫ УПРАВЛЕНИЯ OLE

### Обзор

---

Элемент управления OLE дает вам возможность поместить в ваше окно объекты OLE (связывание и встраивание объектов) или элементы управления .OCX. Смотрите более полное обсуждение этого вопроса в Связывании и встраивании объектов в Справочнике языка.

### OLE

Используя Сервер OLE, вы можете добавить всю мощь сервера к своему приложению, выполняя со своей стороны очень небольшую работу по программированию. Однако ваши конечные пользователи должны быть лицензированы для использования Сервера OLE, а их технические средства должны поддерживать существенные дополнительные ресурсы, требуемые сервером. Например, ваше Clarion приложение может очень эффективно работать на процессоре 486 с 4 Мб оперативной памяти RAM, но если оно вызывает Excel, PowerPoint и/или Word, ваши конечные пользователи должны иметь лицензии на Excel, PowerPoint и Word, а их технические средства должны обеспечить приемлемый уровень работы этих программ (Pentium с оперативной памятью RAM от 16 до 32 Мб).

### OCX

Для работы с OCX обычно требуется большого количества встроенных исходных кодов программ в точках вставки или написанной вами программы. Однако многие OCX поставляются конечным пользователям без дополнительной лицензионной платы и к тому же большинство OCX требуют значительно меньших ресурсов, чем типичные серверы OLE.

**Внимание: Элемент управления OLE - это не шаблон элемента управления. Следовательно, Генератор приложений не генерирует исходную программу для поддержки конкретной функциональной особенности данного элемента управления. Вы должны встроить некоторую исходную программу или написать свой собственный шаблон для того, чтобы получить необходимые функциональные характеристики от элемента управления OLE, в противном случае это только дисплей.**

### Встроенная программа (Embedded Code)

Когда вы используете элемент управления OLE для получения доступа к OLE Серверу, вы по крайней мере должны будете как минимум встроить программу для активации и деактивации сервера (OCX активируются автоматически). Это открывает вашему

пользователю доступ ко всем функциональным возможностям сервера. Вы можете встроить дополнительную программу для управления поведением сервера и тем, какие функции сервера будут доступны для вашего пользователя. Для получения более полной информации ознакомьтесь с документацией сервера.

Вы можете активировать свой сервер, используя или кнопку или выбора из меню или из списка выборов, щелчка клавиши мыши и т.д. Чтобы активировать сервер, используйте PROP:DoVerb. Например:

!Активировать сервер в его режиме по умолчанию.

?01eControl {PROP : DoVerb } = 0

или

!Активировать сервер в режиме открытия (в его собственном отдельном окне).

?01eControl {PROP : DoVerb } = -2

Для деактивации активированного на месте сервера используйте PROP:Deactivate. Вставьте программу деактивации в точку вставки Window Event Handling - Close Window (управление событием окна - закрыть окно) или Gain Focus (получить фокус). Или же добавьте позицию меню “деактивировать” элементу управления OLE и встройте программу деактивации в точку вставки Control Event Handling - Accepted (управление событием элемента управления - принято) для данной позиции меню. Например:

?01e{PROP:Deactivate} !Де-активировать Сервер OLE

Для подключения OCX требуется значительно большие программные вставки. Требуемая программа характерна для OCX и обычно требует полного понимания принципа действия OCX.

### **Меню элемента управления OLE (OLE Control Menus)**

Элементы управления OLE могут иметь меню (MENUs), подобно тем, которые имеют окна (WINDOWS). Чтобы назначить меню элемента управления OLE, в Форматере окна (Window Formatter) выберите элемент управления OLE, затем выберите Menu > New Menu. Назначьте меню таким же образом, как вы назначаете меню для ваших окон (смотрите Меню и Панели инструментов).

Меню элемента управления OLE сливается с меню данного окна. Во время работы, если вы активируете сервер OLE “на месте”, меню сервера также сливается с меню данного окна.

### **Свойства элемента управления OLE (OLE Control Properties)**

Воспользуйтесь диалоговым окном OLE Properties (свойства OLE) для установки того, как элемент управления OLE объявляется в генерированной исходной программе. Данный раздел описывает каждое приглашение в этом диалоговом окне. А на страницах, следующих за этим разделом, приводятся инструкции по использованию данного диалогового окна для специфических реализаций OLE.

### Закладка “Общая” (General)

1. Поместите метку соответствия в поле Use.

Эта метка соответствия ссылается на элемент управления OLE в операторах программы.

2. Выберите атрибут установки размера из ниспадающего списка Sizing Mode (режим изменения размера). Форматер окна добавляет данный атрибут к объявлению OLE.

Default (по умолчанию) Не добавляет атрибут изменения размера. Режим по умолчанию - увеличение (zoom).

Clip (отсекать) Объявление OLE получает атрибут CLIP. Объект OLE показывает только то, что умещается в площади, определенной атрибутом AT элемента управления контейнера OLE. Если объект больше элемента управления, показывается только его верхний левый угол.

Stretch (растянуть) Объявление OLE получает атрибут STRETCH. Объект OLE растягивается до полного заполнения площади, определенной атрибутом AT элемента управления контейнера OLE. Соотношение размеров объекта теряется.

AutoSize (автоматическое приобретение размера) Объявление OLE получает атрибут AUTOSIZE. Объект OLE автоматически меняет свой размер когда во время работы меняется атрибут AT элемента управления контейнера OLE.

Zoom (увеличивать) Объявление OLE получает атрибут ZOOM. Объект OLE растягивается до полного заполнения площади, определенной атрибутом AT элемента управления контейнера OLE. Отношения размеров объекта сохраняются.

3. Отметьте поле Compatible Mode (совместимый режим) для того, чтобы установить режим совместимости 1 для объектов, которые его требуют (таких, как редактор растра Windows). Очистите это поле для того, чтобы присвоить для режима совместимости по умолчанию 0.

4. Используйте группу элементов управления Control Type (тип элемента управления) для того, чтобы установить тип объекта для объявления элемента управления OLE. Выберите одну из трех радиокнопок: Ole, Document или OCX.

OleСписок Object Type (тип объекта) содержит зарегистрированные OLE объекты. Объявление OLE получает атрибут CREATE если не назначено никакого Storage File (запомнить файл) или атрибут OPEN если Storage File назначено.

Object Type (тип объекта) Выберите из списка зарегистрированных OLE объектов, таких как электронные таблицы Excel, документы Word, слайды PowerPoint и т.д., для того, чтобы СОЗДАТЬ (CREATE) или OPEN (ОТКРЫТЬ).

**Совет: Когда вы выбираете сервер OLE, сервер автоматически загружен Форматером окна. Это может требовать определенного времени в течение процесса конструирования данного окна. Сначала сконструируйте и нарисуйте свое окно, а уж затем назначьте свой элемент управления OLE или установите сервер во время работы с помощью синтаксиса свойств вместо того, чтобы делать это во время конструирования.**

Storage File (запомнить файл)      Устанавливает имя файла для запоминания состава OLE и объект внутри него для открывания (OPEN). Отделите имя файла от имени объекта наклонной чертой влево и восклицательным знаком: FileName\!ObjectName.

Генератор приложений предоставляет величину по умолчанию для этого поля тогда, когда вы используете среду разработки для создания Файла запоминания состава. Смотрите Меню Форматера окна - всплывающее меню - Открыть.

Сервер OLE может получить доступ к объекту и манипулировать им в файле запоминания, но только через ваше приложение. Сервер OLE не может получить доступ к файлу запоминания независимо от вашего приложения. Пользуйтесь файлом запоминания когда вам понадобится ограничить доступ вашего пользователя к данному документу.

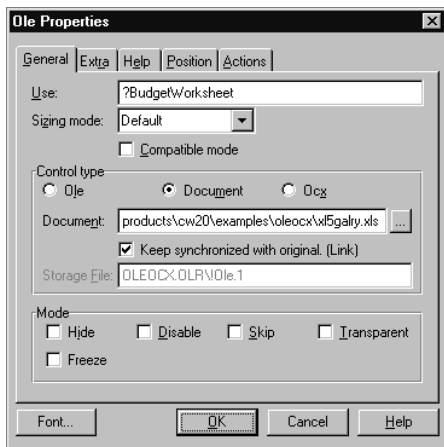
**Совет: Щелкните правой клавишей мыши на элементе управления OLE в Форматере окна и выберите Open (открыть) из всплывающего меню для активации назначенного OLE сервера в режиме открывания. Это дает вам возможность построить данный объект независимо от вашего приложения, кроме того это автоматически назначает по умолчанию filename\!objectname в поле файла запоминания. Когда файл запоминания назначен, объект скорее ОТКРЫТ, а не СОЗДАН.**

Когда данный объект открыт, сохраненная версия свойств контейнера OLE перезагружается, так что нет необходимости назначать свойства для ОТКРЫТЫХ объектов.

Если это поле пустое, OLE объект скорее СОЗДАН (CREATed) чем ОТКРЫТ (OPENed)

Document (документ)      Поле ввода Document (документ) заменяет список Object Type (тип объекта) и появляется поле Keep synchronized with original (поддерживайте синхронизированным с оригиналом). Объявление OLE получает атрибут DOCUMENT или LINK.

Наберите полное имя документного файла или нажмите эллиптическую кнопку (...) для того, чтобы выбрать файл из стандартного диалогового окна. Документный файл - это файл,



который ассоциирован с назначенным OLE сервером, так что приложение может активировать этот сервер во время работы (например, MYBUDGET.XLS ассоциирован с Excel).

Если это имя файла не полное, приложение ищет его в текущей директории. Документный файл должен быть установлен на машине конечного пользователя в назначенном каталоге.

Keep synchronized with original (Link) (поддерживайте синхронизированным с оригиналом (связанным)). Отметьте это поле для того, чтобы придать объекту OLE связь. Это генерирует атрибут LINK (смотрите Справочник языка) для данного элемента управления OLE, который сообщает серверу о необходимости обновления первоначального файла любыми изменениями, сделанными через ваше приложение.

Очистите это поле для того, чтобы придать OLE объекту встраивание. Это генерирует атрибут DOCUMENT (смотрите Справочник языка) для данного элемента управления OLE, который командует серверу не обновлять первоначальный файл изменениями, сделанными через ваше приложение.

Действие, назначенное по умолчанию, DoVerb ({PROP:DoVerb}=0), может зависеть от того, связан ли объект или встроен.

**Storage File (файл запоминания)** Назначает имя файла запоминания состава OLE и объекта внутри, который необходимо открыть (OPEN). Отделите имя файла от имени объекта наклонной чертой влево и восклицательным знаком: FileName\!ObjectName.

Файл запоминания может быть назначен для встроенных документов (DOCUMENT), но не для связанных документов (LINK).

Генератор приложений предоставляет величину по умолчанию для этого поля тогда, когда вы используете среду разработки для создания Файла запоминания состава. Смотрите Меню Форматера окна - всплывающее меню - Открыть.

Сервер OLE может получить доступ к объекту и манипулировать им в файле запоминания, но только через ваше приложение. Сервер OLE не может получить доступ к файлу запоминания независимо от вашего приложения. Пользуйтесь файлом запоминания

когда вам понадобится ограничить доступ вашего пользователя к данному документу.

**Совет:** Щелкните правой клавишей мыши на элементе управления OLE в Форматере окна и выберите Open (открыть) из всплывающего меню для активации назначенного OLE сервера в режиме открывания. Это дает вам возможность построить данный объект независимо от вашего приложения, кроме того это автоматически назначает по умолчанию filename\!objectname в поле файла запоминания. Когда файл запоминания назначен, объект скорее ОТКРЫТ, а не СОЗДАН.

Когда данный объект открыт, сохраненная версия свойств контейнера OLE перезагружается, так что нет необходимости назначать свойства для ОТКРЫТЫХ объектов.

Если это поле пустое, OLE объект скорее СОЗДАН (CREATed) чем ОТКРЫТ (OPENed).  
ОСХСписок Object Type (тип объекта) содержит зарегистрированные ОСХ объекты.

Объявление OLE получает атрибут CREATE или атрибут OPEN.

Object Type (тип объекта) Выберите из списка зарегистрированных ОСХ объектов для того, чтобы СОЗДАТЬ (CREATE) или ОТКРЫТЬ (OPEN).

Storage File (файл запоминания) Назначает имя файла запоминания состава OLE и объекта внутри, который необходимо открыть (OPEN). Отделите имя файла от имени объекта наклонной чертой влево и восклицательным знаком: FileName\!ObjectName.

**Совет:** Щелкните правой клавишей мыши на элементе управления OLE/ОСХ в Форматере окна и выберите из всплывающего меню Custom (покупной) для активации диалогового окна Custom Properties для ОСХ. Это запишет все изменения свойств и добавит их в качестве атрибутов к оператору объявления элемента управления OLE.

Если назначено свойство, которое не может быть представлено в текстовой форме, создается файл запоминания для запоминания данного свойства. Генератор приложений назначает по умолчанию FileName\!ObjectName в поле файла запоминания.

**Совет:** Вы можете создать файл запоминания для ОСХ во время работы, выдав

**?ОСХControl{PROP:SaveAs}='FileName\!ObjectName'**

пока ОСХ активен в вашей программе. Таким способом вы можете снова открыть ОСХ в том состоянии, в котором пользователь оставил его!

Когда данный объект открыт, сохраненная версия свойств контейнера OLE перезагружается, так что нет необходимости назначать свойства для ОТКРЫТЫХ объектов.

Если это поле пустое, ОСХ объект скорее СОЗДАН (CREATed) чем ОТКРЫТ (OPENed).

### **Файл запоминания Связанного или Встроенного документа (Storage File v. Linked or Embedded Document)**

Назначение файла запоминания (OPEN('Filename\!ObjectName')) очень напоминает назначение связанного или встроенного документа (LINK('Filename')). В обоих случаях объектом OLE является связь с внешним файлом. Однако имеются некоторые значительны различия:

- Сервер OLE может работать со связанным или встроенным документным файлом независимо от вашего приложения, но он не может делать этого с файлом запоминания.
- Файл запоминания может содержать многие объекты, например, две электронные таблицы или электронную таблицу и документ word.
- Свойства контейнера OLE сохранены в (и восстанавливаются из) файле запоминания, но не в связанном или встроенном документе.

5. Варианты выбора режима (Mode): смотрите выше Общие свойства элементов управления - Установка режимов элементов управления.

### **Закладка “Дополнительные возможности” (Extra)**

6. По желанию установите цвета для элемента управления OLE.

Наберите допустимую величину соответствия цвета или нажмите эллиптическую кнопку для того, чтобы выбрать цвет . Появится стандартный диалог Color. Смотрите Общие атрибуты элементов управления - Диалоговое окно цвета.

### **Закладка “Помощь” (Help)**

Смотрите Общие атрибуты элементов управления - Установка атрибута Help.

### **Закладка “Позиция” (Position Properties)**

Смотрите Общие атрибуты элементов управления - Установка атрибута AT.

## **Размещение Электронной таблицы/Графика/Документа в вашем приложении (Placing a Spreadsheet/Graph/Document in Your Application)**

Предположим, что у вас имеется одиночный рабочий лист бюджета или может быть документ шаблона слияния, к которым ваш пользователь хочет получить доступ прямо из вашего Clarion приложения. Или может быть вы хотите, чтобы ваш пользователь имел ограниченный доступ к рабочим листам, графику и т.п. Вы можете обеспечить этот доступ с помощью элемента управления OLE следующим образом.

### **Поместите элемент управления OLE**

1. Находясь в Форматере окна, выберите Control > OLE/OCX Control из меню.
2. Перемещайте курсор поперек создаваемого окна и щелкните клавишей мыши. На образце окна появится элемент управления OLE.

### **Захват двойного щелчка на элементе управления OLE (Trap a Double-click on the OLE Control)**

Выполните эти шаги для того, чтобы добавить ALERT(MouseLeft2) к объявлению элемента управления OLE. Это генерирует EVENT:AlertKey в тот момент, когда пользователь дважды щелкнет клавишей мыши на элементе управления и тем самым активирует сервер OLE.

1. Находясь в Форматере окна, щелкните правой клавишей мыши на элементе управления OLE и выберите Properties (свойства) из всплывающего меню.
2. В диалоговом окне Alert Keys нажмите кнопку Add (добавить).
3. В диалоговом окне Input Key (клавиша ввода) отберите Mouse-Left Button (левая клавиша мыши) и Double Click (двойной щелчок).
4. Нажмите кнопку ОК для того, чтобы закрыть оба диалоговых окна.

### **Установить свойства элемента управления OLE (Set the OLE Control's Properties)**

1. В Форматере окна щелкните правой клавишей мыши на элементе управления OLE и выберите Properties (свойства) из всплывающего меню.
2. В поле Use наберите метку соответствия, такую как ?BudgetWorksheet.
3. Отберите радиокнопку Document (документ).  
Появляется поле ввода Document (документ), а также появляется поле флажков Keep



synchronized with original (поддерживайте синхронизированным с оригиналом).

4. Нажмите эллиптическую кнопку (...) около поля ввода Document (документ) для того, чтобы отобразить для доступа рабочий бюджетный лист.

5. Отметьте поле Keep synchronized with original (поддерживайте синхронизированным с оригиналом).

Отметьте это поле для того, чтобы придать объекту OLE связь. Это генерирует атрибут LINK (смотрите Справочник языка) для элемента управления OLE, который командует серверу обновить первоначальный файл любыми изменениями, сделанными через ваше приложение.

**Совет: Отберите закладку Дополнительных возможностей (Extra) и назначьте цвет фона для элемента управления OLE. Это делает границы данного элемента управления более видимыми, а также делает элемент управления выглядящим как действующий, а не пассивный.**

6. Нажмите кнопку ОК для того, чтобы закрыть диалоговое окно.

### Встраиваемая программа для активации OLE сервера (Embed Code to Activate the OLE Server)

1. Щелкните правой клавишей мыши на элементе управления OLE, затем выберите Embeds (вставки) для того, чтобы получить доступ к точкам вставки для данного элемента управления.

2. Вставьте нижеследующий код (SOURCE) в точку вставки Взаимодействие с событием элемента управления, после генерированного кода -AlertKey (Смотрите Генератор приложений - Встроенная исходная программа).

IF KEYCODE( ) = MouseLeft2 !Если клавишей оповещения был двойной щелчок  
?BudgetWorksheet{PROP:DoVerb)=0 !активировать OLE сервер

Ознакомьтесь с вашей документацией сервера относительно реального эффекта различных величин DoVerb. Нуль (0) обычно устанавливает действие сервера по умолчанию, которое может быть активацией на месте. Активация на месте означает, что сервер сливает свои меню и панели инструментов в активном окне и принимает область, определенную атрибутом AT элемента управления OLE. Минус два (-2) обычно устанавливает активацию в режиме открывания, что означает, что сервер открывается в своем собственном отдельном окне.

3. Нажмите кнопку Close (закрыть) для того, чтобы закрыть диалоговое окно Embedded Source (встроенная исходная программа).

### Добавить позицию меню выход к элементу управления OLE (Add an Exit Menu Item to the OLE Control)

Мы воспользуемся этим меню для деактивации OLE сервера после того, как пользователь расстался с ним. Позиция Exit (выход) меню важна для серверов, активируемых на месте, так как это самый простой способ решить, когда пользователь расстался с сервером; однако этого не требуется для серверов, активируемых в режиме открывания.

1. При отобранном элементе управления OLE выберите Menu > New Menu для того, чтобы открыть редактор меню Menu Editor.

Смотрите главу Меню и панели инструментов.

2. Нажмите кнопку Item (позиция) для того, чтобы добавить новую позицию меню.

3. В поле Menu Text (текст меню) наберите E&xit.

4. В ниспадающем списке Position (позиция) отберите First (первый).

Во время активации на месте меню элемента управления OLE сливается с меню OLE сервера. Выбор Exit (выход) более видим для пользователя в первой (крайней левой) позиции. Во время активации в режиме открывания OLE сервер игнорирует меню элемента управления OLE, однако OLE сервер обеспечивает его обычную способность выхода, и таким образом мы можем деактивировать сервер каждый раз, когда наше окно получает фокус.

### Встроенный код для деактивации OLE сервера (Embed Code to Deactivate the OLE Server)

1. На закладке Actions (действия) нажмите кнопку Embeds (вставки) для того, чтобы получить доступ к точкам вставки для данной позиции меню.

2. Вставьте нижеследующий код SOURCE в точку вставки Взаимодействие с событием элемента управления, после генерированного кода -AlertKey (Смотрите Генератор приложений - Встроенная исходная программа).

?BudgetWorksheet{PROP:Deactivate} !Деактивировать OLE сервер

**Совет: Если вы активируете сервер в режиме открывания (в его собственном отдельном окне), встройте вместо этого этот код в точку вставки Window Event Handling (взаимодействие с событием окна) после generated code-GainFocus (генерированный код-получить фокус). Это деактивирует OLE сервер независимо от того, каким способом пользователь выходит из него!**

3. Нажмите кнопку Close (закрыть) для того, чтобы закрыть диалоговое окно Embedded Source (встроенная исходная программа) и Редактор меню.

## **Размещение в каждой записи электронной таблицы/Графика/ Документа (Placing a Spreadsheet/Graph/Document in Each Record)**

Теперь давайте предположим, что вы хотите записать в память объект OLE, такой как рабочий лист инвестиционных целей, для каждой записи в файле. Вы можете обеспечить такой доступ с помощью элемента управления OLE следующим образом.

### **Добавить поле или BLOB к вашему файлу (Add a Field or BLOB to Your File)**

Вы должны обеспечить место для сохранения объекта для каждой записи. Если вы используете файловую систему TopSpeed, вы можете сохранить каждый из объектов внутри вашего .TPS файла в качестве BLOB. Или же вы можете сохранить каждый объект в файле запоминания состава и сохранить имя файла в каждой записи. Оба метода иллюстрируются ниже.

1. Откройте ваш словарь данных и отберите файл, которому необходим объект для каждой записи.

2. Нажмите кнопку Fields/Keys (поля/ключи) для того, чтобы открыть диалоговое окно Field/Key Definition (поле/ ключа определение ).

3. Нажмите кнопку Insert для того, чтобы открыть диалоговое окно New Field Properties ( свойства нового поля).

Чтобы использовать BLOB

4. В поле Field Name (имя поля) наберите подходящее имя, такое как InvestmentGoals (ЦелиИнвестиций).

5. В ниспадающем списке Data Type (тип данных) отберите BLOB.

Поле флажков Binary (бинарный) (атрибут BINARY) не оказывает эффекта на поведение BLOB.

6. Нажмите кнопку ОК.

Чтобы использовать Файл запоминания состава

4. В поле Field Name (имя поля) наберите подходящее имя, такое как InvestmentGoalsStoregeFile (ФайлЗапоминанияЦелиИнвестиций).

5. В ниспадающем списке Data Type (тип данных) отберите STRING (СТРОКА СИМВОЛОВ).

6. В поле Characters (символы) установите подходящую длину.

**Совет: Описание Файла запоминания состава принимает форму Filename\!ObjectName. Имя файла может быть названием с полным путем и может включать расширение файла. ObjectName (имя объекта) может такой длины, как вам хочется. Двадцати пяти символов, по-видимому, достаточно, если не устанавливается путь. Добавьте от двадцати до сорока символов для включения пути.**

7. На закладке Options (опции) отметьте поле Do Not Auto-Populate This Field (не автогенерировать это поле).

Это скажет соответствующему Мастеру процедур игнорировать данное поле при построении Browsers (просмотров), Forms (форм) и Reports (отчетов). Имеется вероятность, что вы генерируете это имя в вашей программе, так что пользователю никогда не нужно будет его знать.

8. Нажмите кнопку ОК.

### **Поместите элемент управления OLE (Place the OLE Control)**

1. Используйте шаблон Мастера Поля Просмотра (Browse Wizard) или Мастера Формы (Form Wizard) для генерации процедуры обновления для вашего файла.

Смотрите главу Мастера и Шаблоны процедур.

2. Находясь в Window Formatter (Форматере окна), выберите Control > OLE/OCX Control из меню.

3. Перемещайте курсор над создаваемым окном и одновременно щелкните клавишей мыши.

На образце окна появится элемент управления OLE.

### **Захват двойного щелчка на элементе управления OLE (Trap a Double-click on the OLE Control)**

Выполните эти шаги для того, чтобы добавить ALRT(MouseLeft2) к объявлению элемента управления OLE. Это генерирует EVENT:AlertKey в тот момент, когда пользователь дважды щелкнет клавишей мыши в элементе управления и тем самым активирует сервер OLE.

1. Находясь в Форматере окна, щелкните правой клавишей мыши на элементе управления OLE и выберите Properties (свойства) из всплывающего меню.

2. В диалоговом окне Alert Keys (клавиши оповещения) нажмите кнопку Add (добавить).

3. В диалоговом окне Input Key (клавиша ввода) отберите Mouse-Left Button (левая клавиша мыши) и Double Click (двойной щелчок).

4. Нажмите кнопку ОК для того, чтобы закрыть оба диалоговых окна.

### **Установить свойства элемента управления OLE (Set the OLE Control's Properties)**

1. В Форматере окна щелкните правой клавишей мыши на элементе управления OLE и выберите Properties (свойства) из всплывающего меню.

2. В поле Use наберите метку соответствия, такую как ?GoalsWorksheet.

**Совет: Отберите закладку Дополнительных возможностей (Extra) и назначьте цвет фона для элемента управления OLE. Это делает границы данного элемента управления более видимыми, а также делает элемент управления выглядящим как действующий, а не пассивный.**

3. Нажмите кнопку ОК для того, чтобы закрыть диалоговое окно.

### **Встраиваемая программа для активации OLE сервера (Embed Code to Activate the OLE Server)**

1. Щелкните правой клавишей мыши на элементе управления OLE, затем выберите Embeds (вставки) для того, чтобы получить доступ к точкам вставки для данного элемента управления.

2. Вставьте нижеследующий код (SOURCE) в точку вставки Взаимодействие с событием элемента управления, после генерированного кода -AlertKey (Смотрите Генератор приложений - Встроенная исходная программа).

IF KEYCODE( ) = MouseLeft2 !Если ключом оповещения был двойной щелчок  
?InvestmentGoals{PROP:DoVerb}=0 !активировать OLE сервер

Ознакомьтесь с вашей документацией сервера относительно реального эффекта различных величин DoVerb. Нуль (0) обычно устанавливает действие сервера по умолчанию, которое может быть активацией на месте. Активация на месте означает, что сервер сливает свои меню и панели инструментов в активном окне и принимает область, определенную атрибутом AT элемента управления OLE. Минус два (-2) обычно устанавливает активацию в режиме открывания, что означает, что сервер открывается в своем собственном отдельном окне.

3. Нажмите кнопку Close (закрыть) для того, чтобы закрыть диалоговое окно Embedded Source (встроенная исходная программа).

### **Добавить позицию меню выход к элементу управления OLE (Add an Exit Menu Item to the OLE Control)**

Мы воспользуемся этим меню для деактивации OLE сервера после того, как пользователь расстался с ним. Позиция Exit (выход) меню важна для серверов, активируемых на месте, так как это самый простой способ решить, когда пользователь расстался с сервером; однако этого не требуется для серверов, активируемых в режиме открывания.

1. При отображенном элементе управления OLE выберите Menu > New Menu для того, чтобы открыть редактор меню Menu Editor.

Смотрите главу Меню и поля инструментов.

2. Нажмите кнопку Item (позиция) для того, чтобы добавить новую позицию меню.

3. В поле Menu Text (текст меню) наберите E&xit.

4. В выпадающем списке Position (позиция) отберите First (первый).

Во время активации на месте меню элемента управления OLE сливается с меню OLE сервера. Выбор Exit (выход) более видим для пользователя в первой (крайней левой) позиции. Во время активации в режиме открывания OLE сервер игнорирует меню элемента управления OLE, однако OLE сервер обеспечивает его обычную способность выхода, и таким образом мы можем деактивировать сервер каждый раз, когда наше окно получает фокус.

Встроенный код для деактивации OLE сервера (Embed Code to Deactivate the OLE Server)

1. На закладке Actions (действия) нажмите кнопку Embeds (вставки) для того, чтобы получить доступ к точкам вставки для данной позиции меню.

2. Вставьте нижеследующий код (SOURCE) в точку вставки Взаимодействие с событием элемента управления, после генерированного кода -Accepted (принято) точка вставки. (Смотрите Генератор приложений - Встроенная исходная программа).

Чтобы сохранить как BLOB

```
?InvestmentGoals{PROP:Deactivate}    !Деактивировать OLE сервер  
CUS: InvestmentGoals{PROP:Handle}= ? InvestmentGoals{PROP:Blob}  
!Присвоить BLOB
```

Чтобы сохранить в Файле запоминания состава

```
?InvestmentGoals{PROP:Deactivate}      !Деактивировать OLE сервер
CUS: InvestmentGoalsStorageFile = |      !Постройте имя файла
'CUS_'&FORMAT(CUS:ID,@N04)&'&.OLR\!Profile'
?InvestmentGoals{PROP:SaveAs} + |        !Затем сохраните его.
CUS: InvestmentGoalsStorageFile
```

**Совет: Если вы активируете сервер в режиме открывания (в его собственном отдельном окне), встройте вместо этого этот код в точку вставки Window Event Handling (взаимодействие с событием окна) после generated code-GainFocus (генерированный код-получить фокус). Это деактивирует OLE сервер независимо от того, каким способом пользователь выходит из него!**

3. Нажмите кнопку Close (закрыть) для того, чтобы закрыть диалоговое окно Embedded Source (встроенная исходная программа) и Редактор меню.

### **Встроенный код для поиска OLE объекта (Embed Code to Retrieve the OLE Object)**

Данный шаг отыскивает OLE объект в BLOB(Большой бинарный объект) или Файле запоминания состава до того, как покажется окно Form, и до того, как будет активирован сервер. Если не существует никакого объекта, создается новый объект.

1. Щелкните правой клавишей мыши на строке заголовка образца окна, затем выберите Embeds (вставки), чтобы получить доступ ко всем встроенным точкам для этой процедуры.

2. Вставьте нижеследующий код SOURCE в точку вставки Взаимодействие с событием элемента управления, после генерированного кода -Accepted (принято) . (Смотрите Генератор приложений - Встроенная исходная программа).

Чтобы найти из BLOB:

(новые записи получают новую пустую электронную таблицу)

```
IF CUS:Profile{PROP:Handle} > 0! Если BLOB существует
?Profile{PROP:Blob} = CUS:Profile{PROP:Handle}      ! найти
ELSE          ! иначе
? Profile{PROP>Create} = 'Excel . Sheet . 5'         ! новая электронная таблица
END
```

или (новые записи получают электронную таблицу шаблона из файла запоминания)

```
IF CUS:Profile{PROP:Handle} > 0! Если BLOB существует
```

```

    ?Profile{PROP:Blob} = CUS:Profile{PROP:Handle}    ! найти
ELSE          ! иначе
    ? Profile{PROP:Open} = 'InvGoals.OLR\!Template'    !открыть файл запоминания
END

```

или (новые записи получают электронную таблицу шаблона из файла Excel)

```

IF CUS:Profile{PROP:Handle} > 0! Если BLOB существует
    ?Profile{PROP:Blob} = CUS:Profile{PROP:Handle}    ! найти
ELSE          ! иначе
    ? Profile{PROP:Document } = 'InvGoals.XLS'    !открыть файл Excel
END

```

Чтобы найти из Файла запоминания состава:

(новые записи получают новую пустую электронную таблицу)

```

IF CUS:StorageFile! Если файл запоминания существует
    ?Profile{PROP:Open } = CUS: StorageFile    !открыть его
ELSE! иначе
    ? Profile{PROP:Create} = 'Excel . Sheet . 5'! новая электронная таблица
END

```

или (новые записи получают электронную таблицу шаблона из файла запоминания)

```

IF CUS:StorageFile    ! Если файл запоминания существует
    ?Profile{PROP:Open } = CUS: StorageFile    !открыть его
ELSE          ! иначе
    ? Profile{PROP: Open } = 'InvGoals.OLR\!Template'! открыть файл запоминания
END

```

или (новые записи получают электронную таблицу шаблона из файла Excel)

```

IF CUS:StorageFile! Если файл запоминания существует
    ?Profile{PROP:Open } = CUS: StorageFile    !открыть его
ELSE          ! иначе
    ? Profile{PROP: Document} = 'InvGoals.XLS'    ! открыть файл Excel
END

```



## **Элементы управления OLE с OCX (OLE Controls with OCXs)**

Вы должны зарегистрировать ваши OCX с вашей операционной системой Windows прежде, чем вы сможете их использовать. Познакомьтесь с OCX документацией, где имеются инструкции по регистрации OCX.

**Совет: Некоторые программы инсталляции OCX регистрируют OCX автоматически при их инсталляции.**

В общем случае ваша программа общается с OCX с помощью приписывания свойств и функций обратного вызова, которые, помимо прочего, захватывают OCX события. Следовательно, вы должны быть знакомы со свойствами OCX, событиями действиями операциями. Ознакомьтесь с информацией об этом в вашей документации.

Библиотека поддержки Clarion вызывает функции обратного вызова во всех случаях, когда необходимо передать информацию, касающуюся OCX. Вы должны регистрировать ваши функции обратного вызова перед тем, как библиотека поддержки может вызвать их. Вы можете, если это необходимо, разрегистрировать функции. Clarion предоставляет процедуры, предназначенные для этого, как показано ниже и в Покупных элементах управления OLE(OCX) в Справочнике языка.

### **Использование OCX с глобальными функциями обратного вызова (Using an OCX with Global Callback Functions)**

---

Следующие примеры показывают, как поддерживать OCX с помощью глобальных функций обратного вызова. При использовании этой техники функции обратного вызова могут быть употреблены для поддержки более чем одного OCX внутри вашего приложения.

#### **Глобальные данные (Global Data)**

Во-первых, мы установим структуру глобальных данных, STRING, с помощью которой надо передать информацию от функций обратного вызова к нашим процедурам.

1. В диалоговом окне Дерева приложения нажмите кнопку Global (глобальный).
2. В диалоговом окне Global Properties (глобальные свойства) нажмите кнопку Data (данные).
3. В диалоговом окне Global Data (глобальные данные) нажмите кнопку Insert (вставить) для того, чтобы открыть диалоговое окно Field Properties (свойства поля).
4. В поле Field Name (имя поля) наберите CallBackData (обратный вызов данных).

5. В поле Characters (символы) наберите 500, а затем нажмите ОК.

Это объявляет глобальное поле, которое мы будем использовать для того, чтобы послать информацию из функций обратного вызова к процедурам нашего приложения. Вы можете предпочесть пользоваться очередью QUEUE вместо STRING для того, чтобы запоминать и отыскивать множественные позиции.

6. Нажмите Cancel, затем нажмите Close для того, чтобы закрыть диалоговое окно Global Data (глобальные данные).

### **Глобальные вставки (Global Embeds)**

Затем, мы включим некоторые прототипы и EQUATES (соответствия) для поддержки встроенных OCX функций поддержки Clarion.

1. В диалоговом окне Global Properties (глобальные свойства) нажмите кнопку Embeds (вставки) и встройте следующую исходную программу в перечисленные глобальные точки вставки.

2. Точка вставки Inside the Global Map (внутри глобальной карты):  
INCLUDE('OCX.CLW') ! OCX процедура и прототипы функций

3. Точка вставки Before File Declarations (перед объявлениями файла):

INCLUDE('OCXEVENT.CLW') !соответствия для обычных OCX событий  
OCXEvent EQUATE(400h) !определяемое пользователем OCX событие

### **Создать функции обратного вызова (Create the Callback Functions)**

Теперь мы используем шаблон Source (исходная программа) для создания функций обратного вызова.

1. В диалоговом окне Дерево приложения выберите Procedure > New.

2. В диалоговом окне New Procedure (новая процедура) наберите EventFunc, затем нажмите ОК.

EventFunc - это имя функции обратного вызова, которая управляет событиями OCX.

3. В диалоговом окне Select Procedure Type (выбор типа процедуры) выберите Source (исходная программа).

Появится диалоговое окно Procedure Properties (свойства процедуры).

4. В поле Description (описание) наберите Callback function to support OCX events

(функция обратного вызова для поддержки событий ОСХ).

5. В поле Prototype (прототип) наберите (\*SHORT Reference,SIGNED OleControl, LONG CurrentEvent), LONG.

Прототип (тип данных, названия параметров и тип возвратных данных) автоматически включается в глобальную карту MAP вашей программы.

6. В поле Parameters (параметры) наберите (\*SHORT Reference,SIGNED OleControl, LONG CurrentEvent).

Названия параметров (и тип данных), которые вы вводите здесь, добавляются к оператору объявления FUNCTION.

7. Нажмите кнопку Embed (встроить) и встройте следующую исходную программу в перечисленные точки вставки.

8. Точка вставки Data Section (сегмент данных):

```
Count          LONG
EventInfo      CSTRING(200)
Parm           CSTRING(30)
```

9. Точка вставки Processed Code (обрабатываемый код):

```
IF CurrentEvent = OCXEVENT:MouseMove    !пропустить событие движения мыши
ELSE
    EventInfo = OleControl{PROP:LastEventName}
    LOOP Count = 1 TO OCXGETPARAMCOUNT(Reference)
        Parm = OCXGETPARAM(Reference,Count-1)    !Получает величину каждого
параметра
        EventInfo = CLIP(EventInfo)&'/'&Parm    ! и конкатенируйте их
    END
    CallBackData = EventInfo    !приписать событие info глобальной переменной,
!так чтобы другие процедуры могли иметь доступ к ней.
    POST(OCXEvent,OleCntrol) !послать событие к активной процедуре для
!обработки
END
RETURN(True)!сказать окну, что мы все сделали.
```

**Совет:** Ваши функции обратного вызова должны отвечать на вызовы OS по возможности быстрее. Следовательно, захватите любую необходимую информацию в функцию обратного вызова и передайте ее вашей ОСХ процедуре драйвера путем присваивания ее структуре глобальных данных, затем отправьте назначенное пользователем событие для того, чтобы сказать процедуре драйвера ОСХ что-нибудь сделать.

Теперь повторите сделанные выше шаги для создания другой функции и процедуры следующим образом.

1. В диалоговом окне Дерево приложения выберите Procedure > New.
2. В диалоговом окне New Procedure (новая процедура) наберите PropEdit, затем нажмите OK.
3. В диалоговом окне Select Procedure Type (выбор типа процедуры) выберите Source (исходная программа).  
Появится диалоговое окно Procedure Properties (свойства процедуры).
4. В поле Description (описание) наберите Callback function to support OCX Property Edits (функция обратного вызова для поддержки редактирования свойств OCX).
5. В поле Prototype (прототип) наберите (SIGNED OleControl,STRING CurrentProp),LONG.
6. В поле Parameters (параметры) наберите (SIGNED OleControl, STRING CurrentProp).
7. Нажмите кнопку Embeds (вставки) и встройте следующую исходную программу в точку вставки Processed Code. Настройте этот код для контроля, какие изменения свойств OCX разрешены в вашем приложении.

```
IF CurrentProp = 'FontSize"  
    RETURN(0)!отменить разрешение на изменение  
ELSE  
    RETURN(1)          !разрешить изменение
```

1. В диалоговом окне Дерево приложения выберите Procedure > New.
2. В диалоговом окне New Procedure (новая процедура) наберите PropChange, затем нажмите OK.
3. В диалоговом окне Select Procedure Type (выбор типа процедуры) выберите Source (исходная программа).  
Появится диалоговое окно Procedure Properties (свойства процедуры).
4. В поле Description (описание) наберите Callback function to support OCX Property Changes (функция обратного вызова для поддержки изменения свойств OCX).
5. В поле Prototype (прототип) наберите (SIGNED OleControl,STRING CurrentProp),LONG.

6. В поле Parameters (параметры) наберите (SIGNED OleControl, STRING CurrentProp).

7. Нажмите кнопку Embeds (вставки) и встройте следующую исходную программу в точку вставки Processed Code. Настройте этот код для принятия соответствующих действий, основанных на изменениях свойств OCX.

```
IF CurrentProp = 'FontSize"  
    OleControl{PROP:Autosize}=TRUE  
END
```

### **Поместить элемент управления OLE (Place the OLE Control)**

Теперь, когда наши функции обратного вызова завершены, давайте установим элемент управления OCX.

1. В Window Formatter (форматере окна) выберите Control > OLE/OCX Control из меню.
2. Перемещайте курсор над создаваемым окном и одновременно щелкните клавишей мыши.  
На образце окна появится элемент управления OLE.

**Совет: Убедитесь, что OCX и ваш проект оба 32-битовые или 16-битовые.**

### **Установить свойства элемента управления OLE (Set the OLE Control's Properties)**

1. В Window Formatter (форматере окна) щелкните правой клавишей мыши на элементе управления OLE и выберите Properties (свойства) из всплывающего меню.
2. В поле USE наберите метку соответствия для ссылки на данный элемент управления в вашей исходной программе.
3. Отберите радиокнопку OCX.  
Список Object Type (тип объекта) содержит зарегистрированные OCX.
4. В ниспадающем списке Object Type (тип объекта) отберите нужный вам для использования OCX.
5. Нажмите кнопку ОК для того, чтобы закрыть диалоговое окно.

### **Установить OCX свойства (Set the OCX's Properties)**

1. В Window Formatter (форматере окна) щелкните правой клавишей мыши на элементе управления OLE и выберите Custom (покупные) из всплывающего меню.

Это открывает диалоговое окно Свойства OCX. Посмотрите информацию об этом диалоговом окне в вашей OCX документации.

### **Добавьте при желании меню к элементу управления OLE (Optionally, Add a Menu to the OLE Control)**

Элементы управления OLE могут иметь меню точно так же, как их имеют окна. Вы можете, когда вам это необходимо, ассоциировать присвоения свойств OCX и другие действия с позициями OLE меню.

1. При отобранном элементе управления OLE выберите Menu > New Menu для того, чтобы открыть редактор меню.

Смотрите главу Меню и панели инструментов. Во время работы меню элемента управления OLE сливается с меню данного окна.

### **Встроить код для сообщения с OCX (Embed Code to Communicate with the OCX)**

Обычно ваша программа общается с OCX через присваивания свойства и функции обратного вызова. Смотрите OLE (.OCX) Покупные элементы управления в Справочнике языка, где имеется более полная информация об этих характеристиках.

### **Присваивание свойств (Property Assignments)**

Присваивание свойства OCX принимает следующую форму

FieldEquateLabel{'OCXProperty'}=assignevalue, где FieldEquateLabel - это метка соответствия данного элемента управления, OCXProperty - это метка свойства, а assignevalue является постоянной, переменной или выражением. Например:

```
?MyOCX{'StartDate'} = TODAY( )
```

Вы можете запросить OCX свойство перемещением OCXProperty направо от знака равенства. Например:

```
DateSelected = ?MyOCX{'SelectedDate'}
```

### **Функции обратного вызова (Callback Functions)**

Вам нужно встроить следующий код, или подобный ему код, в точки вставки, перечисленные ниже, для того, чтобы использовать глобальные функции обратного вызова для сообщения с вашим OCX.

1. Точка вставки After Opening the Window (после открывания окна):  
DO RegisterCallbackFunctions

2. Точка вставки Other Control Event Handling-?MyОСХ (ведение другого события управления - мой ОСХ):

```

IF EVENT( ) = OCXEvent      ! Обработать событие, посланное OCX-объектом
    ...
функцией обратного возврата
END

```

### 3. Точка вставки Procedure Routines (подпрограммы процедур):

```
RegisterCallbackFunctions ROUTINE
OCXREGISTEREVENTPROC(?MyOCX,EventFunc)
OCXREGISTERPROPCHANGE(?MyOCX,PropChange)
OCXREGISTERPROPEEDIT(?MyOCX,PropChangeEdit)
```

## Использование OCX с локальными функциями обратного вызова (Using an OCX with Local Callback Functions)

Вы можете захотеть выполнить некоторые локальные ОСХ для одиночной процедуры. Следующий пример иллюстрирует такое локальное ОСХ исполнение.

### Поместить элемент управления OLE (Place the OLE Control)

1. В Window Formatter (форматере окна) выберите Control > OLE/OCX Control из меню.

2. Перемещайте курсор над создаваемым окном и одновременно щелкните клавишей мыши.

На образце окна появится элемент управления OLE.

**Совет: Убедитесь, что ОСХ и ваш проект оба 32-битовые или 16-битовые.**

### Установить свойства элемента управления OLE (Set the OLE Control's Properties)

1. В Window Formatter (форматере окна) щелкните правой клавишей мыши на элементе управления OLE и выберите Properties (свойства) из всплывающего меню.

2. В поле USE наберите метку соответствия для ссылки на данный элемент управления в вашей исходной программе.

3. Отберите радиокнопку ОСХ.

Список Object Type (тип объекта) содержит зарегистрированные ОСХ.

4. В ниспадающем списке Object Type (тип объекта) отберите нужный вам для использования ОСХ.

5. Нажмите кнопку ОК для того, чтобы закрыть диалоговое окно.

### **Установить ОСХ свойства (Set the OCX's Properties)**

1. В Window Formatter (форматере окна) щелкните правой клавишей мыши на элементе управления OLE и выберите Custom (покупные) из всплывающего меню.

Это открывает диалоговое окно Свойства ОСХ. Посмотрите информацию об этом диалоговом окне в вашей ОСХ документации.

### **С помощью опций добавьте меню к элементу управления OLE (Optionally, Add a Menu to the OLE Control)**

Элементы управления OLE могут иметь меню точно так же, как их имеют окна. Вы можете, когда вам это необходимо, ассоциировать присвоение свойств ОСХ и другие действия с позициями OLE меню.

1. При отобранном элементе управления OLE выберите Menu > New Menu для того, чтобы открыть редактор меню.

Смотрите главу Меню и панели инструментов. Во время работы меню элемента управления OLE сливается с меню данного окна.

### **Встроить код для сообщения с ОСХ (Embed Code to Communicate with the OCX)**

Обычно ваша программа общается с ОСХ через присваивания свойства и функции обратного вызова. Смотрите OLE (.OCX) Покупные элементы управления в Справочнике языка, где имеется более полная информация об этих характеристиках.

### **Присваивание свойств (Property Assignments)**

Присваивание свойства ОСХ принимает следующую форму

1. FieldEquateLabel{'OCXProperty'}=assignevalue, где FieldEquateLabel - это метка соответствия данного элемента управления, OCXProperty - это метка свойства, а assignevalue является постоянной, переменной или выражением. Например:

?MyOCX{'StartDate'} = TODAY ( )

Вы можете запросить ОСХ свойство перемещением OCXProperty направо от знака равенства. Например:

DateSelected = ?MyOCX{'SelectedDate'}



### Функции обратного вызова (Callback Functions)

Вам нужно встроить следующий код, или подобный ему код, в точки вставки, перечисленные ниже, для того, чтобы использовать локальные функции обратного вызова для сообщения с вашим OCX.

1. Точка вставки Module Data Section (модульный сегмент данных) (в диалоговом окне Дерево приложения отберите закладку Module (модуль), затем щелкните правой клавишей на имени модуля, а затем выберите Embeds (вставки) из всплывающего меню):

MAP

INCLUDE('OCX.CLW') !Прототипы OCX процедуры и функции

!Прототип 3 функций обратного вызова:

EventFunc FUNCTION (\*SHORT Reference, SIGNED OleControl, LONG |  
CurrentEvent), LONG

PropChange PROCEDURE (SIGNED OleControl, STRING CurrentProp)

PropEdit FUNCTION (SIGNED OleControl, STRING CurrentProp), LONG  
END

INCLUDE('OCXEVENT.CLW') ! Соответствия для обычных OCX событий

CallBackData STRING(500) ! Данные меняют площадь для функций !обратного  
вызова

OCXEvent EQUATE(400h) !Определенное пользователем OCX событие

2. Точка вставки After Opening the Window (после открывания окна):

DO RegisterCallbackFunctions

3. Точка вставки Other Control Event Handling-?MyOCX (ведение другого события управления - мой OCX):

IF EVENT( ) = OCXEvent ! Обработать событие, посланное ! функцией  
обратного возврата  
END

4. Точка вставки Procedure Routines (подпрограммы процедур):

RegisterCallbackFunctions ROUTINE

OCXREGISTEREVENTPROC(?MyOCX, EventFunc)

OCXREGISTERPROPCHANGE(?MyOCX, PropChange)

OCXREGISTERPROPEdit(?MyOCX, PropEdit)

**Совет:** Так как ниже следующее - это процедуры и функции (не подпрограммы), вам следует встроить их после любых встроенных подпрограмм.

5. Точка вставки на конце Procedure Routines (подпрограмм процедур):

!Функция обратного вызова процессора события

EventFunc FUNCTION |

(\*SHORT Reference,SIGNED OleControl,LONG CurrentEvent)

Count LONG

EventInfo CSTRING(200)

Parm CSTRING(30)

CODE

IF CurrentEvent = OCXEVENT:MouseMove!пропустить события перемещ. мыши

ELSE

EventInfo = OleControl{PROP:LastEventName}

LOOP Count = 1 TO OCXGETPARAMCOUNT(Reference)

EventInfo = CLIP(EventInfo)&'/'&Parm !получить величину каждого параметра

END

CallBackData = EventInfo!сделать событие info доступным

!другим процедурам этого модуля

POST(OCXEvent,OleCntrol)!послать событи моей процедуре OCX для обработки.

END

RETURN(True) !сказать окнам, что мы все сделали.

**Совет: Ваши функции обратного вызова должны отвечать на вызовы OS по возможности быстрее. Следовательно, захватите любую необходимую информацию в функцию обратного вызова и передайте ее вашей OCX процедуре драйвера путем присваивания ее структуре глобальных данных, затем отправьте назначенное пользователем событие для того, чтобы сказать процедуре драйвера OCX что-нибудь сделать.**

!Свойство изменило функцию обратного вызова

PropChange PROCEDURE(SIGNED OleControl,STRING CurrentProp)

CODE

IF CurrentProp = 'FontSize'

OleControl{PROP:AutoSize}=TRUE

END

!функция обратного вызова редактирования свойства

PropEdit FUNCTION(SIGNED OleControl,STRING CurrentProp)

CODE

IF CurrentProp = 'FontSize'

RETURN(0) !отменить разрешение на изменение

ELSE

RETURN(1) !разрешить изменение

END

## Элементы управления VBX (VBX Controls)

Элементы управления VBX - это “добавляемые” элементы управления, продаваемые многими другими фирмами. Они выполняют очень широкий круг задач, от sliders и gauge элементов управления до Twain image capture (захват двойного изображения). Форматер окна позволяет вам разместить эти элементы управления в вашем окне сразу после того, как только вы “зарегистрируете” .VBX библиотеки.

Специфическим форматом элемента управления VBX, поддерживаемым Clarion’ом, является Microsoft Visual Basic формат элементов управления, обычно получающий расширение .VBX. Имеется одно важное ограничение:

- Clarion поддерживает .VBX свойства, совместимые с Microsoft Visual Basic 1.0. Элементы управления VBX., которые требуют VB 2.0 или выше, несовместимы.

Это касается других не-Visual Basic платформ, таких как Microsoft Foundation Classes v. 2.0. Самое большое различие между уровнями один и уровнем два или выше .VBX заключается в том, что последний завязан с механизмом управления базой данных MS Access, который переносится с Visual Basic 2.x и выше. Номер уровня относится к номеру версии VB.

**Совет: Если описание продавца для .VBX не заявляет, что элемент управления спроектирован специально для Visual Basic 1, вы можете немедленно идентифицировать данный .VBX как .VBX уровня два или выше, если они идентифицируют его, как управление “data bound” (ограниченное данными).**

Библиотеки элемента управления VBX обычно требуют лицензионного файла (\*.LIC), прежде чем вы сможете добавить элемент управления к вашему приложению. Продавец библиотеки предоставляет файл при покупке библиотеки. Когда вы распределяете приложение своим конечным пользователям, вы распределяете только файл. VBX, но не лицензионный файл.

Кроме того, когда вы передаете .VBX файл вашим конечным пользователям, следуйте инструкциям продавца библиотеки относительно того, куда помещать файл(ы) .VBX элементы управления.

### Регистрация ваших .VBX

Прежде, чем вы разместите заказной элемент управления VBX в окне, вы должны зарегистрировать файл .VBX, в котором он содержится. Чтобы сделать это:

1. Находясь в главном меню Clarion, выберите Setup > VBX Custom Control Registry.
2. Нажмите кнопку Add в поле диалогового окна VBX Custom Control Registry.
3. Перейдите к файлу .VBX и отберите его в диалоге Add Custom Control Open File и нажмите OK.

Некоторые продавцы .VBX инсталлируют свои .VBX в каталог \WINDOWS\SYSTEM, в то время как другие предпочитают отдельные каталоги. Когда вы инсталлируете свою .VBX библиотеку на ваш жесткий диск, сделайте пометку, куда вы его поместили, так, чтобы вы смогли его найти в диалоге Open File.

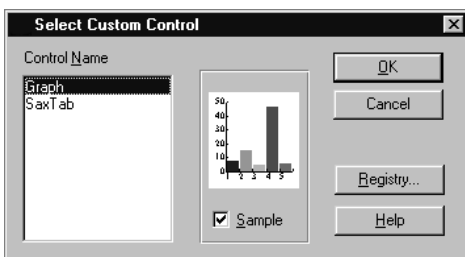
Для Clarion for Windows , .VBX должно быть в каталоге приложения или где-нибудь в системном списке.

4. Нажмите OK для закрытия диалогового окна VBX Custom Control Registry.

### Добавление к окну элементов управления VBX

1. Находясь в Форматере окна, вберите инструмент элемента управления VBX или выберите VBX Control (элемент управления VBX) из списка Control (элемент управления) и затем щелкните клавишей мыши в окне.

Появляется диалоговое окно Select Custom Control (выбор покупного элемента управления). Это диалоговое окно позволяет выбрать элементы управления из VBX Custom Control Registry (Регистр покупных элементов управления VBX). Выделите нужный вам элемент. Когда вы выделите элемент управления, если при этом поле Sample (образец) отмечено, диалоговое поле покажет копию элемента управления в его установках по умолчанию.



2. Нажмите кнопку OK чтобы возвратиться в форматер окна.

3. Щелкните правой клавишей мыши на элементе управления VBX и выберите Properties (свойства) из всплывающего меню.

### **Свойства VBX**

Диалоговое окно VBX Control Properties (свойства элемента управления VBX) содержит следующие приглашения.

1. Если необходимо, наберите метку для элемента управления в поле Text.

Если элемент управления поддерживает метку, она появится как часть этого элемента. На практике большинство элементов потребуют от вас установить метку заголовка, как одного из свойств элемента управления Visual Basic, объясняемое ниже.

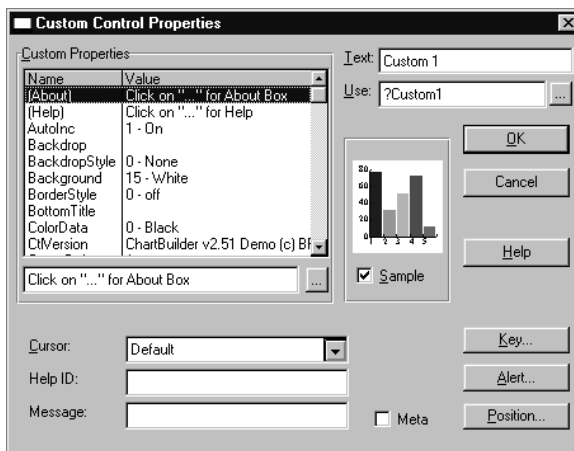
2. Наберите метку соответствия или имя переменной в поле Use.

Переменная номинально получит значение, вырабатываемое данным элементом управления. Если элемент управления принимает ввод пользователя, вы, скорее всего, получите величину, введенную пользователем, получив доступ к свойству элемента управления Visual Basic, описанному ниже.

.VBX также генерируют специфическое событие (EVENT:vbxevent). Событие представляет собой сообщение в виде строки символов, посланное от .VBX к Clarion приложению. Описание этого события описано ниже.

3. Находясь в поле ввода Custom Properties (покупные свойства), установите свойства запуска для элемента управления.

С левой стороны диалогового окна появляется список Custom Properties. Это окно показывает свойства элемента управления Visual Basic и его величины по умолчанию. Если вы введете стартовую величину в диалоговое окно, формater окна автоматически добавит ее к оператору языка Clarion, который размещает элемент в вашем окне.



Некоторые свойства VBX, имеющиеся для конкретного элемента управления.

Когда вы выделяете в списке какое-либо свойство элемента управления Visual Basic, под списком свойств в диалоговом окне появляется либо поле редактирования, либо ниспадающий список. Наберите величину или переменную в поле редактирования или выберите из ниспадающего списка.

Документация, поступающая от продавца библиотеки .VBX, должна описывать свойства элементов управления Visual Basic, которые вы можете установить. Почитайте о том, как изменить их в исполняемой программе, или о том, как изменить эти свойства во время работы и как отыскать ввод пользователя, выполненный из элемента управления VBX.

4. По желанию назначьте внешний вид курсора, идентификатора помощи и текста сообщения.

Смотрите Общие атрибуты элемента управления - Установка атрибута HELP.

5. По желанию назначьте атрибуты KEY и ALRT.

Смотрите Общие атрибуты элементов управления - Установка атрибута KEY и Установка атрибута ALRT.

6. По желанию нажмите кнопку Position (положение) для того, чтобы назначить атрибут AT, а также любые атрибуты режима.

Смотрите Общие атрибуты элементов управления - Установка атрибута AT и Установка атрибута режимов элементов управления.

7. По желанию отметьте поле Sample (образец) для того, чтобы показать данный элемент управления с установками, назначенными по умолчанию.

8. По желанию отметьте поле Meta для генерации метафайла Windows (.WMF) для отчетов.

При добавлении элемента управления .VBX к отчету это устанавливает, что процессор печати генерирует метафайл для представления данного элемента управления.

## **Работа элемента управления VBX**

---

Файл .VBX действует как самостоятельная внешняя библиотека. Когда приложение загружает ее в память, вы можете обмениваться информацией между приложением и заказным элементом управления с его свойствами. Свойства Visual Basic - это карта сообщений.

Свойства .VBX являются наиболее обычными средствами, с помощью которых не- Visual Basic приложение использует функциональные возможности VBX. Рассматривайте свойство как переменную, к которой имеют доступ и приложение и VBX.

Если и приложение и .VBX контролируют некоторое свойство, они могут использовать его для общения между собой. Когда величина свойства изменяется, это является сигналом, что что-то должно быть предпринято. Каждый .VBX имеет свои свойства. Вы узнаете, какие свойства имеются, если прочтете документацию продавца .VBX.

Например, .VBX имеет свойство, называемое 'CellColor' (цвет ячейки), которое указывает, какой цвет у фона ячейки сетки. Если приложение хочет знать, какой текущий цвет, оно отыскивает величину в свойстве, названном 'CellColor'. Обычно это работает и наоборот. Если приложение изменяет величину 'CellColor' с синего на красный, .VBX обновляет управление окна и изменяет цвет.

**Совет: Свойства элемента управления Visual Basic обычно документируются с точкой вначале. Отбросьте эту точку при обращении из приложения Clarion.**

### Элементы управления VBX и синтаксис свойств

В разделе, приведенном выше, отмечается, как установить стартовые свойства для элемента управления с помощью Формatera окна. В другой раз вы захотите поменять свойства в исполняемой программе во время работы и, конечно, отыскать величины после ввода пользователя.

□ Чтобы изменить свойства во время работы, используйте синтаксис свойств. Получите доступ к свойствам данного элемента управления Visual Basic путем ссылки на конкретное свойство в кавычках:

`?vbx{'VBProperty'}= value`

□ Для отыскания текущей величины свойства элемента управления Visual Basic снова используйте синтаксис выражения свойства следующим образом:

`value = ?vbx{'VBProperty'}`

### События .VBX

Кроме свойств, другой “канал”, через который .VBX “общается” с вашим приложением - это общение с помощью событий. .VBX может переключить событие, например, если пользователь дважды щелкнет мышью на нужной его части. Когда событие произошло, .VBX генерирует строку символов (до 255 символов), содержащую имя события. Документация продавца .VBX перечисляет возможные события, которые может генерировать элемент управления.

Ваше приложение может изучить событие и предпринять соответствующее действие, опрашивая PROP:VBX событие. Когда вы работаете с генератором приложений, вы помещаете программу, подобную приводимой ниже, либо в точку вставки, помеченную как “Control Event Handling, before generated code (управление событием элемента управления перед генерацией кода (VBXevent)) или “Control Event Handling, after generated code (VBXevent) (управление событием элемента управления после генерации кода VBXevent). Например, следующее может иметь место в цикле ACCEPT диалогового поля,

содержащего .VBX - элемент управления.

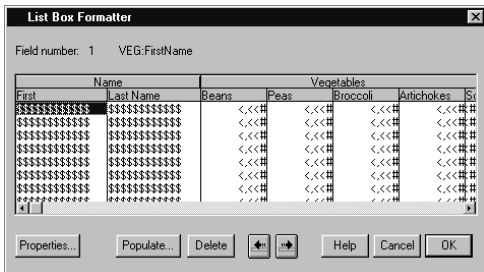
```
SomeString = ?vbx{PROP:VBXevent}  
  IF SomeString = 'UserWantsToDoX'  
    SomeProcedure  
  END
```



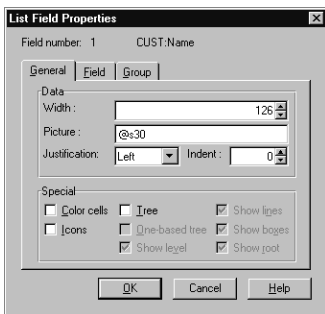
## Глава 11 Форматер поля списка



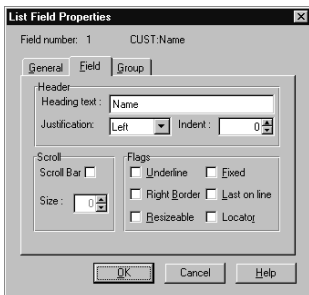
Форматер поля списка представляет собой гибкий инструмент для создания и модификации внешнего вида и функциональных возможностей ваших полей списка. Он демонстрирует образец поля списка, показывающий вам как редактирование влияет на его внешний вид.



Форматер поля списка дает образцы того, как будет выглядеть конструируемое поле списка.



Диалоговое окно “Свойства поля списка” дает вам возможность форматировать по одной колонке за раз.



Групповая закладка “Свойства поля списка” дает вам возможность определить два или более поля, которые имеют общие форматирующие элементы.

Name		Vegetables			
First Name	Last Name	Beans	Peas	Broccoli	Artichokes
Richard	Broadwater	5,795	5,677	9,679	5,458,657
Betzi	Kelton	9,795	5,767	464	34,546
Cindy	Peruzzi	2,455	8	2,345	2,399,945
Sheila	Skaggs	854	3,784	4,428	0,968
Aimee	Slusher	354	6,345	73	5,663,652
Jo	Stramiello	2,344	9,705	43	4,352,438
Jody	Tenner	394	2,383	8,923	5,239
Donald	Wolipka	6,784	641	3,841	465,246

Образцы полей списка и советы по расширению функциональных возможностей ваших элементов управления полем списка.

## Обзор

Понятие о формате поля списка

Диалог Свойства поля списка (List Field Properties Dialog)

Создание групп столбцов

Создание заголовка над двумя соседними столбцами

Создание линейки прокрутки под соседними столбцами

Многострочные записи

Улавливание двойного щелчка на поле списка

Добавление к полю списка способности drag and drop  
(тащить и бросать)

Редактирование на месте

Идентификация выделенной строки

Форматер поля списка обеспечивает высокую степень гибкости при создании и модификации ваших полей списка, выпадающих полей списка и комбинированных полей.

Как только вы определили QUEUE, чтобы обеспечить данные для списка (это делается автоматически шаблоном поля списка), Форматер поля списка представляет вам следующие возможности в настройка вашего списка:

- Вы можете установить количество столбцов в поле списка с изменяющимися или постоянными границами.
- Вы можете установить, чтобы одна запись (строка) из очереди QUEUE занимала более одной строки поля списка.
- Вы можете добавить несколько горизонтальных линеек прокрутки для каждого столбца или группы столбцов в поле списка.
- Вы можете потребовать, чтобы в фокус попадали либо строки, либо индивидуальные ячейки.
- Вы можете установить заголовки для столбцов поля списка.
- Вы можете добавить специальный элемент управления - локатор, который дает возможность пользователю быстро найти нужную ему позицию.
- Вы можете обеспечить возможность выбора многих строк в списке.

## Обзор

Общепринято, что поле списка предназначено показывать данные только для чтения. Оно имеет способность прокрутки и может содержать многие записи и поля. Оно эффективно показывает большие массивы данных.

Также принято, что выпадающее поле списка предназначено для показа только для чтения взаимоисключающих вариантов для выбора. Часто оно бывает прокручивающимся и называется выпадающим списком, так как вначале появляется в виде одной строки, а затем “выпадает” и показывает много строк, как в меню. Оно вынуждает пользователя сделать правильный выбор, обеспечивает визуальный намек, напоминая пользователю, что требуется сделать выбор, предлагает выбор по умолчанию и не занимает на экране много места.

Комбинированное поле - это просто поле списка, обладающее дополнительной способностью принимать ввод от пользователя.

Когда создается поле списка, вы определяете источник его данных, функциональные возможности и его формат. Интегрированная Среда разработки Clarion делит процесс определения этих свойств между несколькими диалогами:

- ◆ Диалоговое окно List Properties (свойства списка) устанавливает файл или очередь, которые поставляют данные, а также устанавливает общую способность прокрутки, то есть, все свойства поля списка, которые не являются специфическими для разных столбцов. Этот диалог обсуждался в предыдущей главе.

- ◆ Диалоговое окно List Box Formatter (форматер поля списка) позволяет добавить, удалить, переупорядочить и изменить размер специфических полей или столбцов, которые показаны в поле списка. Этот диалог обсуждается в данной главе.

- ◆ Диалоговое окно List Field Properties (свойства поля списка) определяет вид и поведение отдельных столбцов поля списка, Например, определяет заголовки, ширины и прокручивание отдельных столбцов. Этот диалог обсуждается в данной главе.

- ◆ Диалоговое окно List Field Properties (свойства поля списка) определяет также вид и поведение групп столбцов внутри поля списка. Например, вы можете распространить заголовок на несколько отдельных столбцов.

После того, как вы начали определять свое поле списка с помощью диалогового окна List Properties, дальше предстоит выполнить несколько шагов для завершения вашего поля списка..

□ Используя Форматер поля списка и диалоговое окно Select Field (выбор поля), добавляйте к вашему списку одно за другим поля.

1. Находясь в Форматере окна, щелкните правой клавишей мыши на элементе

управления “Поле списка” и выберите List Box Format (формат поля списка) из всплывающего меню для показа диалогового окна List Box Formatter (Форматер поля списка).

Форматер поля списка показывает представление поля списка. Каждое поле появляется как столбец в поле списка. Данные в этом столбце представлены символами “\$” для строк символов или символами “<” и “#” для числовых полей.

2. Нажмите кнопку Populate (заселить) чтобы добавить новое поле. (При работе с текстовым редактором кнопка Insert (вставить) заменяет кнопку Populate).

Если вы работаете, находясь в генераторе приложений, выберите поле данных из диалогового окна Select Field. После этого появится диалоговое окно List Field Formatter (форматер поля списка) с вновь добавленным столбцом. В текстовом редакторе диалог Select Field не появляется; переходите прямо в диалог List Field Properties и поэтому пропустите шаг 3.

3. Нажмите кнопку Properties (свойства).

Появится диалоговое окно List Field Properties. Воспользуйтесь этим диалогом, чтобы определить заголовки, ширины, границы, прокручивание и т.п. для столбцов. Форматирование, выполненное для первого столбца, становится форматированием по умолчанию для последующих столбцов.

4. Установите ширину столбца в диалоговых единицах. Дайте порядка четырех диалоговых единиц на каждый символ.

5. Установите шаблон изображения для данных.

Шаблон изображения определяет, как показываются данные. Например, шаблон изображения @P(###)###-####P показывает телефонный номер как (555) 555-5555.

6. Установите дополнительные (необязательные) параметры форматирования.

Вы можете выбрать выравнивание и смещение. Вы можете также установить заголовков столбца, границы, подчеркивание и другое.

7. Определите дополнительные функции.

Например, добавьте линейку прокрутки только для одного столбца. Разрешите поиск столбца. Вы можете установить границы с изменяющимися размерами, которые позволят конечному пользователю во время работы отрегулировать ширины столбцов с помощью мыши.

8. Нажмите кнопку ОК для возврата к диалоговому окну List Box Formatter.

Для каждой модификации, которую вы производите над полем списка на экране, форматер поля списка создает соответствующий атрибут FORMAT для оператора LIST, который определяет ваше поле списка. Оператор LIST, в свою очередь, располагается в структуре WINDOW. Полное описание вы найдете в Справочнике языка.

☐ Если необходимо, сгруппируйте поля.

1. В диалоговом окне List Field Properties выберите закладку Group (группа).

Этим вы устанавливаете, что предыдущее поле и следующее поле, из тех, которое вы добавляете к списку, имеют общие форматирующие элементы.

2. Установите текст заголовка группы.

Простейший общий элемент - это общий заголовок. Заголовок группы из двух полей появляется прямо над заголовками полей.

3. Если необходимо, установите дополнительное форматирование.

Вы можете, например, отформатировать поля так, что никакого разделителя между членами группы не появится, а появится он в конце группы. Чтобы сделать так, проверьте, что поле Right Border (правая граница) нет отметки для первого поля(ей) в группе и отметка установлена только для последнего члена группы.

4. Нажмите кнопку ОК, чтобы закрыть диалог List Field Properties.

☐ По необходимости соберите несколько полей данных в один столбец поля списка.(одно поле над другим “в стопку”)

1. Отметьте поле Last on Line (последний на строке) на закладке Field в диалоговом окне List Field Properties

Этот параметр эквивалентен добавлению символа возврата каретки сразу после текущего поля. Следующее поле группы появится прямо под текущим полем, в том же самом столбце - под заголовком группы.

2. Нажмите кнопку ОК для закрытия диалога List Field Properties .

3. Нажмите кнопку Populate (или Insert ) в диалоговом окне List Box Formatter.

Это даст вам возможность начать форматирование следующего поля. Форматируйте

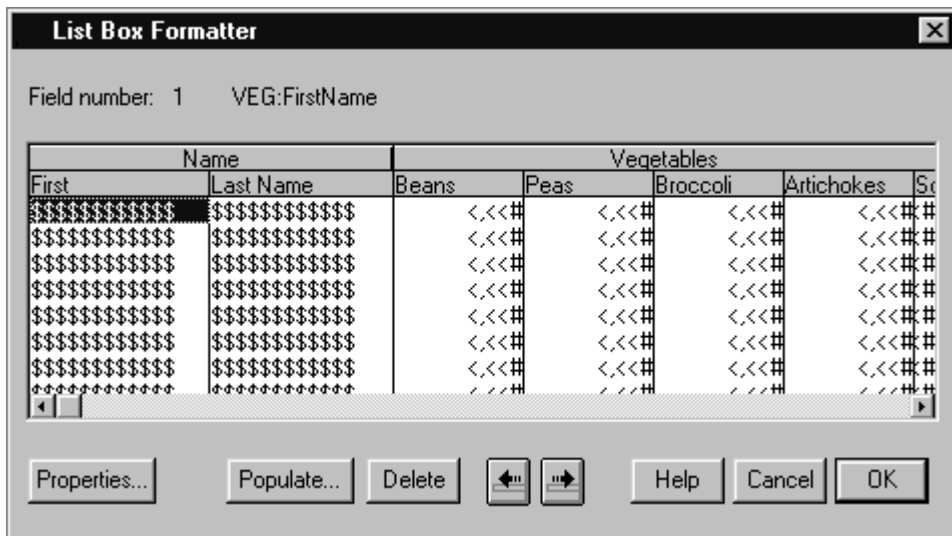
следующее поле, аналогично тому, как это делалось для предыдущих полей. Продолжайте форматирование, пока все поля не будут добавлены к полю списка.

## Работа с форматером поля списка

Форматер поля списка показывает текущий вид создаваемого поля списка. Field Number (Номер поля): приглашение вблизи верхнего края форматера поля списка показывает, какое поле в данный момент выбрано. Каждое поле появляется в поле списка как столбец, в котором данные, отмеченные знаком \$, это строки символов, а знаками < или # - числовые данные. Если какое-либо поле содержит заголовок, над списком появляется строка заголовка.

Вы форматируете поля одно за другим в диалоге List Field Properties (свойства поля списка), обновляющем поле списка. Для вашего удобства форматер поля списка обеспечивает горизонтальную линейку прокручивания, независимо от того, определите вы ее или нет в диалоге List Properties (свойства списка).

Форматер не показывает вертикальной линейки прокрутки, даже если вы отметили поле Vertical (вертикальный) в диалоговом окне List Properties. Однако, вертикальная линейка прокрутки появляется во время работы, если очередь содержит более позиций, чем помещается в поле списка.



Форматер поля списка, показывающий две группы.


Диалоговые кнопки форматера поля списка позволят вам добавить, удалить, перегруппировать, изменить размер и формат полей в поле списка.



☐ Чтобы добавить поле к полю списка, нажмите кнопку Populate (заселить) или Insert (вставить).


Когда форматер поля списка вызывается из генератора приложений, кнопка Populate показывает диалоговое окно выбора поля. Отсюда вы можете указать любое поле из словаря данных или переменную памяти для использования их в качестве источника данных для столбца поля списка. Сгенерированная программа помещает выбранные поля в очередь для использования их в поле списка.



Когда форматер поля списка вызван не из генератора приложений, а из текстового редактора, кнопка Insert заменяет кнопку Populate и Вы сами отвечаете за построение очереди, которая заполняет поле списка.

- ☐ Для удаления поля из поля списка нажмите кнопку Delete.
- ☐ Для вызова помощи нажмите кнопку Help.
- ☐ Для отмены изменений в поле списка нажмите кнопку Cancel.
- ☐ Для подтверждения изменений, сделанных при форматировании поле списка, нажмите кнопку ОК.

☐ Для перемещения выбранного поля налево, щелкните кнопкой  или нажмите SHIFT+”левая стрелка”.

☐ Если выбранное поле является самым левым в группе, кнопка  перемещает поле из группы, однако порядок полей не изменяется. И наоборот, кнопка  передвигает выбранное поле внутрь группы на следующее место слева и порядок полей не изменяется.

☐ Для перемещения выбранного поля в направо, щелкните клавишей  или нажмите SHIFT+”правая стрелка”.

Если выбранное поле является самым правым в группе, кнопка  перемещает поле из группы, однако порядок полей не изменяется. И наоборот, кнопка  передвигает выбранное поле внутрь группы на следующее место справа и порядок полей не изменяется.

☐ Чтобы отформатировать поле, нажмите кнопку Properties.

Диалоговое окно List Field Properties дает вам возможность установить ширину столбца, символный шаблон, текст заголовка, а также другие параметры, такие, как горизонтальная линейка прокрутки для отдельного поля. Кроме того, List Field Properties - диалог дает вам возможность “группировать” поля, что помещает еще один заголовок над сгруппированными столбцами для визуальной демонстрации того, что поля теперь связаны. Подробно это описано ниже.

## Диалог Свойства поля списка (List Field Properties)

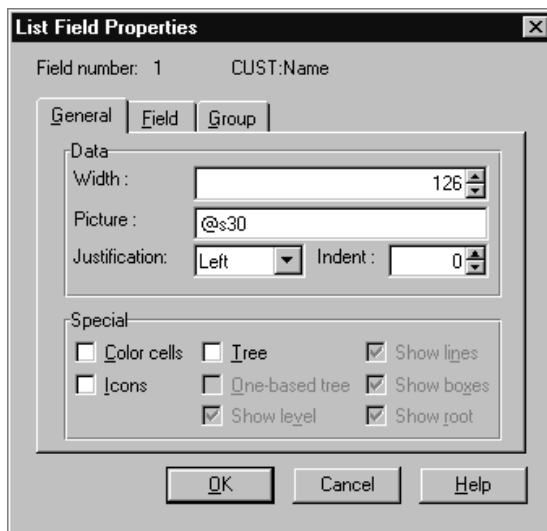
Нажмите кнопку Insert (вставить) или Populate (заселить) для добавления поля к списку, затем отформатируйте это поле в диалоге List Field Properties (свойства поля списка). Для каждого выбора, который вы делаете в диалоге, формater поля списка создает соответствующий атрибут FORMAT для оператора LIST, который определяет вид и поведение вашего поля списка.

### Закладка “Общая” (General)

Диалоговое окно дает вам возможность установить следующие варианты форматирования данных.

Установка свойств поля списка.

Width (ширина)	Установите ширину столбца в диалоговых единицах. По умолчанию Формater устанавливает величину, равную четырехкратному числу символов, установленных в шаблоне поля в словаре данных. Для переменных величина по умолчанию - это четырехкратное число символов в ее шаблоне
----------------	--



**Совет: В качестве грубой ориентировки считайте, что на средний символ приходится четыре диалоговые единицы. Например, если вы хотите иметь столбец шириной в 10 символов, наберите 40 в поле Ширина данных.**

После того, как вы поместили одно поле, диалог List Box Formatter позволит вам также передвинуть мышью разделители столбцов для изменения ширины столбца. Курсор меняется, когда вы помещаете его поверх разделителя, что показывает, что вы можете изменить размер.

Ширина данных, которую вы установили, появляется в строке символов формата для поля предшествующая коду выравнивания, как в “40L”.



**Picture (Шаблон изображения)** Установите шаблон изображения для данных. List Box Formatter показывает данные в соответствии с шаблоном изображения. Например, шаблон изображения @P(###) ###-####P показывает телефонный номер как (555) 555-5555.

Шаблон изображения, который вы устанавливаете, появляется в строке символов формата.

**Justification (выравнивание данных)**

Выберите из выпадающего списка тип выравнивания: левое, правое, по центру или десятичное. Десятичное выравнивание устраивает десятичные числа по их десятичным запятым.

Выравнивание появляется в строке символов формата, следующей за шириной данных, как в “40R”.

**Indent (отступ)**

По желанию установите отступ в диалоговых единицах. Отступ передвигает данные на установленное число диалоговых единиц в направлении, противоположном выравниванию. Отступ из двух (2) на выравненных налево данных улучшает читаемость поля списка.

Отступ появляется в строке символов формата, окруженный круглыми скобками и предваряемый буквой, указывающей тип выравнивания, как в “L(8).”

**Color Cells (цвет ячеек)**

Отметка этого поля показывает этот столбец во время работы со сплошным или заполненным фоном.

Ячейки цвета появляются как значок “\*” в строке символов формата.

**Icons (пиктограммы)** Отметка этого поля создает область налево от данных в столбце, который показывает назначенную вами графическую пиктограмму.

Добавляет “I” к строке символов формата.

### Опции иерархического дерева

**Tree (дерево)**

Отметка этого поля показывает этот столбец в виде иерархического дерева. Смотрите Дерево отношений в главе Шаблоны элементов управления, программы и расширений. Смотрите также Дерево отношений в оперативной помощи.

Добавляет “T” в строку символов формата.

**One-based tree (дерево с одной базой)**

Отметка этого поля позволяет корневому уровню сжаться, то есть, все позиции в этом дереве могут сжаться до одной единственной

строки.

Отметка этого поля добавляет “(1)” к “T” в строке символов FORMAT (формат), а это приводит к тому, что “T(1)” позволяет корневому уровню коллапсировать.

Show Level (показать уровень)

Отметка этого поля заставляет каждый последующий уровень дерева иерархии смещаться направо.

Очистка этого поля приводит к добавлению символа “(I)” к “T” в строке символов формата, что дает в результате “T(I)” и подавляет введение отступа при переходе с одного уровня иерархии на другой, более низкий.

Show Lines (показать линии)

Отметка этого поля добавляет соединительные линии между связанными позициями в диаграмме дерева.

Очистка этого поля приводит к добавлению символа “(L)” к “T” в строке символов формата, что дает в результате “T(L)” и подавляет введение линий разметки в дерево иерархии.

Show Boxes (показать поля)

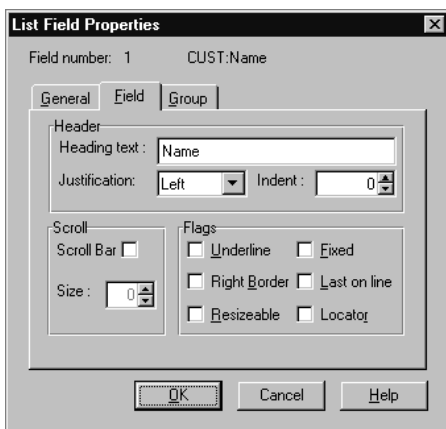
Отметка этого поля добавляет поля расширения (+) и сжатия (-) к диаграмме дерева.

Очистка этого поля приводит к добавлению символа “(B)” к “T” в строке символов формата. что в результате дает “T(B)” и подавляет появление полей.

Show Root (показать корень)

Отметка этого поля показывает корневую позицию для диаграммы дерева.

Очистка этого поля добавляет “(R)” к “T” в строке символов FORMAT (формат), что в результате приводит к тому, что “T(R)” подавляет показ корневой позиции.



### Закладка “Поле” (Field)

#### Heading Text (текст заголовка)

Если необходимо, установите текст заголовка столбца. Заголовок появляется как серая строка над списком позиций поля данных. Чтобы установить отсутствие заголовка, оставьте это поле пустым. Если какое-либо поле, включенное в поле списка, имеет заголовок, заголовок появляется поверх всего поля списка; поля без текста заголовка будут иметь пустую шапку. Заголовок появляется в строке символов формата заключенным между знаками тильды, как в “~My Header~.”

#### Justification (выравнивание данных)

Выберите из выпадающего списка тип выравнивания и установите левое, правое, центральное или десятичное выравнивание заголовка. Тип выравнивания появится в строке символов формата вслед за заголовком, как в “~My Header~L.”

#### Indent (отступ)

Если необходимо, установите отступ для текста заголовка, в диалоговых единицах. Отступ передвигает данные на установленное число диалоговых единиц в направлении, противоположном выравниванию. Отступ из двух (2) на выровненных налево данных улучшает читаемость поля списка. Отступ появится в строке символов формата вслед за заголовком, как в “~My Header~L(8).”

#### Scroll Bar (линейка прокрутки)

Установите флажок в поле линейки прокрутки для задания горизонтальной линейки прокрутки только для этого столбца. Если общее поле списка уже имеет линейку прокрутки, то линейка прокрутки для столбца появится над линейкой поля списка.

**Size (размер)** Устанавливает, в диалоговых единицах, как далеко прокручивается столбец. Эта величина определяет ширину области прокрутки, которая не показывается в поле списка.

Например, если длина вашей позиции данных равна пятидесяти (50) символам, а ширина столбца вашего поля примерно сорок символов (40) (сто шестьдесят (160) диалоговых единиц), вы должны установить Size (Размер) пятьдесят (50). Пятидесяти (50) дополнительных диалоговых единиц достаточно для показа десяти символов, которые выступают за пределы ширины столбца поля списка.

Линейка прокрутки и размер появятся в строке символов формата вместе, как в “S(4).”

### Underline

(подчеркивание) Установите флажок в поле подчеркивания чтобы добавить стиль подчеркивания к тексту поля списка. На деле это создает нижнюю границу для каждой строки в столбце, придавая тем самым вашему полю списка вид таблицы или ячейки.

Строка символов формата включает символ подчеркивания непосредственно перед текстом заголовка, как в “\_~MyHeader~.”

### Right Border (правая граница)

Установите флажок в поле Right Border (правая граница) для установки разделителя столбцов в поле списка во время работы программы.

Строка символов формата включает символ (|),стоящий сразу перед текстом заголовка, как в “|~MyHeader~.”

### Resizable (с изменяемым размером)

Установите флажок в поле Resizable (с изменяемым размером) для установки того, что пользователь может изменить ширину столбцов во время работы программы.

Строка символов формата включает символ “М”, стоящий непосредственно перед заголовком текста, как в “М~MyHeader~.”

**Совет: Во время работы свойство PROP:Format всегда содержит текущий формат поля списка, включая любые изменения пользователя. Чтобы сохранить размеры столбца, установленные пользователем, используйте GETINI и PUTINI для сохранения и восстановления PROP:Format величин:**

PUTINI (‘List Settings’, ‘UserList’,?List(PROP:Format),”MYAPP.INI’)

### Fixed

(фиксированный) Установите флажок в поле Fixed Box (фиксированное поле), чтобы установить, что столбец всегда остается видимым в поле списка.

Строка символов формата включает символ “F, стоящий непосредственно перед заголовком, как в “F~MyHeader~.”

### Last on Line (последний на строке)

Отметка этого поля устанавливает, что следующее поле в группе появится сразу же под текущим полем. Фактически, это выводит два или более полей в один столбец, ниже заголовка группы.

Строка символов формата включает символ “/”, стоящий непосредственно перед заголовком, как в “/~MyHeader~.”

Locator (локатор) Отметьте поле Локатор чтобы установить, что этот столбец работает с элементом управления вводом типа локатор. Когда пользователь набирает символ в поле ввода локатора, поле списка прокручивается до первого совпадения данных в столбце с введенными данными. Строка символов формата включает символ “?”, стоящий непосредственно перед заголовком, как в “?~MyHeader~.”

## Группы столбцов

Группы полей списка содержат два или более полей, которые имеют общие форматирующие элементы, такие, как общий заголовок, или два поля, которые оказываются “сложенными в стопку” внутри одного столбца. Установите групповой заголовок, чтобы визуально связать два соседние поля. Сначала отметьте поле флажков Last on Line (последний на линии) на первом поле, чтобы установить “складывание полей в стопку”.

Вы создаете группу путем выбора первого поля в группе, затем выбираете закладку Group в диалоге List Field Properties (свойства поля списка). Следующее поле вы выбираете из диалогового окна Select Field или следующее поле в очереди QUEUE. Или же вы можете использовать кнопки **а** и **Я** на форматере поля списка для того, чтобы передвинуть поля в существующую группу и из нее.

Из диалогового окна List Field Properties(свойства группы списков) вы можете установить заголовок группы, который появляется над заголовком столбца. Так как атрибут правой границы установлен отдельно от полей, вы можете создать заголовок, который будет расширен на все столбцы.

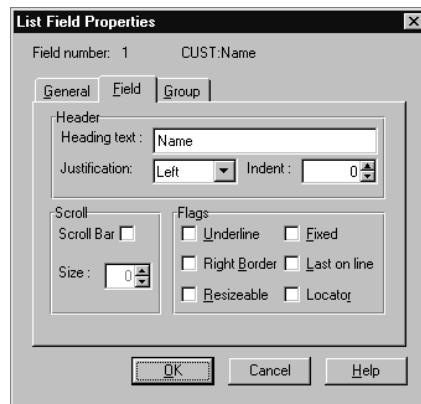
При творческом форматировании вы можете использовать групповые заголовки, для связи родственных данных, находящиеся в разных столбцах. Например, вы можете поместить “имя” в групповой заголовок для столбца один, затем “первый” и “последний” в заголовки полей для первых двух полей. Вы можете также использовать групповой заголовок для разрыва текста заголовка на две строки, когда метка столбца длиннее, чем ширина столбца.

### Закладка “Группа” (Group)

Установка заголовка группы, под названием “Заказчик”.

В диалоговом окне List Group Properties на закладке Group (группа) имеются следующие поля:

Heading Text



(текст заголовка) Если необходимо, определите текст заголовка для группы. Заголовок появляется как серая строка над п о з и ц и я м и данных поля списка. Чтобы установить отсутствие заголовка, оставьте это поле пустым. Если какое-либо поле, включенное в поле списка, имеет заголовок, заголовок появляется над каждым полем в поле списка; поля без текста заголовка будут иметь пустой заголовок.

Justification (выравнивание данных)

Выберите из выпадающего списка установку левого, правого, центрального или десятичного выравнивания.

Indent (отступ)

Если необходимо, установите для заголовка отступ в диалоговых единицах. Отступ передвигает данные на установленное число диалоговых единиц в направлении, противоположном выравниванию. Отступ из двух (2) на выровненных налево данных улучшает читаемость поля списка

Scroll Bar (линейка прокрутки)

Установите флажок в поле Scroll Bar (линейка прокрутки), чтобы установить одну горизонтальную линейку прокрутки для всей группы. Если общее поле списка уже имеет линейку прокрутки, линейка для столбца появляется над линейкой прокрутки поля списка. Если отдельные столбцы в группе имеют линейки прокрутки, они предваряются групповой линейкой прокрутки.

Это является эффективным способом представления в организованном виде связанных позиций данных поля списка.

Size (размер)

Установите диапазон линейки прокрутки. Эта величина определяет ширину области прокрутки, которая не показывается в поле списка. Например, если длина вашей позиции данных равна пятидесяти (50) символам, а ширина столбца вашего поля примерно сорок символов (40) (сто шестьдесят (160) диалоговых единиц), вы должны установить Размер пятьдесят (50). Пятидесяти (50) дополнительных диалоговых единиц достаточно для показа десяти символов, которые выступают за пределы ширины столбца поля списка.

Линейка прокрутки и ее размер появляются вместе в строке символов формата, как в "S(4)."

Underline

(подчеркивание)

Установите флажок в поле Underline (подчеркивание) чтобы добавить стиль подчеркивания к тексту поля списка. На деле это

создает нижнюю границу для каждой строки в столбце, придавая тем самым вашему полю списка вид таблицы или ячейки. Строка символов формата включает символ подчеркивания непосредственно перед текстом заголовка, как в “\_~My Header~.”

**Right Border (правая граница)** Установите флажок в поле Right Border (правая граница ) для установки разделителя столбцов в поле списка во время работы программы. Строка символов формата включает символ (|),стоящий сразу перед текстом заголовка, как в “|~MyHeader~.”

**Resizable (с изменяемым размером)** Установите флажок в Resizable (с изменяемым размером) для установки того, что пользователь может изменить ширину столбцов во время работы программы. Строка символов формата включает символ “М”, стоящий непосредственно перед заголовком текста, как в “М~MyHeader~.”

**Fixed (фиксированный)** Установите флажок в поле Fixed Box (фиксированное поле), чтобы установить, что столбец всегда остается видимым в поле списка. Строка символов формата включает символ “F, стоящий непосредственно перед заголовком,

Name		Vegetables			
First Name	Last Name	Beans	Peas	Broccoli	Artichokes
Kambra	Bolch	3,245	761	556	6,423
Richard	Broadwater	6,785	5,677	9,675	545,667
Betzi	Kelton	9,795	5,767	464	34,546
Cindy	Peruzzi	2,455	8	2,345	2,398,945
Sheila	Skaggs	854	3,784	4,428	0,968
Aimee	Slusher	354	6,345	78	5,663,652
Jo	Stramiello	2,344	9,765	43	4,352,438
Jody	Terrier	984	2,383	8,923	5,239
Donald	Wotipka	6,784	641	3,841	465,246

Создание заголовка над соседними столбцами

Образец элемента управления Поле списка с групповыми заголовками.

1. В форматере поля списка нажмите кнопку Populate (или Insert) для добавления поля к полю списка.

Появится диалоговое окно Select Field (выбор поля).

2. Используйте диалоговое окно выбора поля для того, чтобы выбрать поле и добавить его к вашему списочному полю.

3. Нажмите кнопку Properties (свойства) чтобы открыть диалоговое окно List Field Properties (свойства поля списка).

Используйте диалоговое окно свойств поля списка для нормального форматирования столбца.

- 4. В диалоговом окне List Field Properties выберите закладку Group (группа).
- 5. Нажмите кнопку ОК , когда вас запрашивают, хотите ли вы создать новую группу.
- 6. В поле Heading Text (текст заголовка) наберите текст заголовка группы.

Этот текст общий для всех полей в группе. По умолчанию он центрируется над группой, однако вы можете установить другой вид выравнивания., .

7. Нажмите кнопку ОК для закрытия диалога List Field Properties (свойства группы списков).

Снова появляется диалоговое окно формatera поля списка.

- 8. Нажмите кнопку Populate (или Insert) в диалоговом окне формatera поля списка. Снова появляется диалоговое окно выбора поля..

9. Используйте диалоговое окно выбора поля для выбора поля, которое добавляется к вашему списочному полю.

Поле добавляется к группе, созданной в предыдущих диалогах.

10. Нажмите кнопку Properties (свойства) чтобы открыть диалоговое окно свойств поля списка.

Используйте диалог свойств поля списка для нормального форматирования столбца.

11. Нажмите кнопку ОК для закрытия диалога List Field Properties (свойства поля списка).

Снова появляется форматер поля списка. Дополнительные поля, добавленные таким методом, будут добавлены к группе. Чтобы “окончить” группу, нажмите кнопку а . Это передвигает последнее добавленное поле вовне из группы.

Name		Vegetables				
First Name	Last Name	Beans	Peas	Broccoli	Artichokes	Sc
#####	#####	<.<<#	<.<<#	<.<<#	<.<<#	#
#####	#####	<.<<#	<.<<#	<.<<#	<.<<#	#
#####	#####	<.<<#	<.<<#	<.<<#	<.<<#	#
#####	#####	<.<<#	<.<<#	<.<<#	<.<<#	#
#####	#####	<.<<#	<.<<#	<.<<#	<.<<#	#

**Создание линейки прокрутки под соседними столбцами**

Добавление линейки прокрутки для группы “овощи”. Данными служат случайные символы и цифры.

1. В форматере поля списка нажмите кнопку Populate (или Insert) для добавления поля к полю списка.

Появится диалоговое окно Select Field (выбор поля).

2. Используйте диалоговое окно выбора поля для того, чтобы выбрать поле и добавить



его к вашему списочному полю.

3. Нажмите кнопку Properties (свойства) чтобы открыть диалоговое окно List Field Properties (свойства поля списка).

4. В диалоговом окне List Field Properties выберите закладку Group (группа).

5. Нажмите кнопку OK, когда вас спрашивают, хотите ли вы создать новую группу.

6. На закладке Group (группа) отметьте поле Scroll Bar (линейка прокрутки).

7. Нажмите кнопку OK для закрытия диалога List Field Properties (свойства группы списков).


Снова появляется диалоговое окно форматера поля списка.

8. Нажмите кнопку Populate (или Insert) в диалоговом окне форматера поля списка чтобы добавить второе поле к списочному полю.

Это поле добавлено к группе, установленной в предыдущем диалоге.

9. Используйте диалог свойств поля списка для нормального форматирования поля.

10. Нажмите кнопку OK для закрытия диалога List Field Properties (свойства поля списка).

Снова появляется форматер поля списка. Дополнительные поля, добавленные таким методом, будут добавлены к группе. Чтобы “окончить” группу, нажмите кнопку . Это передвигает последнее добавленное поле вовне из группы.

**Совет: Скорее всего вы оставите поле флажков Right Border (правая граница) неотмеченным для первых полей в группе; и отмеченным для последнего поля. Это отделяет группу, как единое целое, от других полей в поле списка.**

Name	Vegetables				
First Last	Beans	Peas	Broccoli	Artichokes	Squash
Kambra	3,245	761	556	6,423	0
Bolch					
Richard	6,785	5,677	9,675	545	6,667
Broadwater					
Betzi	9,795	5,767	464	34	6,546
Kelton					
Cindy	2,455	8	2,345	2,399	9,945
Peruzzi					
Sheila	854	3,784	4,428	0	968
Skaggs					

## Многострочные записи

---

Тот же образец поля списка, показывающий многострочные записи.

1. В форматере поля списка нажмите кнопку **Populate** (или **Insert**) для добавления поля к полю списка.

Появится диалоговое окно **Select Field** (выбор поля).

2. Используйте диалоговое окно выбора поля для того, чтобы выбрать поле и добавить его к вашему списочному полю.

3. Нажмите кнопку **Properties** (свойства) чтобы открыть диалоговое окно **List Field Properties** (свойства поля списка).

4. В диалоговом окне **List Field Properties** отметьте поле **Last on Line** (последний на строке).

5. В диалоговом окне свойств поля списка выберите закладку **Group** (группа).

6. Нажмите кнопку **OK**, когда вас спрашивают, хотите ли вы создать новую группу.

7. Нажмите кнопку **OK** для закрытия диалога **List Field Properties** (свойства группы списков).

Снова появляется диалоговое окно форматера поля списка.

8. В форматере поля списка нажмите кнопку **Populate** или **Insert** для того, чтобы добавить второе поле к полю списка.

Поля добавляется к группе, установленной в предыдущем диалоге.

9. Используйте диалоговое окно свойств поля списка для того, чтобы отформатировать его нормально.

10. Нажмите кнопку **OK** для закрытия диалога **List Field Properties** (свойства поля списка).

Снова появляется формater поля списка. Дополнительные поля, добавленные таким методом, будут добавлены к группе. Чтобы “окончить” группу, нажмите кнопку **а**. Это передвигает последнее добавленное поле вовне из группы.

События списочного поля и другие функциональные характеристики.

Если вы остаетесь верным шаблонам процедур, вам может никогда не потребуется управлять событиями поля списка. Например, шаблон просмотра данных (**Browse**) дает вам возможность объявить процедуру обновления. Это избавляет вас от нудного программирования двойных щелчков мыши. Если вам нужно расширить функциональные возможности поля списка: добавить возможность **drag and drop** или множественный выбор,

вам нужно добавить операторы к циклу ACCEPT данного окна, чтобы контролировать специфические события поля списка.

**Совет:** Некоторые из нижеследующих разделов имеют мало отношения к диалогу форматирования поля списка. Они, однако, остаются в этой главе потому, что мы думали, что вы сначала просмотрите их.

### **Улавливание двойного щелчка на поле списка**

Способность улавливания двойного щелчка клавиши мыши встроена в шаблоны просмотра Clarion. Чтобы уловить двойной щелчок на списке закодированном вручную:

1. Установите на данном элементе управления списком ALRT(Double-click).

```
MyList LIST,AT( ), ALRT(MouseLeft2)
```

или

```
?MyList(PROP:Alrt)=MouseLeft2
```

2. Поставьте ловушку для события EVENT:AlertKey на данном элементе управления списком, как это показано в следующем примере.

3. Поставьте ловушку для двойного щелчка мыши MouseLeft2 как в следующем примере:

```
ACCEPT
```

```
CASE FIELD( )
```

```
OF ?List
```

```
CASE EVENT ( )
```

```
OF EVENT: AlertKey
```

```
IF KEYCODE( ) = MouseLeft2
```

```
CurrentSel = CHOICE(?List) ! выбрать текущую строку из списка
```

```
GET(TheQUEUE , CurrentSel)! выбрать соответствующие данные из очереди
```

```
....
```

Приведенный выше код обнаруживает, на какой позиции пользователь щелкнул дважды (используется функция CHOICE()) и отыскивает эту позицию в очереди (используется функция GET()).

Вы можете добавить две строки кода в вышеприведенной структуре IF в точке вставки Обработка двойного щелчка просмотра для обработки двойных щелчков мыши для списков, заселенных с помощью шаблона элемента управления полем просмотра в генераторе приложения.

### **Способность drag and drop (тащить и бросать)**

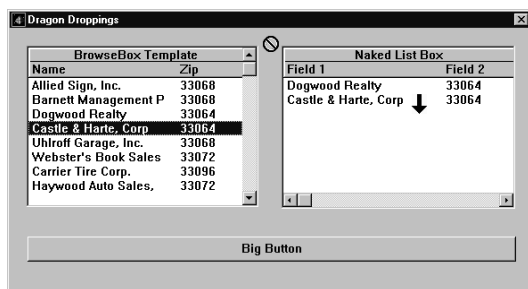
Drag and drop означает, что конечный прользователь может выбрать объект в одном окне или элементе управления и удерживая нажатой левую клавишу мыши, “перетащить” этот объект в другое окно, где освободив клавишу мыши, “бросить” его в там в окне или на

элемент управления в нем, которые в свою очередь затем могут исследовать “сброшенный” на них объект и если необходимо обработать его каким либо образом. Обсуждение общих принципов Drag and drop можно найти в Справочнике языка- Обработка с помощью функции Drag and drop, DRAGID, DROPID, SETDROPID, CLIPBOARD и SETCLIPBOARD

Добавление Drag and Drop к полю списка Clarion for Windows представляет собой простую операцию. В этом разделе дается пример протаскивания позиции из одного поля списка в другое в пределах одного приложения. Вы можете сделать “Drag and Drop” между приложениями - например, из Диспетчера файлов к вашему приложению - смотрите подробности в Справочнике языка.

Выполнение Drag and Drop в Clarion приложении включает два процесса:

- Назначение элементов управления “откуда перетащить и куда бросить”.
- Выполнение обмена данными после того, как пользователь инициирует процедуру “тащить и бросать”, управляя событиями “тащить и бросать”.



Два поля списка, оба из них могут быть источниками или целями операции Drag and Drop. Курсор изменяется на пиктограмму “Не бросать” над теми областями, где нельзя бросать.

Затем изменяет свой вид, превращаясь в стрелку, направленную вниз, указывая на область, в которой можно опустить.

### Назначение места, откуда необходимо “тащить”

1. Щелкните правой клавишей мыши на элементе управления списком, который является источником перетаскиваемого объекта, и выберите Properties (свойства) из всплывающего меню.

2. Отберите закладку Extra (дополнительные возможности).

3. В поле Drag ID (идентификация процедуры “тащить”) наберите имя переноса “тащить и бросать”.

Это может быть метка соответствия элемента управления списка LIST, из которого переносится данная позиция. Например, для списка LIST, который показывает записи заказчика и имеет метку соответствия ?Customers, наберите customer в поле Drag ID, так как вы будете “тащить” заказчиков из этого списка.

4. Нажмите кнопку ОК.

## Назначение мишени для операции бросания drop

```
IF DRAGID ( ) ! Проверяет на совпадение сигнатур переноса  
    GET(Queue:Browse,CHOICE(?Customer)) ! Размещение к  
                                           !"перетаскиваемой" строке  
! ?Customer{PROPLIST:MouseDownRow} можно заменить на функцию CHOICE или  
!DO BRW1::NewSelection ! или для шаблонов BrowseBox используйте эту  
                        !подпрограмму  
SETDROPID( 'string to drug and drop' ) ! Передавая перетаскиваемое поле  
                                         !в область обмена данными. Или же  
                                         !используйте SETCLIPBOARD или  
!переменную Local/Global.  
END
```

Этот код обнаруживает событие “Drag” - в тот момент, когда пользователь отпускает кнопку мыши над правильной целью бросания и пересылает “перетаскиваемые” данные с функцией SETDROPID.

### **Собирание и обработка данных на событии бросания Drop**

1. Щелкните правой клавишей на списочном элементе управления, к которому вы хотите протаскивать, и выберите Embeds (вставки) из всплывающего меню.

2. Установите место события “Обработки события управления после генерированного кода;” точка вставки Drop (бросить) и вставьте следующий код:

```
IF DRAGID( )           !проверяет на совпадение сигнатур пересылки
  MyQ:CUST:Name = DROPID( )  ! находит “переташенные” данные
  ADD(MyQ)               ! сделать что-нибудь с этим, как, например,
                          !добавить это к очереди списка.

END
```

Этот код обнаруживает событие бросания - в тот момент, когда пользователь отпускает кнопку мыши над правильной целью бросания - и восстанавливает строку символов с помощью функции DROPID.

Обратите внимание, что данный пример бросает данные на список LIST, кодированный вручную. Обычно в этом нет смысла и не практично бросать данные на шаблон поля просмотра BrowseBox, так как BrowseBox List является элементом управления только для просмотра записей из файла. Для обновления файла с помощью процедуры drag and drop попробуйте бросить эти данные на окно процедуры формы ( Form).

### **Поддержка редактирования на месте**

Вы можете добавить характеристику редактирования на месте к своему списочному полю с помощью синтаксиса свойства. Вы должны, прежде чем сможете это выполнить, установить свойства списка LIST:

☐ Добавьте атрибут COLUMN (поле флажков Select Columns (выбор столбцов) ) если это многостолбцовое списочное поле, так чтобы пользователь мог редактировать по одному полю за раз.

☐ Добавьте атрибут IMM (поле флажков Immediate (ближайший)), таким образом вы можете получить текущий выбор очереди QUEUE каждый раз, когда конечный пользователь перемещает строку выбора.

Реальное “редактирование” происходит в поле ввода, которое вы можете сначала спрятать, а затем открыть тогда, когда конечный пользователь дважды щелкает мышью на своем выборе. Смотрите выше Захват двойного щелчка мыши на списочном поле.

Этот пример предполагает QUEUE, называемую MyQueue (моя очередь), которая содержит два поля, называемые Field1 и Field2. Это предполагает, что метка соответствия для элемента управления списочного поля - это ?MyList.

1. Добавьте элемент управления ввода ENTRY к вашему окну, используя для этой цели Форматер окна. Установите его атрибут HIDE (спрятать) и назначьте ?EditEntry в качестве его атрибута USE (вы можете использовать величину иную, чем ?EditEntry; просто с этим именем образец кода проще читается).

Библиотека поддержки автоматически открывает его, изменяет размер и меняет положение на основе синтаксиса свойства PROP:edit.

2. Вставьте следующий код в точку встраивания New Selection события элемента управления для ?MyList:

```
IF ?MyList{PROP:edit,?MyList{PROP:column}}
    GET(MyQueue, CHOICE( ))           !получить соответствующий
                                      !элемент QUEUE
END
```

Это помещает текущий выбор в буфер очереди QUEUE, где он оказывается доступным, если конечный пользователь дважды щелкнет клавишей мыши для редактирования выбора.

3. Вставьте следующий код в точку встраивания Browse Double Click Handler (обслуживание двойного щелчка поля просмотра) для ?MyList:

```
GET(MyQueue, CHOICE( )_
?EditEntry{PROP:text} = ?MyList{PROPLIST:picture,?MyList{PROP:column}}
CASE ?MyList{PROP:column}
OF 1
    ?EditEntry{PROP:use} = Field1      !загрузить ввод содержимым очереди
    ?MyList{PROP:edit,1} = ?EditEntry !активировать редактирование для
                                      !столбца один
OF 2
    ?EditEntry{PROP:use} = Field2
    ?MyList{PROP:edit,2} = ?EditEntry ! активировать редактирование для
                                      !столбца два
END
SELECT(?EditEntry)                   !дать фокус элементу управления
                                      !ввода
```

4. Вставьте следующий код в точку встраивания Selected (выбрано) события элемента управления для ?EditEntry:

```
?MyList{PROP:Touched} = 1
```

5. Вставьте следующий код в точку встраивания Accepted (принято) события элемента управления для ?EditEntry:

PUT(MyQueue)	!обновить очередь QUEUE
?MyList{PROP:edit, ?MyList{PROP:column}} = 0	!выключить редактирование на
	!месте
SELECT(?MyList)	
DISPLAY	

Этот код выявляет двойной щелчок на списочном поле и получает отобранную “ячейку”. Она получает содержимое и помещает его в поле ввода, которое его показывает. Затем он проверяет на наличие события Accepted (принято) в поле редактирования. Если это принято, он берет содержимое поля редактирования и использует его для обновления очереди QUEUE.

Ваше дело добавить код для обновления записи базы данных, так как Browse (просмотр) обычно не пишет записи на диск!

## Идентификация выбранных строк

---

Если вы следуете шаблонам, вам обычно нет необходимости находить отобранную строку в списочном поле. Однако, если такая необходимость возникнет, имеется несколько альтернатив:

```
CurrentRow = ?MyList{PROP:SelStart}  
CurrentRow = ?MyList{PROPLIST:MouseDownRow}  
CurrentRow = CHOICE(?MyList)
```



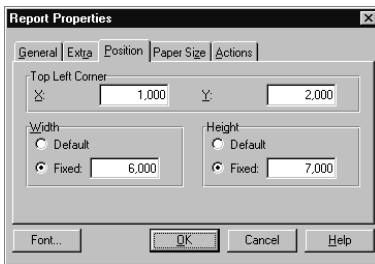
## Глава 12 Форматер отчета



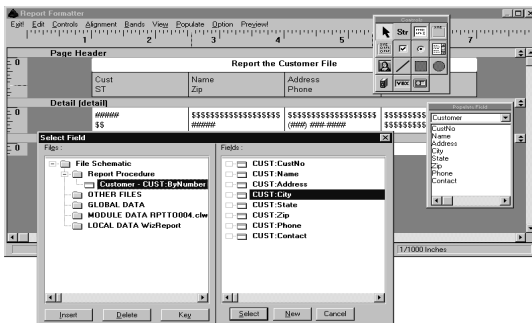
Визуально проектирует отчеты вашего приложения. Форматер отчета автоматически генерирует структуру данных ОТЧЕТ (REPORT), которая работает с форматированием страниц и контролирует переполнение страниц.



Установите имя отчета, единицу измерения по умолчанию и ориентацию страницы в диалоговом окне Report Properties (свойства отчета).



Установите в диалоге позиции Position размер полей по умолчанию.



Заселите отчет полями, переменными, линиями и графикой. Сконструируете заголовки, тело отчета и колонтитулы.



Сортировка и подсчет итогов с помощью структуры группы BREAK.

Как пользоваться данной главой  
Процессор отчетов Clarion  
Интерфейс формatera отчета  
Открытие формatera отчета  
Представление отчета в виде полос  
Панели инструментов  
Меню  
Структуры и свойства отчета  
Возможности формatera отчета  
Создание процедуры отчета  
Определение файлов и ключей (порядка сортировки)  
Определение размера и ориентации страницы  
Определение полей страницы отчета  
Позиционирование и выравнивание  
Создание формы отчета  
Определение заголовков и колонтитулов  
Определение заголовков колонок и заголовка отчета  
Определение полей для печати  
Определение групповых прерываний  
Определение поведения групповых прерываний  
Создание итоговых и вычисляемых полей  
Показ номера страницы  
Показ даты печати  
Применение предварительного просмотра  
Печать бирок (динамически изменяемая)  
Печать одной записи на страницу  
Печать слитых документов  
Печать графических элементов  
Печать многострочного текста с переносом слов  
Отчеты, которые выглядят как окна

Используя Форматер отчета, вы визуальнo проектируете отчеты вашего приложения. Форматер отчета автоматическi генерирует и размещает все кодовые структуры, необходимые для изготовления отчетов. Вы можете предварительнo просмотреть отчеты без реального генерирования какого-либо кода или данных.

Эта глава расскажет вам:

- Как пользоваться данной главой
- Как работает процессор печати Clarion
- О каждой характеристике Формatera отчета
- Как получить различные эффекты оформления отчета

## ***Как пользоваться данной главой***

Clarion обладает многими характеристиками, необходимыми для составления отчета, и мы хотим их все по порядку изложить. Однако, вам, разработчику, может быть нужно было подготовить какой-нибудь отчет еще вчера! Может быть поэтому вы хотели бы прочитать только о тех характеристиках, которые помогут вам создать ваш собственный отчет прямо сейчас. Следовательно, эта глава дает вам обзор странично-ориентированного процессора печати Clarion'a, современное трактование каждой характеристики Форматера отчета, а также справки типа “как сделать” для получения обычных эффектов отчета и таких функций, как сортировка, подсчет итога, переход на новую страницу и т.д.

### **Интерфейс форматера отчета**

Раздел данной главы под названием Ссылки интерфейса пользователя систематически описывает каждую характеристику Форматера отчета по мере того, как вы встречаетесь с ней на вашем экране. Раздел Структуры и свойства отчета обсуждают каждую структуру отчета, ее функцию и диалоговые окна, использованные для манипуляции ими.

### **Возможности форматера отчета**

Раздел Справки Задача/Отчет данной главы предоставляет список функций отчета и эффектов, которые разработчики часто или от случая к случаю просят предоставить. В нем дается общее описание способа создания специфических эффектов отчета, а также ссылки на те части Форматера отчета, которые необходимы для получения конкретного эффекта.

Данный раздел также предлагает удобную последовательность шагов построения отчета. Например, мы обсуждаем характеристики, которые определяют файлы и ключи, а также устанавливаем размер страницы, ориентацию и поля до того, как придет время характеристикам, которые макетируют другие части отчета.

Список эффектов не является всеобъемлющим, мы пытаемся быстро ориентировать вас в исполнении базовых функций отчета. Как только вы поняли, как создавать эти основные эффекты, вы будете прочно стоять на пути создания действительно впечатляющих специальных эффектов для своих отчетов.

Другие примеры создания разнообразных эффектов и характеристик отчетов вы можете

найти в главе Создание отчетов книги "Быстрый Старт".

## ***Процессор печати Clarion***

Прежде, чем узнать, как создать отчет с помощью Формatera отчета, важно понять, как Clarion выполняет отчет - иными словами, как происходит разделение труда между процессором печати и вашей исходной программой, а также каков порядок, в котором процессор печати обрабатывает различные разделы вашего отчета.

Как правило в вашей исходной программе должны быть только структура данных REPORT (отчет) и соответствующие файл ввода/вывода и оператор PRINT. Структура данных ОТЧЕТ генерируется для вас Форматером отчета, а Генератор приложений обрабатывает ввода/вывод файла и оператор PRINT, если, однако, вы не предпочитаете кодировать вручную.

### **Структуры REPORT (отчет)**

Структура данных REPORT содержит всю информацию, необходимую для форматирования и печатания каждой страницы отчета. Структура данных REPORT может содержать пять подструктур: FORM, HEADER, DETAIL FOOTER и BREAK (бланк, заголовок, тело отчета и нижний колонтитул, прерывание). Каждая структура BREAK может содержать свой собственный HEADER, вложенные BREAK, DETAIL и FOOTER.

Названия этих структур несут традиционные смысловые позиционные нагрузки, однако Clarion for Windows обеспечивает совершенную гибкость в позиционировании FORM, страничного HEADER и страничных структур FOOTER. Более того, прерывания BREAKs, заголовки HEADERs прерывания, колонтитулы FOOTERs прерывания и структуры DETAIL могут печатать в любом месте в пределах области печати тела отчета, области, определенной атрибутом ОТЧЕТА AT. Атрибут AT отчета REPORT и область печати тела отчета могут быть назначены с помощью диалогового окна Report Properties (свойства отчета) или Report Formatter's Page Layout View (вид макета страницы форматера отчета).

### **Порядок обработки**

То, что вам следует помнить обо всех этих структурах - это не столько то, где они печатаются (они могут быть напечатаны всюду, где вы их поместите), сколько то, когда они составлены, то есть, порядок, в котором они составляются при печати отчета.

Если вы знаете порядок, в котором разделы отчета генерируются во время печати, вы

можете лучше использовать их. Размещение на странице не влияет на порядок, в котором процессор печати генерирует разделы отчета. Вы можете поместить колонтитул FOOTER страницы выше тела отчета DETAIL, а так как FOOTER страницы генерируется только после того, как процессор печати обработает все DETAIL на странице, вы можете поместить итог страницы наверху этой страницы!

Следующая процедура иллюстрирует порядок, в котором составляется каждая структура отчета. Положим, что отчет содержит одну структуру прерывания BREAK с разделом тела отчета DETAIL внутри нее:

1. Процессор печати составляет FORM, но не отправляет ее еще в буфер печати. Раздел FORM генерируется только один раз; процессор печати не составляет его заново для дополнительных страниц.
2. Процессор печати составляет заголовок HEADER страницы.
3. Процессор печати составляет групповой HEADER для первой группы.
4. Процессор печати составляет раздел DETAIL столько раз, сколько это необходимо для того, чтобы заполнить первую страницу.

Если на странице происходит прерывание BREAK:

5. Процессор печати составляет групповой FOOTER для первой группы.
6. Процессор печати составляет групповой HEADER для следующей группы.
7. Процессор печати генерирует раздел DETAIL для следующей группы записей, непрерывно проверяя, нет ли других групповых прерываний BREAKs.

Внизу страницы:

8. Процессор печати проверяет наличие вдов, увеличивает номер страницы и проверяет следующие страницы на наличие сирот.
9. Процессор печати составляет FOOTER страницы.
10. Процессор печати посылает всю страницу в буфер печати.
11. За исключением шага 1, процессор печати повторяет эту процедуру для всех дополнительных страниц.

## **Интерфейс формatera отчета**

### **Открытие формatera отчета**

---

Вы можете получить доступ к Форматеру отчета из генератора приложений или из текстового редактора.

☐ Чтобы открыть Форматер отчета из генератора приложений:

1. Из диалогового окна Procedure Properties нажмите кнопку Report (отчет).

Появляется диалоговое окно Report Formatter (форматер отчета). Теперь вы готовы определить один за другим разделы отчета.

☐ Чтобы открыть Report Formatter из редактора теста:

1. Откройте файл текста исходной программы.

2. Поместите пустую строку в раздел данных и здесь же поместите курсор. Именно здесь форматер отчета размещает структура REPORT языка Clarion.

3. Выберите Edit>Format Structure из меню.

Вы можете нажать ускоряющую клавишу CTRL+F.

4. Когда появится диалоговое окно New Structure (новая структура), выберите Report.

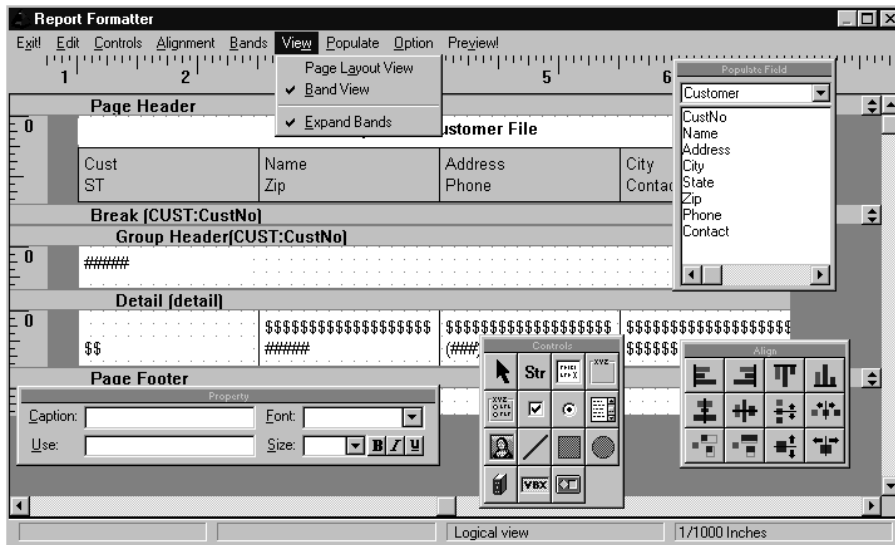
Появляется форматер отчета. Теперь вы готовы определить разделы отчета, установить заголовки, тело отчет и нижние колонтитулы, а также добавить элементы управления.

**Совет: Чтобы отредактировать существующий отчет из текстового редактора, откройте файл исходной программы и поместите курсор на любую строку в пределах структуры REPORT, затем выберите из меню Edit>Format Structure или нажмите ctrl+f.**

### **В виде полос**

---

Когда вы впервые открываете форматер отчета, ваш отчет появляется в виде полос (Band View). А именно, каждый раздел ОТЧЕТА (HEADER, BREAK, DETAIL, FOOTER и FROM) появляется в отдельной “полосе” внутри окна. Это справедливо несмотря на то, что разделы отчета могут в действительности перекрываться при печати.



### Линейки

□ Для того чтобы быстро определить позицию каждого раздела, посмотрите на линейки. Горизонтальная (ось X) линейка показывает положение относительно левого края страницы. Вертикальные (ось Y) линейки показывают положение относительно верха каждой из полос. Единицы измерения установлены в диалоговом окне Report Properties (свойства отчета) (см ниже Структуры и свойства форматера отчета).

### Полосы заголовка

□ Каждый раздел отчета имеет свою собственную полосу заголовка. Каждая полоса заголовка показывает тип полосы и справа кнопку свертки/развертки полосы. Полосы заголовка раздела прерывания BREAK также показывают имя переменной раздела, на которой прерывается данный раздел.

### Кнопка показать/скрыть

□ Чтобы развернуть или свернуть данную полосу отчета, щелкните клавишей мыши над кнопкой свертки/развертки расположенную на правом конце полосы заголовка.

## Панели инструментов

### Панель инструментов “Элементы управления”

Панель инструментов Controls (элементы управления) работает подобно палитре инструментов рисования, таких как панель инструментов в принадлежностях Windows

Paintbrush. Достаточно щелкнуть клавишей на пиктограмме панели инструментов, а затем щелкнуть внутри той полосы, на которой вы хотите разместить данный элемент управления. Верхний левый угол этого элемента управления размещается при щелчке мыши на перекрестии курсора.

Панель инструментов Controls (элементы управления) появляется по умолчанию при запуске Форматера отчета. Спрятать или снова показать панель инструментов Элементы управления можно, выбирая Option > Show Toolbox. Изменить размер панели инструментов можно, поместив курсор на ее краю, а затем “щелкнув клавишей и протаскив” до нужного размера. Перемещать панель инструментов целиком можно, протаскивая ее за полосу заголовка.



- Без инструмента Строка символов Текстовое поле Групповое поле
- Поле опций Поле флажков Радиокнопка Списочное поле
- Изображение Линия Поле Эллипс
- Поле словаря VBX Шаблон элемента управления

Все элементы управления на панели инструментов доступны из меню Controls (элементы управления) и меню Populate (заселять).

**Совет:** Поместите курсор над любым инструментом и подождите полсекунды. появится подсказка инструмента, называющая вам тип элемента управления, который будет создан этим инструментом.



Отменяет текущий выбор элемента управления



Размещает в создаваемом отчете элемент управления STRING (строка символов). См. Элементы управления и их свойства - Свойства Строка.



Размещает в создаваемом отчете элемент управления TEXT (текст). См Элементы управления и их свойства - Свойства элемента управления Текстовое поле.



Размещает в создаваемом отчете элемент управления GROUP (групповое поле). См. Элементы управления и их свойства- Свойства элемента управления Групповое поле..





Размещает в создаваемом отчете элемент управления OPTION (структура OPTION, которая появляется как групповое поле с радиокнопками). См. Элементы управления и их свойства - Свойства элемента управления Поле опций.



Размещает в создаваемом отчете элемент управления CHECK ( поле флажка). См. Элементы управления и их свойства- Свойства элемента управления Поле флажка.



Размещает в создаваемом отчете элемент управления RADIO. См. Элементы управления и их свойства - Свойства элемента управления Радиокнопка.



Размещает в создаваемом отчете элемент управления LIST (списочное поле или ниспадающее поле списка). См. Элементы управления и их свойства - Создание списочных полей.



Размещает в создаваемом отчете элемент управления IMAGE (изображение). См. Элементы управления и их свойства - Свойства элемента управления Изображение.



Размещает в создаваемом отчете элемент управления LINE (линия). См. Элементы управления и их свойства - Свойства элемента управления Линия.



Размещает в создаваемом отчете элемент управления BOX (поле). См. Элементы управления и их свойства - Свойства элемента управления Поле.



Размещает в создаваемом отчете элемент управления ELLIPSE (эллипс). См. Элементы управления и их свойства - Свойства элемента управления Эллипс.



Дает вам возможность выбрать поле, определенное в словаре данных, и помещает назначенный ему элемент управления в создаваемое окно.



Размещает в создаваемом отчете элемент управления VBX ( элемент управления Visual Basic). См Элементы управления и их свойства - Свойства элемента управления VBX.



Дает вам возможность добавить к вашему отчету один или более элементов управления вместе с ассоциированной исходной программой. Clarion не предоставляет шаблоны элементов управления с данным продуктом, однако вы можете написать свой собственный или купить шаблоны у других изготовителей..

### **Панель инструментов “Заселение поля”**

Форматер отчета содержит плавающую панель Populate Field (заселение поля). Эта

панель инструментов дает вам возможность быстро “заселить” отчет полями ваших файлов словаря данных.

Для того, чтобы заселить поле, выберите файл из ниспадающего списка, затем дважды щелкните на поле, которое вы хотите иметь на своем отчете. Поле автоматически размещается и выравнивается. Для размещения поля вручную щелкните один раз клавишей мыши на этом поле, затем щелкните в желаемом месте размещения. Тип элемента управления (строка, поле флажков, радиокнопка и т.д.) определяется установками, назначенными для данного конкретного поля в словаре данных.

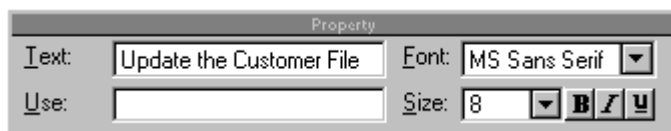


Показать или спрятать панель инструментов Populate Field (заселение поля) можно, выбирая Option>Fieldbox. Изменить размер панели инструментов Заселение полей можно, поместив курсор на границе этой панели. Когда курсор изменит свою форму на двустороннюю стрелку, выполните процедуру “щелкнуть и протаскать”. Перемещать панель инструментов целиком можно, щелкнув и протаскивая ее за полосу заголовка.

Вы можете также заселить свой отчет из файлов с помощью меню Populate (заселить) или с помощью инструмента полей словаря на панели инструментов Controls (элементы управления).

### **Панель инструментов “Свойства”**

Панель инструментов Report Formatter’s Property (свойства форматера отчета) дает вам возможность быстро установить вид и содержание текста для каждого элемента управления в пределах отчета. Управляйте шрифтами, размером, стилем и содержанием всего текста отчета с помощью кнопок и выпадающих списков стандартной системы подготовки текстов.



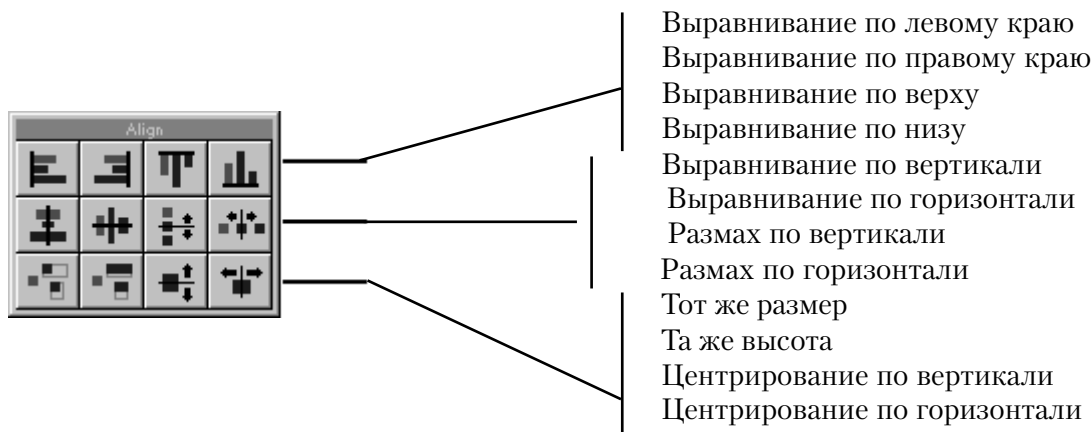
В поле Caption (заголовок) наберите текст элемента управления или шаблон изображения. Смотрите Элементы управления и их свойства - Общие атрибуты элементов управления - Редактор шаблона изображения, где имеется более полная информация о шаблонах изображения.

Вывести панель инструментов Property на экран или спрятать ее вы можете используя Options> Show Propertybox (показать панель свойств). Размеры панели инструментов вы можете менять, помещая курсор на границу панели. Когда курсор становится стрелкой с двумя наконечниками, нажимайте клавишу мыши и тащите. Перемещается панель инструментов с помощью щелчка на полосе заголовка и протаскивания за него.

### Панель инструментов “Выравнивание”

Панель инструментов Report Formatter’s Align (настройка форматера отчета) дает вам возможность быстро, профессионально и точно настроить элементы управления в вашем отчете. Выберите элементы управления для настройки (CTRL+ щелчок позволят выбрать несколько элементов управления, вы можете также выбрать несколько элементов управления с помощью CTRL+ТАЩИТЬ), затем щелкните на соответствующем инструменте выравнивания. Все операции выравнивания доступны также из меню Alignment (выравнивание).

**Совет:** Для большинства функций выравнивания первые выбранные элементы настройки (синие рукоятки) настраиваются по последнему выбранному элементу управления (красные рукоятки ). Это значит, что последний выбранный элемент управления является базовым элементом управления. Он не движется, движутся другие.



Показывайте или прячьте панель инструментов Align путем выбора Options> Show Alignbox (показать панель выравнивания). Размеры панели инструментов вы можете менять, помещая курсор на границу панели. Когда курсор становится стрелкой с двумя наконечниками, нажимайте клавишу мыши и тащите. Перемещается панель инструментов с помощью щелчка на полосе заголовка и протаскивания за него.

**Совет:** Поместите курсор над каким-либо инструментом и подождите

**полсекунды. Появится подсказка инструмента, которая назовет вам тип выравнивания, которое выполнит данный инструмент.**



Выравнивает левые границы выбранных элементов управления с левой границей последнего выбранного элемента управления (красные квадратики - рукоятки ).



Выравнивает правые границы выбранных элементов управления с правой границей последнего выбранного элемента управления (красные квадратики - рукоятки ).



Выравнивает верхние границы выбранных элементов управления с верхней границей последнего выбранного элемента управления (красные квадратики - рукоятки ).



Выравнивает нижние границы выбранных элементов управления с нижней границей последнего выбранного элемента управления (красные квадратики - рукоятки ).



Вдоль вертикальной оси выравнивает центры выбранных элементов управления с центром последнего выбранного элемента управления (красные квадратики - рукоятки ).



Вдоль горизонтальной оси выравнивает центры выбранных элементов управления с центром последнего выбранного элемента управления (красные квадратики - рукоятки ).



Уравнивает вертикальные промежутки между выбранными элементами управления.



Уравнивает горизонтальные промежутки между выбранными элементами управления.



Делает все выбранные элементы управления одинаковыми по высоте и ширине с последним выбранным элементом управления (красные квадратики - рукоятки ).



Делает все выбранные элементы управления одинаковыми по высоте с последним выбранным элементом управления (красные квадратики - рукоятки ).



Как группу (относительные позиции выбранных элементов управления не изменяются) центрирует выбранные элементы управления вертикально внутри полосы отчета.



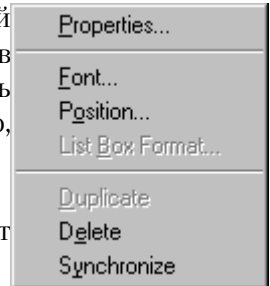
Как группу (относительные позиции выбранных элементов управления не

изменяются) центрирует выбранные элементы управления горизонтально внутри полосы отчета.

## Меню

### Всплывающее меню (POPUP MENU)

Получите доступ к всплывающему меню, щелкнув для этого правой клавишей по полосе или элементу управления. Всплывающее меню в Форматере отчета дает вам возможность манипулировать и настраивать разделы отчета и элементы управления в них, в зависимости от того, какие из них выбраны.



- Чтобы выбрать элемент управления, поместите курсор на этот элемент управления и щелкните правой клавишей мыши.
- Чтобы выбрать раздел отчета, поместите курсор куда-либо на эту полосу, но не на других элементах управления, затем щелкните правой клавишей мыши.

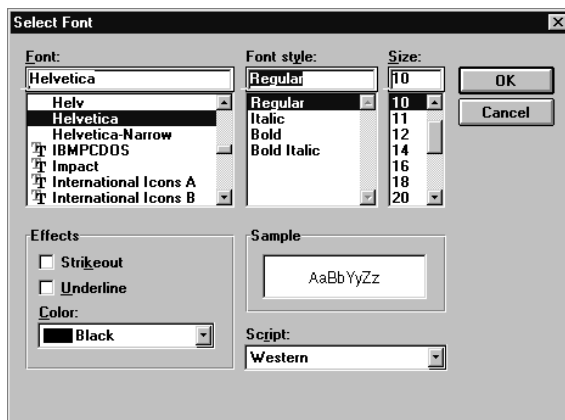
**Совет: Многие команды всплывающего меню доступны также на меню редактирования форматера отчета.**

Далее приводится описание выборов по желанию этого всплывающего меню.

Properties (свойства)	Открывает диалоговое окно свойств для выбранного раздела отчета или элемента управления. Смотрите обсуждение выбранной позиции для получения более полной информации о ее свойствах.
Font (шрифт)	Вызывает диалоговое окно Select Font (выбор шрифта), которое даст вам возможность выбрать шрифт (начертание шрифта), размер, стиль (такой как жирный или курсив), цвет, а также шрифтовые эффекты (подчеркивание и перечеркивание) для всех элементов управления в данном разделе отчета. Вы можете отменить шрифт раздела путем установки другого шрифта в диалоговом окне свойств для любого конкретного элемента управления. Когда вы делаете свой выбор, диалоговое окно показывает образец отобранного шрифта.
Position (положение)	Открывает диалоговое окно свойств на закладке Position (положение). Смотрите обсуждение выбранной позиции для получения более полной информации о ее свойствах. В общем случае вы можете также изменять положение элементов с помощью щелчка клавиши мыши и протаскивания, а также с помощью инструментов выравнивания.
List Box Format (формат	

поля списка)

Назначает внешний вид и функциональные возможности элемента управления “Поле списка”. Дополнительную информацию смотрите в главе Использование формatera поля списка.



Установка шрифтов для текста вашего отчета.

Duplicate (копировать)

Копирует текущий отобранный элемент управления в верхний левый угол того же самого раздела отчета.

Delete (удалить)

Удаляет выбранный элемент управления или полосу отчета. Это можно сделать и другим способом: выберите элемент управления и затем нажмите клавишу DELETE.

Synchronize

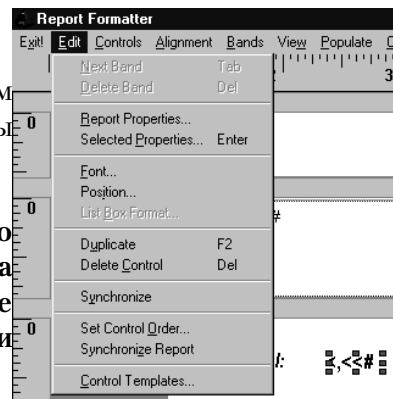
(синхронизировать)

Присваивает атрибуты данного элемента управления, назначенные для него в словаре данных, или, если выбрана полоса, всем элементам управления в данной полосе. Атрибуты присваиваются так, как установлено в закладке Synchronize (синхронизировать) диалогового окна Application Options (опции приложения). Смотрите Генератор приложений - Установка опций приложения.

## **Меню редактирования**

Меню Edit (редактирование) в Форматере отчета дает вам возможность манипулировать и настраивать отчет и элементы управления в этом отчете.

**Совет:** Многие команды меню редактирования доступны также на всплывающем меню, к которому вы можете получить доступ щелкая правой клавишей мыши на элементе управления или полосе отчета.



Next Band (следующая полоса)	В этой версии не реализовано.
Delete Band (удалить полосу)	Удаляет отобранную полосу отчета. Или же вы можете отобрать ее и нажать клавишу DELETE. Это удаляет полосу и все элементы управления в ней.
Report Properties (свойства отчета)	Открывает диалоговое окно Report Properties (свойства отчета).
Selected properties (выбранные свойства)	Открывает диалоговое окно свойств для отобранного раздела или элемента управления. Смотрите обсуждение выбранной позиции для получения более полной информации о ее свойствах.
Font (шрифт)	Вызывает диалоговое окно Select Font (выбор шрифта), которое даст вам возможность выбрать шрифт (начертание шрифта), размер, стиль (такой как жирный или курсив), цвет, а также шрифтовые эффекты (подчеркивание и перечеркивание) для всех элементов управления в данном разделе отчета. Вы можете отменить шрифт раздела путем установки другого шрифта в диалоговом окне свойств для любого конкретного элемента управления. Когда вы делаете свой выбор, диалоговое окно показывает образец отобранного шрифта.
Position (положение)	Открывает диалоговое окно свойств на закладке Position (положение). Смотрите обсуждение отобранной позиции для получения более полной информации о ее свойствах. В общем случае вы можете также изменять положение элементов с помощью щелчка клавиши мыши и протаскивания, а также с помощью инструментов выравнивания.
List Box Format (формат поля списка)	Назначает внешний вид и функциональные возможности элемента управления “Поле списка”. Дополнительную информацию смотрите в главе Использование форматера поля списка.
Duplicate (копировать)	Копирует текущий отобранный элемент управления в верхний левый угол того же самого раздела отчета.
Delete (удалить)	Удаляет выбранный элемент управления или полосу отчета. Это можно сделать и другим способом: выберите элемент управления и затем нажмите клавишу DELETE.
Synchronize (синхронизировать)	Присваивает атрибуты данного элемента управления, назначенные для него в словаре данных, или, если

выбрана полоса, всем элементам управления в данной полосе. Атрибуты присваиваются так, как установлено в закладке Synchronize (синхронизировать) диалогового окна Application Options (опции приложения). Смотрите Генератор приложений - Установка опций приложения.

Set Control Order (установить порядок следования элементов управления)

Это открывает диалоговое окно Order Control (порядок управления), который показывает все элементы управления в отчете в иерархическом списке. Переставьте элементы управления выбирая элемент управления и нажимая кнопки **б** и **и** в для перемещения элемента управления вверх и вниз по списку.

**Совет:** Если один элемент управления перекрывает другой, например, текст перекрывает поле, выберите Edit>Set Control Order для того, чтобы обеспечить, что ниже лежащий элемент управления печатается перед элементом управления, лежащим поверх; в противном случае вышележащий элемент управления может выглядеть неясным.

Synchronize Report (синхронизировать отчет)

Присваивает атрибуты элемента управления, назначенные для него в словаре данных, всем элементам управления в данном отчете. Атрибуты присваиваются так, как установлено в закладке Synchronize (синхронизировать) диалогового окна Application Options (опции приложения). Смотрите Генератор приложений - Установка опций приложения.

Control Templates (шаблоны управления)

Чтобы добавить шаблоны расширений к данной процедуре, или для того, чтобы отредактировать приглашения элемента управления и шаблона расширения, выберите команду Control Templates (шаблоны элемента управления). Это открывает диалоговое окно Extension and Control Templates (шаблоны расширений и элементов управления), где вы можете добавить шаблоны расширений или же вы можете отредактировать любой шаблон элемента управления и расширения. Смотрите главу Шаблоны элементов управления, программы и расширений, где приводится более подробная информация.

### **Меню элементов управления (CONTROLS)**

В меню элементов управления Controls (элементы управления) перечисляются те элементы управления, которые появляются на панели инструментов “Элементы управления”, за исключением шаблона элемента управления и полей словаря (Смотрите



Меню заселения). Исполнение команды из меню “Элементы управления” идентично щелканью клавишей мыши на соответствующей пиктограмме панели инструментов. Меню служит для удобства.

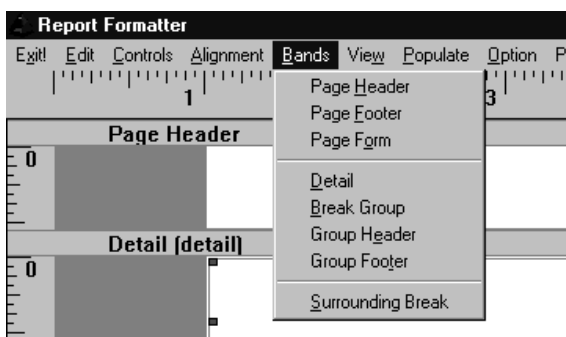
Смотрите выше раздел Панели инструментов. Смотрите также Элементы управления и их свойства.

### **Меню выравнивания**

Меню элементов управления Alignment (выравнивание) перечисляются те инструменты выравнивания, которые появляются в панели инструментов Выравнивание. Исполнение команды из меню Выравнивание идентично щелканью клавиши мыши на соответствующей пиктограмме панели инструментов. Смотрите вышеприведенный раздел Панели инструментов.

### **Меню полос**

Меню полос дает вам возможность добавлять к вашему отчету новые разделы или полосы, Сделайте выбор из нижеследующего:



Page Header (заголовок страницы)

Добавляет полосу заголовка страницы к вашему отчету. Структура HEADER - это первый формируемый раздел данной страницы. Обычно вы помещаете в HEADER титул отчета, графику и другие “вводные” элементы.

Page Footer (нижний колонтитул страницы)

Добавляет полосу нижнего колонтитула страницы к вашему отчету. Структура FOOTER является последним формируемым разделом данной страницы. Обычно вы помещаете в FOOTER номер страницы и ее итоги.

Page Form (форма страницы)

Добавляет полосу формы (бланк) к вашему отчету. Структура FORM составляется однажды и остается постоянной от страницы к странице. Обычно вы показываете “перекрытия” или фиксированные данные, такие как графику и метки полей в слое FORM, затем печатаете переменные данные в других полосах.

**Detail (тело отчета)**

Добавляет строки тела отчета к вашему отчету. Структура DETAIL является “телом” отчета. Она содержит нижний уровень данных для печати.

**Break Group (групповое прерывания)**

Добавляет к вашему отчету новое тело отчета, прерывание, групповой заголовок и групповой нижний колонтитул. Поместите перекрестие туда, где вы хотите чтобы появилась новая группа полос, и щелкните клавишей мыши. Появится диалоговое окно Break Properties (свойства прерывания). Установите переменную, на которой должно быть прерывание и нажмите ОК.

**Group Header (групповой заголовок)**

Добавляет групповой заголовок к существующему прерыванию. Поместите перекрестие на линейку заголовка прерывания, который вы хотите модифицировать, и щелкните клавишей мыши. Появится диалоговое окно Page/Group Header Properties (свойства заголовка страницы/группы).

**Group Footer (Групповой нижний колонтитул)**

Добавляет полосу нижнего колонтитула к существующему прерыванию. Поместите перекрестие на линейку заголовка прерывания, которое вы хотите модифицировать, и щелкните правой клавишей мыши. Появится диалоговое окно Page/Group Footer Properties (свойства страничного/группового нижнего колонтитула).

**Surrounding Break (окружающее прерывание)**

Добавляет прерывание вокруг существующего тела отчета. Поместите перекрестие на тело отчета, которое вы хотите модифицировать, затем щелкните клавишей мыши. Появится диалоговое окно Break Properties (свойства прерывания). Установите переменную, на которой должно быть прерывание, и нажмите ОК.

**Меню видов представления отчета на экране форматера отчета**

Меню видов представления View дает вам возможность переключаться между Band View (представление в виде полос) и Page Layout View (вид представления макет страницы), а также спрятать или показать все полосы сразу.

**Page Layout View (вид представления макет страницы)**

Точно показывает положение разделов отчета на образце страницы. Дает вам возможность перемещать и изменять размер заголовка страницы, колонтитула страницы, формы, а также область печати тела отчета с помощью операции “щелкнуть и тащить”.

**Band View (представление в виде полос)**

Показывает каждый раздел отчета в отдельной полосе без перекрытия. Дает вам возможность добавлять к вашему отчету элементы

управления и установить их свойства.

Expand Bands (развернуть полосы)

Сжимает или разворачивает все полосы сразу.

### **Меню заселения**

Меню заселения Populate Menu появляется в форматере окна только тогда, когда генератор приложений активен. Оно помещает соответствующий элемент управления в отчет для показа отобранного поля или переменной памяти.

Dictionary Field (поле словаря)

Дает вам возможность поместить элемент управления строка, связанный с полем словаря данных или переменной памяти. Когда вы щелкаете клавишей мыши в отчете, появляется диалоговое окно Select Field (выбор поля). Выберите поле или переменную, затем щелкните клавишей мыши в отчете.

Если вы предварительно форматировали поле, на закладке Report (отчет) диалогового окна Свойства поля (например, устанавливая текстовый элемент управления), помещается назначенный вами элемент управления, в противном случае помещается строка символов.

Multiple Fields (множество полей)

То же самое, что и Dictionary Field (поле словаря), но в форме повтора. Нажмите кнопку Cancel (отменить) после того, как вы поместили последнее поле.

Control Template (шаблон элемента управления)

Дает вам возможность добавить один или более элементов управления к вашему отчету вместе с соответствующей исходной программой. Clarion не предоставляет шаблоны элементов управления с настоящим продуктом, однако вы можете написать свой собственный или приобрести шаблон у других изготовителей.

### **Меню опций**

Помимо всего прочего, меню Options (опции) дает вам возможность показать и спрятать различные инструменты и панели инструментов форматера отчета.

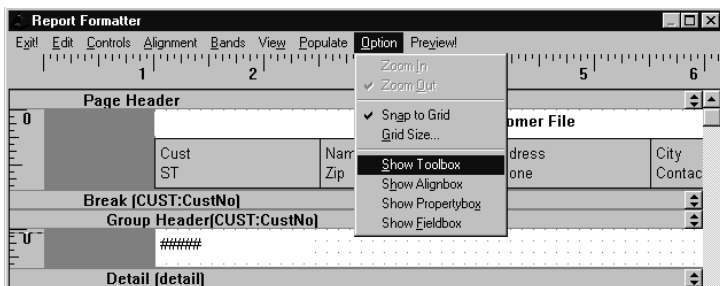
Zoom In (раскрыть) Увеличивает “обзор” в режиме предварительного просмотра.

Zoom out (закрыть) Уменьшает “обзор” в режиме предварительного просмотра.

Snap to Grid (выравнивание по сетке)

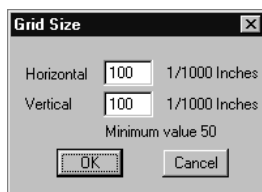
Чтобы переключить выравнивание по сетке в положение включено или выключено, выберите команду Snap to Grid (выравнивание по сетке). Выравнивание по сетке вынуждает верхний левый угол новых

элементов управления настраиваться на совпадение с узлом сетки в окне. Конечный пользователь не видит сетки во время работы; это только конструкторский инструмент.



Grid Size (размер сетки)

Чтобы установить величины размера сетки, выберите команду Grid Size (размер сетки). Вы можете ввести разные значения для осей X и Y. Чтобы установить расстояние между точками сетки, введите значения ширина и высота в диалоге размер сетки.



Show Toolbox

(показать панель инструментов)

Показывает или прячет панель инструментов  
Элементы управления.

Show Alignbox (показать панель  
выравнивания)

Показывает или прячет панель инструментов  
Выравнивание.

Show Propertybox (показать  
панель свойств)

Показывает или прячет панель инструментов  
Свойства.

Show Fieldbox (показать панель  
заселение поля)

Показывает или прячет панель инструментов  
Заселение поля.

### Предварительный просмотр

Предварительный просмотр (Preview!) дает вам возможность поэкспериментировать с различными форматами отчета без реального компилирования и прогонки отчета. Вы можете быстро “посмотреть” альтернативные планы DETAILS, BREAKs, HEADERS, FOOTERS, вы можете также оценить результат действия выбранных вами прерываний страницы.

Форматер отчета поставляет данные для тестирования в формате, который вы установили. Показываются шрифты, размеры, цвета и положения элементов управления отчета.

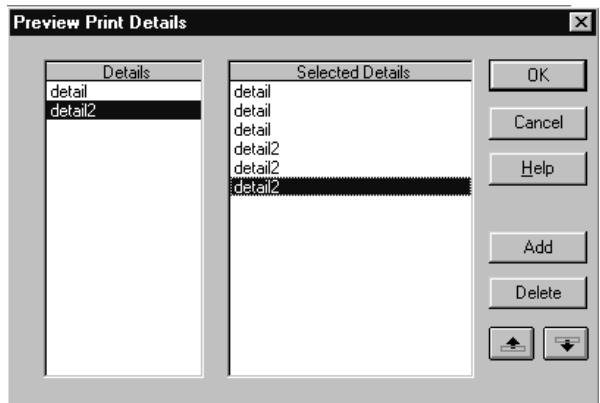
Чтобы генерировать модель отчета, в том виде, как его будет видеть ваш пользователь:

### 1. Выберите команду Preview!

Появляется диалоговое окно Preview Print Details (предпросмотр печати тела отчета). Это диалоговое окно дает вам возможность генерировать данные “заполнителя” для вашего отчета. Данные не имеют величин, они служат в качестве местодержателей, так что вы можете получить ощущение наличия завершенного отчета.

Если у вас более одного DETAIL, выделите один из них с левой стороны диалогового окна.

Экспериментирование с различными форматами и комбинациями форматов тела отчета.



2. Нажмите кнопку Add (добавить) для генерации данных заполнителя.

Генерируется столько, сколько вам нужно. Некоторые отчеты будут иметь только одну запись на страницу, другие будут иметь много записей. Вы можете добавить достаточно записей, чтобы переполнить страницу и просмотреть поведение прерывания вашего отчета.

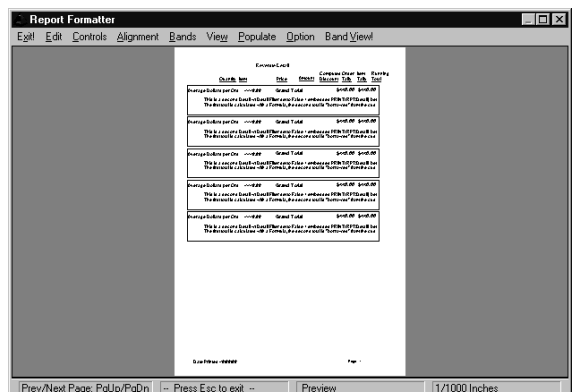
Вы можете даже смешать два или более DETAILS. Используйте кнопки “вверх” и “вниз” для перераспределения положения заполнителей DETAIL.

3. Нажмите кнопку OK для предпросмотра вашего отчета.

Предпросмотр страничных HEADER и групповых HEADER.

Предпросмотр DETAIL и поведения прерывания страницы.

Предпросмотр начертания шрифта,



размера, стиля и цвета.

Предпросмотр ориентации страницы

4. Выберите для увеличения масштаба обзора Option > Zoom.

5. Чтобы выйти из режима Preview!, нажмите ESC или нажмите BAND VIEW! (вид представления полосы)

Снова появится вид представления полосы форматера отчета.

## **Структуры и свойства отчета**

Структура REPORT (отчет) содержит всю информацию, необходимую для форматирования и печати каждой страницы отчета.

Ниже приводится пример структуры REPORT с пустыми верхними и нижними колонтитулами, бланком страницы, прерыванием на “CustNumber” и несколькими переменными строками символов в теле отчета. Эта структура была генерирована форматером отчета Report Formatter.

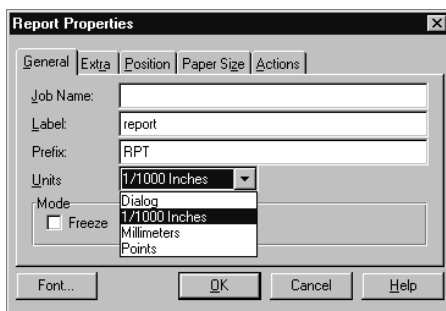
```
Report  REPORT,AT(1000,2000,6000,7000),PRE(RPT),FONT('Arial',10,,),THOUS
        HEADER,AT(1000,1000,6000,1000)
        END
CustBreak  BREAK(CUS:CustNumber)
           HEADER,AT(,,1000)
           END
Detail     DETAIL
           STRING(@n4),AT(125,52),USE(CUS:CustNumber)
           STRING(@S20),AT(125,208),USE(CUS:Company)
           STRING(@S20),AT(125,365),USE(CUS:CustAddress)
           STRING(@S20),AT(125,531),USE(CUS:City)
           STRING(@S2),AT(125,688),USE(CUS:State)
           STRING(@n5),AT(125,844),USE(CUS:ZipCode)
           END
           END
           FOOTER,AT(1000,9000,6000,1000)
           END
           FORM,AT(1000,1000,6000,9000)
           END
END
```

## **Свойства отчета**

Имеются два диалоговых окна Report Properties (свойства отчета). Одно из этих диалоговых окон ассоциировано с шаблоном процедуры отчета. К этому диалоговому окну можно получить доступ с помощью кнопки Report Properties в диалоговом окне Procedure Properties (свойства процедуры); окно Report Properties обсуждается в Мастера и Шаблоны

процедур - Шаблон отчета.

Другое диалоговое окно Report Properties ассоциировано с каждым отчетом, независимо от того, использован или нет шаблон процедуры. Это диалоговое окно может быть открыто из Report Formatter (форматера отчета) при выборе Edit>Report Properties, оно обсуждается здесь. Это диалоговое окно дает вам возможность установить базовые опции отчета, включая такие, как ориентация страницы, единицы измерения, область печати тела отчета и размер бумаги. Мы рекомендуем устанавливать эти опции перед макетированием других частей вашего отчета.



### **Закладка “Общие свойства” (General)**

Job Name (задание для печати)

Название задания для печати, как оно будет высвечено в приложении Windows Диспетчер печати.

Label (метка)

Наберите допустимую метку Clarion для поименования структуры данных ОТЧЕТ.

Prefix (префикс)

Назначает префикс метки для структуры ОТЧЕТ.

Units (единицы измерения)

Назначает единицы измерения по умолчанию для всех элементов управления, помещенных в отчет. Выберите диалоговые единицы, тысячные доли дюйма, миллиметры или пункты.

Freeze (замораживание)

“Замораживает” все элементы управления в данном отчете, так что последующие изменения словаря данных неприменимы к отчету. Вы можете отменить атрибут #Freeze для всех элементов управления или для отдельных элементов управления. Смотрите Генератор приложений - Установка опций приложения.

### **Закладка “Дополнительные возможности” (Extra)**

Preview (предварительный просмотр)

Устанавливает имя очереди QUEUE, которая сохраняет имена файлов для метафайлов (\*.WMF), генерированных для

предпросмотра печати. Смотрите атрибут PREVIEW (предварительный просмотр) в Справочнике языка.

Если вы используете шаблон Процедура отчета, вам нужно только отметить поле Print Preview (предварительный просмотр печати) в другом диалоговом окне Report Properties (свойства отчета) и оставьте это поле ввод пустым.

Colors (цвета) Устанавливает цвет текста или фона для отчета. Смотрите Общие атрибуты элементов управления - Установка атрибута ЦВЕТ.

### **Закладка Положение (Position)**



Определяет местоположение и размер области печати тела отчета для данного документа, заполняя для этого атрибут ОТЧЕТА AT. Все DETAILS (тело отчета), групповые HEADERS (верхние колонтитулы) и групповые FOOTERS (нижние колонтитулы) печатаются в пределах области печати тела отчета с отступом относительно напечатанной ранее позиции.

Единицы измерения для этих величин установлены в закладке General (Общая).  
Top Left Corner (верхний левый угол)

Для того, чтобы установить точную исходную точку для области печати тела вашего отчета относительно левого верхнего угла страницы, назначьте координаты Верхнего левого угла с помощью данного диалогового окна. Реально это устанавливает верхнее и левое поля для области печати тела вашего отчета. Эти установки могут быть также сделаны визуальным путем протаскивания области печати тела отчета и его границ в Report Formatter's Page Layout View (вид макета страницы форматера отчета).

Width/Height (ширина/высота)

Чтобы установить размер области печати тела отчета, выберите из следующих опций для ширины (Width) и высоты (Height).

**Совет: При изменении ориентации отчета с книжной на альбомную, или наоборот, вы одновременно должны изменить величины ширины и/или высоты в этом же диалоговом окне.**

Default (по умолчанию)

Устанавливает величину, основанную на формате бумаги.

Fixed (фиксированный)

Чтобы установить особый размер, отметьте поля фиксированные выборы.

### **Закладка "Формат бумаги" (Paper Size)**

Paper Size (формат бумаги)

Выберите из 40 стандартных размеров или же выберите Other (иной) для установки вашего собственного размера.



Width (ширина)	Устанавливает пользовательский размер ширины страницы в единицах, назначенных на закладке General (общие свойства).
Height (высота)	Устанавливает пользовательский размер высоты страницы в единицах, назначенных на закладке General (общие свойства).
Landscape (альбом)	Назначает ориентацию текста относительно бумаги. “Альбомная” (отмеченное поле) означает, что текст отчета выровнен параллельно большему размеру бумаги. “Книжный” (очищенное поле) означает, что текст отчета выровнен параллельно меньшему размеру бумаги.

**Совет:** Перед тем, как изменять формат бумаги или ориентацию на существующем отчете, передвиньте элементы управления и измените размер ФОРМЫ, ЗАГОЛОВКОВ, КОЛОНТИТУЛОВ и т.д. до соответствия их с новыми размерами страницы. Иначе эти позиции окажутся вне границ, доступных Форматеру Отчета.

### Закладка “Действия” Actions

Files (файлы)	Открывается доступ к схеме файлов для отчета.
Embeds (вставки)	Открывается доступ к диалоговому окну Вставки программного текста для данного отчета.

### Закладка “Шрифт” (Font)

Чтобы установить шрифт по умолчанию для всех элементов управления в данном отчете, нажмите кнопку Font (шрифт), затем выберите шрифт и стиль в диалоговом окне Select Font (выбрать шрифт). Вы можете переопределить назначение по умолчанию путем назначения иного шрифта в диалоговом окне Properties (свойства) для любого конкретного элемента управления. Опции, которые вы выбираете в диалоговом окне, становятся параметрами для атрибута ШРИФТ. Когда вы отбираете опции, диалоговое окно показывает образец форматирования.

### Форма (Form)

Чтобы назначить постоянный текст или графику, которые печатаются на каждой странице, поместите их в ФОРМУ. Процессор печати составляет ФОРМУ в начале задания печати; он не обновляет их с каждой новой страницей. Следовательно, ФОРМА не подходит для изменяющихся данных, и даже для номера страницы. Вы можете, однако, печатать поля из файла данных если вы хотите печатать одного и того же содержание поля на каждой странице.

Форма обычно перекрывает другие разделы и может быть использована в качестве слоя для графических рамок или другого “предпечатного” материала, на который накладываются данные из других разделов. Вы можете, например, использовать линии и поля для деления тела отчета на отсеки, группирующие данные для пользователя. Вы можете даже создать эффект “зеленой полосы” путем чередования серых и светло-зеленых цветных блоков. Другое применение ФОРМЫ - имитация водяного знака.

**Совет:** Для получения лучших результатов при использовании инструмента рисования для создания водяного знака на, например, принтере с разрешением порядка 300 точек на дюйм, установите заполнение для элемента водяного знака на 10% серого или светло-серого цвета. При более высоком уровне разрешения принтера попробуйте 20% серого.

Чтобы добавить форму к вашему отчету, выберите Bands> Page Form.

### **Свойства Формы**

Щелкните правой клавишей мыши на полосе формы, а затем выберите Properties (свойства) из всплывающего меню.

### **Закладка “Общие свойства” (General)**

- |                          |   |
|--------------------------|---|
| Use (метка соответствия) | Наберите метку соответствия для ссылки на ФОРМУ в вашей исходной программе.   |
| Freeze (замораживать)    | “Замораживает” все элементы управления на данной полосе, так что последующие изменения словаря данных неприменимы. Вы можете отменить атрибут #Freeze для всех элементов управления или для отдельных элементов управления. Смотрите Генератор приложений - Установка опций приложения. |

### **Закладка “Дополнительные возможности” (Extra)**

- |                |   |
|----------------|---|
| Colors (цвета) | Устанавливает цвет текста или фона для данной полосы. Смотрите Общие атрибуты элементов управления - Установка атрибута ЦВЕТ. |
|----------------|---|

### **Закладка Положение (Position)**

Дает вам возможность установить местоположение и размер формы путем заполнения атрибута AT. Единицы измерения для этих величин установлены на закладке General (общие свойства) диалогового окна Report Properties (свойства отчета).

- Top Left Corner (верхний левый угол)

Для того, чтобы установить точную исходную точку для вашей формы относительно левого верхнего угла страницы, назначьте координаты Верхнего левого угла с помощью данного диалогового окна. Реально это устанавливает верхнее и левое поля для вашей формы. Эти установки могут быть также сделаны визуально путем перемещения формы и ее границ в Report Formatter's Page Layout View (вид макета страницы форматера отчета).

Width/Height (ширина  
/высота)

Чтобы установить размер формы страницы, выберите из следующих опций для ширины (Width) и высоты (Height).

Default (по умолчанию)

Устанавливает величину, основанную на формате бумаги.

Fixed (фиксированный)

Чтобы установить особый размер, отметьте фиксированные выборы.

**Совет: При изменении ориентации отчета с книжной на альбомную, или наоборот, вы одновременно должны изменить величины ширины и/или высоты в этом же диалоговом окне.**

### Закладка “Действия” Actions

Files (файлы)

Открывается доступ к схеме файлов для отчета.

Embeds (вставки)

Открывается доступ к диалоговому окну Вставки программного текста для данного отчета.

### Закладка “Шрифт” (Font)

Вызывает диалоговое окно Select Font (выбрать шрифт), которое дает вам возможность отобрать шрифт (начертание шрифта), размер, стиль (например, жирный или курсив), цвет и эффекты шрифта (подчеркивание и перечеркивание для всех элементов управления в разделе отчета. Вы можете отменить шрифт раздела путем назначения иного шрифта в диалоговом окне Properties (свойства) для любого конкретного элемента управления. Когда вы отбираете опции, диалоговое окно показывает образец выбранного шрифта.

## Заголовок страницы

Для того чтобы установить текст и данные для составления в самом начале каждой страницы, поместите их в страничный HEADER (заголовок). Помните, заголовок страницы может быть физически помещен в любое место страницы, а не только именно на верх.

Обычно HEADER включает название отчета и номер страницы. Это также полезное место для показа вашего фирменного знака.

Чтобы добавить заголовок страницы к вашему отчету, выберите Bands>Page Header.

### Свойства заголовка страницы

Щелкните правой клавишей на полосе заголовка, а затем выберите из всплывающего меню Properties (свойства).

### Закладка “Общие свойства” (General)

Use (метка соответствия)

Наберите метку соответствия для ссылки на HEADER (заголовок) в вашей исполняемой программе.

Freeze (замораживать)

“Замораживает” все элементы управления на данной полосе, так что последующие изменения словаря данных

неприменимы. Вы можете переопределить атрибут #Freeze для всех элементов управления или для отдельных элементов управления. Смотрите Генератор приложений - Установка опций приложения.

### **Закладка “Дополнительные возможности” (Extra)**

- Colors (цвета)      Устанавливает цвет текста или фона для данной полосы. Смотрите Общие атрибуты элементов управления - Установка атрибута ЦВЕТ.
- Alone (одиноким)      Не оказывает воздействия на Заголовок Страницы. Смотрите Групповой заголовок.
- Absolute  
(абсолютный)      Не оказывает воздействия на Заголовок Страницы. Смотрите Групповой заголовок.

### **Закладка Положение (Position)**

Дает вам возможность установить местоположение и размер заголовка страницы путем заполнения атрибута AT. Единицы измерения для этих величин установлены на закладке General (общие свойства) диалогового окна Report Properties (свойства отчета).

Top Left Corner (верхний левый угол)

Для того, чтобы установить точную исходную точку для вашего заголовка страницы относительно левого верхнего угла страницы, назначьте координаты Верхнего левого угла с помощью данного диалогового окна. Реально это устанавливает верхнее и левое поля для вашего заголовка страницы. Эти установки могут быть также сделаны визуально путем протаскивания заголовка страницы и его границ в Report Formatter's Page Layout View (вид макета страницы форматора отчета).

Width/Height (ширина/высота)

Чтобы установить размер заголовка страницы, выберите из следующих опций для ширины (Width) и высоты (Height).

Default (по умолчанию)

Устанавливает величину, основанную на формате бумаги.

Fixed (фиксированный)

Чтобы установить особый размер, отметьте фиксированные выборы.

**Совет: При изменении ориентации отчета с книжной на альбомную, или наоборот, вы одновременно должны изменить величины ширины и/или высоты в этом же диалоговом окне.**

### **Закладка “Действия” Actions**

- Files (файлы)      Открывается доступ к схеме файлов для данного отчета.
- Embeds (вставки)      Открывается доступ к диалоговому окну Вставки программного текста для данного отчета.

**Закладка “Шрифт” (Font)**

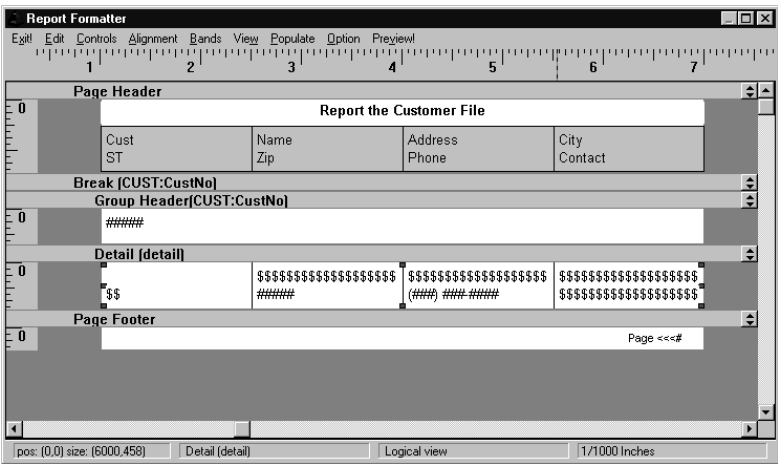
Вызывает диалоговое окно Select Font (выбрать шрифт), которое дает вам возможность отобрать шрифт (начертание шрифта), размер, стиль (например, жирный или курсив), цвет и эффекты шрифта (подчеркивание и перечеркивание для всех элементов управления в разделе отчета. Вы можете отменить шрифт раздела путем назначения иного шрифта в диалоговом окне Properties (свойства) для любого конкретного элемента управления. Когда вы отбираете опции, диалоговое окно показывает образец выбранного шрифта.

**Group Breaks (Групповые прерывания)**

Групповые прерывания обеспечивают средства группирования данных отчета по разделам и, если надо, обеспечения показа подзаголовков, подитогов или другой информации, связанной с подгруппой. Каждая группа состоит из набора записей, каждая из которых имеет одно и то же значение поля BREAK.

Когда величина переменной прерывания изменяется, оканчивается старая группа и начинается новая. Окончание группы означает, что последний элемент DETAIL (тело отчета) в группе обработан и составлен FOOTER (нижний колонтитул группы). Начало группы означает, что составлен HEADER (заголовок) группы и обработан первый элемент в DETAIL (тело отчета) в новой группе. Для того чтобы составить значащие группы, записи должны быть отсортированы по тем же полям, по которым объявлены BREAKs. Смотрите ниже Назначение порядка сортировки и Назначение вложенных групповых прерываний.

В пределах отчета вы можете визуально разделить эти группы пробелами, промежуточными суммами, заголовками или другой суммарной информацией либо сверху группы, либо под группой, либо и там и там. Показ суммарной (итоговой) информации для группы выполняется помещением элементов управления текстом или графикой в ЗАГОЛОВОК или КОЛОНТИТУЛ.



Структура BREAK (прерывание) может содержать большинство тех же самых элементов что и структура отчет: групповые ЗАГОЛОВКИ, ТЕЛА ОТЧЕТА, групповые КОЛОНТИТУЛЫ, а также ПРЕРЫВАНИЯ. Такие прерывания могут быть вложенными, давая несколько уровней группировки записей. Форматер отчета показывает групповые прерывания в структуре отчета со смещением вправо, которое дает вам возможность легко визуализировать вложенные групповые прерывания.

### **Свойства группового прерывания**

Данное диалоговое окно дает вам возможность редактировать свойства группового ПРЕРЫВАНИЯ. Щелкните правой клавишей мыши на полосе прерывания, а затем выберите из всплывающего меню Properties (свойства).

### **Закладка “Общие свойства” (General)**

Label (метка)	Наберите допустимую метку Clarion’a для структуры ПРЕРЫВАНИЕ.
Variable (переменная)	Нажмите эллиптическую (...) кнопку для того, чтобы отобразить поле прерывания. Когда величина этой переменной изменяется, заканчиваются старые группы и начинаются новые группа.
Use ( метка соответствия)	Наберите метку соответствия для ссылки на BREAK (прерывание) в вашей исполняемой программе.
Freeze (замораживать)	“Замораживает” все элементы управления в данной структуре, так что последующие изменения словаря данных неприменимы. Вы можете отменить атрибут #Freeze для всех элементов управления или для отдельных элементов управления. Смотрите Генератор приложений - Установка опций приложения.

### **Закладка “Действия” Actions**

Files (файлы)	Открывается доступ к схеме файлов для данного отчета.
Embeds (вставки)	Открывается доступ к диалоговому окну Вставки программного текста для данного отчета.

**Совет:** Если переменная прерывания является глобальной или локальной переменной, вы должны быть уверены, что исполняемая программа обновляет ее значение, так что она может генерировать групповые прерывания.

### **Групповой Заголовок**

Хотя они печатаются на одной и той же странице одновременно, процессор печати составляет групповой ЗАГОЛОВОК до группового ТЕЛА ОТЧЕТА. Групповой ЗАГОЛОВОК является хорошим местом для идентификации группы, но не таким хорошим местом для показа групповых итогов (так как группа еще не была обработана).

### **Свойства Группового ЗАГОЛОВКА**

Это диалоговое окно дает вам возможность отредактировать свойства группового ЗАГОЛОВКА. Щелкните правой клавишей мыши на полосе заголовка, а затем выберите из всплывающего меню Properties (свойства).

#### **Закладка “Общие свойства” (General)**

- Use ( метка соответствия)      Наберите метку соответствия для ссылки на групповой HEADER (заголовок) в вашей исполняемой программе.
- Page Before (страница до)      Для того, чтобы печатать структуру группового ЗАГОЛОВКА на новой странице, отметьте это поле. Это устанавливает атрибут PAGEBEFORE. Процессор печати генерирует переход на новую страницу перед печатью заголовка страницы. Номер страницы автоматически меняется если вы не переустанавливаете его. Для переустановки номера страницы на величину, назначенную вами, наберите эту величину в соответствующем поле New Page No: (новый номер страницы).
- Page After (страница после)      Для того, чтобы печатать структуру группового ЗАГОЛОВКА, а затем форсировать переход на новую страницу, отметьте это поле. Это устанавливает атрибут PAGEAFTER. Процессор печати генерирует прерывание страницы сразу после печатания заголовка страницы. Номер страницы автоматически меняется если вы не переустанавливаете его. Для переустановки номера страницы на величину, назначенную вами, наберите эту величину в соответствующем поле New Page No: (новый номер страницы).
- With Prior (с предыдущей)      Чтобы предотвратить появление элементов “сирот” в распечатке, наберите цифровую величину в этом поле. Это устанавливает атрибут WITHPRIOR (с предыдущей). “Сиротский” элемент печати - это элемент, являющийся последним в группе, и отделенный от остальной группы прерыванием страницы. Данная величина устанавливает число предшествующих позиций группы, которые должны появиться вместе на одной странице с последней позицией.
- With Next (с последующей)      Чтобы предотвратить появление элементов “вдов” в распечатке, наберите цифровую величину в этом поле. Это устанавливает атрибут WITHNEXT (с последующей). “Вдовый” элемент печати - это элемент, являющийся первым в группе, и отделенный от остальной группы прерыванием страницы. Данная величина устанавливает число последующих позиций

Freeze (замораживать) группы, которые должны появиться вместе на одной странице с первой позицией.

“Замораживает” все элементы управления на данной полосе, так что последующие изменения словаря данных неприменимы. Вы можете отменить атрибут #Freeze для всех элементов управления или для отдельных элементов управления. Смотрите Генератор приложений - Установка опций приложения.

### **Закладка “Дополнительные возможности” (Extra)**

Colors (цвета) Устанавливает цвет текста или фона для данной полосы. Смотрите Общие атрибуты элементов управления - Установка атрибута ЦВЕТ.

Alone (одинокый) Устанавливает, что раздел Групповой ЗАГОЛОВОК всегда печатается в одиночестве на странице, без формы, заголовка страницы или колонтитула страницы. Это добавляет к структуре атрибут ALONE (одинокий), что полезно при печати страниц оглавлений и общих итогов.

Absolute (абсолютный) Устанавливает, что раздел Групповой ЗАГОЛОВОК всегда печатается на одной и той же фиксированной позиции на странице. Это добавляет к структуре атрибут ABSOLUTE (абсолютный). Смотрите следующий раздел Положение.

**Совет: Эти установки полезны, когда ваш отчет имеет более одного тела отчета. Тела отчета могут быть напечатаны условно и могут быть использованы для печати только одноразовых страниц (страницы оглавления или страницы общего итога)**

### **Закладка Положение (Position)**

Дает вам возможность установить местоположение и размер группового заголовка путем заполнения атрибута AT. Местоположение - это отступ относительно верхнего левого угла области печати тела отчета или относительно последней позиции, отпечатанной в области печати тела отчета. Единицы измерения для этих величин установлены на закладке General (общие свойства) диалогового окна Report Properties (свойства отчета).

Top Left Corner (верхний левый угол)

Для того, чтобы установить точную исходную точку для вашего группового заголовка относительно левого верхнего угла области печати тела отчета или относительно последней позиции, отпечатанной в области печати тела отчета, назначьте координаты Верхнего левого угла с помощью данного диалогового окна.

Width/Height (ширина/высота)



Default (по умолчанию)	Чтобы установить размер заголовка страницы, выберите из следующих опций для ширины (Width) и высоты (Height). Устанавливает величину, основанную на формате бумаги и области печати тела отчета (смотрите также выше Свойства Отчета)..
Fixed (фиксированный)	Чтобы установить особый размер, отметьте фиксированные выборы.

**Совет: При изменении ориентации отчета с книжной на альбомную, или наоборот, вы одновременно должны изменить величины ширины и/или высоты в этом же диалоговом окне.**

### Закладка “Действия” Actions

Files (файлы)	Открывается доступ к схеме файлов для данного отчета.
Embeds (вставки)	Открывается доступ к диалоговому окну Вставки программного текста для данного отчета.

### Закладка “Шрифт” (Font)

Вызывает диалоговое окно Select Font (выбрать шрифт), которое дает вам возможность отобрать шрифт (начертание шрифта), размер, стиль (например, жирный или курсив), цвет и эффекты шрифта (подчеркивание и перечеркивание для всех элементов управления в разделе отчета. Вы можете отменить шрифт раздела путем назначения иного шрифта в диалоговом окне Properties (свойства) для любого конкретного элемента управления. Когда вы отбираете опции, диалоговое окно показывает образец форматирования.

### Detail (тело отчета)

Чтобы установить данные для тела отчета, поместите их в DETAIL. Здесь обычно печатается информация низшего уровня. Информация высокого уровня, повторяющаяся или суммарная информация более подходят для структур HEADER и FOOTER. Чтобы добавить DETAIL, выберите Bands> Detail.

Обратите внимание на то, что отчет может иметь много тел отчета DETAILS, которые могут быть отпечатаны по условиям (смотрите Мастера и шаблоны процедур - Шаблон отчета - Фильтры тела отчета). Это полезно для печати только одноразовых страниц, таких как страницы оглавления или страницы полного итога (смотрите Создание итогов и вычисляемых полей). Вы можете также использовать операторы встроенной программы для контроля того, какое тело отчета печатать во время работы. Каждая структура Тела отчета требует своего собственного оператора PRINT (печать).

Тело отчета печатается внутри области печати тела отчета, определенной атрибутом отчета AT. Кроме того, любые групповые ЗАГОЛОВКИ, групповые КОЛОНТИТУЛЫ или групповые ПРЕРЫВАНИЯ также печатаются внутри области печати тела отчета.

**Совет:** Для лучшей автоматической обработки, когда наступает время размещения структур на странице, вложите вашу **DETAIL** внутрь всех других структур. Например, если у вас две структуры **BREAK**, одна вложенная в другую, удалите все структуры **DETAIL** за исключением одной, вложенной внутрь самого глубокого **BREAK**.

### Свойства тела отчета **DETAIL**

Щелкните правой клавишей мыши на полосе тела отчета, а затем выберите из всплывающего меню **Properties** (свойства).

#### Закладка “Общие свойства” (**General**)

**Label** (метка)                      Наберите допустимую метку Clarion’a, называющую структуру ТЕЛА ОТЧЕТА (**DETAIL**).

**Use** ( метка соответствия)                      Наберите метку соответствия для ссылки на ТЕЛО ОТЧЕТА (**DETAIL**) в вашей исполняемой программе.

**Page Before** (страница до)                      Для того чтобы печатать структуру ТЕЛА ОТЧЕТА на новой странице, отметьте это поле. Это устанавливает атрибут **PAGEBEFORE**. Процессор печати генерирует прерывание страницы перед печатанием тела отчета.

Номер страницы автоматически меняется если вы не переустанавливаете его. Для переустановки номера страницы к величине, назначенной вами, наберите эту величину в соответствующем поле **New Page No:** (новый номер страницы).

**Page After** (страница после)                      Для того, чтобы печатать структуру ТЕЛА ОТЧЕТА, а затем форсировать переход на новую страницу, отметьте это поле. Это устанавливает атрибут **PAGEAFTER**. Процессор печати генерирует прерывание страницы сразу после печатания тела отчета.

Номер страницы автоматически меняется если вы не переустанавливаете его. Для переустановки номера страницы к величине, назначенной вами, наберите эту величину в соответствующем поле **New Page No:** (новый номер страницы).

**Совет:** Чтобы печатать на отдельной странице каждую запись, поместите переменные строки символов и/или элементы управления, которые вам нужны, в тело отчета, а после этого отметьте поле **PAGEAFTER** (страница после) в диалоговом окне **Свойства тела отчета**.

**With Prior** (с предыдущей)

Чтобы предотвратить появление элементов “сирот” в распечатке, наберите цифровую величину в этом поле. Это устанавливает атрибут

WITHPRIOR (с предыдущей). “Сиротский” элемент печати - это элемент, являющийся последним в группе, и отделенный от остальной группы прерыванием страницы.

Данная величина устанавливает число предшествующих позиций группы, которые должны появиться вместе на одной странице с последней позицией.

**Совет:** При размещении промежуточных сумм или итогов в тело отчета используйте атрибут **WITHPRIOR (с предыдущей)** для обеспечения их печатания вместе по крайней мере с одним членом столбца, расположенным выше, в случае наступления прерывания страницы.

With Next (с последующей)

Чтобы предотвратить появление элементов “вдов” в распечатке, наберите цифровую величину в этом поле. Это устанавливает атрибут WITHNEXT (с последующей). “Вдовый” элемент печати - это элемент, являющийся первым в группе, и отделенный от остальной группы прерыванием страницы.

Данная величина устанавливает число последующих позиций группы, которые должны появиться вместе на одной странице с первой позицией.

Freeze (замораживать)

“Замораживает” все элементы управления на данной полосе, так что последующие изменения словаря данных неприменимы. Вы можете отменить атрибут #Freeze для всех элементов управления или для отдельных элементов управления. Смотрите Генератор приложений - Установка опций приложения.

### **Закладка “Дополнительные возможности” (Extra)**

Colors (цвета) Устанавливает цвет текста или фона для данной полосы. Смотрите Атрибуты обычного элемента управления - Установка атрибута ЦВЕТ.

Alone (одинокый) Устанавливает, что раздел ТЕЛО ОТЧЕТА (DETAIL) всегда печатается в одиночестве на странице, без формы, заголовка страницы или колонтитула страницы. Это добавляет к структуре атрибут ALONE (одинокый).

Absolute (абсолютный)

Устанавливает, что раздел ТЕЛО ОТЧЕТА (DETAIL) всегда печатается на одной и той же фиксированной позиции на странице. Это добавляет к структуре атрибут ABSOLUTE (абсолютный). Смотрите следующий раздел Положение.

**Совет:** Эти установки полезны, когда ваш отчет имеет более одного тела отчета. Тела отчета могут быть напечатаны при выполнении условий и могут быть использованы для печати только одноразовых страниц ( титульной страницы или страницы общего итога).

### Закладка Положение (Position)

Дает вам возможность установить местоположение и размер тела отчета путем заполнения атрибута AT. Местоположение - это отступ относительно верхнего левого угла области печати тела отчета или относительно последней позиции, отпечатанной в области печати тела отчета. Единицы измерения для этих величин установлены на закладке General (общие свойства) диалогового окна Report Properties (свойства отчета).

Top Left Corner (верхний левый угол)

Для того, чтобы установить точную исходную точку для вашего тела отчета относительно левого верхнего угла области печати тела отчета или относительно последней позиции, отпечатанной в области печати тела отчета, назначьте координаты Верхнего левого угла с помощью данного диалогового окна.

Width/Height (ширина/высота)

Чтобы установить размер тела отчета, выберите из следующих опций для ширины (Width) и высоты (Height).

Default (по умолчанию)

Устанавливает величину, основанную на формате бумаги и области печати тела отчета (смотрите также выше Свойства Отчета)..

Fixed (фиксированный)

Чтобы установить особый размер, отметьте фиксированные выборы.

**Совет: При изменении ориентации отчета с книжной на альбомную, или наоборот, вы одновременно должны изменить величины ширины и/или высоты в этом же диалоговом окне.**

### Закладка “Действия” Actions

Files (файлы)

Открывается доступ к схеме файлов для данного отчета.

Embeds (вставки)

Открывается доступ к диалоговому окну Вставки программного текста для данного отчета.

### Закладка “Шрифт” (Font)

Вызывает диалоговое окно Select Font (выбрать шрифт), которое дает вам возможность отобрать шрифт (начертание шрифта), размер, стиль (например, жирный или курсив), цвет и эффекты шрифта (подчеркивание и перечеркивание для всех элементов управления в разделе отчета. Вы можете отменить шрифт раздела путем назначения иного шрифта в диалоговом окне Properties (свойства) для любого конкретного элемента управления. Когда вы отбираете опции, диалоговое окно показывает образец форматирования.

## Групповой Колонтитул (Group Footer)

---

Групповой КОЛОНТИТУЛ (FOOTER) составляется после группового ТЕЛА ОТЧЕТА. Это удобное место для показа групповых итогов или счетчиков.

### Свойства Группового Колонтитула (Group Footer)

Это диалоговое окно дает вам возможность редактировать свойства группового КОЛОНТИТУЛА. Щелкните правой клавишей мыши на полосе колонтитула, а затем выберите из всплывающего меню Properties (свойства).

### Закладка “Общие свойства” (General)

- Use ( метка соответствия) Наберите метку соответствия для ссылки на ГРУППОВОЙ КОЛОНТИТУЛ (Group FOOTER) в вашей исполняемой программе.
- Page Before (страница до) Для того, чтобы печатать структуру групповой КОЛОНТИТУЛ на новой странице, отметьте это поле. Это устанавливает атрибут PAGEBEFORE. Процессор печати генерирует прерывание страницы перед печатанием группового колонтитула.  
Номер страницы автоматически меняется если вы не переустанавливаете его. Для переустановки номера страницы к величине, назначенной вами, наберите эту величину в соответствующем поле New Page No: (новый номер страницы).
- Page After (страница после) Для того, чтобы печатать структуру группового КОЛОНТИТУЛА, а затем форсировать переход на новую страницу, отметьте это поле. Это устанавливает атрибут PAGEAFTER. Процессор печати генерирует прерывание страницы сразу после печатания группового КОЛОНТИТУЛА.  
Номер страницы автоматически меняется если вы не переустанавливаете его. Для переустановки номера страницы к величине, назначенной вами, наберите эту величину в соответствующем поле New Page No: (новый номер страницы).
- With Prior (с предыдущей) Чтобы предотвратить появление элементов “сирот” в распечатке, наберите цифровую величину в этом поле. Это устанавливает атрибут WITHPRIOR (с предыдущей).  
“Сиротский” элемент печати - это элемент, являющийся последним в группе, и отделенный от остальной группы прерыванием страницы.  
Данная величина устанавливает число предшествующих элементов группы, которые должны появиться вместе на одной странице с последним элементом.

- With Next (с последующей)** тобы предотвратить появление элементов “вдов” в распечатке, наберите цифровую величину в этом поле. Это устанавливает атрибут WITHNEXT (с последующей). “Вдовый” элемент печати - это элемент, являющийся первым в группе, и отделенный от остальной группы прерыванием страницы. Данная величина устанавливает число последующих позиций группы, которые должны появиться вместе на одной странице с первой позицией.
- Freeze (замораживать)** “Замораживает” все элементы управления на данной полосе, так что последующие изменения словаря данных неприменимы. Вы можете отменить атрибут #Freeze для всех элементов управления или для отдельных элементов управления. Смотрите Генератор приложений - Установка опций приложения.

### **Закладка “Дополнительные возможности” (Extra)**

- Colors (цвета)** Устанавливает цвет текста или фона для данной полосы. Смотрите Общие атрибуты элементов управления - Установка атрибута ЦВЕТ.
- Alone (одиночный)** Устанавливает, что раздел групповой КОЛОНТИТУЛ (group FOOTER) всегда печатается в одиночестве на странице, без формы, заголовка страницы или колонтитула страницы. Это добавляет к структуре атрибут ALONE (одиночный).
- Absolute (абсолютный)** Устанавливает, что раздел групповой КОЛОНТИТУЛ (group FOOTER) всегда печатается на одной и той же фиксированной позиции на странице. Это добавляет к структуре атрибут ABSOLUTE (абсолютный). Смотрите следующий раздел Положение.

### **Закладка Положение (Position)**

Дает вам возможность установить местоположение и размер группового КОЛОНТИТУЛА путем заполнения атрибута АТ. Местоположение - это отступ относительно верхнего левого угла области печати тела отчета или относительно последней позиции, отпечатанной в области печати тела отчета. Единицы измерения для этих величин установлены на закладке General (общие свойства) диалогового окна Report Properties (свойства отчета).

**Top Left Corner (верхний левый угол)**

Для того, чтобы установить точную исходную точку для вашего группового колонтитула относительно левого верхнего угла области печати тела отчета или относительно последней позиции, отпечатанной в области печати тела отчета, назначьте координаты Верхнего левого угла с помощью данного диалогового окна.

Width/Height (ширина/высота)

Чтобы установить размер группового колонтитула, выберите из следующих опций для ширины (Width) и высоты (Height).

Default (по умолчанию)

Устанавливает величину, основанную на формате бумаги и области печати тела отчета (смотрите также выше Свойства Отчета)..

Fixed (фиксированный)

Чтобы установить особый размер, отметьте фиксированные выборы.

**Совет: При изменении ориентации отчета с книжной на альбомную, или наоборот, вы одновременно должны изменить величины ширины и/или высоты в этом же диалоговом окне.**

### Закладка “Действия” Actions

Files (файлы)

Открывается доступ к схеме файлов для данного отчета.

Embeds (вставки)

Открывается доступ к диалоговому окну Вставки программного текста для данного отчета.

### Закладка “Шрифт” (Font)

Вызывает диалоговое окно Select Font (выбрать шрифт), которое дает вам возможность отобрать шрифт (начертание шрифта), размер, стиль (например, жирный или курсив), цвет и эффекты шрифта (подчеркивание и перечеркивание для всех элементов управления в разделе отчета. Вы можете отменить шрифт раздела путем назначения иного шрифта в диалоговом окне Properties (свойства) для любого конкретного элемента управления. Когда вы отбираете опции, диалоговое окно показывает образец форматирования.

## Колонтитул страницы (Page Footer)

Чтобы назначить текст и данные, которые необходимо генерировать в конце каждой страницы, поместите их в КОЛОНТИТУЛ страницы. Помните, что колонтитул страницы может быть физически расположен в любом месте страницы, а не только внизу ее. Обычно страничный КОЛОНТИТУЛ включает итоги, номера страниц, данные печати и т.д.

Чтобы добавить колонтитул страницы к вашему отчету, выберите Bands> Page Footer.

### Свойства КОЛОНТИТУЛа страницы

Щелкните правой клавишей мыши на полосе колонтитула, а затем выберите из всплывающего меню Properties (свойства).

### Закладка “Общие свойства” (General)

Use (метка соответствия)

Наберите метку соответствия для ссылки на страничный КОЛОНТИТУЛ (page FOOTER) в вашей исполняемой программе.

**Freeze (замораживать)** “Замораживает” все элементы управления на данной полосе, так что последующие изменения словаря данных неприменимы. Вы можете отменить атрибут #Freeze для всех элементов управления или для отдельных элементов управления. Смотрите Генератор приложений - Установка опций приложения.

### **Закладка “Дополнительные возможности” (Extra)**

**Colors (цвета)** Устанавливает цвет текста или фона для данной полосы. Смотрите Общие атрибуты элементов управления - Установка атрибута ЦВЕТ.

**Alone (одиначный)** Не оказывает никакого воздействия на страничный КОЛОНТИТУЛ. Смотрите Групповой КОЛОНТИТУЛ.

**Absolute (абсолютный)** Не оказывает никакого воздействия на страничный КОЛОНТИТУЛ. Смотрите Групповой КОЛОНТИТУЛ.

### **Закладка Положение (Position)**

Дает вам возможность установить местоположение и размер страничного КОЛОНТИТУЛА путем заполнения атрибута AT.

Единицы измерения для этих величин установлены на закладке General (общие свойства) диалогового окна Report Properties (свойства отчета).

**Top Left Corner (верхний левый угол)**

Для того, чтобы установить точную исходную точку для вашего страничного колонтитула относительно левого верхнего угла страницы, назначьте координаты Верхнего левого угла с помощью данного диалогового окна.

**Width/Height (ширина/высота)**

Чтобы установить размер страничного колонтитула, выберите из следующих опций для ширины (Width) и высоты (Height).

**Default (по умолчанию)**

Устанавливает величину, основанную на формате бумаги

**Fixed (фиксированный)**

Чтобы установить особый размер, отметьте фиксированные выборы.

**Совет: При изменении ориентации отчета с книжной на альбомную, или наоборот, вы одновременно должны изменить величины ширины и/или высоты в этом же диалоговом окне.**

### **Закладка “Действия” Actions**

**Files (файлы)**

Открывается доступ к схеме файлов для данного отчета.

**Embeds (вставки)**

Открывается доступ к диалоговому окну Вставки программного текста для данного отчета.



### **Закладка “Шрифт” (Font)**

Вызывает диалоговое окно Select Font (выбрать шрифт), которое дает вам возможность отобрать шрифт (начертание шрифта), размер, стиль (например, жирный или курсив), цвет и эффекты шрифта (подчеркивание и перечеркивание для всех элементов управления в разделе отчета. Вы можете отменить шрифт раздела путем назначения иного шрифта в диалоговом окне Properties (свойства) для любого конкретного элемента управления. Когда вы отбираете опции, диалоговое окно показывает образец форматирования.

## ***Возможности форматера отчета***

### **Создание Процедуры отчета**

Если вы не определили процедуру отчета, вы можете сделать это, открыв диалоговое окно Application Tree (дерево приложений) и нажав клавишу INSERT, или же вы можете выбрать из меню Procedure> New. Когда появится диалоговое окно Select Procedure Type (выбор типа процедуры), очистите поле Use Procedure Wizard (использование мастера процедуры) и отберите Report (отчет). Появится диалоговое окно Procedure Properties (свойства процедуры). Вы должны установить файлы отчета и ключи в соответствии с приводимыми ниже указаниями.

Другой способ включает использование Procedure Wizard, который поможет вам определить процедуру вашего отчета. Смотрите Мастера и шаблоны процедур - Мастер отчета.

### **Назначение файлов и ключей (Порядок сортировки)**

Первым логическим шагом в подготовке отчета является назначение файлов и ключей, которые будут использоваться вашей процедурой отчета. Вы можете сделать это из диалогового окна Procedure Properties (свойства процедуры) даже до запуска Report Formatter (форматера отчета).

Файлы, которые вы назначаете, будут, в свою очередь, определять, какие поля словаря данных появятся на вашем отчете. Вы можете назначить более одного файла для отчета, а именно, первичный файл плюс вторичные файлы. Вторичные файлы должны быть связаны с первичным файлом общим полем. Смотрите Добавление или модифицирование отношений в главе 4 данной книги.

☐ Чтобы назначить файлы для вашего отчета:

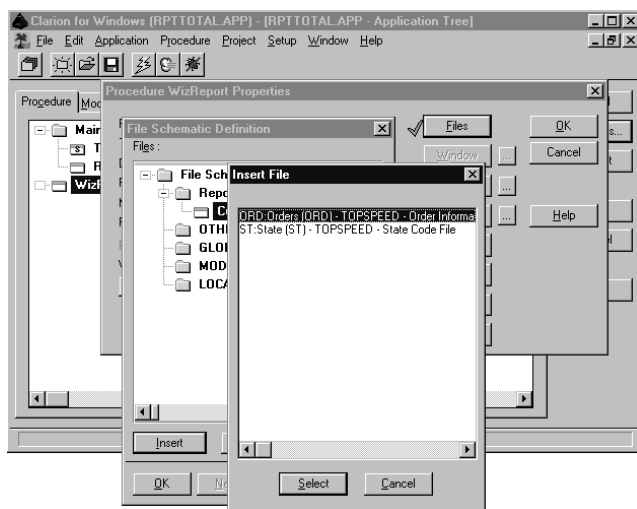
1. Находясь в диалоговом окне Application Tree (дерево приложения), дважды щелкните клавишей мыши на имени процедуры отчета.

Появится диалоговое окно Procedure Properties (свойства процедуры).

2. Нажмите кнопку Files (файлы).

Появится диалоговое окно File Schematic Definition (определение схемы файлов). Воспользуйтесь этим диалоговым окном для того, чтобы указать Генератору приложений, какие файлы и ключи использует ваша процедура отчета.

3. Дважды щелкните клавишей мыши на позиции ToDo для вашей процедуры отчета.



Появится диалоговое окно Insert File (вставить файл).

Назначение файла(ов), о котором вы будете делать отчет.

4. Дважды щелкните клавишей мыши на файле, из которого вы хотите делать отчет.

Появляется диалоговое окно File Schematic Definition (определение схемы файлов). Позиция ToDo заменена первичным файлом, выбранным вами для этой процедуры отчета.

### **Печать из более чем одного файла (добавление вторичных файлов)**

1. Из диалогового окна File Schematic Definition (определение схемы файлов) дважды щелкните на первичном файле, уже назначенном для вашей процедуры отчета.

Появляется диалоговое окно Insert File (вставить файл), перечисляющее только файлы, связанные с первичным файлом.

2. Дважды щелкните клавишей мыши на дополнительном файле, который вы хотите использовать в отчете.

Появляется диалоговое окно File Schematic Definition (определение схемы файлов). Повторите эти шаги для любых других связанных файлов. Добавляя файлы вы можете также дважды щелкнуть на вторичном файле, затем выбрать из файлов, относящихся только

к данному вторичному файлу.

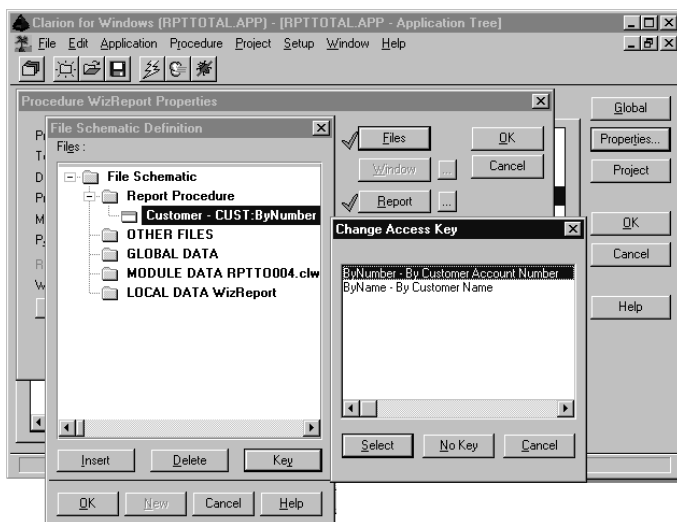
### Назначение порядка сортировки

Ключи (или индексы), которые вы назначаете для вашей процедуры отчета, будут определять последовательность, в которой появляются записи. Ключи должны также определять переменные прерывания и порядок вложения любых групповых прерываний.

Добавление вторичных файлов (смотрите выше) к вашей процедуре дает вам также другое логическое поле для прерывания - а именно, общие поля, связывающие два файла.

□ Чтобы назначить ключи для вашего отчета:

1. Находясь в диалоговом окне File Schematic Definition (определение схемы файлов), нажмите кнопку Key для того, чтобы назначить, какой ключ используется для этой процедуры.



Появится диалоговое окно Change Access Key (изменение ключа доступа):

Назначение ключа сортировки для вашего отчета.

2. Дважды щелкните мышью на ключе, который вам нужен для этого отчета.

Снова появляется диалоговое окно File Schematic Definition (определение схемы файлов).

3. Нажмите кнопку OK.

## Назначение размера и ориентации бумаги

---

Формат бумаги и ориентация текста по отношению к бумаге должны быть тщательно продуманы и установлены как можно раньше в процессе разработки отчета. Изменение этих установок после того, как были распланированы поля отчета, заголовки и т.д., приведет в результате к переделке большей части макета.

Чтобы установить формат и ориентацию, выберите Edit> Report Properties, а затем отберите закладку General (общие характеристики). Выберите диалоговые единицы, тысячные доли дюйма, миллиметры или пункты из ниспадающего списка Units (единицы измерения).

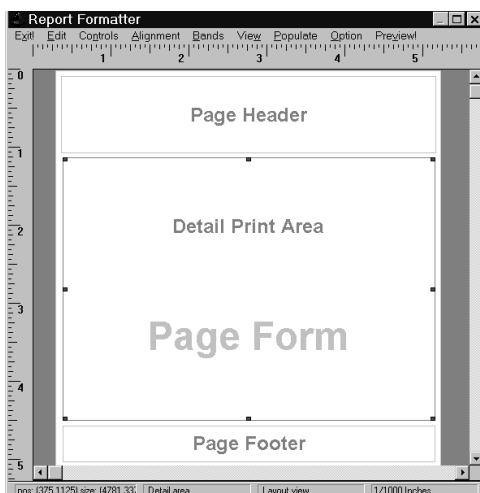
Диалоговые единицы (смотрите Глоссарий) определяются как одна четвертая средней ширины символа на одну восьмую средней высоты символа. Размер диалоговой единицы зависит от размера шрифта, назначенного по умолчанию для вашего отчета. Эта единица измерения основана на шрифте, установленном в атрибуте ШРИФТ отчета или от шрифта принтера по умолчанию.



## Назначение полей отчета

---

Поля для ФОРМ (FORMs), страничных ЗАГОЛОВКОВ (page HEADERS), страничных КОЛОНТИТУЛОВ (page FOOTERS) и область печати тела отчета все устанавливаются независимо друг от друга, следовательно, нет ни одной установки полей, которая применялась бы ко всему отчету. Вы можете установить эти поля визуально с помощью Report Formatter's Page Layout View (вид макета страницы формatera отчета). Обратите, пожалуйста, внимание на то, что границы каждой из этих структур могут перекрываться.



Вид макета страницы форматера отчета.

Этот прямоугольник представляет границы ЗАГОЛОВКА страницы. Перемещать ее или менять ее размеры можно с помощью операции “тащить и бросать”.

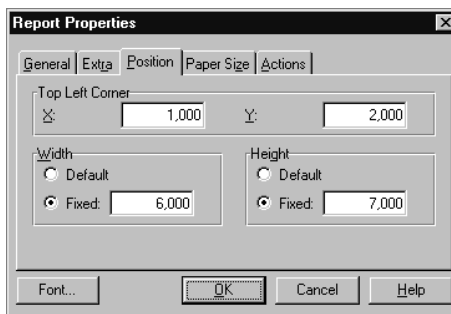
Серая область, ограниченная линейками, представляет границы бумаги.

Этот прямоугольник представляет границы области печати тела отчета. Перемещать ее или менять ее размеры можно с помощью операции “тащить и бросать”.

Этот большой (белый) прямоугольник представляет границы ФОРМЫ. Перемещать ее или менять ее размеры можно с помощью операции “тащить и бросать”.

Этот прямоугольник представляет границы КОЛОНТИТУЛА страницы. Перемещать ее или менять ее размеры можно с помощью операции “тащить и бросать”.

Вы можете также назначить поля ФОРМЫ, страничного КОЛОНТИТУЛА, страничного ЗАГОЛОВКА, а также область печати тела отчета на закладке Position (положение) соответствующих диалоговых окон Form Properties (свойства формы), Page/Group Header Properties (свойства страничного/группового заголовка), Page/Group Footer Properties (свойства страничного/группового колонтитула), а также Report Properties (свойства отчета).



### **Область печати тела отчета**

Область печати тела отчета определяется атрибутом AT структуры REPORT (отчет). Вот те четыре правила, которые вам следует понять и запомнить об области печати тела отчета:

- ☐ Каждое тело отчета (DETAIL), групповой ЗАГОЛОВОК (HEADER) и групповой КОЛОНТИТУЛ (FOOTER) печатаются внутри границ области печати тела отчета.

- ☐ Границы области печати тела отчета могут быть установлены с помощью Report Formatter's Page Layout View (вида макета страницы формatera отчета) или с помощью закладки Position (положение) диалогового окна Report Properties (свойства отчета).

- ☐ Каждая позиция внутри области печати тела отчета печатается относительно предыдущей отпечатанной позиции.

- ☐ Относительное положение позиций, печатаемых в пределах области печати тела отчета, может быть установлено с помощью закладки Position (положение) диалогового окна Detail Properties (свойства тела отчета) или с помощью протаскивания ручек DETAILS при просмотре отчета в виде полос.

**Совет:** При выборе просмотра отчета в виде полос белая область полосы тела отчета представляет собой область печати тела отчета. Прямоугольник, ограниченный ручками, представляет структуру тела отчета и его относительный отступ.

## **Установка положения и выравнивание**

### **Атрибут AT**

Используйте закладку Position (положение) для установки атрибута AT для различных структур отчета. Атрибут AT в структурах печати выполняет две различные функции, в зависимости от структуры, на которую он помещен.

Будучи помещенным в ФОРМУ, страничный ЗАГОЛОВОК или страничный КОЛОНТИТУЛ, атрибут АТ определяет положение и размер данной структуры. Положение, установленное параметрами  $x$  и  $y$ , является относительным к верхнему левому углу страницы.

Если он помещен в ТЕЛО ОТЧЕТА, ПРЕРЫВАНИЕ, групповой ЗАГОЛОВОК или групповой КОЛОНТИТУЛ, структура печатается относительно предыдущей напечатанной позиции, в соответствии со следующими правилами (если не присутствует атрибут АБСОЛЮТНО):

- Параметры ширины и высоты атрибута АТ определяют минимальный размер структуры.
- Структура в действительности печатается в следующей доступной позиции внутри области печати тела отчета.
- Положение, назначенное параметрами  $x$  и  $y$  атрибута АТ структуры является отступом от следующей доступной позиции печати внутри области печати тела отчета.
- Первая позиция печатается в верхнем левом углу области печати тела отчета (с отступом, назначенным ее атрибутом АТ).
- Следующая и последующие позиции печатаются относительно конечного положения предыдущей позиции:

Если имеется место для печати следующей позиции кроме предыдущей позиции, то она печатается здесь.

Если нет, она печатается ниже предыдущей позиции.

### **Точное позиционирование и выравнивание**

Имеются четыре главных инструмента, которые помогут вам точно выравнивать данные вашего отчета: выравнивание по сетке, инструменты выравнивания, закладка Position (положение) соответствующих диалогов свойств, а также ограниченное перемещение. Выравнивание по сетке обсуждалось выше в разделе Меню - Использование опций меню. Инструменты выравнивания обсуждались выше в Панели инструментов - Панель инструментов выравнивания.

**Совет:** Для сверхточного позиционирования используйте закладку “Положение” соответствующих диалоговых окон свойств. Когда вы позиционируете структуру на экране, наименьшая единица, на которую вы ее можете подвинуть обычно равна 1/96 дюйма. Однако закладка “Положение” дает вам возможность установить положение с точностью до тысячных дюйма.

### **Ограниченное перемещение**

Нажмите и удерживайте клавишу SHIFT пока вы передвигаете элемент управления; это ограничит перемещение данного элемента управления движением по одной оси. То есть,

SHIFT+перемещение перемещают элемент управления либо горизонтально, либо вертикально, но не по обоим осям.

## **Назначение формы “предпечать”**

---

Смотрите выше раздел Структуры и свойства отчета - Форма.

В Report Formatter's Band View (форматера отчета в виде полос) поместите в полосе формы нужные вам текстовые или графические элементы управления. Это установит постоянные текст или графику, которые печатаются на каждой странице. Эта “лежащая ниже” “форма” печатается независимо и “под” всеми другими позициями на отчете.

## **Определение страничных заголовков и колонтитулов**

---

Смотрите выше раздел Структуры и свойства отчета -Страничный заголовок и Страничный колонтитул. Смотрите также Меню - Использование меню полос.

В Report Formatter's Band View (представление форматера отчета в виде полос) поместите в полосе страничного заголовка или полосе страничного колонтитула нужные вам текстовые или графические элементы управления. Используйте для добавления меню Bands (полосы).

Страничные заголовки состояются прежде тел отчета и прерываний. Страничные колонтитулы состояются после тел отчета и прерываний. Как ЗАГОЛОВКИ, так и КОЛОНТИТУЛЫ могут быть помещены в любом месте страницы.

Данные, печатаемые в страничных заголовках и колонтитулах, могут быть постоянными или переменными. Страничные заголовки и колонтитулы обычно используются для представления оглавлений отчета, фиксированных заголовков столбцов, номеров страниц, данных печати, фирменного знака и т.п.

## **Определение заголовков столбцов и заголовков отчета**

---

Смотрите Панели инструментов - Панели инструментов элементов управления и Панели инструментов свойств. Смотрите также Элементы управления и их свойства - Свойства элемента управления строка символов.

Статические строки символов подходят для печати заголовков столбцов, заголовков отчета и любых других текстов, которые не изменяются. Обычно вам нужно поместить ваши оглавления и заголовки в страничный заголовок или страничную форму. Если вам нужно печатать изменяющиеся оглавления или заголовки, смотрите ниже Установка полей для печати.

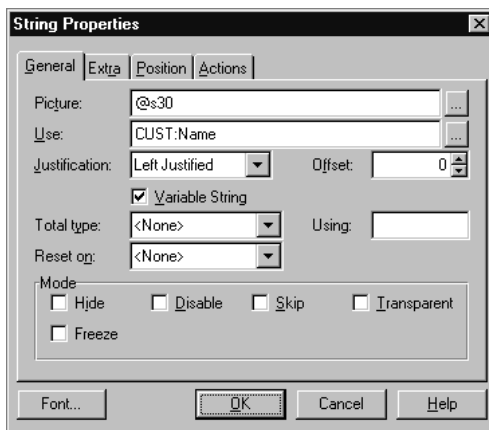
☐ Для того, чтобы поместить в ваш отчет элемент управления в виде статической строки символов:

1. Отберите инструмент Строка символов из панели инструментов Элементы управления или выберите Controls> String.



2. Щелкните в полосе, которая будет содержать заголовок (обычно это страничный заголовок).

Report Formatter (форматер отчета) помещает элемент управления СТРОКА символов в структуру отчета. Центр перекрестия определяет положение верхнего левого угла строки СИМВОЛОВ.



3. Дважды щелкните клавишей мыши на элементе управления строка, или же щелкните правой клавишей мыши на данном элементе управления и выберите из всплывающего меню Properties (свойства).

4. Наберите текст нужного вам оглавления для прямого показа в поле Text (текст). Кавычки не нужны.

5. Нажмите кнопку Font (шрифт) для того, чтобы назначить начертание шрифта, размер, цвет и стиль текста.

**Совет: Вы можете также назначить текст и его шрифт с помощью панели инструментов Свойства. Выберите Option> Show Propertybox.**

6. По желанию назначьте выравнивание текста с помощью выпадающего списка Justification (выравнивание).

Для специальных эффектов:

7. Выберите закладку Extra (дополнительные возможности) для установки цвета и угла.

8. Нажмите кнопку OK и закройте диалоговое окно String Properties.

## Назначение полей для печати (переменный текст)

Смотрите Панели инструментов - Панель инструментов заселение полей и Меню - Использование меню заселения. Смотрите также Элементы управления и их свойства - Свойства строки символов.

Элементы управления в виде переменной строки символов представляют собой базовые единицы для печати в отчете переменных данных. Переменные строки символов используются также для показа итогов и других вычисляемых полей.

Используя переменную USE, процедура отчета получает доступ к переменной памяти или полю словаря данных, которое вы хотите напечатать. Report Formatter (форматер отчета) форматирует данные в соответствии с назначенным вами шаблоном изображения.

□ Чтобы поместить в ваш отчет элемент управления в виде переменной строки символов:

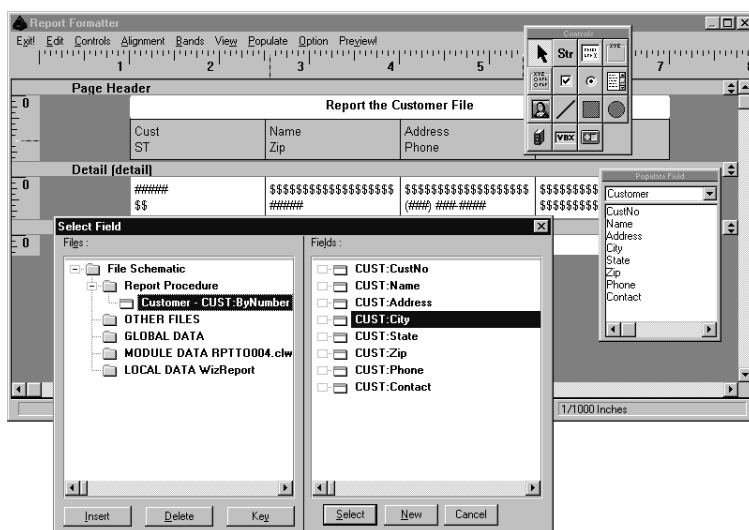
1. Отберите инструмент Полей словаря из панели инструментов “элементы управления” или выделите поле на панели инструментов “заселение поля” (выберите Option>Show Fieldbox).

Использование панели инструментов дает вам возможность поместить только поле словаря данных, не покидая при этом Форматер отчета. Использование панели инструментов “поля словаря” дает вам возможность сделать выбор из полей словаря данных и переменных памяти с помощью диалогового окна Select Fields (выбор полей).

2. Щелкните клавишей мыши в полосе, которая будет содержать переменную строку (обычно это тело отчета, групповой заголовок или групповой колонтитул).

Если вы используете инструмент Поля словаря, отберите вашу переменную с помощью диалогового окна Select Fields (выбор полей). Центр креста курсора позиционирует левый верхний угол строки символов.

Используйте инструмент Поля словаря для того, чтобы поместить в свой отчет поле словаря данных или переменную памяти.



Используйте панель инструментов Заселение поля для того, чтобы поместить в свой отчет поле словаря данных. Дважды щелкните на поле, которое вы хотите напечатать.

Форматер отчета помещает элемент управления СТРОКА в структуру данных ОТЧЕТ с переменной в качестве атрибута Use и устанавливает следующие свойства строки: поле Variable String (переменная строка) отмечено, маркер Picture (шаблон изображения) и величина Justification (выравнивание) обеспечены на основе информации словаря данных для выбранного поля или переменной.

Или же вы можете поместить элемент управления в виде строки символов, затем вручную установить свойства строки для получения того же результата: отметьте поле Variable String (переменная строка), наберите имя переменной в поле Use, наберите шаблон изображения в поле Picture (шаблон изображения) и отберите величину Justification (выравнивание) из ниспадающего списка.

3. Нажмите кнопку Font (шрифт) для того, чтобы назначить начертание шрифта, размер, цвет и стиль текста.

**Совет: Другой способ: вы можете назначить текст и его шрифт с помощью панели инструментов Property (свойства). Выберите Option>Show Propertybox.**

Для получения специальных эффектов:

4. Отберите закладку Extra (дополнительные возможности) для того, чтобы установить цвет текста и угол.

5. По желанию отберите тип итога из ниспадающего списка Total Type (тип итога) для создания сумм, средних, счетчиков, номеров страниц и т.д. Смотрите ниже Создание итогов.

6. Нажмите кнопку ОК для того, чтобы закрыть диалоговое окно String Properties (свойства строки).

## **Назначение Групповых прерываний (Group Breaks)**

---

Для того чтобы иметь значащие группы в отчете, записи отчета должны быть должным образом отсортированы. Смотрите выше Назначение порядка сортировки. Смотрите также Свойства структур отчета - Групповые прерывания.

☐ Чтобы создать групповое прерывание:

1. Находясь в Форматере отчета, выберите Bands>Surrounding Break.

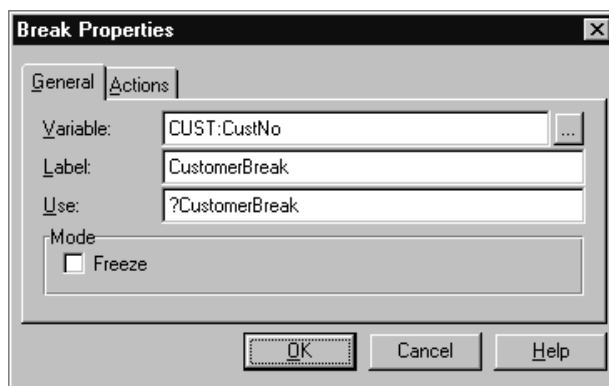
2. Перемещайте курсор над полосой тела отчета; когда курсор меняет свою форму на крест, щелкните клавишей мыши в полосе ТЕЛО ОТЧЕТА.

Появляется диалоговое окно Break Properties (свойства прерывания).

3. В поле Variable (переменная) нажмите эллиптическую кнопку (...) для того, чтобы выбрать переменную прерывания из диалогового окна Select Fields (выбор полей).

Когда величина переменной меняется, начинается новая группа.

4. В поле Label (метка) наберите допустимую Clarion метку для использования ее в качестве метки для структуры ПРЕРЫВАНИЕ.



На метку могут ссылаться атрибуты RESET (сброс) и TALLY (подсчет).

5. В поле Use наберите метку соответствия для ссылки на ПРЕРЫВАНИЕ (BREAK) в вашей исходной программе.

6. Нажмите кнопку ОК.

Это действие вставляет групповое ПРЕРЫВАНИЕ. Когда величина переменной изменяется, отчет составляет групповой КОЛОНТИТУЛ и следующий групповой ЗАГОЛОВОК, назначенный для данного прерывания.

### **Назначение Групповых заголовков и колонтитулов (Group Headers and Footers)**

Смотрите Свойства структур отчета - Групповые прерывания - Групповые заголовки и Групповые колонтитулы.

Процессор печати составляет групповой ЗАГОЛОВОК прежде группового ТЕЛА ОТЧЕТА. Групповой ЗАГОЛОВОК является хорошим местом для идентификации группы, например, с помощью статической строки символов, говорящей "Заказчик:", за которой следует переменная строка символов, показывающее поле имени заказчика.

Процессор печати составляет групповой КОЛОНТИТУЛ после группового ТЕЛА ОТЧЕТА. Групповой КОЛОНТИТУЛ - это хорошее место для суммирования группы, например, с помощью статической строки символов, говорящей “Итог:” с последующей переменной строкой символов, показывающей сумму. Смотрите ниже Создание итогов.

1. Во-первых, в соответствии со сказанным определите групповое прерывание.

2. Выберите Bands>Group Header из меню для назначения группового ЗАГОЛОВКА для данного ПРЕРЫВАНИЯ.

3. Перемещайте курсор поперек полосы тела отчета; когда курсор меняет свою форму на крест, щелкните клавишей мыши в строке заголовка ПРЕРЫВАНИЯ.

Появится диалоговое окно Page/Group Header Properties (свойства страничного/группового заголовка). Назначьте метку соответствия и любое специальное поведение прерывания страницы. Смотрите ниже Назначение прерываний страницы.

4. Нажмите кнопку ОК.

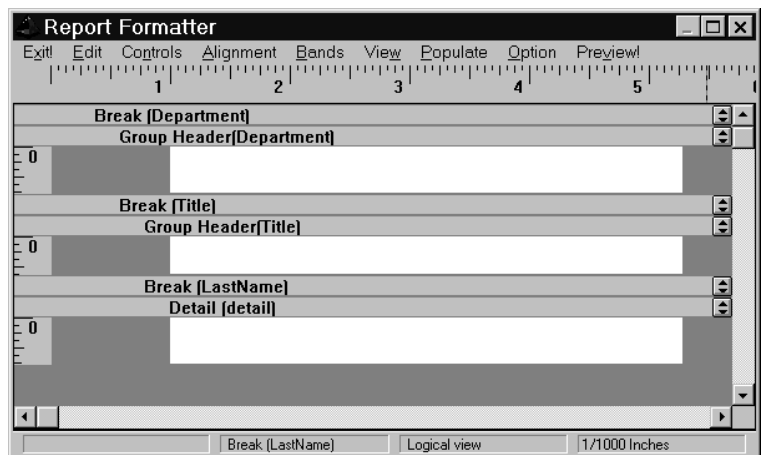
Это действие вставляет полосу группового ЗАГОЛОВКА. Вы можете поместить здесь элементы управления, точно так же, как и в любую другую полосу отчета. Групповые колонтитулы добавляются аналогичным образом, с помощью Bands>Group Footer из меню.

### **Назначение вложенных Групповых прерываний (Nested Group Breaks)**

Смотрите выше Назначение порядка сортировки и Назначение групповых прерываний.

Вложенное прерывание создается таким же образом, как и первое прерывание. Второе прерывание может быть вложено “внутри” первого прерывания помещением второго прерывания на тело отчета внутри первого прерывания. Или же второе прерывание может быть добавлено “снаружи” первого прерывания помещением второго прерывания на первое прерывание.

При установлении вложенных прерываний вы должны координировать порядок вложения прерываний с порядком сортировки обрабатываемых файлов. Например, если вы выбрали ключ, составленный из полей Отдел, Название и Фамилия, вы можете подобным же образом прервать на Фамилии, внутри Названия, внутри



Отдела.

Полосы просмотра показывают вложенные прерывания в макете отчета с отступом.

Вы можете обозреть компоненты своего ключа в диалоговом окне словаря данных Field/Key Definition (определение поля/ключа).

Помните, что добавление вторичных файлов к вашей процедуре также дает вам еще одно логическое поле для прерывания на нем: это общие поля, связывающие два файла.

## **Назначение поведения страничных прерываний (Page Breaking Behavior)**

---

Смотрите Структуры и свойства отчета - Групповой заголовок, е групповое тело отчета, групповой колонтитул.

Отредактируйте свойства групповых ЗАГОЛОВКОВ, ТЕЛ ОТЧЕТА и/или групповых КОЛОНТИТУЛОВ. Опциями при этом являются Страница до, Страница после, Со следующей и С предыдущей.

## **Создание Итогов и Вычисляемых полей (Totals and Calculated Fields)**

---

Смотрите выше Назначение полей для печати.

Итоговое поле - это просто элемент управления “переменная СТРОКА символов” с добавленным атрибутом SUM. Аналогично атрибуты AVE, CNT, MAX и MIN создают поля средних значений, счетов (подсчетов), максимумов и минимумов. Эти атрибуты могут быть добавлены путем выбора из ниспадающего меню Total Type (тип итога) в диалоговом окне String Properties (свойства строки).

Кроме этого, вы можете точно управлять поведением суммирования путем назначения атрибута RESET (сброс) из ниспадающего списка и назначением атрибута TALLY (подсчет) из списка Tallies (подсчеты) на закладке Extra (дополнительные возможности). И наконец, вы можете получить окончательный элемент управления путем назначения переменной для получения промежуточных величин для операций SUM, AVE, CNT, MAX и MIN с помощью поля Using. Вы можете выполнить вычисления во встроенной исходной программе с использованием для этого промежуточной переменной для получения нужного вам результата.

В общем случае вы помещаете итоговое поле в страничный или групповой КОЛОНТИТУЛ, так чтобы он смог подытожить записи с начала отчета, с начала страницы или с начала группового ПРЕРЫВАНИЯ. Однако вы можете также поместить итоговое поле в структуру ТЕЛО ОТЧЕТА для обеспечения текущих подитогов. Поле подсчета

(CNT) в ТЕЛЕ ОТЧЕТА может нумеровать записи по мере их появления на отчете.

□ Чтобы поместить поле итога:

1. Поместите элемент управления переменная строка в соответствии с описанием в Назначении полей для печати.

Например, если вы хотите подытожить продажи из системы ввода заказов, отберите поле словаря данных, содержащее величины продаж.

2. Дважды щелкните клавишей мыши на строковом элементе управления для того, чтобы открыть диалоговое окно String Properties (свойства строки).

3. Выберите тип итога из ниспадающего списка Total Type (тип итога). Сделайте выбор из Суммы, Среднего, Минимума, Максимума, Счетчик и Номера страницы.

4. По желанию используйте ниспадающий список Reset on (сбросить) для того, чтобы сбросить итог на нуль перед каждой страницей или перед каждым прерыванием.

Например, если ваш отчет постраничный, так что вам нужен итог для каждой страницы, выберите Page (страница) из ниспадающего списка Reset on (сбросить). Если вы хотите получить общий итог, выберите <None> (никакой) из ниспадающего списка Reset on. Смотрите ниже Общие итоги.

5. По желанию воспользуйтесь списком Tallies (подсчеты) на закладке Extra (дополнительные возможности) для того, чтобы назначить, когда происходит вычисление.

По умолчанию назначенное вычисление (SUM, CNT, MAX, MIN, AVG) производится каждый раз, когда печатается ТЕЛО ОТЧЕТА. Это справедливо даже если вычисляемое поле не находится в полосе ТЕЛА ОТЧЕТА и даже если вычисленное поле не в дочерней записи нижнего уровня многоуровневого отношения файлов.

Иными словами, используйте список Tallies (подсчеты), чтобы ограничить выполнение расчета до моментов печати прерываний более высокого уровня. Например, в отчете, который печатает три уровня отношений файлов (Заказчик>Заказ>Позиция), где поле для итога находится в среднем уровне (Заказ) и отчет печатает все три уровня, выберите OrderBreak (прерывание Заказ) в списке Tallies для изменения итога каждый раз, когда печатается ПРЕРЫВАНИЕ Заказ, а не каждый раз, когда печатается ТЕЛО ОТЧЕТА файла Позиция.

**Совет: Не выбирайте то же самое прерывание для Reset on и Tallies. Если вы так сделаете, результат каждый раз будет нуль.**

6. По желанию используйте поле Using для назначения переменных, которая получит промежуточные величины для операции SUM, AVE, CNT, MAX или MIN.

В действительности это дает вам доступ к возможности получения подитогов Процессора отчета. Вы можете использовать назначенные переменные в пределах ваших расчетов во встроенной исходной программе. Пожалуйста, обратите внимание, что Процессор отчета только пишет в промежуточную переменную, он не читает и не использует повторно переменную для последующих вычислений. Следовательно, вы не можете изменить итог, генерированный Процессором отчета, однако вы можете СПРЯТАТЬ поле, вычисленное Процессором отчета, и заменить его со своим собственным вычисленным полем.

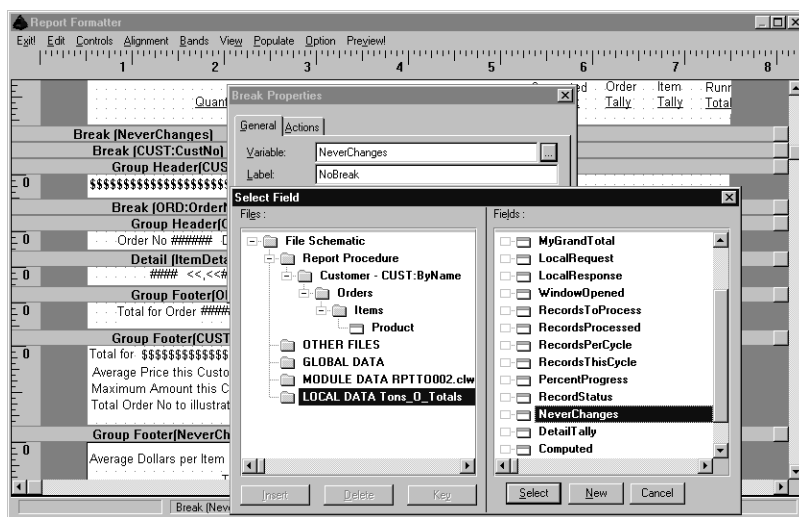
### Промежуточные суммы и страничные итоги

Промежуточные итоги создаются простым помещением итогового поля в пределах страничного или группового колонтитула и переустановкой итога на нуль в начале каждой страницы или группы. Используйте ниспадающий список Reset (сбросить) в диалоговом окне String Properties (свойства строки символов) для сброса итога на нуль перед каждой новой страницей или группой.

### Общие (суммарные) итоги (Grand Totals)

В действительности общие итоги - это просто промежуточные суммы, основанные на переменной прерывания, которая никогда не изменяется.

Чтобы создать для вашего отчета общий итог, добавьте групповое прерывание на переменную, которая не изменяется на протяжении всего отчета. Мы рекомендуем прерывание на переменной Локальные данные, называемой RecordStatus (статус записи). Любые другие групповые прерывания должны быть вложены в пределы этого группового прерывания "общие итоги".



Чтобы распечатать общий итог, создайте групповой итог по переменной, которая никогда не изменяется.



Затем добавьте колонтитул к групповому прерыванию общего итога. И наконец, добавьте промежуточный итог к групповому колонтитулу в соответствии с вышесказанным. Не сбрасывайте итог на нуль, то есть выберите <None> из ниспадающего списка Reset (сбросить) в диалоговом окне String Properties (свойства строки символов).

### **Построчные итоги (Row Totals)**

Для показа построчного итога (или любого другого вычисления) требуется выполнить два шага: присваивание величины переменной, а затем показ величины переменной в строковом элементе управления (смотрите выше Назначение полей для печати). Хороший пример этого процесса приводится в главе Заставим ее работать книги Начинаем.

Присваивание может быть выполнено с помощью Редактора формул ( смотрите главу Редактор формул) или путем вставки вручную в код оператора присваивания, смотрите Генератор приложений - Назначение встроенного исходного кода программы.

Используете ли вы Редактор формул, или делаете вставку программного текста, ключевым моментом, который необходимо знать, является то, что присваивание должно быть сделано сразу перед оператором PRINT . Для формул вам следует выбрать класс Before Print Detail (до печати тела отчета). Для встроенной исходной программы вам следует выбрать точку вставки Before Printing Detail Section (перед печатью раздела тело отчета).

### **Показ номеров страниц (Displaying Page Numbers)**

□ Чтобы поместить элемент управления, который печатает номер страницы.

1. Выберите Controls> String или выберите инструмент строки на панели инструментов “Элементы управления”, а затем щелкните клавишей мыши в полосе страничного заголовка или колонтитула.

2. Дважды щелкните клавишей мыши на элементе управления строка , который вы только что поместили.

Появится диалоговое окно String Properties (свойства строки символов).

3. Отметьте поле Variable String (переменная строка символов).

4. В поле Picture (шаблон изображения) диалогового окна String Properties (свойства строки символов) наберите @pPage<<#p.

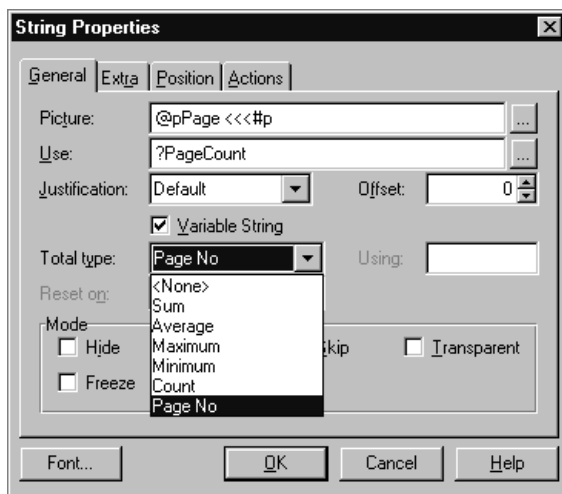
Этот шаблон изображения подавляет ведущие нули и показывает номер страницы, следующий за словом “Page” (страница).

5. В поле Use наберите метку соответствия для ссылки на строку в исходной программе.

Например, наберите ?PageCount (счетчик страниц). Не следует использовать переменную. Процессор отчета генерирует свою собственную переменную для номера страницы.

6. Из ниспадающего списка Total Type (тип итога) выберите Page No (номер страницы).

7. Нажмите кнопку ОК для того, чтобы закрыть диалоговое окно String Properties (свойства строки символов).



Чтобы напечатать номер страницы, отметьте поле Переменная строка символов, затем выберите PageNo из ниспадающего списка Total Type.

## Показ Дат печати (Displaying Print Dates)

☐ Чтобы поместить в отчет “Дату печати”

1. Выберите Controls> String, затем щелкните клавишей мыши в соответствующей полосе отчета.

Для даты печати вы обычно используете заголовок страницы или колонтитул страницы.

2. Дважды щелкните клавишей мыши на строчном элементе управления, который вы только что поместили.

Появится диалоговое окно String Properties (свойства строки символов).

3. В поле Text (текст) наберите шаблон изображения даты (@d1, например) или нажмите эллиптическую кнопку для использования Редактора шаблона изображения.

4. В поле Use нажмите эллиптическую кнопку (...) для создания локальной переменной,

называемой PrintDate (дата печати).

Появится диалоговое окно Select Field (выбор поля).

5. Выделите Локальные данные (Local Data), затем нажмите кнопку New (новое).

Появится диалоговое окно New Field Properties (свойства нового поля).

6. В поле Name (имя) наберите PrintDate (дата печати).

7. Выберите Long из ниспадающего списка Data Type (тип данных).

8. Нажмите ОК для того, чтобы закрыть диалоговое окно New Field Properties (свойства нового поля).

9. Нажмите ОК для того, чтобы закрыть диалоговое окно String Properties (свойства строки символов).

10. Нажмите Exit! (выход) для выхода из Форматера отчета.

☐ Чтобы присвоить величину TODAY( ) (сегодня) для PrintDate:

11. Щелкните кнопкой Embeds ( точки вставки) в диалоговом окне Procedure Properties (свойства процедуры).

12. В диалоговом окне Embedded Source (встроенная исходная программа) дважды щелкните клавишей мыши на точке вставки Procedure Setup (запуск процедуры).

13. В диалоговом окне Select embed type (выбор типа вставки) дважды щелкните на SOURCE (исходная программа).

14. В Редакторе текста наберите:

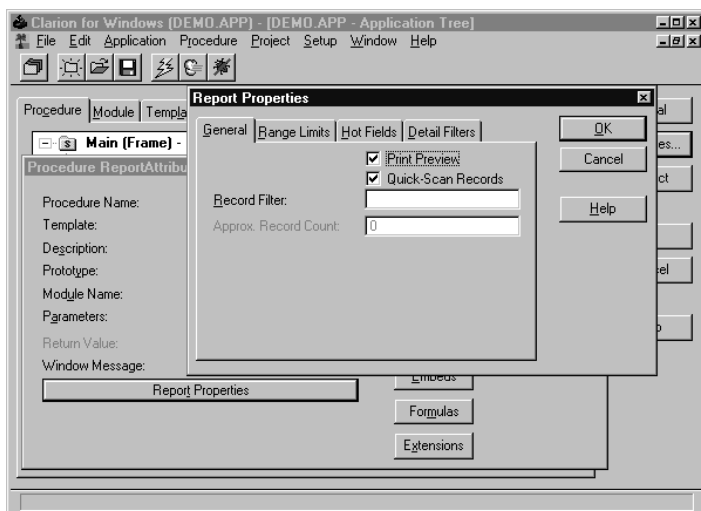
PrintDate = TODAY() (дата печати = сегодня)

15. Нажмите Exit! (выход) и сохраните вашу встроенную исходную программу.

## Выполнение предварительного просмотра печати (Implementing Print Preview)

---

Генерируйте процедуру отчета с помощью шаблона Отчет или Мастера отчета (смотрите Мастера и Шаблоны процедур). В диалоговом окне процедуры отчета Procedure Properties (свойства процедуры) нажмите кнопку Report Properties (свойства отчета), затем отметьте поле Print Preview (предварительный просмотр печати) в диалоговом окне Report Properties (свойства отчета) (смотрите выше Свойства отчета).



Выполнение предварительного просмотра печати в вашем отчете.

Нажмите кнопку Report Properties (свойства отчета), затем отметьте поле Print Preview (предварительный просмотр печати)

Если вы предпочитаете кодировать вручную свой процесс предварительного просмотра печати, смотрите PREVIEW (ПРЕДВАРИТЕЛЬНЫЙ ПРОСМОТР) в Справочнике языка, где приведена более полная информация и примеры.

## **Печатание бирок (динамически масштабируемых) (Printing Labels)**

Печать бирок просто означает печать отчета со многими столбцами так, чтобы строки и столбцы отчета совпадали с формами коммерческих бирок, используемыми вами.

В реальном мире различные пользователи будут иметь различную бумаги в разное время, поэтому в идеале пользователь должен суметь назначить разметку бумаги во время работы, а отчет должен подходить для назначенной бумаги.

- Чтобы печатать динамически масштабируемые бирки:
  - Добавьте файл Бирки к вашему словарю данных.
  - Создайте процедуры Просмотр (Browse) и Обновление (Update) для файла Бирки.
  - Создайте процедуру Отчет для печати бирок.
  - Добавьте файл Бирки к Схеме файлов Отчета.
  - Встройте код для изменения размеров во время работы.

### **Добавьте файл Бирки к вашему словарю данных**

Файл может использовать любую файловую систему, которую вы предпочтете, и он

должен содержать следующие поля. Никакие ключи не требуются, использование LabelType в качестве единственного ключа обеспечивает сортировку по алфавиту и предотвращает дублирование вводов. Смотрите Редактор словаря - Добавление файлов к словарю и Добавление или модификация полей.

Имя	Тип	Длина	Начальная величина
LabelType	String	15	
PageWidth	Decimal	5,2	
PageHeight	Decimal	5,2	
LabelWidth	Decimal	5,2	
LabelHeight	Decimal	5,2	
TopMargin	Decimal	5,2	
LeftMargin	Decimal	5,2	
FontSize	Byte		11

### **Создать Процедуру Просмотра и Форму Обновления для файла бирки**

Самый быстрый способ построения этих процедур - это использование Мастера процедуры. Смотрите Мастера и Шаблоны процедур - Мастер Процедуры Просмотра.

1. Находясь в Дереве приложения, нажмите клавишу INSERT.
2. В диалоговом окне New Procedure (новая процедура) наберите имя (BrowseLabels ), затем нажмите OK.
3. В диалоговом окне Select Procedure Type (выбор типа процедуры) отметьте поле Use Procedure Wizard (использование мастера процедур), а затем выберите Browse (просмотр).
4. Ответьте на вопросы Мастера:  
Выберите файл Бирки. Разрешите пользователю обновлять записи. Предоставьте кнопку "Select" (выбор). Мастер строит обе процедуры, затем открывает диалоговое окно Procedure Properties (свойства процедуры) для процедуры Просмотр.
5. Нажмите кнопку OK чтобы покинуть диалоговое окно Procedure Properties (свойства процедуры).

### **Создать процедуру отчет для бирок**

Ваш отчет должен соблюдать эти соглашения. Поля отчета должны иметь Высоту и Ширину по умолчанию (Default), должно быть только одно поле в строке. Так что если вам нужно показать Имя (FirstName) и Фамилию (LastName) на одной и той же строке, вы должны конкатенировать эти поля в одно поле. Опционные поля, такие как вторая адресная строка, должны быть показаны с помощью СТРОКИ (STRING) с атрибутом SKIP (пропустить).

1. Создайте отчет для вашего файла адресов. Используйте, если хотите, Мастер отчета, но пока не беспокойтесь о форматировании. Просто убедитесь, что отчет содержит все именные и адресные поля, которые вам нужны для ваших бирок.

Назовите отчет таким образом, как вы бы это сделали для любого другого отчета. Смотрите Генератор приложений - Добавление процедуры к вашему приложению и Меню и линейки инструментов - Создание меню вашего приложения.

2. Находясь в Дереве приложения, щелкните правой клавишей мыши на вашей процедуре отчета и выберите из всплывающего меню Report (отчет).

3. Удалите все другие разделы, за исключением раздела Тело отчета.

Щелкните на линейке заголовка раздела, затем нажмите клавишу DELETE.

4. Для строк отчета с более чем одним полем конкатенируйте поля и удалите конечные пробелы.

Этот шаг улучшает внешний ваших бирок удалением ненужных пробелов. Этот шаг требуется также для того, чтобы поддержать встроенный код, рекомендуемый ниже.

Следуйте за шагами, описанными в теме Как обрезать и конкатенировать поля среди Общих вопросов интерактивной помощи.

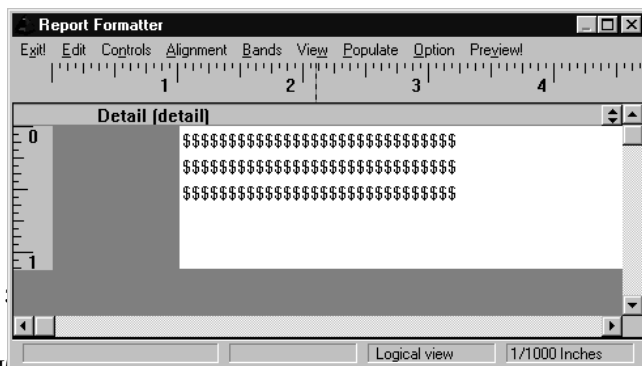
5. Установите свойства для каждого поля отчета.

**Совет: Используйте Панель инструментов “Свойства” для удобной идентификации полей: Выберите Option>Propertybox.**

Сгруппируйте ваши именные и адресные поля в вертикальном формате, то есть, один под другим, начиная с верхнего левого угла полосы Тела отчета. Используйте инструменты Выравнивание для точного выравнивания. Смотрите выше Назначение полей для печати.

Не забудьте установить Высоту и Ширину для каждого поля на умолчание (Default) на закладке Position (положение) и отметьте поле SKIP на закладке General (общие свойства) для опционных полей, таких как второе адресное поле.

Ваш отчет должен выглядеть подобно:



6. Назначьте метку соответствия (атрибутирование) для каждого поля отчета.

Щелкните клавишей мыши над телом отчета, затем наберите ?detail в поле Свойства поля Use.

7. Выйдите (Exit!) из Форматера отчета и сохраните свои изменения.

### **Добавьте файл Бирки к схеме файлов отчета**

Этот шаг гарантирует, что процедура отчета открывает файл Бирки и, следовательно, имеет доступ к буферу записи файла бирки. Описываемый ниже встроенный код использует величины позиции и размера в этом буфере записи для присваивания рабочих текущих позиций и размеров отчету.

1. Находясь в Дереве приложения, щелкните правой клавишей мыши на вашей процедуре отчета и выберите из всплывающего меню Properties (свойства).

2. Из диалогового окна Procedure Properties (свойства процедуры) нажмите кнопку Files (файлы).

3. Из диалогового окна File Schematic (схема файла) дважды щелкните клавишей мыши на Other Files (другие файлы).

4. Из диалогового окна Insert File (вставить файл) дважды щелкните на Labels (бирки).

5. Нажмите ОК чтобы выйти из диалоговых окон.

### **Встройте код динамического изменения размеров**

Теперь мы встроим код в три точки в процедуре отчета:

- Начало процедуры - вызовите процедуру BrowseLabels чтобы разрешить пользователю использовать определенную бумагу.
- После открывания отчета - вызовите подпрограмму ResizeTheReport для присваивания размера бирки, назначенного пользователем.
- Подпрограммы процедуры - подпрограмма ResizeTheReport.

1. Находясь в Дереве приложения, щелкните правой клавишей на вашей процедуре отчета, затем выберите Embeds (вставки) из всплывающего меню.

2. Из диалогового окна Embedded Source (встроенная исходная программа) дважды щелкните на точке вставки Beginning of Procedure (начало процедуры), After Opening Files (после открывания файлов).

3. Из диалогового окна Select Embed Type (выбор типа вставки) дважды щелкните на SOURCE и наберите следующие операторы:

!Разрешить динамическое назначение разметки бумаги  
 GlobalRequest = SelectRecord      !разрешить выбор кнопки  
 BrowseLabels                      !вызвать процедуру печати бирок

4. Выйдите из редактора текста и сохраните ваши изменения.

5. Находясь в диалоговом окне Embedded Source (встроенная исходная программа), дважды щелкните на точке вставки After Opening Report (после открывания отчета).

6. Находясь в диалогового окна Select Embed Type (выбор типа вставки) дважды щелкните на SOURCE и наберите следующие операторы:  
 DO Resize The Report

7. Выйдите из текстового редактора и сохраните свои изменения.

8. Находясь в диалоговом окне Embedded Source (встроенная исходная программа), дважды щелкните на точке вставки Procedure Routines (подпрограммы процедуры).

9. Находясь в диалогового окна Select Embed Type (выбор типа вставки) дважды щелкните на SOURCE и наберите следующие операторы.

Если вы назначили метку для своей структуры отчета REPORT, измените метки “report” внизу для соответствия вашей метке.

ResizeTheReport    ROUTINE

!!=====

!! Изменить отчет для получения соответствия с назначенной бумагой.

!!Предполагается, что единица измерения отчета равна 1/1000 дюйма.

11=====

SETTARGET(report) !Сделать отчет мишенью для присвоения свойства

!!=====

!!Определить атрибут At ОТЧЕТА ( области печати тела отчета).

report{PROP:Width} = LAB:PageWidth \* 1000

report{PROP:Height} = LAB:PageHeight \* 1000

!!Отрегулировать поля для центрирования адресного текста в пределах к а ж д о й бирки.

report{PROP:Xpos} = LAB:LeftMargin \* 1000

report{PROP:Ypos} = LAB:TopMargin \* 1000

!!=====

!!Определить атрибут AT ТЕЛА ОТЧЕТА.

!!ТЕЛО ОТЧЕТА должно быть одного размера с каждой индивидуальной



биркой

```
?detail{PROP:Width} = LAB:PageWidth * 1000
?detail{PROP:Height} = LAB:PageHeight * 1000

!!=====
!!Настроить размер шрифта и вертикальное положение полей отчета.
LOOP i# = 2 TO LASTFIELD ( ) !Пропуск #1, полосы тела отчета.
  i#{PROP:FontSize} = LAB:FontSize !Присвоить шрифт из файла бирки.
IF i# > 2 !Не перемещайте первое поле.
  Bottom0LastField# = (i#-1){PROP:Ypos}+(16*LAB:FontSize)
  i#{PROP:Ypos} = Bottom0LastField# !Изменение позиции, основанное
    !на размере шрифта.
END
END
SETTARGET(report) !Сбросить мишень по умолчанию на активное окно.
EXIT !Выйти из ROUTINE (подпрограммы).
```

Вот оно! Сделайте и прогоните ваше приложение. Когда будете прогонять свой отчет для печати бирок, вы будете иметь возможность вводить размеры ваших различных бумаг. Размеры запоминаются в файле бирок, где они доступны для вашего выбора или модификации каждый раз, когда вы печатаете бирки.

### **Печать одной записи на страницу (Printing One Record per Page)**

---

Чтобы напечатать отдельную страницу для каждой записи, отметьте поле PAGEAFTER (страница после) в диалоговом окне Detail Properties (свойства тела отчета). Смотрите выше Структуры и свойства отчета - Тело отчета.

### **Печать слитых документов (Printing Mail-Merge Documents)**

---

В случае слияния документа вы обычно помещаете поля имя и адрес в ЗАГОЛОВОК (HEADER), а затем резервируете ТЕЛО ОТЧЕТА (DETAIL) для многострочного текстового элемента управления. Смотрите выше Страничный Заголовок. Смотрите ниже Многострочный текст.

### **Печатание графики**

---

Графические элементы украшают отчет и направляют взгляд читателя к данным. Описываемые ниже элементы управления позволяют вам добавить изображения, линии и формы к вашему отчету. Смотрите главу Установка свойств элемента управления, где приводится более полная информация об описываемых элементах управления.

Используйте изображения,

поля

линии

и другие средства для  
оформления своего отчета.



## Banyan Supply INVOICE

### Изображение

Скорее всего, вы захотите поместить изображение, например фирменный знак, в HEADER. Вы можете выбрать любые форматы графических файлов, поддерживаемых для элементов управления окна; однако, печать больших изображений, в особенности файлов .JPG, может создать проблемы для некоторых принтеров.

Наиболее важным обстоятельством при размещении растровых изображений является их размер - Clarion автоматически изменяет размер растровой картинку таким образом, чтобы он подходил к размеру, выделенному под эту картинку в отчете. Это может вызвать искажения. Файлы WMF искажают меньше всего, однако самый простой способ предотвратить искажения - это сохранение отношения высоты к ширине для ваших изображений.

Например, для изображения размером в 640x480 пиксел, определим для него отношение высоты к ширине; оно равно 4:3. Спланируйте поле изображения в том же отношении - например, 2000x1500 тысячных дюйма, что составляет на странице 2 x 1.5. дюйма

**Совет: Всякий раз, когда это возможно, используйте векторную графику, такую как Windows Metafile Format (\*.WMF). Когда вам нужно сжать или растянуть изображения, вид таких изображений менее подвержен искажениям, чем для растровых.**

Чтобы разместить элемент управления IMAGE (изображение) в вашем отчете:

1. Выберите элемент управления Изображение из панели инструментов элементов управления или выберите Controls > Image.

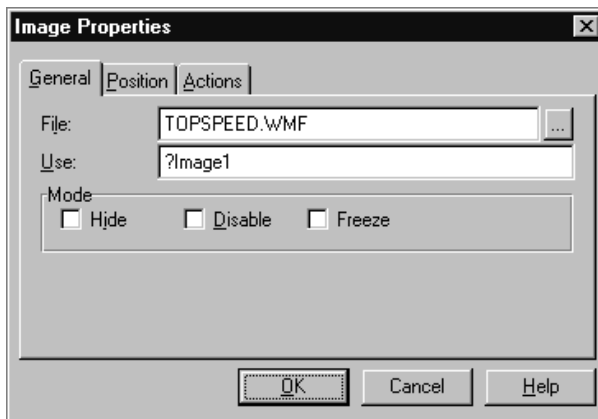
2. Щелкните мышью в полосе, в которой вы хотите поместить изображение.

Форматер отчета помещает элемент управления IMAGE в структуру отчета. Центр

перекрестия курсора позиционирует верхний левый угол элемента управления.

3. Дважды щелкните клавишей мыши на элементе управления, который вы только что поместили.

Появляется диалоговое окно Image Properties (свойства изображения).



4. Наберите полное (включающее путь) имя файла в поле File или нажмите эллиптическую (...) кнопку рядом с этим полем и выберите файл из стандартного диалога открывания файла.

Clarion автоматически связывает файл изображения с исполняемой программой, когда файл полностью поименован в элементе управления, поэтому вам нет необходимости поставлять файл изображения отдельно.

5. В поле USE наберите метку соответствия для ссылки на элемент управления Изображение в исходной программе.

6. Нажмите кнопку Position.

7. Введите точный размер изображения в фиксированных полях Width(ширина) и Height(высота).

Вы можете также изменить размер изображения, щелкнув на нем и протаскив его за ручки.

8. Нажмите ОК.

### **Линия**

Линии представляют собой простейшее средство визуального отделения разделов или полей внутри отчета. Чтобы поместить линию:

1. Выберите инструмент “Линия” из панели инструментов элементов управления или

выберите Controls > Line - курсор примет форму перекрестия.

2. Щелкните мышью в полосе, в которой вы хотите поместить линию.

Центр крестообразного курсора определяет положение крайней левой точки. Форматер отчета помещает элемент LINE в структуру отчета. Перемещайте линию и меняйте ее размер используя “Рукоятки” линии.

3. Дважды щелкните клавишей мыши на только что размещенной линии.

Появится диалоговое окно Line Properties (свойства линии).

4. Наберите имя метки соответствия в поле Use для ссылки на линию в тексте программы.

5. Нажмите закладку Position, если вы хотите установить точные координаты линии.

Чтобы установить горизонтальную линию, обязательно отметьте поле Fixed (фиксированная) в группе Height (высота) и наберите нуль в соседнее с этим поле. Высота является мерой вертикального расстояния между начальной и конечной точкой; для горизонтальной линии она равна нулю. В группе Width (ширина) наберите длину линии в поле Fixed (фиксированная).

Чтобы установить вертикальную линию, обязательно отметьте поле Fixed в группе Width (ширина) и наберите нуль в соседнее с этим поле. Ширина является мерой горизонтального расстояния между начальной и конечной точкой; для вертикальной линии она равна нулю. В группе Height (высота) наберите длину линии в поле Fixed.

Чтобы установить горизонтальную или вертикальную линию в полную ширину или высоту раздела, отметьте Full (полный).

**Совет: Чтобы создать горизонтальную разделительную линию, используемую, например, для отделения раздела HEADER от раздела DETAIL установите отметку Full Width (полная ширина) и установите фиксированную высоту Fixed Height равной нулю.**

6. Чтобы установить цвет линии, нажмите кнопку Line Color (цвет линии) на закладке Extra. Затем выберите цвет из диалогового окна Line Color.

### **Прямоугольник (BOX)**

Вы можете выделить поле отчета, помещая под ним серое поле. Вы можете поместить весь отчет в рамку, размещая в поле отчета незакрашенный прямоугольник достаточного

размера.

**Совет:** В случае перекрывания одного элемента управления другим выберите **Edit>Set Control Order** чтобы обеспечить порядок, при котором нижележащий элемент управления печатается прежде лежащего поверх него элемента управления; в противном случае вышележащий элемент управления может быть затушеван.

Чтобы разместить прямоугольник:

1. Выберите инструмент “прямоугольник” из панели инструментов элементов управления или выберите **Controls > Box**.

2. Щелкните мышью в полосе, в которой вы хотите поместить прямоугольник.

Центр крестообразного курсора определяет верхний левый угол прямоугольника. После вашего щелчка форматер отчета помещает элемент управления Прямоугольник в структуру отчета. Используйте “Рукоятки” для изменения размера прямоугольника и его положения.

3. Дважды щелкните мышью только что размещенный прямоугольник. Появится диалоговое окно **Box Properties** (свойства прямоугольника).

4. Наберите метку соответствия в поле **Use** для ссылки на прямоугольник в тексте программы.

5. Отберите закладку **Extra**.

6. Установите атрибут закрашивания **COLOR**.

Если вам нужно сплошное цветное поле, наберите код цвета в поле флажков **Fill Color** (заполнить цветом) либо нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна **Fill Color**.

Если вы хотите иметь цветную границу, наберите код цвета в поле флажков **Border Color** (цвет границы) либо нажмите эллиптическую (...) кнопку для того, чтобы выбрать цвет из диалогового окна **Border Color**.

7. Если вы хотите иметь у вашего прямоугольника закругленные края, установите флажок в поле **Round(закругление)** для установки атрибута **ROUND**.

8. Чтобы установить размер поля не приблизительно, мышью, а точно, с клавиатуры, отберите закладку **Position**.

Наберите нужные вам размеры в полях **Fixed Width(фиксированная ширина)** и **Fixed Height(фиксированная высота)**.

**Совет:** Чтобы создать кромку или “рамку” вокруг всего отчета, разместите прямоугольник в полосе **Form (бланк)**. Убедитесь, что бланк (**FORM**)

занимает всю страницу. Создайте поле с рамкой, но без заполнения, и установите ширину и высоту Full(полная).

9. Нажмите кнопку ОК, чтобы закрыть диалоговое окно Box Properties.

### Эллипс

При размещении в отчете эллипса (ELLIPSE) следуйте тем же процедурам, что и при размещении прямоугольника.

## **Печатание многострочного текста с помощью автоматического перехода на новую строку (Printing Multi-line Text with Word-wrap)**

Элемент управления “многострочный текст” может печатать длинную строку символов (такую, как МЕМО), автоматически переходя на новую строку и печатая столько строк, сколько требует содержание МЕМО.

1. Отберите инструмент Текст (Text) с панели инструментов “Элементы управления” или выберите Controls> Text Field.

2. Щелкните клавишей мыши в полосе, которая будет содержать данный элемент управления.

Центр перекрестия курсора указывает положение верхнего левого угла данного элемента управления. Появляется диалоговое окно Select Field (выбор поля). Воспользуйтесь этим диалоговым окном для того, чтобы отобрать (или создать) поле словаря данных или переменную памяти для печати.

3. Нажмите кнопку Select (выбрать).

Форматер отчета (Report Formatter) помещает элемент управления ТЕХТ (текст) в структуру данного отчета. Изменить размер текстового поля можно с помощью мыши.

4. Щелкните правой клавишей мыши на данном текстовом элементе управления и выберите Properties (свойства) из всплывающего меню.

5. Отберите закладку Extra (дополнительные возможности), затем отметьте поле Resize (изменение размера).



Во время работы текст расширяется вниз, расширяя при необходимости тело отчета, для размещения всего текста мемо! Атрибут RESIZE (изменить размер) только регулирует высоту ТЕКСТА и ТЕЛА ОТЧЕТА.

## **Отчеты, которые выглядят как окна (Reports That Look Like Windows)**

Форматер отчета дает вам возможность размещать на странице отчета практически все то же самое, что вы можете поместить на экран. Это могут быть специализированные элементы управления, такие как поля списка, поля флажков, групповые поля, радиокнопки и т.д.

В большинстве случаев установка свойств элемента управления для отчета идентична установке свойств элемента управления для окна. Дополнительную информацию по каждому из нижеприведенных элементов управления вы можете получить в главе Элементы управления и их свойства.

### **Поле списка**

Когда данные, которые вам нужны в отчете, существуют в QUEUE, вы можете поместить в отчет поле списка. Поле списка, которое появляется на странице, подобно элементу управления LIST, который появляется на экране, хотя у него, очевидно, будут не те же самые функциональные возможности - печатная страница не поддерживает, например, линейки прокрутки.

Так как для поля списка требуется очередь и так как шаблон отчета Clarion'a обычно не использует очередь, несколько встроенных операторов исходной программы следует добавить к генерированной шаблоном процедуре отчета для того, чтобы использовать поле списка. Или же вы можете вручную кодировать процедуру отчета или использовать шаблон отчета, который предусматривает употребление очереди.

Во время печати, когда первый раз программа делает цикл через очередь, она печатает заголовок LIST и первую позицию в очереди. Каждый дополнительный цикл печатает следующий элемент очереди queue без повторения заголовка header. Колонтитул (FOOTER) списка LIST печатается в конце очереди

Чтобы разместить список в вашем генерированном шаблоне отчета:

1. Выберите инструмент “Поле списка” в панели инструментов элементов управления или выберите Controls > List Box.

2. Щелкните мышью в полосе, которая должна содержать данный элемент.

Центр крестообразного курсора определяет верхний левый угол списка. После вашего щелчка появляется Форматер поля списка. Используйте форматер поля списка так, как если бы вы конструировали поле списка для экрана. Смотрите главу Форматер поля списка.

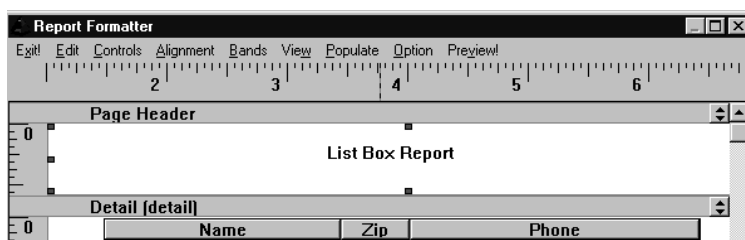
Закончив, нажмите кнопку ОК.

3. Сделайте данное поле списка высотой в одну (1) строку.

Протащите нижнюю ручку до тех пор, пока не останутся видимыми только заголовки поля списка. Это ликвидирует пустое пространство между строками.

4. Сделайте полосу тела отчета той же высоты, что и поле списка.

Перетащите поле списка вверх полосы тела отчета, затем потяните нижнюю ручку полосы тела отчета до низа поля списка. Это ликвидирует пустое пространство между строками. Ваш отчет должен теперь выглядеть похожим на эту иллюстрацию.



5. Дважды щелкните мышью в поле списка, которое вы только что разместили, для того, чтобы открыть диалоговое окно List Properties (свойства списка).

6. Наберите метку соответствия в поле Use для ссылки на элемент управления в исходной программе.

7. В поле From (откуда) наберите метку очереди, которую показывает данное поле списка, или нажмите эллиптическую (...) кнопку для того, чтобы выбрать или назначить очередь.

8. Нажмите кнопку ОК чтобы закрыть диалоговое окно List Properties.

### **Необходимые вставки исходного кода:**

Вообще говоря, операторы встроенной исходной программы должны делать три вещи: объявлять очередь QUEUE, загружать эту очередь данными и читать эту очередь вместо генерированного шаблоном VIEW.

В приведенном ниже примере я позаимствовал исходную программу, генерированную шаблоном элемента управления FileDrop для объявления и загрузки очереди. Вы можете генерировать подобный код, создавая процедуру с шаблоном элемента управления FileDrop. Заселите поле списка FileDrop теми же полями, которые вы разместили в поле списка отчета, затем генерируйте исходную программу.



1. Объявите QUEUE (очередь) в точке вставки Declaration Section (объявление раздела), что-то вроде этого:

```
Customer View VIEW(Customer)!Объявить VIEW для файла Customer
      !(заказчик)
      PROJECT(CUST:Name)
      END
```

```
CustomerQ  QUEUE,PRE           !Объявить очередь QUEUE,
      !основанную на VIEW
Q:CUST:Name  LIKE(CUST:Name)
      END
QItemsProcessed  LONG
```

2. Загрузите очередь в точке вставки After Turning QuickScan On (после включения быстрого сканирования), подобно нижеследующему:

```
DO LoadCustomerQ
```

И в точку вставки Procedure Routines (подпрограммы процедуры), подобно следующему:

```
LoadCustomerQ  ROUTINE
```

```
SET(CUST:ByName)
OPEN(CustomerView)
IF ERRORCODE( )
      StandardWarning(Warn:ViewOpenError)
END
LOOP
      NEXT(CustomerView)
      IF ERRORCODE( )
            IF ERRORCODE( ) = BadRecErr
                  BREAK
            ELSE
                  StandardWarning(Warn:RecordFetchError,'Customer')
                  POST(Event:CloseWindow)
                  EXIT
            END
      END
      Q:CUST:Name = CUST:Name
      ADD(CustomerQ)
END
CLOSE(CustomerView)
```

3. Читайте из очереди, а не из генерированного шаблоном VIEW.

В точке вставки Top of GetNextRecordRoutine (верх подпрограммы “получить следующую запись”):

```
OMIT('End_Omit')           !Опустить генерированное шаблоном
                           !чтение
```

и в точке вставки GetNextRecordRoutine, Next failed (получить следующую запись, следующая отказывает):

```
!End_Omit                 !Читать очередь вместо View
QItemsProcessed +=1       !увеличение счетчика записи
Get(CustomerQ,QItemsProcessed) !получить следующую запись из
                           !очереди
IF ERRORCODE( )           !Начать проверку ошибки. Шаблон
                           !заканчивает оператор IF.
```

### **Поле опций**

В вашем отчете вы можете напечатать структуру OPTION. Она появляется на странице точно также как на экране - как поле опций или необязательных выборов. Вы можете поместить структуру опций на страницу только для того, чтобы поддержать радиокнопки. Вы можете спрятать рамку поля, так чтобы она не печаталась на странице.

1. Выберите инструмент Option Box (Поле опций) из панели инструментов элементов управления или выберите Controls > Option Box.

2. Щелкните мышью в полосе, которая должна содержать структуру OPTION.

Центр крестообразного курсора определяет положение верхнего левого угла поля. Форматер отчета помещает структуру OPTION внутрь структуры отчета. Используйте “Рукоятки” для изменения размера поля и его положения.

3. Дважды щелкните клавишей мыши на только что размещенном поле. Появляется диалоговое окно Options Properties(свойства поля опций).

4. В поле Text (текст) наберите заголовок для поля опций.

Если вы решите не прятать поле параметров при печати, заголовок появится в верхнем левом углу поля, точно так же, как на экране.

5. В поле Use наберите метку соответствия поля для ссылки на данный элемент управления в тексте программе.

6. Нажмите закладку Extra.

7. Очистите от флажков поле **Boxed** (в прямоугольнике) чтобы спрятать поле, но не радиокнопки.

8. Нажмите **ОК**.

Теперь вы должны добавлять каждую радиокнопку по отдельности, помещая их в поле **OPTION**.

### **Радиокнопка**

Размещение кнопок **RADIO** в печатном отчете обеспечивает визуальную помощь для пользователя, показывая выбранную величину а также все возможные величины для данного поля.

Прежде чем поместить радиокнопки в отчет, вы должны сначала, используя команду **Controls > Option Box**, разместить в отчете структуру **OPTION**. Кнопка **RADIO** должна быть помещена внутри группового поля, представляющего структуру **OPTION**. Если вы попытаетесь поместить радиокнопку без структуры **OPTION**, интегральная среда разработки выдаст сообщение об ошибке.

1. Разместите поле опций.

2. Выберите инструмент “Радиокнопка” из панели инструментов элементов управления или выберите **Controls > Radio Button**.

3. Щелкните мышью внутри поля опций, которое вы только что поместили.

Центр крестообразного курсора определяет верхний левый угол радиокнопки. Форматер отчета помещает элемент **RADIO** в структуру **OPTION**.

4. Дважды щелкните клавишей мыши на кнопке, которую вы только что поместили. Появится диалоговое окно **Radio Button Properties** (свойства радиокнопки).

5. В поле **Параметр** наберите заголовок для радиокнопки.

Заголовок появится около кнопки, так же как происходит на экране.

6. В поле **Use** наберите метку соответствия для ссылки на элемент управления в исходной программе.

Радиокнопка автоматически “включается и выключается” в соответствии с величиной переменной, установленной в атрибуте **USE** поля опций.

7. Нажмите **ОК**.

## **Поле флажков**

Поле флажков (элемент управления CHECK) обеспечивает привлекательный способ показа выбора да/нет для поля - иначе может понадобиться целый столбец, в котором повторяется “один”, “да” или даже “.Т.” для каждой записи.

Напечатанное поле флажков выглядит подобно аналогичному экранному полю. Чтобы разместить поле флажков:

1. Выберите инструмент “Поле флажков” из панели инструментов элементов управления или выберите Controls > Check Box.

2. Щелкните мышью внутри полосы, где будет находиться данный элемент.

Центр крестообразного курсора определяет верхний левый угол поля флажков. Появляется диалоговое окно Select Field (выбор поля). Используйте это диалоговое окно для выбора (или создания) поля словаря данных или переменной памяти, показываемой для этого элемента управления. Это должна быть цифровая переменная, переводящая поле в положения вкл/выкл. Величина нуль указывает на то, что поле неотмечено; другая любая величина - отмечено.

3. Нажмите кнопку Select (выбрать).

Форматер отчета помещает структуру поля CHECK внутрь структуры отчета.

4. Дважды щелкните клавишей мыши на данном элементе управления или щелкните на нем правой клавишей и выберите из всплывающего меню Properties (свойства).

5. В поле Parameter наберите заголовок для поля флажков.

Заголовок появится около поля флажков, точно так, как это происходит на экране.

6. Нажмите ОК.

## **Групповое поле**

Главная причина использования групповых полей в отчете - это необходимость того, чтобы группа элементов управления на бумаге напоминала их вид на экране.

Чтобы разместить элемент управления GROUP в отчете:

1. Выберите инструмент “Поле группы” из панели инструментов элементов управления или выберите Controls > Group Box.

2. Щелкните мышью внутри полосы, в которую вы хотите поместить данное групповое поле.

Центр крестообразного курсора определяет верхний левый угол группового поля. Форматер отчета помещает структуру GROUP в структуру отчета.

3. Дважды щелкните клавишей мыши на элементе управления или щелкните на нем правой клавишей и выберите Properties (свойства) из всплывающего меню.

Появится диалоговое окно Group properties (свойства группы).

4. В поле Parameter (параметр) наберите текст заголовка для группового поля.

Когда печатается отчет, этот текст появляется слева наверху от поля группы, если вы отмаркировали поле флажков Boxed.

5. В поле Use наберите метку соответствия для ссылки на элемент управления в исходной программе.

6. Выберите закладку Extra.

7. Очистите поле флажков Boxed чтобы спрятать поле, но не внутренние элементы управления.

8. Нажмите ОК.

9. Добавьте дополнительные элементы управления к группе.

### **Покупные элементы управления**

Вы можете поместить в ваш отчет заказной элемент .VBX CUSTOM. Имеется большое число заказных библиотек элементов управления, которые очень подходят для отчетов , включая диаграммы и другие визуальные элементы. Чтобы разместить эти элементы:

1. Выберите инструмент “.VBX” из панели инструментов элементов управления или выберите Controls > Custom Control.

2. Щелкните мышью внутри полосы, в которой будет размещаться данный элемент.

Центр крестообразного курсора определяет верхний левый угол заказного элемента. После вашего щелчка, появляется диалоговое окно Select Custom Control (выбор покупного элемента управления. Используйте это диалоговое окно для выбора покупного элемента управления.

3. Нажмите кнопку ОК. Форматер отчета помещает элемент CUSTOM внутрь структуры отчета. Для изменения размеров и положения покупного элемента управления используйте его “рукоятки”.

4. Дважды щелкните клавишей только что размещенный элемент управления.

Появится диалоговое окно Custom Control Properties (свойства покупного элемента управления.

5. Наберите заголовок элемента в поле Text.

.VBX - элемент может показывать, а может и не показывать свой заголовок на странице отчета, это зависит от того, каким .VBX вы пользуетесь.

6. Наберите имя переменной в поле Use .

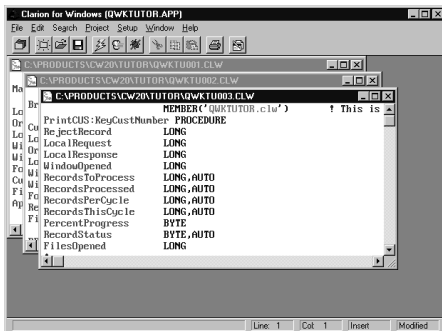
Тип переменной зависит от .VBX элемента. Величина переменной передается к .VBX элементу управления. Дополнительные подробности об этом вы найдете в главе Элементы управления и их свойства.

7. По желанию отметьте поле Meta для печати данного элемента управления как метафайл.

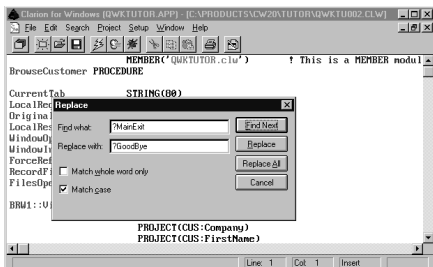
## Глава 13 Текстовый редактор



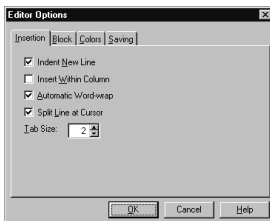
Используете ли вы текстовый редактор для добавления вставок программного текста в процедуру или для написания программы “вручную”, текстовый редактор является мощным и удобным для программиста редактором.



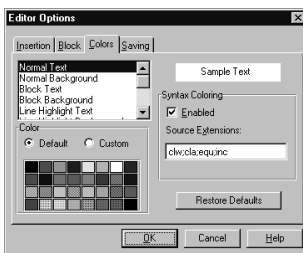
Текстовый редактор выделяет цветом элементы синтаксических конструкций, облегчая тем самым чтение вашего кода.



Текстовый редактор оснащен полным набором средств поиска и замены.



Установите параметры настройки редактирования в соответствии с вашим вкусом.



Настройте по своему вкусу цветовое выделение синтаксиса.

- Открываем текстовый редактор
- Управление окнами текстового редактора
- Использование инструментов текстового редактора
- Использование меню редактирования
- Использование линейки инструментов
- Поля
- Использование меню поиска
- Использование файлового меню
- Отступ блока
- Макросы
- Настройка текстового редактора
- Опции вставки
- Опции блока
- Опции цвета
- Опции сохранения
- Редактирование ошибок

Эта глава представляет текстовый редактор. Если вы позволите генератору приложений писать большую часть вашей программы, вы вероятно будете использовать текстовый редактор только для написания ваших вставок. Если вы пишете свою программу “вручную”, вы вероятно будет интенсивно использовать текстовый редактор для создания и обслуживания текста вашей программы. Чтобы помочь выполнить вашу задачу, текстовый редактор имеет следующие возможности:

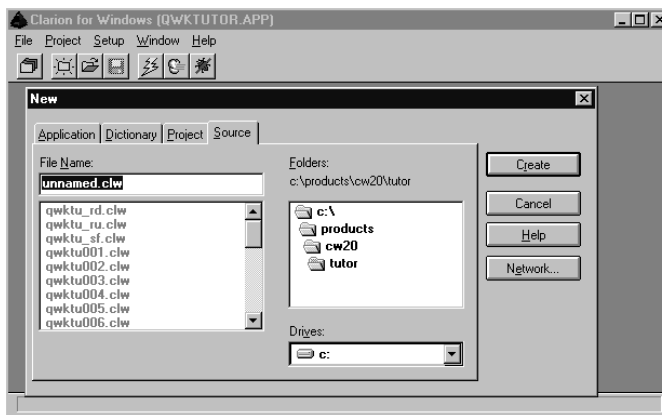
- Множественные документные окна, в которых вы можете редактировать столько документов, сколько позволяет ваша система.
- Цветовое выделение синтаксиса, которое облегчает чтение отдельных строк кода.
- Всегда доступны поиск и замена для любых строк символов.
- Авто-отступ, облегчающий чтение кода.
- Локатор следующей ошибки и предыдущей ошибки.
- Текущую позицию курсора (строка и столбец), высвечиваемую в строке статуса.



## Открываем текстовый редактор

Каждый раз, когда вы просматриваете текст исходной программы Clarion, вы используете текстовый редактор. Имеется несколько способов открыть файл исходного текста:

□ Сначала выполните команду **File** → **New**, а затем выберите закладку **Source** (источник) в диалоговом окне **New** (новый). Пройдите к каталогу вашего источника и заполните имя нового файла в этом стандартном диалоге. Затем нажмите кнопку **Create** (создать). Это открывает пустой файл текста исходной программы.



Диалоговое окно **New** дает вам возможность создать новый Clarion-файл исходного текста.

□ Выполните команду **File** → **Open**, а затем выберите закладку **Source** и дважды щелкните клавишей мыши на файле исходной программы в стандартном диалоговом окне **Open File**.

□ Используйте команду **File** → **Pick** для просмотра самых последних из отредактированных вами файлов. Выберите закладку **Source**, выделите файл исходной программы и затем нажмите кнопку **Select** (выбрать).

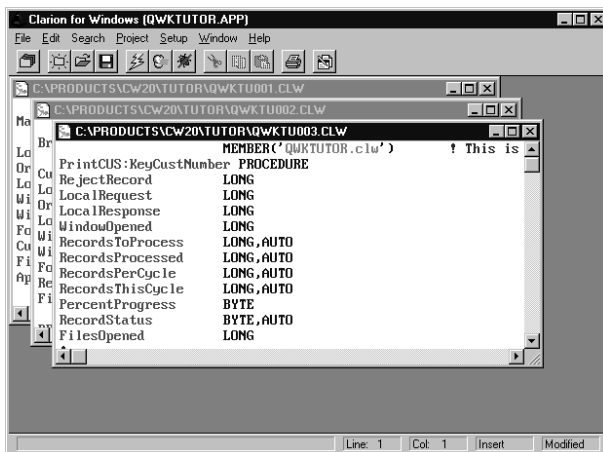
□ Внутри диалогового окна **Project Editor** (редактор проекта), выделите (.CLW) файл исходной программы и нажмите кнопку **Edit** (редактор). Кнопка **Edit** включена только для проектов, кодированных вручную.

□ После компилирования, если обнаружены ошибки, нажмите кнопку **Edit Errors** (редактирование ошибок).

## Управление окнами текстового редактора

Каждый файл исходной программы появляется в отдельном документном окне. В данном разделе дается краткий обзор действий, которые вы можете предпринять, чтобы изменить план этих окон:

Документное окно редактора показывает цветное кодированное синтаксиса.



Close a window  
(закрыть окно)

Выберите File III Close из главного меню или выберите Close из системного меню окна или дважды щелкните мышью в системном меню окна или нажмите CTRL+F4.

Activate a window  
(активизировать  
окно)

Щелкните мышью где-нибудь внутри окна; или выберите имя документа из меню Window. Другой вариант: нажимайте CTRL+F6 или CTRL+TAB и подождите пока активизируется нужное вам окно.

Move a window  
(переместить окно)

Передвиньте мышью строку заголовка документного окна. Или выберите Move из системного меню окна, а затем используйте клавиши курсора и нажмите ENTER, чтобы установить окно на место.

Resize a window  
(изменить размер  
окна)

Передвиньте границу с помощью мыши. Или выберите Size из системного меню окна, используйте клавиши курсора для изменения размера, затем нажмите ENTER для возвращения к редактированию.

Maximize a window  
(увеличить окно до  
предела)

Нажмите кнопку максимизации в строке заголовка документного окна; или выберите Maximize из системного меню окна.

Iconize a window (свернуть  
окно до пиктограммами)

Нажмите кнопку минимизации на линейке заголовка окна; или выберите Minimize из системного меню окна.

Restore iconized (восстановить  
свернутое до пиктограммы  
окно)

Чтобы восстановить свернутое до пиктограммы документное окно, щелкните дважды мышью на пиктограмме окна; или выберите Restore из системного меню пиктограммы.

Cycle to next window  
(циклическое переключение  
окон)

Для переключения к следующему окну нажмите CTRL+F6 или CTRL+TAB.

Tile the Windows  
(размещение окон  
“Черепицей”)

Чтобы расположить все открытые документные окна рядом, выберите Window Tile Vertically (вертикальная черепица) Window Tile Horizontally (горизонтальная черепица) из главного меню. Это обеспечивает удобный доступ к документам, как показано на приводимой ниже иллюстрации.

Cascade Windows  
(каскадные окна)

Чтобы расположить все открытые окна так, чтобы были видны строки заголовков всех окон, выберите Window Cascade из главного меню.

## ***Использование инструментов текстового редактора***

Ввод и редактирование текста исходной программы с помощью текстового редактора подобно набивке документов с помощью большинства программ систем подготовки текстов. Набирайте код как если бы вы печатали на машинке, затем используйте различные инструменты текстового редактора и его команды для компоновки, копирования и модификации вашего кода.

### **Меню редактирования**

---

Для использования команды редактирования, такой как, например Cut или Copy, выделите текст, над которым вы хотите произвести действие команды, а затем выберите команду из меню или панели инструментов.

Когда вы хотите вставить текст, укажите щелчком мыши с курсором в виде луча место вставки и затем набивайте или Paste (вставить из буфера) новый текст.

Меню Edit показывает первичные команды редактирования. В следующих разделах последовательно обсуждаются все команды из меню Edit:

Undo (отмена последней выполненной программы)

Чтобы отменить последнее действие редактирования, выберите команду Undo. Эта команда меню изменяет к состоянию, когда действие не было сделано. Если вы наберете строку текста, данная позиция меню покажет, что вы можете Undo Line Edits (отменить редактирование строки). Если вы удалите строку программы, она позволит вам Undo Block Delete (отменить удаление блока). Действие некоторых команд, таких, как сохранение файла или заменить все, не может быть отменено.

Cut (вырезание блока)

Чтобы удалить выделенный текст из документа и удержать его в буфере обмена, выберите команду Cut. Ускоряющая клавиша для данной команды - CTRL+X. Кнопка панели инструментов с пиктограммой ножниц также активизирует эту команду.

Copy (копирование блока)

Чтобы скопировать выделенный текст и сохранить его в буфере, используйте команду Copy. Ускоряющая клавиша для данной команды - CTRL+C. Кнопка панели инструментов с перекрывающимися страницами на пиктограмме также активизирует эту команду.

Paste (вставка блока)

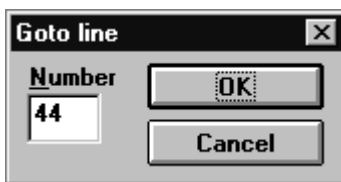
Чтобы поместить содержание буфера обмена (только текст) в документ в точку вставки, используйте команду Paste. Ускоряющая клавиша для данной команды - CTRL+V. Кнопка панели инструментов с пиктограммой страницы в раскрытой папке также активизирует эту команду.

Select All (отметить все)

Чтобы выделить весь текст в документе так, чтобы следующая команда подействовала на весь документ, выберите команду Select all.

Goto Line (перейти к строке)

Чтобы перейти к определенной строке исходной программы для последующего редактирования, выберите Goto Line... Ускоряющая клавиша для данной команды - CTRL+G.



Наберите номер строки в диалоге Goto Line.

Текстовый редактор помещает точку вставки в первый столбец с номером строки, который вы набрали в диалоговом поле. Строка статуса показывает текущие номера строки и столбца для позиции точки вставки.

Goto Next Error (перейти к

следующей ошибке) Чтобы переместить точку вставки к следующей ошибке компилятора, выберите эту команду. Редактор помещает курсор на ту часть оператора, где он нашел ошибку. Эта команда активизирована только после работы компилятора, который обнаружил ошибки.

Goto Previous Error (перейти к

предыдущей ошибке)

Чтобы переместить точку вставки к предыдущей точке, в которой исходная программа генерировала ошибку компилятора, выберите эту команду. Редактор помещает курсор на ту часть оператора, где он установил местонахождение ошибки. Эта команда включена только после компиляции, генерировавшей ошибку.

Set/Clear Tabstop (установка

/снятие табуляции) Помещает или снимает табуляционную метку в точку вставки.

Duplicate Line (дублировать строку)

Чтобы сдублировать всю строку и поместить копию на следующую строку, выберите эту команду или нажмите CTRL+2. Первоначальную строку не нужно выделять; просто поставьте курсор в любое место этой строки.

Toggle Case (смена

регистра)

Чтобы изменить регистр следующего символа после точки вставки, выберите эту команду или нажмите CTRL+/. Буква нижнего регистра становится буквой верхнего регистра и наоборот.

Delete Line (удалить строку)

Чтобы удалить всю строку, на которой находится точка вставки, выберите эту команду или нажмите CTRL+Y.

Delete Word (удалить слово)

Чтобы удалить слово, следующее за точкой вставки, выберите эту команду или нажмите CTRL+T.

Format Structure (форматировать

структуру)

Представляйте это себе как “визуальное редактирование” окна или отчета. Поместите точку вставки на любую строку в структуре и

выберите эту команду или нажмите CTRL+F. Кнопка панели инструментов с пиктограммой карандаша и бумаги также активирует эту команду. Форматер окна (или форматер отчета) показывает визуальное представление структуры, готовое для редактирования. Когда вы выходите из форматера окна (или форматера отчета), ваша исходная программа отражает сделанные вами изменения. Это обеспечивает высокий уровень взаимодействия на уровне исходной программы с инструментами визуального конструирования. Вы можете также поместить точку вставки в пустую строку, затем вызвать форматер окна для создания новой структуры. Когда вы возвращаетесь к текстовому редактору, текст исходной программы будет содержать новую структуру, которую вы создали с помощью форматера окна (или форматера отчета). Ваше дело убедиться, что вы вставили структуру в раздел данных программы.

## Линейка инструментов

Линейка инструментов текстового редактора обеспечивает быстрый доступ к большинству часто используемых команд редактирования: Cut, Copy, Paste, Format Structure. Это те же самые команды, доступ к которым достигается через меню редактирования. Кроме того, имеется кнопка печати для вызова стандартного диалогового окна печати окон. Щелкните клавишей мыши на этих кнопках для быстрого доступа к вашей команде.



Вырезать/ Копировать / Вставлять из буфера / Печать / Структура формата /  
Предыдущая ошибка / Следующая ошибка

## Поля

Список полей текстового редактора обеспечивает быстрый доступ к полям словаря данных и переменным памяти. Поместите курсор в точку вставки, затем щелкните клавишей мыши на поле или переменной в списке полей. Редактор текста вводит полностью уточненное имя в точку вставки!

## Меню поиска

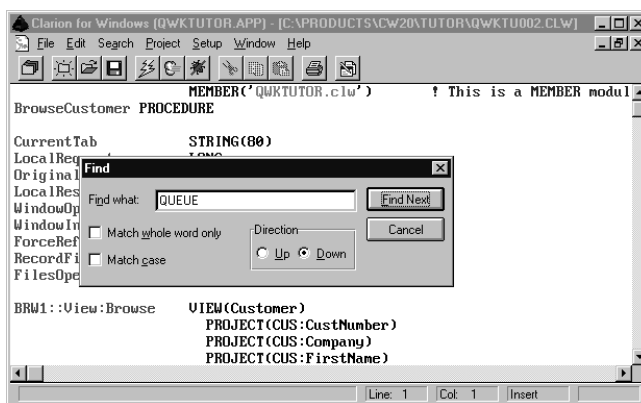
Меню Search (поиск) облегчает обнаружение и изменение фрагментов текста в вашей исходной программе. Вы можете искать конкретный текст, сделать разовые или многократные изменения текста во всем документе или просто выделить переменную, затем сделать прыжок к следующему месту ее появления в коде.

Команды меню поиска Search:

### **Find (Найти)**

Чтобы найти следующее место появления слова, наберите его в диалоговом окне Find (найти) и нажмите кнопку Find Next (найти следующую). Быстрой клавишей является ALT+F3.

Диалог Find является немодальным. Это значит, что диалог будет оставаться на экране, так что вы сможете легко возобновить поиск.



Диалог Find в поиске “QUEUE”.

1. В поле Find What (что найти) наберите текст для поиска.

Содержание по умолчанию для поля Find What (что искать)- последний отыскиваемый текст.

2. По желанию отметьте поле Match whole word only (совпадение только всего слова), поле Match case (совпадение регистра) или и то и другое.

Например, если вы ведете поиск слова ‘find’ с помощью Match whole word only, то слово ‘findings’ обнаружено не будет. Если вы ищете ‘Find’ с помощью Match case, вы не получите ‘find’.

3. Установите, направление поиска: вперед или назад.

Нажмите кнопку Find Next (искать следующего) для начала поиска.

## **Replace (Заменить)**

Чтобы заменить конкретную строку символов, в диалоговом окне Replace наберите оригинальный текст и заменяющий его текст. Вы можете сделать изменения по одному за раз, в выделенном блоке текста или во всем документе.

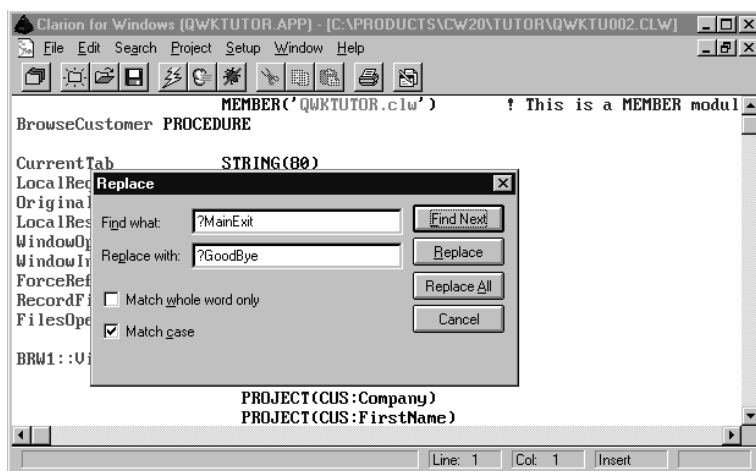
Диалог Replace немодальный. Это значит, что диалог будет оставаться на экране, так что вы сможете легко возобновить поиск и замену.

1. В поле Find What наберите подлежащий поиску текст.

Содержание поля Find What (найти что) по умолчанию - последний отыскиваемый текст.

2. Наберите замещающий текст в поле Replace with (заменить на).

Текст по умолчанию поля Replace with - предыдущий текст замещения.



Диалог Replace - изменение имени с ?MainExit на ?GoodBye.

3. По желанию отметьте поле флажков Match whole word only (совпадение только всего слова), поле Match case (совпадение регистра) или и то и другое.

Например, если вы ведете поиск слова 'find' с помощью Match whole word only, то слово 'findings' обнаружено не будет. Если вы ищете 'Find' с помощью Match case, вы не получите 'find'.

4. Для показа следующего места появления первоначального текста и остановки перед его изменением нажмите кнопку Find Next.



Диалоговое окно попросит вас подтвердить замену.

5. Чтобы изменить встретившийся следующий раз заменяемый текст без подтверждения, нажмите кнопку Replace.

6. Чтобы изменить все вхождения заменяемого текста в выбранном блоке (или во всем документе, если вы выбрали все или ничего), нажмите кнопку Replace All (заменить все).

Диалог Replace All работает только на выбранном блоке текста. Если текст не выбран, он работает на всем документе.

### **Find Next (найти следующую)**

Чтобы искать тот же текст, который вы искали последний раз, выберите эту команду или нажмите клавишу F3. Это поиск в направлении “вперед”.

### **Find Previous (Найти предыдущее)**

Чтобы искать тот же текст, который вы искали последний раз, причем в направлении назад, выберите эту команду или нажмите SHIFT+F3.

### **Find Marked Text (Найти отмеченный текста)**

Чтобы быстро найти следующее место появления отмеченного в данный момент текста, выберите команду или нажмите CTRL+F3. Это эквивалентно исполнению команды Find, ввода текущего выделенного текста в поле Find What и назначения поиска вперед.

## **Файловое меню**

Меню File (файл) текстового редактора имеет следующие специальные команды, ориентированные на файл.

### **Save All (Сохранить все)**

Чтобы сохранить все файлы исходного текста, выберите эту команду.

### **Import File (Импортировать файл)**

Вызывает диалоговое окно открывания файла, что позволяет вам вставить содержание файла в текущий документ исходной программы в точке вставки.

### **Export Block (Экспортировать блок)**

Сохраняет текущий выделенный текст в новом документе исходной программы под выбранным вами новым именем.

## Отступ блока

---

Текстовый редактор поддерживает отступы блока, то есть вы можете сдвигать много строк кода в левую или правую сторону, так что все эти докучливые IF выстаиваются в одну линию с их END.

Чтобы передвинуть блок текста, достаточно выбрать блок, а затем нажать клавишу табуляции tab. Появится диалоговое окно Block Indent (отступ блока). Назначьте величину и направление отступа, а затем нажмите кнопку ОК.

## Макросы

Текстовый редактор поддерживает простые макросы.

Чтобы создать макрос:

1. Нажмите CTRL+=.

То есть, удерживая нажатой клавишу CTRL, нажмите клавишу =. Появится диалоговое окно Press Key for Macro (нажмите клавишу для макроса).

2. Нажмите клавишу, которая будет вызывать макрос, например CTRL+Z.

3. Нажмите ОК.

Активизируется записыватель макроса и записывает вашу комбинацию клавиш.

4. Выполните операции, которые затем сохранятся.

Наберите текст, TAB, ENTER, функциональные клавиши и т.д.

5. Когда закончите, нажмите снова CTRL+=.

Макрос сохранен и может быть вызван нажатием назначенной вами клавиши.

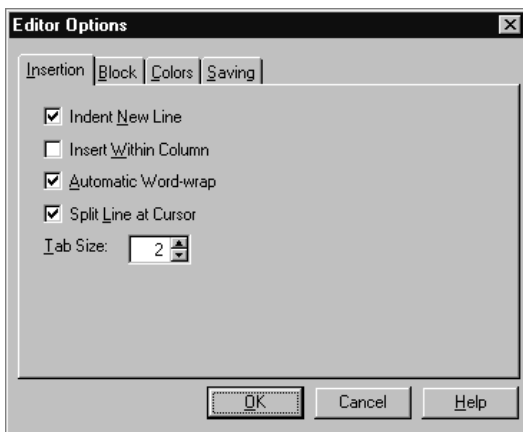
**Совет: Макрос не сохраняется между сеансами и имеется только пока вы не закроете Clarion for Windows.**

## **Настройка текстового редактора**

Чтобы персонализировать вашу среду редактирования, настройте внешний вид редактора и поведение курсора в нем с помощью диалогового окна Editor Options (параметры редактора). Чтобы вызвать это диалоговое окно, выберите Setup III Editor Options. Выберите соответствующую закладку для установки соответствующих опций текстового редактора.

## Опции вставки

---



Indent New Line (отступ  
новой линии)

Чтобы установить автоматическое появление отступа у новых строк, отметьте поле Indent New Line (отступ новой строки). Это сделает вашу программу более читаемой.

Insert Within Column  
(вставка в столбце)

Когда точка вставки находится в середине строки, ENTER добавляет новую строку после текущей строки.

Automatic Word-wrap  
(автоматический переход  
на новую строку)

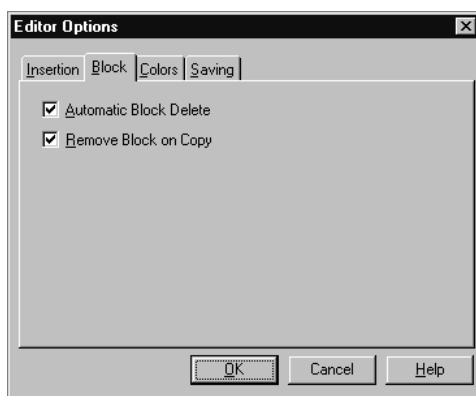
Чтобы вызвать автоматический обрыв строки на столбце 70, отметьте это поле флажков.

Split Line at Cursor  
(разбиение на курсоре)

Когда отмечено это поле флажков, ENTER будет разбивать текущую строку в точке вставки (курсора). Вторая часть строки появится на новой строке. Когда это поле не отмечено, ENTER вставит пустую строку ниже текущей, не разбивая текущую строку.

Tab Size (размер табуляции) Чтобы установить расстояние между табуляционными метками по умолчанию, введите число в поле Tab Size.

## Опции блока



Automatic Block Delete  
(автоматическое удаление  
блока)

Чтобы удалить выделенный текст при вставке, отметьте поле Automatic Block Delete (автоматическое удаление блока).  
Чтобы вставить перед выбранным блоком, очистите это поле.

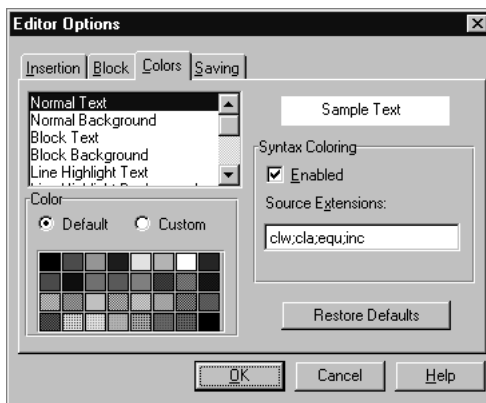
Remove Block On Copy  
(удалить блок при  
копировании)

Чтобы удалить выбранный текст при копировании, отметьте поле Remove Block On Copy (удалить блок при копировании).

## Опции цвета

Эти опции позволяют установить цвет для двадцати одного различного элемента языка Clarion. Заставьте появляться ключевые слова Clarion в красном цвете, или стандартные мнемонические метки - в зеленом.

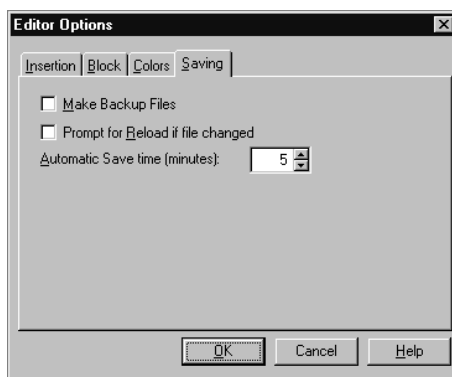
Выберите языковый или текстовый элемент в списочном поле Color Groups, затем щелкните клавишей мыши на поле выбора цвета. Образец текста покажет вам, как выбранный языковый элемент будет выглядеть в текстовом редакторе.



Color Groups (цветовые группы)	Выделение элемента языка или текста для назначения
Color (цвет)	Для присваивания цвета выбранному элементу языка, щелкните клавишей мыши на поле выбора цвета.
Default (по умолчанию)	Чтобы присвоить цвет, принятый по умолчанию для выбранного элемента языка, отметьте это поле.
Custom (заказной)	Чтобы восстановить заказной цвет для выбранного элемента языка, отметьте это поле.
Sample Text (образец текста)	Показывает, как выделенный языковой элемент будет выглядеть в текстовом редакторе.
Enabled (разрешенный)	Чтобы придать выделение в цветовом синтаксисе типам файлов, перечисленных в поле Source Extensions (расширения исходного текста), отметьте это поле.
Source Extensions (расширения исходного текста)	Чтобы установить типы файлов, к которым применено выделение в цветовом синтаксисе, наберите список расширений файлов, разделенных точками с запятой.
Restore Defaults (восстановить умолчания)	Чтобы присвоить цвета по умолчанию всем элементам языка и текста, отметьте это поле.

## Опции сохранения

---



Make Backup Files (изготовление резервных файлов)	Чтобы текстовый редактор изготовлял резервный файл (.BAK) каждый раз, когда вы явно сохраняете файл исходного
--	---

текста, отметьте это поле флажков. Файл .ВАК содержит исходный текст в том виде, в каком он был сохранен предыдущий раз.

### Prompt for Reload if file changed

(приглашение к перезагрузке  
если файл изменен)

Чтобы получить сообщение “исходный .CLW был изменен на диске. Хотите перезагрузить?” каждый раз как текстовый редактор находит такое изменение, отметьте это поле.

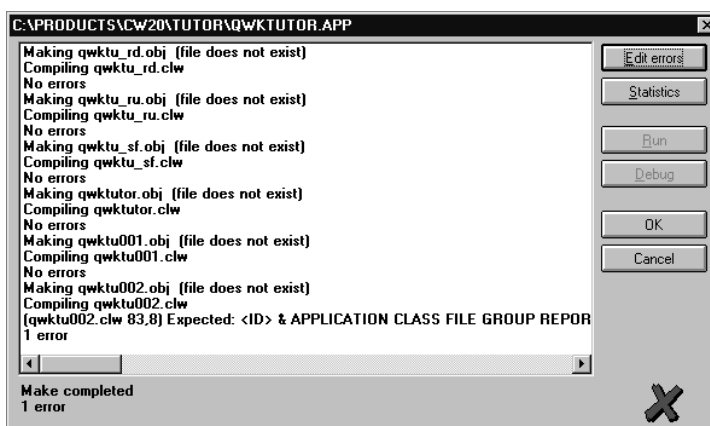
### Automatic Save TIME (minutes)

(автоматический интервал между  
сохранениями (в минутах))

Чтобы установить временной интервал между автоматическими сохранениями, наберите число в этом поле.

## Редактирование ошибок

Одна из главных точек входа в текстовый редактор - через диалоговое окно результатов компилирования - когда ошибка в программе прерывает работу компилятора.



Кнопка Ошибки редактирования в диалоговом окне результатов компилирования открывает текстовый редактор.

Кнопка Edit Errors (редактирование ошибок автоматически вызывает текстовый редактор и помещает курсор в позицию, где компилятор нашел ошибку. Вы можете затем отредактировать исходную программу для исправления ошибки. Команда Edit III Goto Next Error служит для перехода к следующей ошибке компиляции после исправления первой ошибки.

**Совет:** При входе в текстовый редактор через диалоговое окно результатов компилирования любые изменения, сделанные сгенерированному коду, будут переписаны следующими процедурами генерации или изготовления проекта.

## Глава 14 Редактор формул



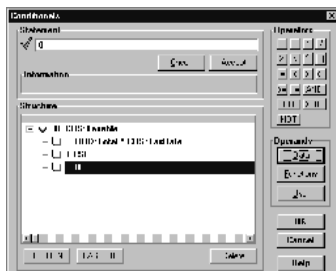
Редактор формул помогает вам быстро генерировать оператор, присваивающий значение выражения переменной. Вы можете использовать Редактор формул для создания вычисляемых или условных полей.



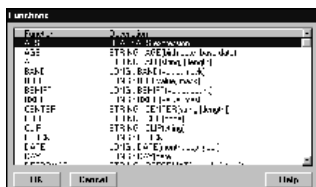
Диалоговое окно **Формулы** организует формулы и управляет ими для вашей процедуры.



Редактор формул создает правильные выражения нажатием кнопок.



Диалог **Условия** создает структуры управления для условных формул.



Диалог функций обеспечивает быстрый доступ к функциям Skarion.

- Компоненты выражения
- Инструменты редактора формул
- Диалог Formulas (Формулы)
- Редактор формул
- Диалог Условия (Conditionals)
- Создание выражения присваивания
- Условные выражения
- Создание структуры IF
- Создание структуры CASE
- Вложенные структуры



Редактор формул помогает вам быстро генерировать оператор, присваивающий переменной значение выражения. Вы можете использовать Редактор формул для создания вычисляемых полей или условных полей.

◆ Computed field (вычисляемое поле) получает вычисленное значение выражения. Иными словами, вычисляемое поле является получателем простого оператора присваивания: переменная=выражение. Например, вычисляемое поле, называемое GrossPrice (оптовая цена), может получить результат сложения двух полей, называемых BasePrice (базисная цена) и Tax (налог).

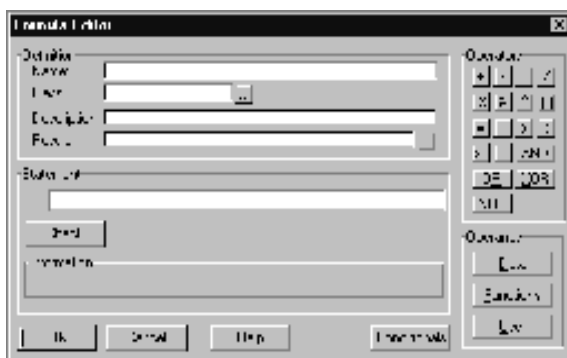
Вы можете использовать вычисляемое поле везде в тех местах, где программа должна выполнять вычисления.

◆ Conditional field (условное поле) является вычисляемым полем с несколькими возможными вариантами присваивания. Имеется два типа условных полей - структуры IF и CASE. Присваивание, которое выполнил оператор, зависит от оценки условий IF или CASE. Например, условное поле, называемое "Tax", может быть 0, если значение логической переменной "Taxable" (обложение налогом) (условие IF) False, или "Tax" может быть равным Цене, помноженной на Налоговую ставку, если "Taxable" равно True.

Вы можете использовать условное поле везде в тех местах, где программа должна выполнять различные вычисления, основанные на условии.

Диалоговое окно Formula Editor (редактор формул) обеспечивает доступ к полям словаря данных, а также к глобальным и локальным переменным памяти и облегчает создание синтаксически правильных выражений. Одним из главных достоинств редактора формул является автоматическая проверка синтаксиса.

Чтобы создать выражение, вы нажимаете кнопку с целью добавления компонентов выражения к строке Statement (оператор). Вы можете также набрать ваше выражение и проверить синтаксис по завершении набора.



## ***Компоненты выражения***

Выражение составлено из двух типов компонент: операндов и операторов. Операторы выполняют операцию (такую как сложение, вычитание и т.д.) над одним или более компонентами выражения. Операндами являются компоненты, над которыми выполняются операции. Операнды либо хранят, либо возвращают величину. Константы, поля словаря данных, переменные памяти и функции являются примерами операндов. Операнд может быть составлен из более чем одного компонента, таких, как, например, функция и ее параметры.

Редактор формул позволяет вам выбирать операторы и операнды, а затем вставлять их в строку Statement.

В приводимой ниже таблице приводятся все компоненты, используемые в Clarion выражениях.

### **Математические операторы**

- +      Знак плюс: складывает два операнда
- Знак минус: Вычитает один операнд из другого
- \*      Знак звездочка: Умножает один операнд на другой
- /      Знак наклонная черта вправо: Делит один операнд на другой
- %      Знак процента: Возвращает остаток от операции деления (модульное деление).
- &      Конкатенация: добавляет одну строку символов к другой.
- ^      Возведение в степень. Возводит один операнд в степень, равную другому операнду.
- ( )    Символы группирования. скобки группируют компоненты в выражении вместе.

### **Логические операторы:**

- =      Равняется: Оценивает, равно ли одно выражение другому.
- <      Меньше чем: Оценивает, не меньше ли одно выражение, чем другое.
- >      Больше чем: Оценивает, не больше ли одно выражение, чем другое.
- <>    Не равняются: Оценивает, что одно выражение не равно другому.

- >=** Больше или равно: Оценивает, не больше ли или равно одно выражение другому.
- <=** Меньше или равно: Оценивает, не меньше ли или равно одно выражение другому.
- AND** Соединяет два логических выражения вместе. Выражение, содержащее AND, вырабатывает значение True если оба выражения, соединяемые AND, равны True.
- OR** Соединяет два логических выражения вместе. Выражение, содержащее OR, вырабатывает значение True если хотя бы одно из выражений, соединяемых OR, равно True.
- XOR** Соединяет два логических выражения вместе. Выражение, содержащее XOR, вырабатывает значение True если только одно из выражений, соединяемые XOR равно True.
- NOT** Обращает оценку выражения.

### **Операнды:**

- |                                       |   |
|---------------------------------------|---|
| Data (данные)                         | Включает поля словаря данных, глобальные и локальные переменные памяти.   |
| Functions (функции)                   | Любая из встроенных функций языка программирования Clariion. Все эти функции выполняют некоторую операцию над параметрами (другими операндами) и возвращают величину в выражение.   |
| User (пользовательские)               | Любая FUNCTION в вашем приложении. Эти функции выполняют некоторую операцию над параметрами (другими операндами) и возвращают величину в выражение.   |
| Constant Text<br>(постоянный текст)   | Вы можете набрать в строке Statement ( ) постоянный текст, окруженный одинарными кавычками (' A').  |
| Constant Number<br>(постоянное число) | Вы можете набрать на строке Statement ( ) постоянные числа. Постоянные числа могут быть представлены в любом законном формате, таком как десятичный (1 или 1.2345), научный (22e4), бинарный (0101b) или шеснадцатеричный(1AFFh). |

## Инструменты редактора формул

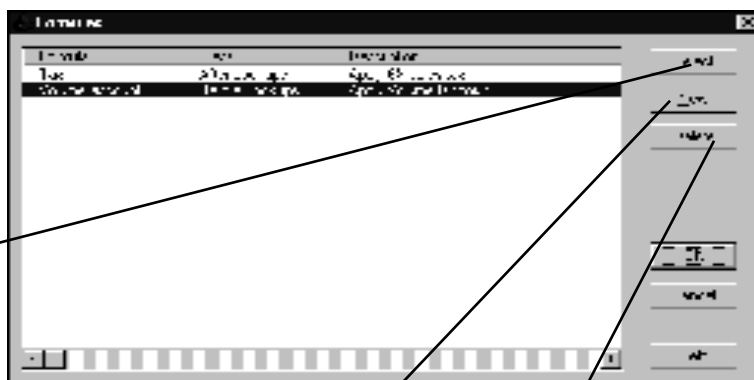
Редактор формул состоит из трех диалоговых полей:

Formulas (формулы)	Управляет всеми формулами, которые вы создали для процедуры.
Formula Editor (редактор формул)	Создает простые операторы присваивания.
Conditionals (условные)	Создает условные структуры (IF..THEN или CASE..OF).

## Диалог Formulas (Формулы)

Слева:  
Список формул для процедуры с классом исполнения и описанием

Справа:  
Отбирает выделенную формулу для редактирования



Дает возможность создать новую формулу

Удаляет выделенную формулу

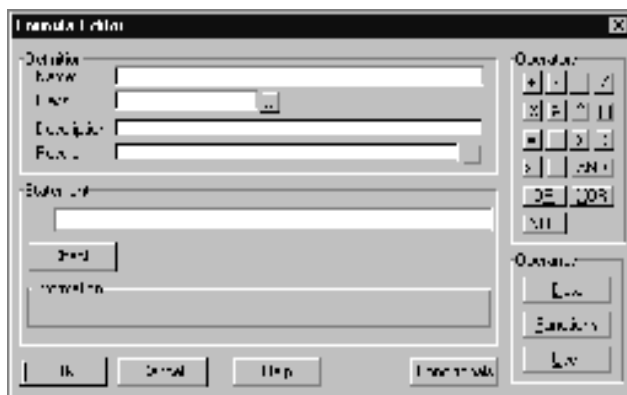
## Редактор формул

Name - Описательная метка

Class - Определяет в каком месте программы выражение вычисляется

Result - Переменная которой присваивается величина

Check - Проверяет синтаксис вашего выражения





Если же это первая формула в этой процедуре, диалоговое окно Formulas (формулы) не появится. Появится диалоговое окно Formula Editor (редактор формул), поэтому пропустите шаг 2.

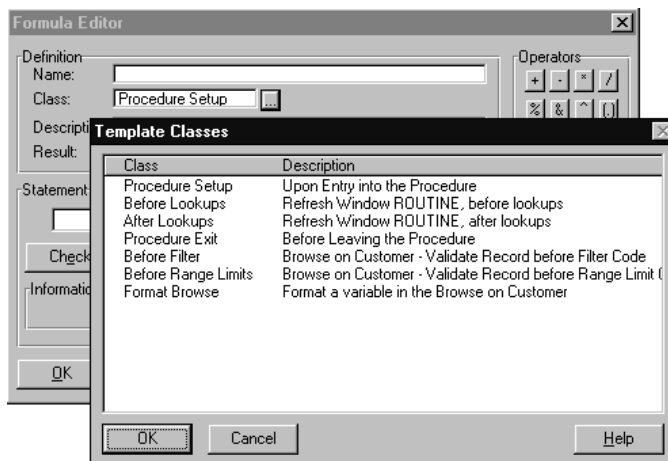
2. Нажмите кнопку New (новый).

Появится диалоговое окно Formula Editor (редактор формул).

3. Наберите имя формулы в поле Name (имя).

4. Нажмите эллиптическую (...) кнопку рядом с полем Class (класс) для выбора класса формулы.

Выбор класса формулы.



Класс формулы определяет, где в генерированной исходной программе будет выполняться вычисление формулы. Каждый шаблон процедуры Clarion имеет свой собственный набор классов. Например, в шаблоне формы есть класс, называемый “После Поиска”, который сообщает генератору приложений о необходимости вычислить формулу после того, как все поиски во вторичных файлах для процедуры будут завершены.

**Совет: Не путайте классы формул с классами шаблонов. Класс шаблона - это просто группа шаблонов, производства Clarion или иного разработчика, в регистре шаблонов. Класс формулы - это точка внутри шаблона процедуры, где формула вычисляется.**

5. Введите описание формулы в поле Description (описание).

6. В поле Result наберите переменную, которой присваивается результат вычисления выражения или нажмите эллиптическую (...) кнопку, чтобы выбрать переменную из диалогового окна Select Field (выбор поля).

Вы можете выбрать локальную, модульную или глобальную переменную, или поле файла данных. Это имя появится в списке диалогового окна Формулы.

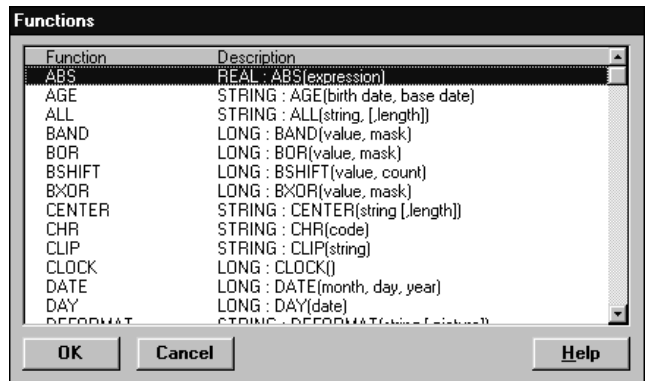
### 7. Создайте свою формулу в поле Statement(оператор).

Вы можете набрать выражение, используя для этого кнопки Редактора формулы, или сделать и то и другое. Первой компонентой выражения должен быть операнд, левая скобка или унарный минус (знак минуса). Например, нажмите кнопку Data (данные) и выберите переменную, нажмите кнопку умножения (\*), затем нажмите кнопку Functions (функции) чтобы выбрать встроенную функцию Clarion.

Отсюда вы можете выбрать встроенные функции Clarion.

8. Нажмите кнопку Check (проверить) для проверки синтаксиса вашего выражения.

Если синтаксис правилен, слева от поля Statement(оператор) появляется большая зеленая метка, сигнализирующая об отсутствии ошибок. Если обнаружены синтаксические ошибки, появляется большое красное X.



9. Нажмите кнопку OK.

## Условные выражения

Создание условных выражений с помощью редактора формулы в действительности создает в тексте исходной программы структуры управления. Имеются две структуры, которые вы можете создать с помощью редактора формулы - структура IF и/или структура CASE. Вы можете также вложить любую из этих структур, создавая тем самым сложные условные выражения.

Структура IF присваивает значение результирующей переменной на основе оценки истина/ложь одиночного логического выражения. Имеется только два возможных присваивания, так как только одно условие испытано. Если испытанное условие истинно, одно присвоение сделано, если - не истинно (ложно), тогда делается другое присваивание.

Вкладывание структур IF делает возможными дополнительные альтернативные присваивания. Однако, структура CASE предлагает менее сложный метод присваивания величин, основанный на оценке множественных логических выражений.

Структура CASE селективно присваивает значение результирующей переменной, основанной на оценки множественных выражений OF относительно выражения CASE.

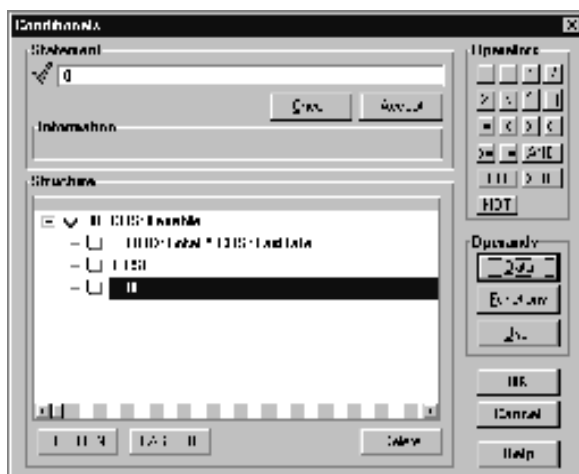
Практически говоря, имеются неограниченные альтернативные присваивания, так как любое число выражений может быть оценено. Смотрите дополнительную информацию в Справочнике языка.

## Создание структуры IF

Используйте структуру IF, чтобы присвоить одно из двух значений полю Результата в зависимости от условия. Например, вы можете захотеть определить налог на заказ. Величина Налога (Tax) зависит от условия - облагается клиент налогами или нет. Результирующая управляющая структура будет:

```
IF CUS:Taxable          ! условное выражение
  TAX= ORD:Total * CUS:TaxRate    ! выражение истинного присвоения
ELSE
  TAX= 0                    ! выражение ложного присваивания
END
```

Эта структура в диалоговом окне “Условия” будет выглядеть как на иллюстрации 3 в начале главы.



□ Чтобы создать условную формулу IF (из диалогового окна Procedure Properties (свойства процедуры)):

1. Нажмите кнопку Formulas (формулы).

Если у вас уже есть формулы в процедуре, появится диалоговое окно Formulas (формулы), содержащее формулы, спроектированные ранее.

Если же это первая формула в этой процедуре, диалоговое окно Formulas (формулы) не появится. Появится диалоговое окно Formula Editor (редактор формул) и поэтому



пропустите шаг 2.

2. Нажмите кнопку New (новый).

Появится диалоговое окно Formula Editor (редактор формул).

3. Наберите имя формулы в поле Name (имя).

4. Нажмите эллиптическую (...) кнопку рядом с полем Class (класс) для выбора класса шаблона.

Класс формулы определяет, где в генерированной исходной программе будет выполняться вычисление выражения. Каждый шаблон процедуры Clarion имеет свой собственный набор классов. Например, в шаблоне формы есть класс, называемый “После Поиска”, который сообщает генератору приложений о необходимости вычислить формулу после того, как все поиски во вторичных файлах для процедуры будут завершены.

**Совет: Не путайте классы формул с классами шаблонов. Класс шаблона - это просто группа шаблонов, производства Clarion или иного разработчика, в регистре шаблонов. Класс формулы - это точка внутри шаблона процедуры, где формула вычисляется.**

5. Введите описание формулы в поле Description (описание).

6. В поле Result наберите переменную, которой присваивается результат вычисления выражения или нажмите эллиптическую (...) кнопку, чтобы выбрать переменную из диалогового окна Select Field (выбор поля).

Вы можете выбрать локальную, модульную или глобальную переменную, или поле файла данных. Это имя появится в списке диалогового окна Формулы.

7. Нажмите кнопку Conditionals (условия).

8. Нажмите кнопку IF.THEN (если..то) .

Структура оператора IF появится в окне Структура.

9. На строке Statement (оператор) введите для вычисления условие IF.

Вы можете набрать выражение или можете использовать кнопки Operators (операторы) или Operands (операнды) для выбора компонент выражения, или же вы можете сделать и то и другое.

10. Нажмите кнопку Check (проверить) для проверки синтаксиса вашего выражения.

11. Нажмите кнопку **Ассерпт (принять)** , чтобы вставить ваше выражение в структуру.

12. Выделите строку под строкой **IF** в окне Структура.

Это то место, куда вводится выражение присваивания , если значение условия “Истина”.

13. На строке **Statement (оператор)** введите выражение присваивания, когда условие “Истинно”.

Опять, вы можете набрать выражение или можете использовать кнопки **Операторы** и **Операнды** для выбора компонент выражения, или же вы можете сделать и то и другое. Если условие **IF** истинно, это выражение вычисляется и результирующая величина присваивается переменной **Результат**.

Наличие выражения не обязательно. Если не введено никого выражения, значит никого присваивания не выполняется..

14. Нажмите кнопку **Check (проверить)** для проверки синтаксиса вашего выражения.

15. Нажмите кнопку **Ассерпт (принять)** , чтобы вставить ваше выражение в структуру.

16. Выделите строку под строкой **ELSE** в окне Структура.

Это то место, куда вводится выражение присваивания , если значение условия “ложь”.

17. На строке **Statement (оператор)** введите выражение присваивания, когда значение условия “Ложь”.

Опять, вы можете набрать выражение или можете использовать кнопки **Операторы** и **Операнды** для выбора компонент выражения, или же вы можете сделать и то и другое. Если условие **IF** ложно, это выражение вычисляется и результирующая величина присваивается переменной **Результат**.

Наличие выражения не обязательно. Если не введено никого выражения, значит никого присваивания не выполняется..

18. Нажмите кнопку **Check** для проверки синтаксиса вашего выражения.

19. Нажмите кнопку **Ассерпт** чтобы вставить ваше выражение в структуру.

20. Когда ваша управляющая структура будет завершена, нажмите кнопки **ОК** в диалоговых окнах **Conditionals (условия)**, **Formula Editor (редактор формул)** и **Formulas (формулы)**.

## Создание структуры CASE

Простая структура CASE может быть использована для присваивания одного или нескольких значений полю результата в зависимости от того, какое условие OF равно выражению CASE. Например, вы можете захотеть предложить различные скидки за большие покупки в зависимости от кода скидки покупателя. Результирующая структура CASE может быть такова:

CASE CUS:DiscountCode !Case выражение, сравниваемое с выражением OF

OF 'A' ! 1-ое выражение сравнения OF

Discount = 0 ! 1-ое выражение присваивания OF

OF 'B' ! 2-ое выражение сравнения OF

Discount = ORD:Total \* .1 ! 2-ое выражение присваивания OF

OF 'C' ! 3-ее выражение сравнения OF

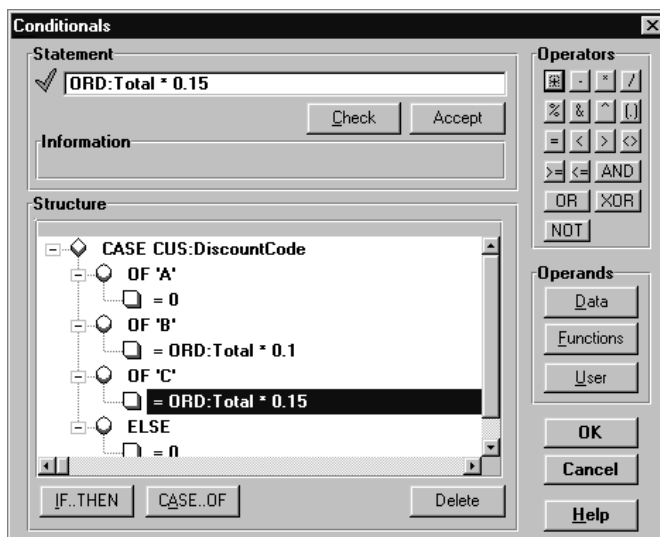
Discount = ORD:Total \* .15 ! 3-е выражение присваивания OF

ELSE

Discount = 0 ! собрать все присваивание

END

Эта управляющая структура появляется в редакторе формул как показано на следующей иллюстрации.



□ Чтобы создать условную формулу CASE (из диалогового окна Procedure Properties (свойства процедуры):

1. Нажмите кнопку Formulas (формулы).

Если у вас уже есть формулы в процедуре, появится диалоговое окно Formulas (формулы), содержащее формулы, спроектированные ранее.

Если же это первая формула в этой процедуре, диалоговое окно Formulas (формулы) не появится. Появится диалоговое окно Formula Editor (редактор формул) и поэтому пропустите шаг 2.

2. Нажмите кнопку New (новый).

Появится диалоговое окно Formula Editor (редактор формул).

3. Наберите имя формулы в поле Name (имя).

4. Нажмите эллиптическую (...) кнопку рядом с полем Class (класс) для выбора класса шаблона.

Класс формулы определяет, где в генерированной исходной программе будет выполняться вычисление выражения. Каждый шаблон процедуры Clarion имеет свой собственный набор классов. Например, в шаблоне формы есть класс, называемый “После Поиска”, который сообщает генератору приложений о необходимости вычислить формулу после того, как все поиски во вторичных файлах для процедуры будут завершены.

**Совет: Не путайте классы формул с классами шаблонов. Класс шаблона - это просто группа шаблонов, производства Clarion или иного разработчика, в регистре шаблонов. Класс формулы - это точка внутри шаблона процедуры, где формула вычисляется.**

5. Введите описание формулы в поле Description (описание).

6. В поле Result наберите переменную, которой присваивается результат вычисления выражения или нажмите эллиптическую (...) кнопку, чтобы выбрать переменную из диалогового окна Select Field (выбор поля).

Вы можете выбрать локальную, модульную или глобальную переменную, или поле файла данных. Это имя появится в списке диалогового окна Формулы.

7. Нажмите кнопку Conditionals (условия).

8. Нажмите кнопку CASE..OF .

Структура оператора CASE появится в окне Структура.

9. На строке Statement (оператор) введите выражение CASE, которое сравнивается с множественными выражениями OF.

Вы можете набрать выражение или можете использовать кнопки Operators (операторы) или Operands (операнды) для выбора компонент выражения, или же вы можете сделать и то и другое.

10. Нажмите кнопку Check (проверить) для проверки синтаксиса вашего выражения.

11. Нажмите кнопку Ассерпт (принять) , чтобы вставить ваше выражение в структуру.

12. Выделите строку OF под строкой CASE в окне Структура.

Это то место, куда вводится первое выражение OF которое сравнивается со значением выражения CASE.

13. На строке Statement (оператор) введите выражение сравнения OF.

Опять, вы можете набрать выражение или можете использовать кнопки Операторы и Операнды для выбора компонент выражения, или же вы можете сделать и то и другое. Во время прогона если значение выражения CASE совпадает со значением выражения OF, тогда последующее выражение вычисляется и результирующая величина присваивается переменной Результат.

14. Нажмите кнопку Check (проверить) для проверки синтаксиса вашего выражения.

15. Нажмите кнопку Ассерпт (принять) , чтобы вставить ваше выражение в структуру.

16. Выделите строку под строкой OF в окне Структура.

Это то место, куда вводится присваиваемое выражение OF.

17. На строке Оператор введите присваиваемое выражение OF.

Опять, вы можете набрать выражение или можете использовать кнопки Операторы и Операнды для выбора компонент выражения, или же вы можете сделать и то и другое. Во время прогона если значение выражения CASE совпадает с выражением из строки OF, тогда присваиваемое выражение вычисляется и результирующая величина присваивается переменной Результат.

18. Нажмите кнопку Check для проверки синтаксиса вашего выражения.

19. Нажмите кнопку Ассерпт чтобы вставить ваше выражение в структуру.

□ Чтобы добавить дополнительные выражения OF:

1. Выделите строку OF в окне Structure (структура).
2. Нажмите кнопку Case..OF.
3. Вставьте ваши выражения таким же образом, как описано выше.
4. Когда ваша структура CASE будет завершена, нажмите кнопки ОК в диалоговых окнах Conditionals (условные), Formula Editor (редактор формул) и Formulas (формулы).

## Вложенные структуры

Любая из приведенных выше структур может быть вложена одна внутри другой. Это дает вам возможность легко создавать очень сложные структуры.

Допустим вы хотели определить налоговую ставку, основываясь на следующей логике:

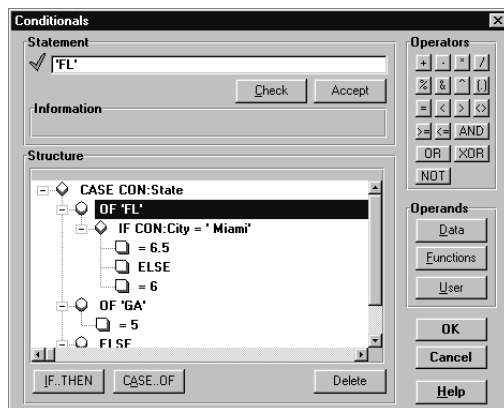
Если штат - Флорида, налоговая ставка равна 6%, кроме города Майами, который требует 6.5%.

Если штат - Джорджия, налоговая ставка 5%.

Вы создадите структуру CASE с вложенной структурой IF, как показано ниже.

□ Чтобы добавить вложенную структуру управления:

1. Выделите строку присваивания в окне Structure.



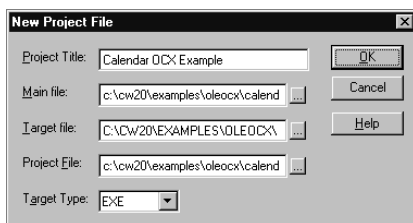
2. Нажмите кнопку CASE..OF (случай ..) или IF..THEN (если..то).

3. Вставьте выражения на соответствующие строки, следуя инструкциям, приведенным в предыдущих разделах.

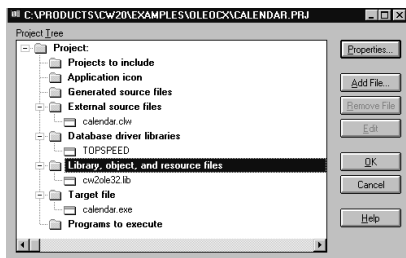
## Глава 15 Система управления проектом



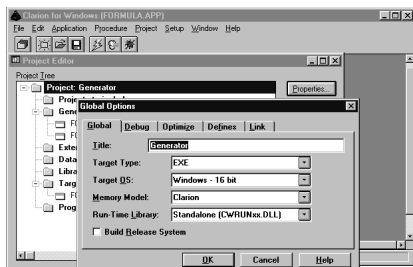
Данная глава покажет вам, как использовать Систему управления проектом. Система управления проектом управляет параметрами компиляции и редактирования связей. В данной главе также содержится информация о распределении вашего приложения.



Начните с того, что дайте вашему проекту имя. Выбирая ваш основной файл исходной программы из каталога, Clarion автоматически заполняет большинство позиций диалогового поля.



Добавление, удаление и редактирование свойств исходной программы, внешних библиотек и объектных файлов.



Установка глобальных параметров компилятора и оптимизации.

Меню проекта

Редактирование файла переадресации

Создание файла проекта для приложения, закодированного вручную

Сопровождение проекта

Пиктограмма приложения

Добавление файлов исходной программы

Добавление ресурсов, объектных файлов и библиотек

Добавление внешних ресурсов

Добавление других проектов

Добавление исполняемых программ

Распределение файлов

Выбор конфигурации

Инсталлирование и доступ к DLLs вашего приложения

Установка целевого файла

Файлы .LIB

Файлы .DLL

Установка параметров файла проекта

Глобальные опции компиляции и компоновки

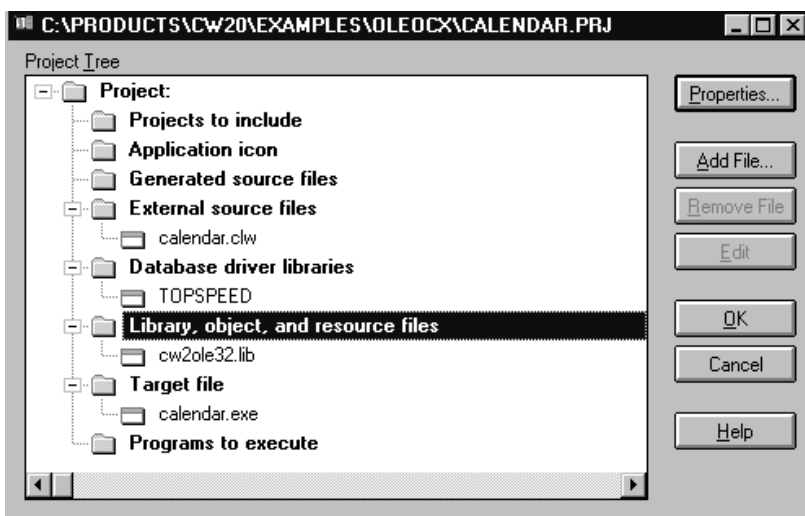
Опции компилирования отдельных исходных модулей



Файл проекта (.PRJ) прослеживает все компоненты, которые использованы для создания окончательной исполняемой программы (объектный файл) для вашего приложения. Он также сохраняет опции компилятора от, например, включать или нет код отладки, до установки предпочтительного метода оптимизации. Работа компилятора и компоновщика зависят от того, что сообщает им файл проекта о том что и как компилировать и компоновать. Файл проекта функционально эквивалентен файлу MAKE (делай) компиляторов других языков.

Система Управления Проектом Clarion визуально управляет файлом проекта. Она ведет диаграммы дерева файлов исходного текста, внешних библиотек, ресурсов и других компонент проекта.

Диалог Project Editor (редактор проекта) содержит древовидный список проекта. Ветви



дерева сворачиваются и разворачиваются, когда вы щелкаете на них мышью. Развернутые, они показывают список файловых компонентов. Когда свернуты - виден лишь прямоугольник, содержащий крестик, который показывает, что вы можете развернуть эту ветвь.

В этой главе рассматривается, как управлять системой проекта и связанные с этим вопросы. В данной главе:

- Описаны различные команды меню, которые устанавливают параметры режима работы Системы управления проектом.
- Показано, как добавить ваши файлы с текстами исходной программы к дереву проекта.
- Показано, как добавить к дереву внешние библиотеки и как получить доступ к их функциям и процедурам в вашей исходной программе.
- Показано, как установить тип и параметры объектного файла, а также как настроить компилятор. Объектный файл - это конечная исполняемая программа, созданная для вашего приложения.

## Меню проекта

Меню интегральной среды разработки Clarion обеспечивает несколько команд меню, которые имеют доступ к системе проекта или влияют на нее. В этом разделе приводится список команд с указанием того, что они делают. Чтобы получить доступ к этим командам, выберите Project (проект) из линейки действий.

Меню проекта



- |                      |  |
|----------------------|--|
| Set (установить)     | Устанавливает текущий проект, так что последующие команды проекта, такие как Run и Make, действуют на выбранный проект. Появляется стандартный диалог Select Project (выбор проекта); выберите из поля списка файл .PRJ или .APP.  |
| New (новый)          | Создает новый проект. В диалоговом окне New Project File (файл нового проекта) заполните поле заголовка Project Title (заглавие проекта) и поле Main File (главный файл)   |
| Load (загружать)     | Делает проект исполняемым, так что последующие команды проекта, такие как Run и Make, действуют на выбранный проект. Если нет текущего проекта, появляется диалоговое окно Select Project (выбрать проект); выберите из поля списка файл .PRJ или .APP. Показывает либо диалоговое окно Application Tree (дерево приложения) или диалоговое окно Project Editor (редактор проекта), в зависимости от того, какой файл был выбран, .PRJ или .APP. |
| Edit (редактировать) | Дает вам возможность в диалоговом окне Project Editor (редактор проекта) редактировать файл текущего проекта.  |
| Make (создать)       | Дает вам возможность компилировать и компоновать текущий   |

проект. Вы можете также нажать кнопку Make (создать) на панели инструментов .

Run (выполнить)	Дает вам возможность компилировать, компоновать и затем выполнить текущий проект.
Debug (отладка)	Дает вам возможность компилировать и загрузить текущий проект в отладчик.
Make Statistics (выполнить статистику)	Дает вам возможность просмотреть последние данные по статистике программы. Данные о размере каждого модуля, включая код и размер данных, появятся в окне Make Statistics.
Auto make before run (создать автоматически перед выполнением)	Переключает установки системы проекта, которые заставляют делать перекompilирование каждый раз, когда вы выбираете команду Run.
File Save before run (сохранить файл перед выполнением)	Переключает установки системы проекта, которые сохраняют файл исходного текста каждый раз как вы выбираете команду Run.
Minimize on Run (минимизировать при работе)	Переключает установки системы проекта, которые минимизируют интегральную среду разработки перед показом приложения каждый раз как вы выбираете команду Run.
Wait for termination on run (ожидать окончания при работе)	Переключает установку системы проекта, которая задерживает интегральную среду разработки до тех пор, пока вы не закончите приложение по исполнению с помощью команды Run.
Generate (генерировать)	Генерирует исходную программу для любой процедуры в проекте, которая изменилась.
Generate All (генерировать все)	Генерирует исходную программу для всех процедур в проекте.

Properties (свойства)

Редактирует файл текущего проекта с помощью диалогового окна Project Editor (редактор проекта).

## **Редактирование файла переадресации**

Интегральная Среда разработки Clarion устанавливает в качестве рабочего каталога тот, в котором находится текущий файл .APP или .PRJ. Кроме того, Clarion for Windows использует файл переадресации (CW20.RED) для хранения списка каталогов для компонентов интегральной Среды разработки. Этот файл сообщает среде разработки, где искать файлы и где создавать новые файлы. Каждая строка файла переадресации записана в формате:

```
filepattern = directory1 [;directory2]...[;directoryn]
```

Filepattern - это или имя файла или маска файла, использующая стандартные универсальные DOS-символы: \* и ?.

Например:

```
*.dbd = c:\cw20\obj  
*.dll = .;c:\cw20\bin  
*.lib = c:\cw20\obj;c:\cw20\lib  
*.res = c:\cw20\obj;c:\cw20\lib  
*.obj = c:\cw20\obj;c:\cw20\lib  
*.rsc = c:\cw20\obj  
*.ico = .;c:\cw20\template  
*.wmf = .;c:\cw20\template  
*.cur = .;c:\cw20\template  
*.bmp = .;c:\cw20\template  
*.tpl = c:\cw20\template  
*.tpw = c:\cw20\template  
*.trf = c:\cw20\template  
  
*.* = .; c:\cw20\examples; c:\cw20\libsrc
```

```
QCKSTART.TXA = c:\cw20\TEMPLATE
```

```
QCKSTART.TXD = c:\cw20\TEMPLATE
```

Первый каталог - это каталог, в котором создается любой новый файл того типа, который установлен с помощью маски файла filepattern. Это справедливо только для файлов, созданных и сохраненных интегральной средой разработки, таких как .OBJ, .DBD, .LIB, .EXE, .CLW. Последующие имена каталогов являются путями, по которым Clarion будет искать существующие файлы.

**Внимание: Резервные файлы всегда создаются в том каталоге, где размещен оригинальный файл.**

Чтобы отредактировать файл переадресации, выберите Setup Edit Redirection File. Текстовый редактор открывает файл CW20.RED для редактирования. Маски файлов появляются слева; пути каталога - справа.

□ Чтобы изменить путь по умолчанию для типа файла, отберите текущий путь дважды щелкнув клавишей, затем наберите поверх него новый путь.

□ Чтобы добавить дополнительный подкаталог к пути поиска для файла, добавьте точку с запятой в конце текущего пути, затем добавьте подкаталог.

## ***Создание файла проекта для приложения, закодированного вручную***

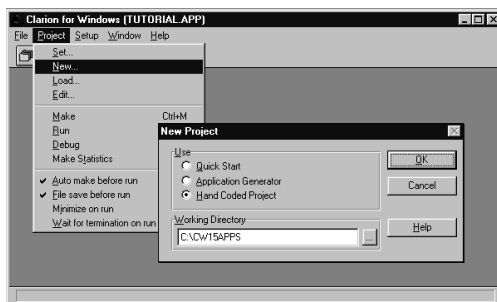
В этом разделе дается обзор шагов, необходимых для создания файла проекта (\*.PRJ). Файл проекта отслеживает все компоненты, которые используются для создания исполняемой программы вашего приложения. он также устанавливает опции компилятора, в пределах от опции, включать код отладки или нет, до установки предпочтительного метода оптимизации.

Если для создания исходного текста вашей программы вы используете генератор приложений, отдельный файл .PRJ не создается и единственное, для чего вы, по-видимому, будете использовать систему проекта, это установка параметров отладки. Генератор приложений заботится почти обо всем остальном. Следовательно, в этой главе описывается, как использовать Систему проекта для приложений, запрограммированных вручную.

Диалоговое окно Project Tree (дерево проекта) организует все компоненты системы проекта и обеспечивает доступ к другим диалоговым окнам, которые управляют вашим файлом проекта

□ Чтобы создать файл проекта. выполните следующие этапы:

1. Выберите Project New.



Появится диалоговое окно New Project (новый проект).

2. В поле USE щелкните клавишей на Hand Coded Project (вручную кодированный проект)

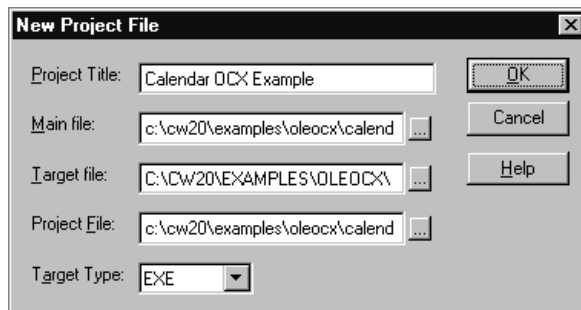
3. В комбинированном поле Working Directory (рабочий каталог) наберите маршрут каталога или нажмите эллиптическую (...) кнопку чтобы выбрать каталог стандартным диалоговым окном открывания файла.

4. Нажмите кнопку ОК.

Появится диалоговое окно New Project File (новый файл проекта).

5. Наберите описательное имя вашего проекта в поле Project Title (заглавие проекта).

После того как вы нажмете эллиптическую кнопку и выберете Main File (главный файл), последние три поля заполнятся автоматически.



6. В поле Main File (главный файл) наберите имя вашего главного файла исходного текста или нажмите эллиптическую (...) кнопку для выбора исходного файла, используя стандартное диалоговое окно Open File (открыть файл).

Этот этап автоматически заполняет имена Target file (целевой файл) и Project file (файл проекта). Если вы хотите установить другое имя для каждого из них, наберите новые имена или нажмите соответственно эллиптические кнопки (...) для выбора файла, используя стандартное диалоговое окно Open File (открыть файл).

7. Нажмите кнопку ОК.

Появляется диалоговое окно Project Tree (дерево проекта).

8. Щелкните клавишей мыши на Database driver libraries (библиотеки драйверов базы данных), затем нажмите кнопку Add File (добавить файл).

Появится диалоговое окно Select Driver (выбор драйвера).

Действие кнопки Add File различное в зависимости от того, на какой папке дерева

проекта выделение. В данном случае она вызывает диалоговое окно Select Driver, поэтому вы можете выбрать из списка правильные Clarion драйверы базы данных. Во всех других случаях кнопка Add File открывает стандартное диалоговое окно открывания файла, чтобы найти файл для включения в процесс компилирования и компоновки вашего проекта. Если вы выберете неправильный тип файла, Система проекта добавляет файл к соответствующей папке дерева проекта.

9. Выберите драйвер базы данных и нажмите кнопку ОК.

10. Щелкните клавишей мыши на Project:(первая) строка и нажмите кнопку Properties (свойства).

Появляется диалоговое окно Global Options (глобальные опции). Этот диалог устанавливает различные опции компилирования и компоновки для всего вашего проекта, включая метод оптимизации, тип созданной исполняемой программы, необходимость использовать отладчик и т.п.

Как и кнопка Add File, кнопка Properties ведет себя по-разному в зависимости от того, какая папка дерева проекта отмечена. Когда выделен исходный файл, кнопка Properties вызывает диалоговое окно Compile Options (параметры компилирования). Этот диалог устанавливает опции компилирования для выделенного конкретного исходного файла. Установленные вами опции компилирования имеют преимущество перед глобальными опциями компилирования.

А пока, нажмите закладки в диалоговом окне Global Options для того, чтобы получить представление о доступных опциях, или нажмите кнопку Help (помощь) для просмотра описания каждой опции. Закончив, нажмите кнопку ОК. Подробности об установке этих опций компилирования и компоновки смотрите ниже в Установка опций файла проекта.

11. Нажмите кнопку ОК, затем щелкните клавишей мыши на Yes, в ответ на вопрос, хотите ли вы сохранить файл проекта.

## **Сопровождение проекта**

В данном разделе дается обзор процедур, необходимых для сопровождения файла проекта. Сопровождение файла проекта включает добавление и удаление исходных файлов, объектных файлов, библиотек и т.д. из процесса компилирования и компоновки. Кроме того, вы можете установить как глобальные, так и локальные опции компилирования и компоновки в файле проекта.

### **Пиктограмма приложения**

---

Вы можете указать файлы пиктограмм которые необходимо включить в вашу исполняемую программу. Благодаря этой возможности вам не нужно поставлять их по-

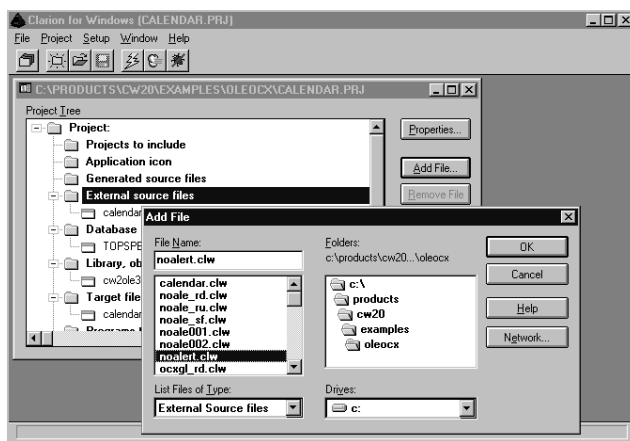
отдельности. Чтобы скомпоновать файл пиктограммы с вашим приложением, выделите Application Icon (пиктограмма приложения). Затем нажмите кнопку Add File (добавить файл) и отберите нужный вам файл для добавления с помощью стандартного диалогового окна Open File (открыть файл).

**Внимание:** Clarion for Windows автоматически привязывает пиктограммы, объявленные в структуре данных Clarion. Сюда входят пиктограммы, назначенные через Window Formatter (форматер окна). Используйте пиктограмму приложения для привязывания пиктограмм, не объявленных в структуре данных.

## Добавление файлов исходной программы

Файлы исходной программы - это файлы, которые содержат ваши операторы языка Clarion (или операторы на других языках если у вас используются компиляторы TopSpeed для их поддержки). Чтобы добавить файл исходной программы к диалогу редактора проекта, нужно только дважды щелкнуть мышью на имени файла.

□ Чтобы добавить “закодированные вручную” файлы исходной программы, выделите External Source Files (внешние файлы исходной программы). Затем нажмите кнопку Add File (добавить файл) и выберите нужный вам файл, чтобы добавить его, используя стандартное диалоговое окно Open File (открыть файл).



Если вы выбрали неправильный тип файла (например, генерированный исходный файл или файл .LIB), система проекта добавляет файл к соответствующей папке дерева проекта.

.APP файлы, Generated source files (генерированные исходные файлы) не могут быть добавлены или удалены из системы проекта. Они могут быть только добавлены или удалены из генератора приложений. Любая попытка добавить исходные модули к Generated source files (генерированные исходные файлы) добавит исходный файл к External source files.



## Добавление ресурсов, объектных файлов и библиотек

Объектный (.OBJ) файл содержит объекты (подпрограммы, функции или процедуры), которые могут быть добавлены к вашей программе во время процесса компоновки. Файл библиотеки (.LIB) - это просто файл, который содержит несколько объектов. Ресурсный файл - это любой другой файл (.RSC, .ICO, .BMP), который должен быть добавлен к вашей программе. Будучи правильно скомпонованной, ваша программа может вызвать эти объекты для выполнения некоторых задач. Вы можете создать файлы .LIB для использования в ваших приложениях Clarion. Смотрите ниже Стратегии разработки и развертывания.

**Внимание:** Для 16-битовых проектов с библиотекой времени исполнения, установленной на Local (локальный), вы должны включить любые файлы .LIBs , файлы .RSC (по одному для каждого .CLW файла) и любые .ICO, .BMP и т.п. файлы, используемые с LIBs.

Объекты, используемые таким образом, называются внешними, так как исходная программа для этих объектов является внешней по отношению к вашему проекту. То есть, вам не нужно иметь исходную программу объекта для того, чтобы использовать объект. Не нужно также, чтобы объекты были написаны в том же самом языке программирования. Ваши Clarion программы могут вызывать объекты, скомпилированные из C, C++, Pascal и т.д.

Ваше приложение может вызвать подпрограммы драйвера базы данных TopSpeed для доступа к вашим файлам TopSpeed. Эти подпрограммы имеются в библиотеках, поставляемых вместе с Clarion for Windows, они помещены в подкаталог CW20\ LIB с помощью программы Setup Clarion. Во время процесса компоновки проекта ссылки на эти внешние подпрограммы могут быть разрешены только если библиотека, содержащая подпрограммы, добавлена к вашему файлу проекта.

□ Чтобы скомпоновать библиотеку драйвера базы данных, выделите Database driver libraries (библиотеки драйвера базы данных) в списке дерева проекта. Затем нажмите кнопку Add File и отберите драйвер, который вы хотите добавить, из диалогового окна Select Driver (выбор драйвера).

Если вы выбрали неправильный тип файла, система проекта добавляет файл к соответствующей папке дерева проекта.

□ Чтобы скомпоновать другую библиотеку, объектный или ресурсный файлы, выделите Library and object files (файлы библиотеки и объекта) в списке дерева проекта. Затем нажмите кнопку Add File и выберите файл, который вы хотите добавить, используя стандартное диалоговое окно Open File (открыть файл)

Файлы .LIB, .OBJ, .RSC и т.п. появляется в дереве проекта и любые объекты из файла,

которые имеют подходящую ссылку в вашей исходной программе, связываются в ваш целевой (исполняемый) файл.

## Добавление внешних ресурсов

Одна из наиболее вероятных задач, которую вы захотите решить с помощью системы проекта, связана с подключением внешних ресурсов к исполняемой программе. Сюда прежде всего относится графика (а именно, файлы .BMP, .ICO и .WMF) и ресурсы окон (.RSC). Подключая их к исполняемой программе, вы можете избежать необходимости поставлять их как отдельные, внешние файлы.

Если вы непосредственно сошлетесь на графический файл в структуре данных, компилятор автоматически свяжет график с вашим приложением, и таким образом в вас не возникнет необходимости добавлять графический файл к вашему дереву проекта. Например, если вы помещаете элемент управления Image в окно и устанавливаете файл по имени в диалоговом окне Image Properties (свойства изображения), редактор связей автоматически подключает этот файл в вашу исполняемую программу. Но если вы назначите другую графику к элементу управления, используя механизм свойств, например, компоновщик включит новый файл в вашу исполняемую программу, только если вы добавите файл к вашему дереву проекта.

☐ Чтобы подключить графические файлы к исполняемой программе:

1. Выделите Library, object and resource files (библиотека, объектные и ресурсные файлы) и щелкните клавишей на кнопке Add File.

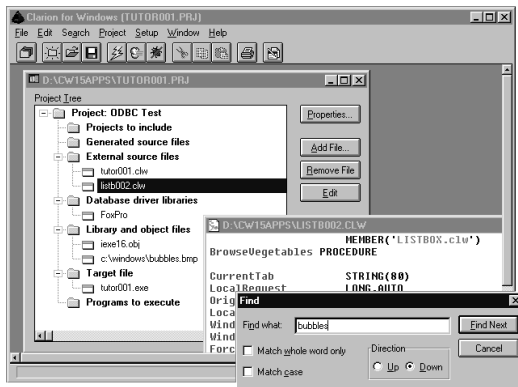
Выберите растр, пиктограмму или графический метафайл, используя стандартное диалоговое окно Open File (открыть файл).

2. Нажмите кнопку ОК чтобы вернуться к диалоговому окну Project Editor (редактор проекта).

3. Выделите файл исходной программы, который ссылается на графику, и щелкните клавишей на кнопке Edit.

Текстовый редактор открывает файл исходной программы.

4. Поместите знак тильды (~) перед именем графического файла в соответствующем операторе присваивания исходной программы (не в разделе данных).



Например: замените ?Image{PROP:Text} = 'I.ICO' на ?Image{PROP:Text} = '~I.ICO.' Знак тильда указывает, что программа должна найти позицию как скомпонованную в ресурсе, а не как внешний файл.

При желании выберите Search Find для нахождения имени файла.

5. Выберите File Exit, затем в ответ на вопрос о необходимости сохранения сделанных вами изменений, щелкните клавишей мыши на клавише Yes.

Теперь, когда вы перекомпилируете вашу программу, исполняемая программа не будет больше требовать внешний графический файл.

## **Добавление других проектов**

---

Система Управления Проектом может компилировать и компоновать другие проекты, упоминаемые в текущем файле проекта. Другой проект может даже вызывать еще один, в каскадной последовательности компиляционных ссылок.

Каскадные проекты дают вам возможность расщепить процесс разработки на отдельные проекты, а затем связать их всех вместе, когда все будет готово.

1. Чтобы добавить проект к дереву проекта, выделите Projects to Include (проекты для включения) в списке дерева проекта. Затем нажмите кнопку Add File (добавить файл) и выберите файл проекта, который вы хотите добавить, используя стандартное диалоговое окно Open File (открыть файл).

## **Добавление исполняемых программ**

---

Добавление исполняемых программ к вашему проекту дает вам возможность настроить процесс компилирования и компоновки. Это могут быть файлы .BAT или более сложные файлы .EXE, которые выполняют любые дополнительные задачи, устанавливаемые вами как часть процесса компилирования и компоновки. Программы выполняются в том порядке, в каком они появления в дереве проекта, начинаясь сразу после того, как целевой файл изготовлен. То есть, после завершения всего процесса компилирования и компоновки.

□ Чтобы добавить программу для исполнения к дереву проекта, выделите Programs to execute (программы для исполнения) в списке дерева проекта. Затем нажмите кнопку Add File и используя стандартное диалоговое окно Open File (открыть файл) выберите программный файл, который вы хотите добавить к дереву проекта.

## Распределение файлов

### Выбор конфигурации

---

Это раздел включен сюда для того, чтобы помочь вам решить, какого вида целевой файл установить для вашего проекта. Смотрите также Стратегии разработки и развертывания.

Clarion for Windows производит настоящие исполняемые файлы приложения, которые вы можете распределить без лицензионных сборов. Приложения, которые вы распределяете, требуют Windows 3.10, 3.11, 95 или NT.

Исполняемые программы Clarion изготавливаются в двух версиях: файлы .EXE и файлы .DLL. Файл .EXE является просто исполняемой программой. Файл .DLL (динамическая библиотека) является исполняемым кодом, подключается к файлу .EXE во время прогона. Это отличается от случая файлов .OBJ и .LIB, которые компонуется в файл .EXE во время компилирования и редактирования связей. Наиболее очевидное преимущество .DLL - это то, что он обеспечивает метод модификации .EXE без переделывания (компилирования и компонования) .EXE.

Исполняемые программы Clarion могут быть распределены в следующих четырех конфигурациях, где xx - "16" или "32" в зависимости от того, на какую операционную систему рассчитано приложение - на 16-битовую (Windows 3.10, 3.11) или 32-битовую (Windows 95 или Windows NT):

- ◆ \*.EXE

Состоящий из одного куска .EXE будет как правило больше, чем .EXE, распределенный с помощью .DLL(s). Однако Состоящий из одного куска .EXE будет, вероятно, меньше, чем суммарный размер .EXE и его ассоциированных .DLL(s).

Состоящий из одного куска .EXE делается по возможности небольшим с помощью Clarion-процесса интеллектуальной компоновки, при котором в исполняемый код связываются только те функции, которые действительно вызываются программой приложения (в то время когда .DLL содержит фиксированный набор функций, независимо от того, вызываются ли они действительно вашей программой).

В состоящем из одного куска .EXE не может возникать конфликтов или проблем, возникающих от связывания с неправильным .DLL во время работы.

**Время изготовления (компилирование и компоновка) для состоящего из одного куска .EXE больше, чем для .EXE вместе с .DLL.**

- ◆ \*.EXE + CWRUNxx.DLL

Расщепление исполняемых программ между .EXEs и DLLs позволяет более эффективно использовать пространство на диске. Многие приложения Clarion (.EXEs) могут делить одиночный CWRUNxx.DLL. Или, комплект одиночного приложения с несколькими .EXEs может иметь общий CWRUNxx.DLL. Однако вы, как разработчик, должны обеспечить, чтобы ваше приложение имело доступ к точной версии CWRUNxx.DLL.

Примером использования .DLL является типичная система учета где .EXEs управляют главным меню системы и вызывают подотделы системы, такие как Счета на получение и Счета на оплату от отдельных .DLLs. Этот метод распределения позволяет частям программы быть проданными и обслуживаемыми по отдельности.

**Совет: Расщепление исполняемых программ между .EXEs и .DLLs позволяет более эффективно использовать пространство на диске, но менее эффективно при этом использовать оперативную память. Это вызвано тем, что Windows загружает дополнительный CWRUNxx.DLL в память для каждой активной исполняемой программы Clarion for Windows, а также потому, что CWRUNxx.DLL содержит некоторые функции, которые ваш .EXE никогда не будет вызывать.**

**Совет: Чтобы сделать .EXE+CWRUNxx.DLL: в диалоговом окне Global Options (глобальные опции), установите Target Type (тип мишени) на .EXE а также установите библиотеку на Standalone (автономная).**

◆ \*.EXE + CWRUNxx.DLL+\*.DLL1 +.....+\*.DLLn

Эта конфигурация дает те же преимущества и неудобства, что и конфигурация .EXE+CWRUNxx.DLL. Она приведена здесь для иллюстрации того, что вы не ограничены одним .DLL, не ограничены вы также Clarion .DLLs. Ваши приложения Clarion могут использовать .DLLs, скомпилированные из других языков, наряду с CWRUNxx.DLL. Дополнительную информацию о драйверах базы данных и их файлах смотрите в Приложении. Драйверы базы данных.

◆ \*.EXE+\*.DLL1+... +\*.DLLn

Эта конфигурация дает большинство тех же преимуществ и недостатков, как и конфигурация .EXE+CWRUNxx.DLL. Здесь это дается для иллюстрации того, что CWRUNxx.DLL может быть скомпонован в другой .DLL. Эта техника “прячет” CWRUNXX.DLL и обеспечивает, что ваше приложение никогда не получит неправильную версию CWRUNxx.DLL, так как, технически, оно не ищет CWRUNxx.DLL.

**Совет: Чтобы “спрятать” CWRUNxx.DLL в другой .DLL в диалоговом окне Global Options (глобальные опции), выполните шаги, описанные в приложении Стратегии разработки и развертывания.**

## Инсталлирование и доступ к DLLs вашего приложения

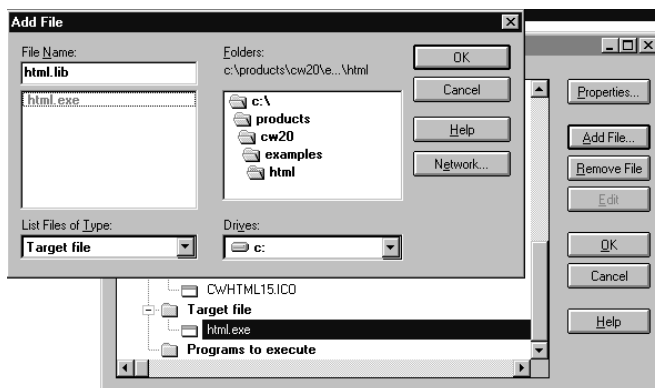
Если распределяется CWRUNxx.DLL, он должен находиться в том же самом каталоге, что и приложение, в подкаталоге Windows\System или в любом каталоге, на который есть ссылка в DOS PATH. TopSpeed рекомендует, чтобы вы инсталлировали CWRUNxx.DLL в каталог приложения при создании программы Setup для распределения ваших приложений.

Помните, что многие приложения Clarion for Windows могут использовать один и тот же файл CWRUNxx.DLL, избегая ненужного дублирования и тем самым экономить пространство на жестком диске пользователей. С другой стороны, использование только одного CWRUNxx.DLL повышает вероятность конфликтов среди приложений, разработанных под различными версиями Clarion for Windows. Чтобы избежать возможных конфликтов, инсталлируйте отдельный CWRUNxx.DLL для каждого каталога приложения или распределяйте приложение как один файл \*.EXE, или скомпонуйте CWRUNxx.DLL в другой .DLL, который уникален для вашего приложения.

## Целевой файл

Используя систему проекта, вы можете легко назначить выбор типа создаваемых файлов .EXE, .LIB или .DLL. В данном разделе приводится информация о выборе типа создаваемых целевых файлов, а также краткие объяснения характерных особенностей типов файлов. Дополнительную информацию можно найти в разделах Добавление объектных файлов и библиотек и Распределение файлов. Смотрите также Стратегии разработки и развертывания.

□ Система Управления Проектом предполагает по умолчанию, что вы хотите создать стандартный исполняемый файл (.EXE + CWRUNxx.DLL). Когда вы называете файл проекта в диалоге New Project File (файл нового проекта), он автоматически устанавливает для Target File (целевого файла) расширение .EXE, а Run-Time Library (библиотеку



поддержки) на Standalone (автономная).

□ Чтобы создать библиотеку (.LIB файл), просто измените расширение целевого файла в дереве проекта. Выделите Target file (целевой файл) в дереве проекта и нажмите кнопку Add File (добавить файл). Затем наберите имя файла .LIB, который вы хотите создать, включая расширение. Нажмите кнопку ОК.

□ Чтобы создать .DLL (динамически подключаемая библиотека), измените целевое расширение в дереве проекта. Выделите Target file (целевой файл) в дереве проекта и нажмите кнопку Add File (добавить файл). Затем наберите имя файла .DLL, который вы хотите создать, включая расширение. Нажмите кнопку ОК.

**Внимание: Когда вы разрабатываете приложения с помощью Генератора приложений, очень важно, чтобы вы установили Destination Type (тип назначения) приложения, а не Project's Target Type (тип цели проекта), так как Генератор приложений и шаблоны выполняют решения генерации кода на основе значения типа назначения приложения, а не типа цели проекта.**

## Файлы .LIB

Библиотечные файлы содержат (внешние) процедуры и функции, которые подключаются к вашему приложению во время компиляции. Чтобы создать библиотечные файлы, к которым можно иметь доступ через Clarion for Windows, или любые компиляторы TopSpeed, просто укажите в качестве целевого файла файл с расширением .LIB.

Чтобы использовать процедуры и функции из заранее откомпилированного файла .LIB, вы должны описать прототипы всех внешних процедур и функций, вызываемых вашей программой. Макетирование достигается добавлением структуры MODULE к карте вашего приложения. Чтобы вызвать внешнюю процедуру .LIB из “вручную закодированной” программы:

1. Добавьте структуру MODULE к MAP вашего приложения.

MODULE должна ссылаться на внешний библиотечный файл. В генераторе приложений вы можете поместить это в точку вставки Inside the Global Map (Внутри глобальной карты).

2. Добавьте прототипы функции или процедуры:

```
MAP
  MODULE('EXTERNAL.LIB')
    ExtProc(*CSTRING),RAW      ! макет процедуры
    ExtFunc(USHORT,*BYTE[ ]),USHORT ! макет функции
  END
END
```

Каждый прототип устанавливает имя процедуры или функции, типы данных любых параметров (в скобках) и тип возвращаемых данных (если это функция).

В вышеприведенном примере процедура (называемая здесь ExtProc) ожидает в качестве параметра адрес переменной типа CSTRING(без длины, отсюда атрибут RAW) .

Функция (здесь называемая ExtFunc) ожидает величину переменной USHORT, адрес массива BYTE и возвращает USHORT.

3. Чтобы установить дополнительные соглашения о вызове, добавьте их к описанию прототипа.

Вы можете использовать файлы .LIB или .OBJ, созданные другими компиляторами.

Модифицируя вышеприведенные примеры, получим, что первая строка из двух приведенных ниже, идентифицирует процедуру, как ожидающую C - соглашений вызова. Вторая строка идентифицирует функцию как ожидающую соглашения вызова PASCAL, которое является стандартным соглашением Windows:

```
ExtProc(*CSTRING), C, RAW  
ExtFunc(USHORT, *BYTE[ ]), USHORT, PASCAL
```

4. Чтобы, если необходимо, установить чужой идентификатор компоновщика, добавьте его к прототипу.

Некоторые компиляторы, особенно компиляторы языка 'C', добавляют лидирующий символ подчеркивания к имени процедур и функций во время компилирования. Приведенные ниже примеры добавляют атрибут NAME:

```
ExtProc(*CSTRING), C, RAW, NAME(' ExtProc' )  
ExtFunc(USHORT, *BYTE[ ]), USHORT, PASCAL, NAME( '_ExtFunc' )
```

Более полную информацию о MODULE, MAP, PROCEDURES, FUNCTIONS и прототипах вы можете найти в Справочнике языка.

## Файлы .DLL

Динамически вызываемые библиотеки (.DLL) содержат процедуры и функции, которые подключаются к вашей программе во время исполнения. Чтобы создать динамически вызываемую библиотеку, просто установите для вашего целевого файла расширение .DLL .

Чтобы вызвать внешнюю .DLL-процедуру, повторите шаги, описанные выше для вызова .LIB процедуры.

**Внимание:** Когда вы разрабатываете приложения с помощью Генератора



приложений, очень важно, чтобы вы установили **Destination Type** (тип назначения) приложения, а не **Project's Target Type** (тип цели проекта), так как Генератор приложений и шаблоны выполняют решения генерации кода на основе значения типа назначения приложения, а не типа цели проекта.

## Файл списка загрузки

---

Для генерированных приложений Clarion for Windows автоматически создает файл загрузки (.SHP), который содержит имена файлов, которые необходимы для работы вашего приложения. Файл называется application.SHP и находится в том же подкаталоге, что и ваш файл .APP.

Этот список включает только те файлы, которые являются видимыми для шаблонов Clarion for Windows. Любые DLLs, загруженные в файлы EMBEDs или INCLUDE могут быть невидимы для шаблонов и могут не присутствовать в списке.

В случае модулей Внешней библиотеки файл .LIB будет в перечислении. Некоторые из .LIBs (например, WINDOWS.LIB) не имеют ассоциированных с ними DLLs, однако в большинстве случаев вам нужно поставлять ассоциированный DLL.

В случае модуля Внешней библиотеки, генерированного Clarion for Windows, вы должны обеспечить, чтобы все файлы в списке загрузки для этого LIB/DLL были также включены.

**Совет: Вы можете модифицировать список загрузки вашего приложения путем встраивания текста в глобальной точке встраивания Inside the Shipping List (внутри списка загрузки).**

В этом разделе описываются отдельные компоненты файла проекта и показывается, как модифицировать их свойства.

## Глобальные опции компиляции и компоновки

---

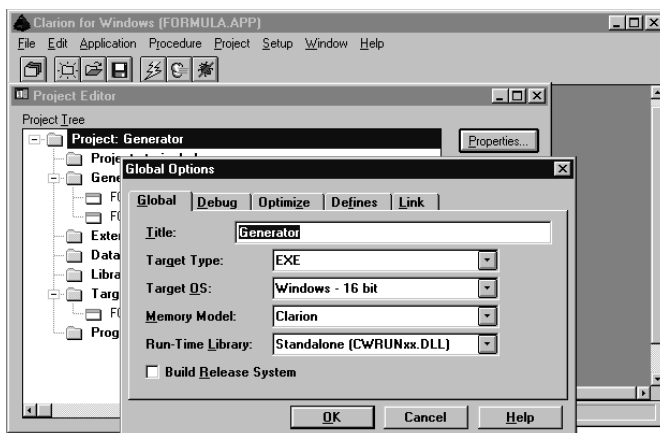
Выберите первую строку из дерева проекта и нажмите кнопку Properties (свойства), чтобы открыть диалоговое окно Compile Options (параметры компилирования). В это окно включены следующие поля:

### Закладка “Глобальные” (Global)

**Title (заголовок)** Чтобы добавить краткое текстовое описание, введите его в поле Title (заголовок). Система проекта поместит описание рядом с именем проекта в дерева проекта.

**Target Type (тип цели)** Чтобы установить тип исполняемого файла, выберите .EXE, .LIB, или

.DLL из выпадающего списка Target Type (тип цели).



Target OS (целевая Операционная Система)

Чтобы установить целевую операционную систему исполняемой программы, выберите Windows 16-бит или Windows 32-бит из выпадающего списка Target OS.

**Внимание: Вы можете компилировать и компоновать 32-битовые исполняемые программы с помощью Windows 3.1 если у вас установлен Win32S, однако у вас должны быть Windows 95 или Windows NT для работы с ними.**

Memory Model (модель памяти)

Модель Clarion не является опционной.

Run-Time Library (библиотека времени исполнения)

Чтобы установить, как библиотека времени исполнения вызывается целевым файлом, выберите Standalone, Local или External из выпадающего списка Run-Time Library (библиотека времени исполнения). Смотрите более полную информацию в Стратегии разработки и развертывания.

Standalone (одиночный)

Создает целевой файл, поэтому вызывает библиотеки поддержки как CWRUNxx.DLL.

Local (локальный)

Создает целевой файл с библиотекой поддержки, скомпонованной внутренне (исполняемая программа “из одного куска”).

External (внешний)

Компонуется приложение так, что оно вызывает библиотеку поддержки из .DLL, который вы создали с помощью библиотеки поддержки, отдельно скомпонованной и экспортированной. Более полную

информацию об экспортном файле (.EXP) вы можете найти в Руководстве программиста.

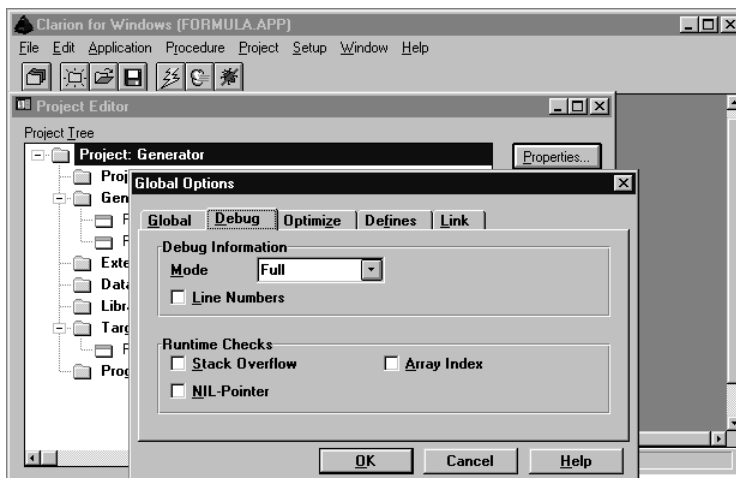
### Build Release System (построить систему редакции)

Чтобы создать исполняемую программу для редактирования, отметьте поле Build Release System. Чтобы создать исполняемую программу для использования с отладчиком, очистите поле флажков Build Release System.

### Закладка “Отладка” (Debug)

#### Debug Mode (режим отладки)

Чтобы определить уровень возможностей отладчика, выберите из выпадающего списка Mode: Off (выключено), Min (минимальный) или Full (полный).



#### Line Numbers (номера строк)

Чтобы установить число строк, которые должны быть встроены в объектном файле, отметьте это поле. Это не обязательно для отладчика Clarion, но может быть полезно при использовании других отладчиков.

#### Stack Overflow (переполнение стека)

Чтобы активизировать механизм выдачи предупреждающих сообщений во время работы, отметьте поле Stack Overflow.

#### Nil-Pointer (Нулевой указатель)

Для выдачи предупреждающих сообщений компилятора о неразрешенных ссылках, отметьте это поле.

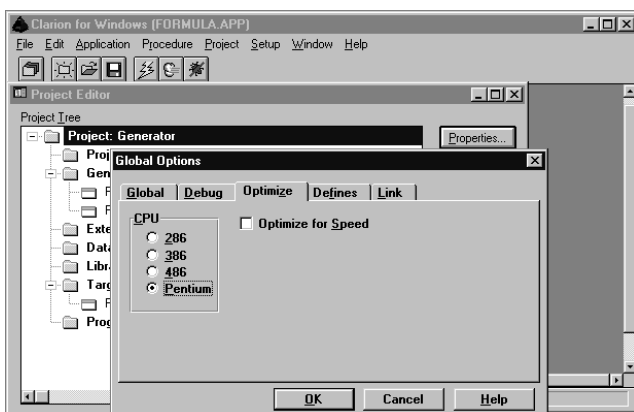
## Array Index (индекс массива)

Для проверки выхода индексов за границы массива, отметьте это поле.

## Закладка “Оптимизация” (Optimize)

### CPU (процессор)

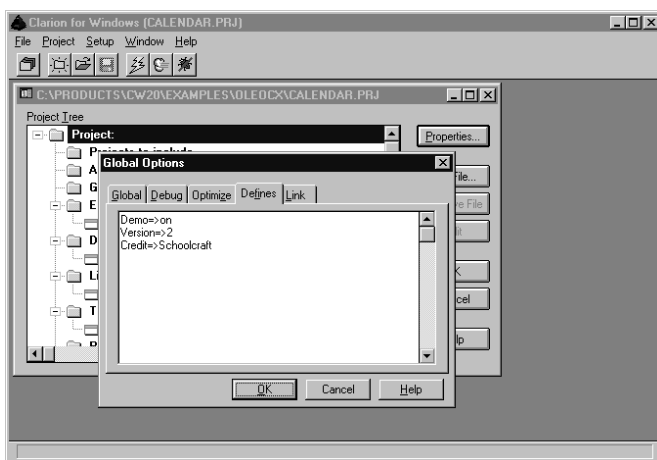
Чтобы установить оптимизацию в зависимости от типа микропроцессора, выберите или 286 или 386 или 486. Оптимизация под Pentium в настоящее время не поддерживается. Пользователи Pentium должны назначить 486..



### Optimize Speed (оптимизировать скорость)

Чтобы увеличить скорость программы за счет размеров исполняемого модуля, отметьте это поле.

## Закладка “Определения” (Defines)



## Defines (определения)

Чтобы определить переключатель или переключатели для использования с директивами транслятора COMPILE (компилировать) и OMIT (пропустить), наберите список правильных меток Clarion, разделенных запятыми. Каждая метка определяет отдельный переключатель. Величина по умолчанию для каждого переключателя - “включено”. Чтобы назначить для каждого переключателя величину не по умолчанию, наберите label=>value.

Defines ссылается на оператор языка системы проекта #PRAGMA DEFINE(). Оператор #PRAGMA DEFINE() создает переключатель, который может быть коммутирован в положения вкл. и выкл. Смотрите более полную информацию о #PRAGMA DEFINE() в Руководстве программиста. Переключатель можно затем быть опрошен с помощью директив компилирования COMPILE и OMIT.

Например, наберите ‘Demo’ в поле Defines. Система проекта создаст переключатель, называемый Demo, и переведет его во “вкл”. Теперь вы можете использовать переключатель в условных операторах COMPILE и OMIT внутри исходной программы. Например:

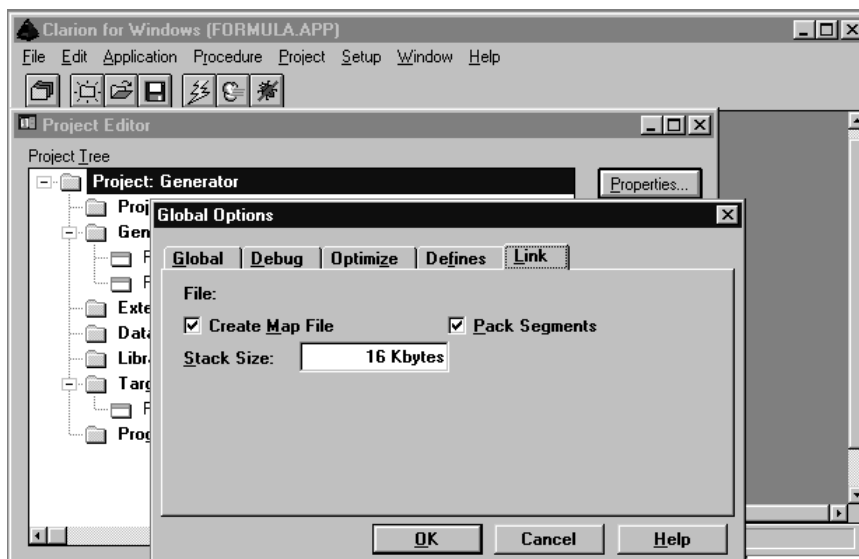
```
COMPILE('END COMPILE',DEMO=ON)
IF TODAY() > FirstRunDate + 30
    #ReturnCode = MESSAGE('Beta period
expired')
    RETURN
END
END COMPILE
```

## Zero Divide (деление на ноль)

Чтобы сделать возможным деление на ноль с использованием переменных с плавающей точкой (REAL, SREAL, BFLOAT4 и т.д.) в подвыражениях Clarion’a, переключатель zero\_divide (деление на ноль) должен быть установлен в файле проекта приложения.

Наберите “zero\_divide” в поле Defines(определения). Система проекта создаст переключатель и поставит его на “включено”. Скомпилируйте ваше приложение, а затем, во время прогона, деление на ноль вернет ноль вместо исключительной ситуации с плавающей запятой.

## Закладка “Компоновка” (Link)



### Create Map File (создать Map файл)

Чтобы создать Map файл, который содержит информацию о размерах сегментов и общих функциях, установите флажок в это поле. Map файл может быть использован с отладчиками третьих фирм.

### Pack Segments (упаковать сегменты)

Чтобы упаковать сегменты данных и программы в файл .EXE, установите флажок в это поле.

### Stack Size (размер стека)

Чтобы установить размер стека в килобайтах, введите в это поле необходимое число.

## **Опции компилирования отдельных исходных модулей**

Вы можете установить опции компилирования, как для отдельных исходных модулей, так и для всего проекта в целом. Индивидуальные установки имеют преимущество перед глобальными установками компилирования. Настраивая установки компилирования для отдельных исходных модулей, вы можете установить необходимость полномасштабной отладки только для одного модуля и ни для какого другого.

Выделите файл исходной программы в диалоговом окне дерева проекта, затем нажмите кнопку Properties (свойства). Появится диалоговое окно Compile Options (параметры компилирования), показывающие большинство тех же элементов управления, как и диалог Global Option. Это диалоговое окно устанавливает опции компилирования для индивидуальных выделенных исходных модулей.

Информацию о использовании этого диалога вы можете найти выше в Закладке “Отладка”, Закладке “Оптимизация” и Закладке “Определения”.

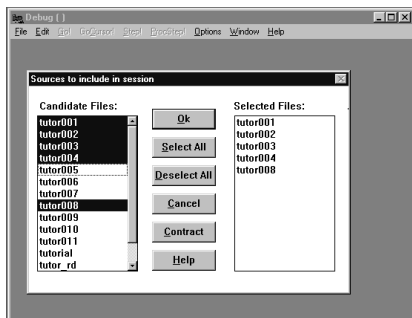




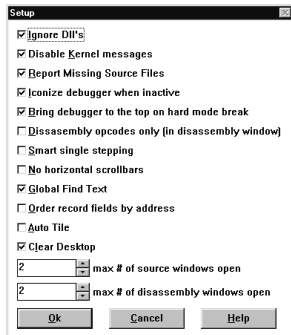
## Глава 16 Отладчики



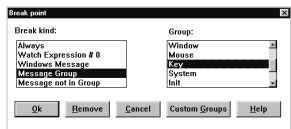
Отладка программы обычно требует выполнения программы с многократными ее остановками для выяснения текущих значений отдельных переменных. Отладчик Clarion предоставляет все инструменты, необходимые для отслеживания ошибок вашей программы.



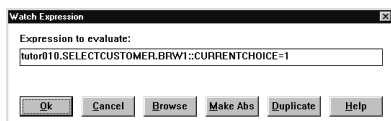
Запускайте отладчик из IDE или из Windows. В одном из окон вы увидите вашу исходную программу во время исполнения.



Настройте отладчик так, чтобы он соответствовал вашим рабочим условиям.



Как только у вас возникнут подозрения о причине ошибки, установите точку прерывания. Вы можете затем исполнить программу и ее выполнение будет автоматически остановлено в этой точке для проверки и изменения величин переменных.



Установите точки прерывания на простых или сложных выражениях.

Обзор: Процесс отладки  
Подготовка ваших проектов для отладки  
16-битовый отладчик  
Запуск отладчика  
Загрузка исходных файлов  
Установка параметров отладки  
Окна отладчика  
Установка точек прерывания  
Выполнение программы  
Работа с исходной программой  
Редактирование выражений наблюдения  
Редактирование переменных во время прогона  
32-битовый отладчик  
Запуск отладчика  
Загрузка исходных файлов  
Установка опций отладчика  
Окна отладчика  
Назначение точек прерывания  
Исполнение программы  
Редактирование переменных во время работы

Clarion for Windows предоставляет два отладчика: 16-битовый отладчик для 16-битовых приложений и 32-битовый отладчик для 32-битовых приложений. И тот и другой являются мощными инструментами для обнаружения и диагностики ошибок в ваших приложениях. Вы можете проверить исходную программу и данные во время исполнения вашей программы и проверить поведение всех элементов управления при выполнении вашей программы.

В этой главе будет рассказано о том, как:

- ◆ Подготовить ваши проекты для отладки.
- ◆ Как запустить отладчик.
- ◆ Как настроить работу отладчика на ваши рабочие условия.
- ◆ Как контролировать исполнение вашей программы и проверить ее состояние в специфических точках путем установки точек прерывания и наблюдением за значениями переменных.

## Процесс отладки

Отладчики очень гибки и достаточно сложны, здесь имеется множество окон, опций и рабочих характеристик. Данный обзор процесса отладки предполагает общую последовательность шагов, которые вводят вас в наиболее важные особенности отладчиков с минимальной путаницей. Имейте в виду эту последовательность шагов когда вы будете изучать отладчики.

1. Закройте другие приложения, затем запустите отладчик.

Это дает два преимущества. Во-первых, большинство ресурсов системы оказываются доступными для вашего приложения и отладчика. Во-вторых, вы не потеряете данные из других активных приложений если крах системы случится во время процесса отладки.

2. Загрузите только те исходные файлы, которые вам нужно отладить.

Каждый выбранный вами исходный файл становится дочерним окном в отладчике. Чем меньше исходных файлов вы отберете, тем меньше беспорядка будет на экране отладчика и тем меньше будет накладок, которые должен будет обработать отладчик.

3. Установите параметры отладчика.

Потратьте несколько минут на то, чтобы почитать об опциях запуска. Такие опции как Clarion Soft Mode (мягкий режим), Autotile (автоуправление окнами), Clean Desktop(чистый стол), отладчик наверху(On Top), Global Find Text (глобальное обнаружение текста) и другие помогут читать данные (16-битового) отладчика и работать с ним.

**Совет: Мягкий режим Clarion рекомендуется для большинства 16-битовых проектов. Однако в Windows 95 вам нужно использовать жесткий режим. В жестком режиме вся другая деятельность системы задерживается на то время, пока работает отладчик. Это означает, что “рабочий стол” не перекраивается, иначе это могло бы смутить вас если вы этого не ожидали.**

4. Назначьте точку прерывания.

5. Запустите ваше приложение (отлаживаемое) с помощью команд Go (пуск) или Step (шаг).

6. Выберите и настройте окна отладчика.

Многие окна отладчика будут пустыми до тех пор, пока ваше приложение не остановится в точке прерывания. Как только ваше приложение останавливается и окна заполняются, содержимое окон станет иметь больше смысла для понимания и работы. Превратите в

пиктограмму или закройте окна, которые вам не нужно видеть.

7. Назначьте точки прерывания, установите выражения наблюдения и измените величины переменных.

8. Запустите ваше приложение с помощью команд Go (пуск) или Step (шаг).

9. Повторите шаги 7 и 8 при необходимости.

10. Выйдите из вашего приложения (отлаживаемого).

Очень важно, чтобы вы вышли из отлаживаемой программы прежде, чем вы выйдете из отладчика. Выход из отладчика пока отлаживаемое приложение еще активно может вызвать нарушение работы всей системы.

## **Подготовка ваших проектов для отладки**

Система проекта дает вам возможность установить опции отладки для всех программ вашего приложения в диалоговом окне Global Options (глобальные опции). Чтобы сделать вашу исполняемую программу (.EXE или .DLL) подходящей для отладки:

1. Создайте файл вашего проекта и сделайте его текущим проектом (как это сделать, объясняется в главе Использование системы проекта).

2. Выберите Project > Edit для вызова диалогового окна Project Editor (дерево проекта).

3. Выберите верхний уровень дерева, который содержит имя проекта и нажмите кнопку Properties (свойства).

4. Когда появится диалоговое окно Global Options (глобальные опции), выберите закладку Debug (отладка), затем выберите Full (полный) выпадающего списка Debug Mode (режим отладки).

5. Если необходимо, отметьте поле Line Numbers (номера строк).

Номера строк автоматически доступны для отладчика Clarion, однако если вы используете другой отладчик, отметка этого поля сделает номера строк доступными для этого другого отладчика.

6. Нажмите кнопку ОК для того, чтобы закрыть диалоговое окно Global Options, а затем и диалоговое окно Project Editor,

7. Нажмите кнопку Make (выполнять) на панели инструментов для компилирования и

компонования вашего приложения.

Приложение теперь включает информацию, необходимую для отладчика.

Вы можете также определить информацию об отладке для отдельного модуля в проекте. Это уменьшает накладные расходы отладчика. Чтобы сделать это, следуйте шагам приведенным выше для Global Options, за исключением None из выпадающего списка Debug Mode. Затем следуют такие шаги:

1. Выберите Project > Edit для просмотра диалогового окна Project Editor (дерево проекта).
2. Выберите только тот исходный модуль, который вам нужно отладить, и нажмите кнопку Properties (свойства).
3. Когда появится диалоговое окно Compile Options (параметры компилирования), выберите Full (полный) из выпадающего списка Debug Mode (режим отладки).
4. Нажмите кнопку ОК для того, чтобы закрыть диалоговое окно Project Editor и диалоговое окно Compile Options.
5. Нажмите кнопку Make на панели инструментов для компилирования и компонования приложения.  
Это включает информацию отладки только для одного модуля.

## 16-битовый отладчик

### Запуск отладчика

16-битовый отладчик работает как отдельное приложение, но вы можете запустить его либо внутри интегральной Среды разработки или прямо из Windows. Если вы начинаете из Windows, с незагруженной средой разработки, это означает, что больше ресурсов системы будет доступно для вашего приложения и отладчика.

□ Чтобы запустить отладчик находясь внутри интегральной среды разработки, либо:

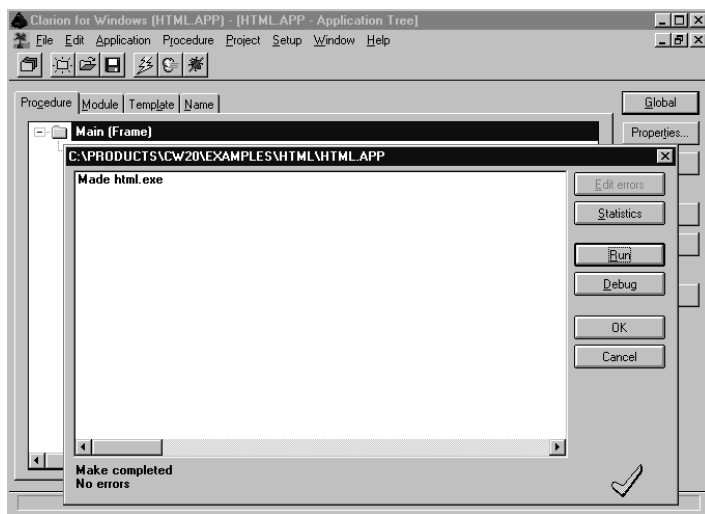
1. Выберите Project > Debug или нажмите кнопку Debug (отладка) на панели инструментов.

Интегральная Среда разработки проверяет информацию проекта для того, чтобы определить, является ли ваше приложение 16-битовым или 32-битовым, и запускает соответствующий отладчик

либо...

2. Компилируйте и компоуйте ваше приложение, нажав кнопку Make (выполнить), затем, при еще открытом диалоговом окне результатов компилирования, нажмите кнопку Debug (отладка).

Два способа запуска отладчика.



□ Чтобы запустить отладчик из среды Windows:

1. Переключитесь на диспетчер программ (Windows 3.x) или нажмите кнопку Start (старт) на линейке задач (Windows 95) и откройте программную группу Clarion.

2. Дважды щелкните мышью на 16-битовой пиктограмме (Windows 3.x) Clarion Debugger (отладчик Clarion) или выберите 16-битовый отладчик Clarion из меню программ (Windows 95).

3. С запущенным отладчиком выберите File > File to Debug, затем в диалоговом окне Open File (открыть файл) выберите файл .EXE.

Вы можете загрузить отладчик, затем отладить программу, которая уже запущена и работает к моменту загрузки отладчика. Это полезно для ситуаций, в которых разрабатываемая программа “плохо себя ведет”, но еще не произвела фатальной ошибки.

Запустите отладчик как обычно и выполните команду File > File to Debug (файл для отладки). Выберите из диалога Open File (открыть файл) файл .EXE для выполнения программы. Отладчик попросит вас подтвердить, что вы хотите отладить работающую в данный момент программу.

**Совет:** При выполнении отладки активизируйте только отладчик и отлаживаемые программы. Поступая так, вы не потеряете данные в других приложениях в случае если разрушение системы случится во время процесса отладки.

## Загрузка исходных файлов

Когда вы работаете с отладчиком, вы должны отобрать для отладки файлы исходной программы. Для этого автоматически после запуска отладчика появляется диалоговое окно Sources to include in session (исходные для включения в сеанс).

□ Чтобы загрузить файлы исходной программы после появления отладчика:

1. Выберите файлы исходной программы в диалоговом окне Sources to include in session (источники для включения в сеанс), щелкнув для этого на них клавишей мыши.

Отладчик сохраняет выбранные вами файлы между сеансами отладки.

2. Чтобы включить все исходные файлы проекта, нажмите кнопку Select All.

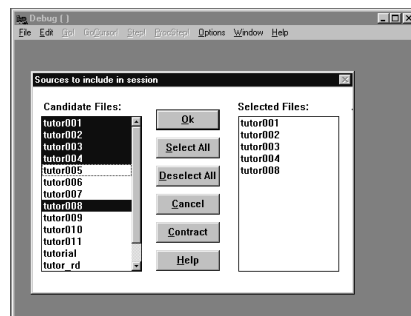
3. Нажмите кнопку Expand/Contract (расширить/сжать) для показа списка только тех исходных файлов, которые были выбраны.

4. Нажмите кнопку ОК.

Открываются окна отладчика.

Если приложение не включает информацию отладки, отладчик пропускает этот шаг и открывает окно дисассемблера (объясняется дальше).

Отладчик подсказывает вам выбрать файлы исходной программы для проекта.





## Установка параметров отладки

---

Из-за своей большой гибкости отладчик довольно сложен, поэтому установка некоторых базовых опций перед использованием отладчика может оплатиться сокращением периода обучения. В частности, мы рекомендуем включение Clarion Soft Mode (мягкого режима) для большинства проектов, разрабатываемых в Windows 3.1.

Меню Options (параметры) отладчика предоставляют несколько параметров - переключателей, которые могут помочь тонко настроить метод отладки вашего проекта.

### Soft Mode

(мягкий режим)      Переключение между режимами жесткой и мягкой отладки.

В мягком режиме, когда отлаживаемая программа остановлена в отладчике, часть отладчика попытается воспроизвести поведение отлаживаемой программы.

В жестком режиме, когда отлаживаемая программа остановлена в отладчике, единственное окно, с которым можно работать, это отладчик. Все другие активные процессы приостанавливаются. Одно из следствий этого - это то, что экран не перерисовывается. Другое - это то, что другие активные приложения будут недоступны до тех пор, пока отладчик не вернет управление отлаживаемой программе.

**Совет: Работая в жестком режиме, наберите D чтобы перевести отладчик наверх.**

### Clarion Soft Mode (мягкий режим Clarion)

Отладчик будет использовать часть библиотеки времени исполнения для имитации поведения отлаживаемой программы. Этот режим рекомендуется для большинства проектов.

### Extended Stack Trace (расширенная трассировка стека)

Отладчик показывает информацию о процедурах, когда нет никакой информации об отладке. Открывается окно дисассемблера, содержащее нужный сегмент.

### Disassembly On (включен дисассемблер)

Окно дисассемблера “затеняет” окно активного исходного текста. Когда вы выбираете исходный текст, курсор в Окне Дисассемблера перемещается к строке, соответствующей этому тексту.

Это меню - параметр переключатель. Если Окно Дисассемблера закрыто в то время, когда вы включаете данный режим, вы можете открыть его двойным щелчком мыши на строке исходного текста, затем нажатием Cancel (прервать) в диалоговом окне Break Point

(точка прерывания).

Assembly Single Step (пошаговый режим для программы на языке ассемблера)

Переключает шаговый режим для точек прерывания ассемблера.

Когда исполнение достигает точки прерывания ассемблера, устанавливается шаговый режим. Когда исполнение достигает точки прерывания указанной в исходном тексте, он выключается.

Control Panel

(панель управления) Показывает окно панели инструментов с кнопками, соответствующими четырем командам Go! Следующий раз, когда отлаживаемая программа будет остановлена, панель управления получит фокус.

**Совет:** При отладке в жестком режиме, когда вы переключаете активное окно на главное окно отладчика, вы не сможете получить доступ к панели управления.

Setup

Открывает диалоговое окно Setup. См. ниже.

### Опции Setup отладчика

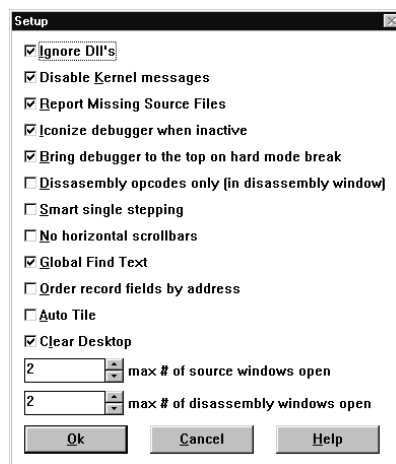
Получите доступ к диалоговому окну отладчика Setup с помощью команды Options > Setup. Этот диалог обеспечивает установку следующих опций:

Ignore .DLLs (игнорировать файлы .DLL)

Инструктирует отладчик игнорировать информацию об отладке в файлах .DLL. Это уменьшает время старта отладчика.

Disable Kernel messages (сделать недоступными сообщения ядра)

Если вы используете отладчик Windows (имеющийся в Microsoft Windows 3.1 SDK), отладчик автоматически захватит сообщение об ошибке, посланное ядром (одна из трех главных библиотек динамических связей, используемых Windows). Вы можете установить местонахождение таких ошибок с помощью команды Find Last Error (найти последнюю ошибку).



Если вы не используете устройство AUX для подачи сообщений, добавьте строку OutputTo=NUL в секцию [DEBUG] вашего файла SYSTEM.INI.

Report Missing Source Files (сообщить о пропавшем исходном файле)

Отладчик автоматически подсказывает имена исходных файлов, местонахождение которых он не может установить.

Iconize debugger when inactive (Сворачивать до пиктограммы неактивный отладчик)

Автоматически свертывает отладчик до пиктограммы, когда он неактивен.

Bring debugger to the top on hard mode break (поместить отладчик над всеми окнами при прерывании в жестком режиме)

Отладчик появляется над любым другим открытым окном, если он активен; относится только к жесткому режиму отладки.

**Совет: При работе в жестком режиме наберите D, чтобы вывести отладчик наверх.**

Disassembly opcodes only (in disassembly window)(только коды дисассемблера в его окне)

Окно дисассемблера содержит только коды операций, удаляя промежутки, занятые двоичными кодами.

Smart Single Stepping (быстрый одношаговый режим)

В одношаговом режиме при достижении строки, содержащей вызов процедуры, загружается информацию об отладке для целевой процедуры, если таковая имеется. Эта возможность распространяется на .DLL с информацией об отладке.

No Horizontal Scrollbars (без горизонтальной прокрутки)

Прячет линейки горизонтальной прокрутки.

Global Find Text (Глобальный поиск текста)

Когда недоступно, каждое окно текста исходной программы “помнит” свою собственную поисковую строку символов. Когда сделано доступным, поисковый текст по умолчанию будет тем же самым, что и последний поиск, независимо от окна.

Order record fields by address (упорядочение полей записи по адресам)

Упорядочивает переменные в структуре RECORD по адресам памяти.

Auto Tile (автоматическое упорядочивание окон)

Упорядочивает окна отладчика.

### Clear Desktop (Очистить экран)

Уменьшает до предела все работающие приложения (кроме отлаживаемого) при активации отладчика.

### Max # of source windows open (максимальное число открытых окон исходного текста)

Устанавливает максимальное число окон исходного текста, которые отладчик откроет за раз.

### Max # of disassembly windows open (максимальное число окон дисассемблирования)

Устанавливает максимальное число окон дисассемблера, которые отладчик откроет за раз.

## Дополнительные команды отладчика

В дополнение к нормальным параметрам окна и запуска отладки из данного меню, вы можете активировать специальные режимы и параметры :

### Redirection (переадресация)

Чтобы использовать с отладчиком файл переадресации, отличающийся от CW20.RED, выберите File > Load Redirection. Файл Redirection (переадресации) помогает отладчику найти такие файлы, как \*.DBD или \*.CLW. Смотрите главу Использование системы проекта, Редактирование файла переадресации.

### Active DLL's (активные DLL)

Чтобы добавить к сеансу отладки динамически вызываемую библиотеку (\*.DLL), выберите File > Debug Active DLL. Выберите файл из диалога Active Module (активный модуль). Эта возможность может быть доступна только для жесткого режима отладки.

### Sleep (спящий режим)

Чтобы перевести отладчик в спящий режим, в котором он ожидает общую ошибку защиты (GPF), CTRL+ALT+SYS REQ или INTERRUPT(INT3), выберите File > Sleeper Mode. Эта возможность доступна только для режима жесткой отладки.

Вы можете запустить отладчик в спящем режиме из командной строки DOS путем добавления / S к команде запуска отладчика.

**Совет: Если отлаживаемая программа попадает в бесконечный цикл, выйти из него можно при помощи CTRL+ALT+SYS RQ.**

### Restart (повторный запуск)

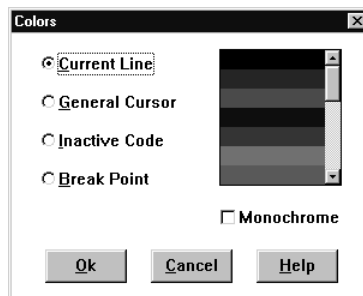
Чтобы начать сеанс отладки с выражениями наблюдения и точками остановки из ранее законченной сессии, выберите File Restart. Эта

опция работает только при условии, что исходная программа не изменилась.

**Position (позиция)** Чтобы установить размер окна отладчика до максимального размера экрана, не занятого отладчиком, выберите **Window Position Debuggee**. Эта возможность не действует, когда окно отладчика увеличено до предельного размера.

**Message Groups (группы сообщений)** Чтобы установить ваши собственные группы сообщений для наблюдения, выберите **Options Custom Groups**. Введите имя группы и выберите сообщения **Windows** из поля списка в диалоговом окне **Selective Break Point Groups** (группы селективных точек прерывания)

**Colors (цвета)** Чтобы настроить палитру отладчика, выберите **Options Custom Colors**. Выберите цвета для **Current Line** (текущая строка), **General Cursor** (главный курсор), **Inactive Code** (неактивный код) и **Break Points** (точки прерывания) в диалоговом окне **Color** (цвет).



## Окна отладчика

Отладчик состоит из набора дочерних окон, которые прослеживают для вас различную информацию о программе. Этими окнами являются:

- Окно исходной программы
- Окно выражений наблюдения
- Окно глобальных переменных
- Окно активных процедур
- Окно дисассемблера
- Окно регистров процессора
- Окно библиотечных состояний
- Окно сообщений окон

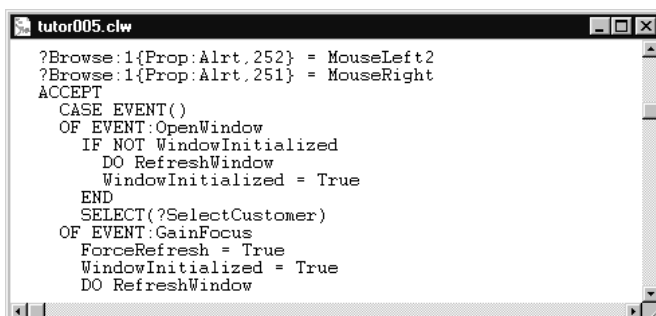
После того, как вы запустили отладчик, остановитесь и настройте различные окна на удобный для вас формат. Расположите наиболее важные окна там, где вы можете быстро просматривать нужную вам информацию. Закройте или превратите в пиктограммы ненужные вам окна.

### Окна по умолчанию

Сначала отладчик открывает четыре окна:

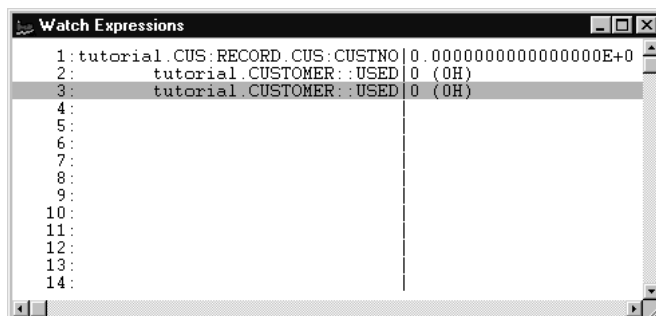
❑ Окна Source code (исходная программа) показывают документы исходного текста программы. Строка заголовка показывает имя модуля исходного текста. По умолчанию следующая строка, которую нужно исполнять, зеленая. Вручную выбранные вами строки имеют зелено-голубой (cyan) цвет.

**Совет:** Если отладчик открывается без вывода каких-либо исходных текстов в диалоговом окне Sources to include in session (источники для включения в сеанс), наиболее вероятной причиной этого является то, что ни один из исходных файлов, перечисленных в дереве проекта, не имеет информации об отладке. Проверьте файлы .DBD в C:\CW20\OBJ.



```
tutor005.clw
?Browse:1{Prop:Alrt,252} = MouseLeft2
?Browse:1{Prop:Alrt,251} = MouseRight
ACCEPT
CASE EVENT()
  OF EVENT:OpenWindow
    IF NOT WindowInitialized
      DO RefreshWindow
      WindowInitialized = True
    END
  SELECT(?SelectCustomer)
  OF EVENT:GainFocus
    ForceRefresh = True
    WindowInitialized = True
    DO RefreshWindow
```

❑ Окна Watch Expressions (Выражения наблюдения) показывают текущее значение переменных и выражений. Установите выражение наблюдения чтобы увидеть, как при исполнении вашей программы меняется переменная или выражение.



Watch Expressions	
1:	tutorial.CUS:RECORD.CUS:CUSTNO 0.0000000000000000E+0
2:	tutorial.CUSTOMER::USED 0 (0H)
3:	tutorial.CUSTOMER::USED 0 (0H)
4:	
5:	
6:	
7:	
8:	
9:	
10:	
11:	
12:	
13:	
14:	

\* Строка заголовка - Watch Expressions (выражения наблюдения). Смотрите раздел Editing Watch Expressions (редакция выражений наблюдения) для изучения синтаксиса выражений наблюдения (ниже). Чтобы добавить переменную к списку наблюдения:

1. Дважды щелкните мышью на пустой строке в окне Watch Expressions (Выражения наблюдения).

2. Когда появится диалоговое окно Watch Expressions (выражения наблюдения), наберите имя переменной (как оно появляется в списке глобальных переменных) и нажмите ОК.

3. Или нажмите кнопку Browse (просмотр), выберите переменную из списка, затем нажмите дважды ОК.

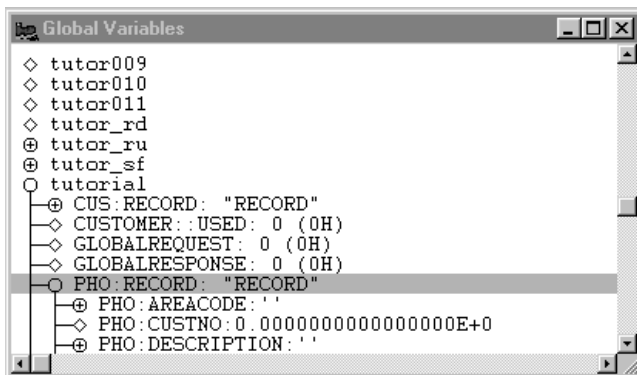
В окне Watch Expressions появятся выражение или переменная, а рядом с ней появится ее текущее значение.

Вы можете также добавить переменную к окну Watch Expressions, щелкнув для этого дважды мышью на переменной в любом из окон глобальных переменных или активных процедур.

Вы можете отредактировать переменную в окне Watch Expressions с помощью двойного щелчка мышью на ней и набрав выражение в диалоговом окне Watch Expressions (выражения наблюдения). Смотрите ниже раздел Редактирование выражений наблюдения.

**Совет: Чтобы быстро добавить структурированную переменную (такую как запись, строка символов или массив) к списку наблюдения, щелкните дважды мышью на ней в окнах глобальных переменных или активных процедур, затем нажмите кнопку Copy Variable to Watch (копировать переменную для наблюдения).**

□ Окно Global Variables (Глобальные переменные) показывает вам текущее значение каждой компоненты каждой глобальной переменной. Например, переменная в виде строки из восьми символов показывается на восьми отдельных строках, показывающих содержание

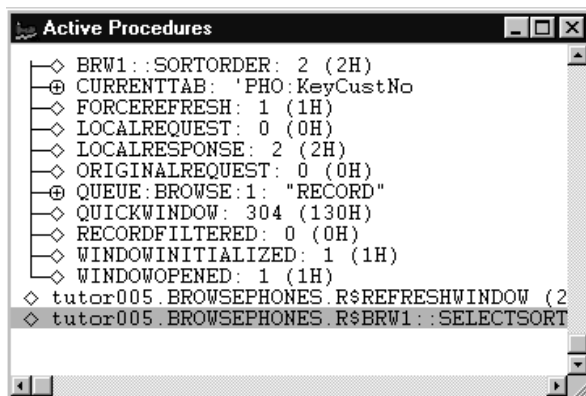


каждой позиции в строке символов.

Окно Глобальные переменные содержит элементы управления дерева, так что вы можете расширить только те переменные, которые вы хотите исследовать. Элементы управления, имеющие (+), расширяются при щелкании на них клавишей. Элементы управления, содержащие (-), сжимаются при щелкании на них.

Верхний уровень - это модуль исходной программы, который содержит переменную. Следующий - имя переменной.

□ Окно Active Procedures (Активные процедуры) дает список исполняемых в данный момент процедур, который позволяет вам контролировать вызовы вложенных процедур. Список появляется в формате дерева. Верхние уровни представляют имена процедур, а нижние уровни представляют переменные.



Щелкните дважды мышью на активной процедуре и появится или исходный текст или результат дисассемблирования исполняемого кода. Двойной щелчок мышью на переменной копирует ее в диалоговое окно Watch Expressions (выражения наблюдения).

Окно Активные процедуры показывает информацию только для текущей линии процесса.

### Другие окна

Другие окна отладки обеспечивают другие виды информации:

□ Окно Disassembly (Дисассемблер) является необязательным для проекта с информацией отладки: выберите Options Disassembly On для его показа.

Если вы прогоняете отладчик на программе без какой-либо информации об отладке, окно Дисассемблер автоматически покажет инструкции языка ассемблера. Выбрана текущая инструкция.



```

5397:0166 16          push    ss
5397:0167 8D 46 F8     lea     ax, [bp][-8]
5397:016A 50          push    ax
5397:016B 9A DC 1C FF 2A call    far 0036/2AFF:1CD
5397:0170 83 C4 18     add     sp, 18H
5397:0173 8D 46 DC     lea     ax, [bp][-24H]
5397:0176 8C D3       mov     bx, ss
5397:0178 9A B8 0F 0F 61 call    far 0023/610F:0FB
5397:017D 85 C0       test    ax, ax
5397:017F 74 03       je      184H
5397:0181 E9 D7 02     jmp     near 45BH
5397:0184 B8 67 54     mov     ax, 5467H
5397:0187 8E C0       mov     es, ax
5397:0189 8C 86 D5 F1   mov     [bp][-0E2BH], es
5397:018D 26 A1 11 00   mov     ax, es:[11H]
5397:0191 26 8B 1E 13 00 mov     bx, es:[13H]

```

Двойной щелчок мышью (или нажатие клавиши ввода) на строке в окне Дисассемблер, которое содержит инструкцию перехода или вызова, передвигает курсор к месту размещения цели. Нажатие клавиши ESC возвращает курсор в первоначальное место.

Клавиша INS добавляет точку безусловного прерывания в позиции курсора. DEL удаляет точку прерывания. Нажатие SPACE BAR (пробела) показывает диалоговое окно Break Point (точка прерывания).

☐ Окно Machine Registers (Машинные регистры) показывает величины текущих регистров; для вызова окна выберите Window → Registers.

Окно Машинные регистры показывает регистр в левом столбце, а его содержимое - справа.

☐ Окно Library States (Библиотечные состояния) показывает возвращаемые величины функций библиотеки для вызова окна выберите Window → Library State. Эти функции представляют все полевые и другие события. К функциям относятся ACCEPTED, SELECTED, FIELD, CUS, FIRST-FIELD, LASTFIELD, ERRORCODE, и ERRORFILE.

```

AX: 0
BX: 0
CX: 0
DX: 241F
SI: 0
DI: 5467
DS: 535F
ES: 5467
SS: 241F
SP: B4F8
BP: C34A
CS: 5397
IP: 1B8
f1: 246
fp(0): 0.700000
fp(1): 0.200000

```

Clarion;

Э Т И М  
F O -

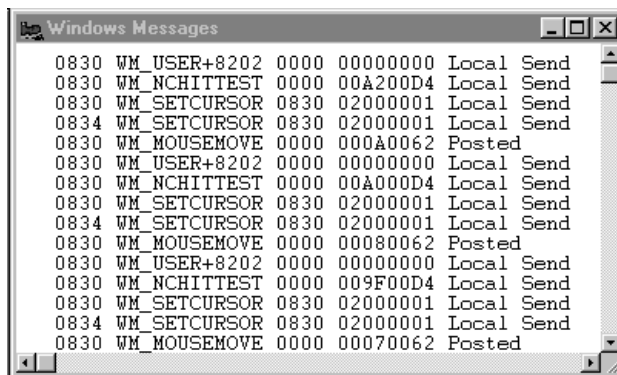
```

Library State
ACCEPTED : 0
SELECTED : 0
FIELD : 0
FIRSTFIELD : 0
LASTFIELD : 0
EVENT : 0
ERRORCODE : 0
KEYCODE : 0
MOUSEX : 0
MOUSEY : 0
FOCUS : 0
THREAD : 2

```

Имена, перечисленные в EQUATES.CLW и KEYCODES.CLW, появляются рядом с возвращаемыми величинами.

□ Окно Windows Messages (Сообщения Windows) показывает до 200 самых последних сообщений о событиях, сгенерированных или направленных вашему приложению; для вызова окна выберите Window Messages.



Отладчик добавляет строку-разделитель (“—”) для показа возникшей точки прерывания.

Каждое действие, предпринятое пользователем - от перемещения мыши до команд меню - сначала обрабатывается Windows. Если Windows решает, что действие относится к вашему приложению, он передает информацию вашему приложению с помощью сообщения. Например, если пользователь набивает букву “A”, он направляет сообщение WM\_KEYDOWN вашему приложению с ключевым кодом для “A” в качестве первого параметра сообщения.

**Совет: Если вы включили услуги DDE в ваше приложение, мы рекомендуем проверить вашу программу с помощью другого DDE приложения, контролируя сообщения этого DDE. Более полную информацию вы получите в Microsoft Windows 3.1 Programmers Reference, Volume 3, распространяемый Microsoft Press.**

## Установка точек прерывания

Обыкновенно, при отладке вашей программы вы будете идентифицировать ту небольшую часть программы, которая дает неточный результат или вообще терпит крах. Процесс отладки для такой ситуации потребует, вероятно, прогона как раз этой части программы и остановки ее в одной или более точках для проверки ее состояния.

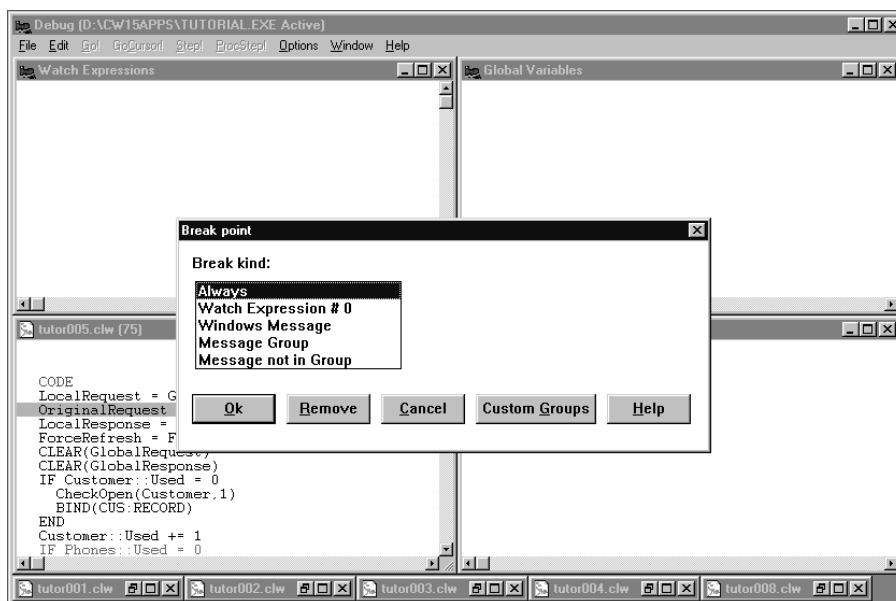
Точки прерывания позволяют автоматически останавливать исполнение на строке кода, в которой (или около которой), как вы думаете, возникла ваша проблема. Ваша программа идет до точки прерывания, затем останавливается и возвращает управление отладчику. Вы можете затем проверить содержание переменных и выражений для идентификации причины проблемы.

Вы можете также установить условия для точки прерывания, говоря программе продолжать исполнение, если условие ложно, или вернуть управление отладчику, если справедливо.

Когда вы устанавливаете точку прерывания, строка исходной программы с точкой прерывания становится лиловой (magenta) в окне источника.

### **Безусловные точки прерывания**

Безусловная или “липкая” точка прерывания помещена на строку исходной программы и останавливает ее выполнение каждый раз, когда программа встречает этот оператор:



Установка точки прерывания. Отладчик будет всегда останавливаться в этой точке прерывания.

1. Откройте окно исходной программы или дисассемблированной процедуры.
2. Установите строку кода, где необходимо вставить точку прерывания и щелкните на ней дважды мышью.

Появляется диалоговое окно Break Point (точка прерывания).

3. Выберите Always (всегда) и нажмите кнопку OK.

Когда вы введете команду Go! (выполнить!), программа будет выполняться до тех пор,

пока она не достигнет точки прерывания, в этот момент она остановится.

**Совет:** Когда активным окном является окно исходной программы или окно дисассемблера, нажмите клавишу **INSERT** для добавки безусловной точки прерывания или клавишу **DELETE** для удаления ее, если она уже размещена.

### Условные точки прерывания

Чтобы сузить поиск ошибок, вы можете дать указание отладчику делать останов программы только тогда, когда существуют определенные условия. Условие принимает форму выражения, которое может включать программные переменные, операторы и константы. Вы можете также приказать отладчику выполнить останов при совершении некоторого события в программе или при получении сообщения для приложения из Windows.

☐ Чтобы создать условную точку прерывания, реагирующую на изменение значения переменной или выражения:

1. Сформулируйте выражение наблюдения в соответствии с правилами, описанными в разделе Редактирование выражений наблюдения, приводимом ниже.

2. Установите строку, в которой нужно создать точку прерывания и щелкните на ней дважды мышью.

3. Когда появится диалоговое окно Break Point (точка прерывания), выберите Watch Expression #0 (выражение наблюдения).

4. Введите номер выражения наблюдения (из окна выражений наблюдения) в поле Watch#0 (наблюдать#0).

5. Нажмите кнопку ОК.

Когда вы выполняете команду Go! (ход!), программа работает до тех пор, пока она не достигнет точки прерывания, далее отладчик оценивает выражение наблюдения и затем останавливает выполнение программы, если выражение справедливо, т.е. оценка дает ненулевой результат.

Например, если переменная X должна иметь максимальную величину 999, но тем не менее возрастает до 1000, вызывая нарушение работы программы, вы можете приказать отладчику прервать выполнение при 999, перейти к пошаговому режиму работы, чтобы увидеть, когда и как переменная X достигает 1000.

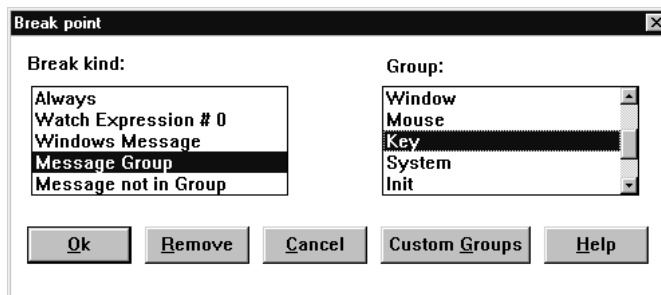
☐ Чтобы установить точку прерывания с реакцией на заданное сообщение Windows:

1. Установите строку кода, в которой необходимо создать точку прерывания, и щелкните на ней дважды мышью.
2. Когда появится диалоговое окно Break Point (точка прерывания), выберите Windows Message (сообщение Windows).
3. Выберите сообщение Windows из комбинированного поля.
4. Нажмите кнопку ОК.

Например, вы можете поместить точку прерывания в цикл, который проверяет сообщение WM\_RBUTTONDOWN, которое посылает Windows, когда пользователь щелкает правой клавишей мыши в вашем окне. Когда выполняется ваша программа, и вы нажимаете правую клавишу мыши внутри окна этой программы, Windows посылает сообщение WM\_RBUTTONDOWN вашему приложению. В этот момент условие точки прерывания будет справедливо.

☐ Чтобы установить условную (или нет) точку прерывания, реагирующую на получение одного из нескольких сообщений Windows:

1. Установите строку кода, в которой необходимо создать точку прерывания и щелкните на ней дважды мышью.
2. Когда появится диалоговое окно Break Point (точка прерывания), выберите Message Groupe (группа сообщений) или Message not in Groupe (сообщение вне группы).



Установка точки прерывания на получение любого связанного с KEY сообщением из Windows.

3. Выберите группу сообщений из списка. Это может указывать категорию сообщений, такую, как сообщения мыши или клавиатуры. Вы можете также установить заказную группу сообщений (нажатием кнопки Custom Groups (заказные группы) , чтобы запомнить несколько особых сообщений, так что точка прерывания возникнет только на одном из этих сообщений.

#### 4. Нажмите ОК.

Например, вы можете поместить точку прерывания в последовательность операторов, которая проверяет сообщения от клавиатуры. Если при выполнении вашей программы вы нажимаете какую-либо клавишу на клавиатуре, Windows посылает сообщение вашему приложению и условие точки прерывания будет справедливым.

□ Чтобы установить точку прерывания, которая прерывает работу программы, когда вы получаете незапланированное ранее, неожиданное сообщение, то есть, сообщение, которое не принадлежит к выделяемой вами группе:

1. Установите строку кода, в которой необходимо создать точку прерывания, и щелкните на ней дважды мышью.

2. Когда появится диалоговое окно Break point, выберите Message not in Group (сообщение вне группы).

3. Выберите группу сообщений из комбинированного поля. Точка прерывания возникает только тогда, когда приложение получает сообщение не из этой группы.

#### 4. Нажмите кнопку ОК.

### Выполнение программы

---

Команды Go!, GoCursor!, Step! и ProcStep! исполняют ваше приложение, в то время как отладчик контролирует его на заднем плане. Они позволяют вам испытать ваше приложение в контролируемой среде, которая помогает вам быстрее идентифицировать ошибки.

**Совет:** Эти команды являются командами верхнего уровня меню. Ниже их не появляется никакого выпадающего меню; для выполнения команды просто поместите курсор на команду меню и щелкните мышью или нажмите ALT плюс подчеркнутая буква.

**Go! (выполнить)**      Чтобы прогнать программу из ее текущего состояния к следующей точке прерывания, выберите команду Go!

Когда окно исходного текста или дисассемблера активно, клавиша G исполняет команду.

**GoCursor! (выполнить до курсора)**

Чтобы прогнать программу от ее текущего состояния к выбранной строке исходного или ассемблерного текста в окне исходной программы или дисассемблера, выберите GoCursor!

Когда окно исходного текста или дисассемблера активно, клавиша C исполняет команду.

Step! (пошаговое выполнение)

Чтобы продвигать выполнение программы от текущей выбранной строки исходного или ассемблерного текста, по одной строке кода за один раз, выберите Step!

Когда окно исходного текста или дисассемблера активно, клавиша S исполняет команду.

ProcStep!

Чтобы продвигать выполнение программы от текущей выбранной строки исходного или ассемблерного текста, по одной строке кода за один раз, не останавливаясь внутри процедур, выберите ProcStep!

Когда окно исходного текста или дисассемблера активно, клавиша P исполняет команду.

## Работа с исходной программой

---

Когда активно окно исходной программы, вы можете путешествовать по тексту исходной программы с помощью меню редактирования Edit.

**Совет: Двойной щелчок мыши на строке текста исходной программы, содержащей вызов в процедуру, приводит вас к первой строке этой процедуры. ESC возвращает вас обратно.**

Имеются следующие команды:

Find Text (найти текст)	Найти местоположение строки, которая содержит текст, вводимой вами в диалоговом окне Find Text (найти текст).
Find Next (найти следующую)	Найти местоположение строки, которая содержит текст, разыскиваемый ранее вами с помощью команды Find Text.
Find Procedure (найти процедуру)	Находит местоположение первой строки исходной программы, содержащей процедуру, которую вы отобрали из диалогового окна Find Procedure (найти процедуру). Текст процедуры появится в комбинированном поле внутри диалога.
Goto Line (перейти к строке)	Передвигает курсор к строке, номер которой вы установили.

**Current Line**

(текущая строка) Передвигает курсор к строке исходной программы, которая содержит следующий исполняемый оператор.

**Find Last Error (найти**

последнюю ошибку) Помещает курсор на последнюю ошибку.

Эта команда будет работать после большинства ошибок General Protection Fault. Курсор появится на строке исходной программы, где была обнаружена ошибка времени исполнения, или на строке, содержащей вызов вызвавшей проблемы функции.

**Break Points (точки прерывания)**

Показывает диалоговое окно Breakpoints (точки прерывания), который дает список точек прерывания, которые вы установили для данной сессии отладки. Точки прерывания появляются в формате SourceModule: Procedure: Line Number. Выберите точку прерывания из списка, затем нажмите одну из следующих кнопок: Locate, Delete, Edit, OK или Help..

**Locate (найти)**

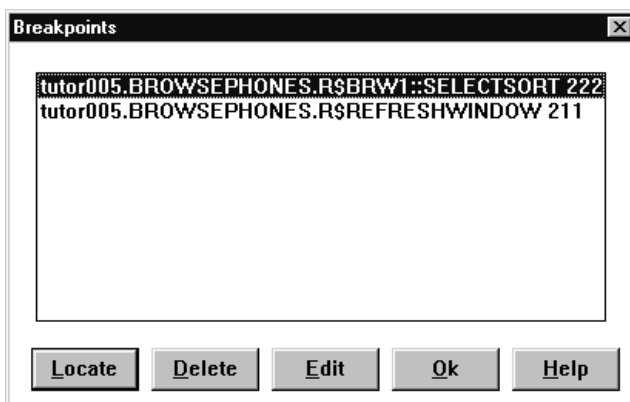
Прокручивает окно исходного текста до строки, содержащей точку прерывания.

**Delete (удалить)**

Удаляет точку прерывания.

**Edit (редактировать)**

Вызывает диалоговое окно Breakpoint, смотрите выше Установка точек прерывания.



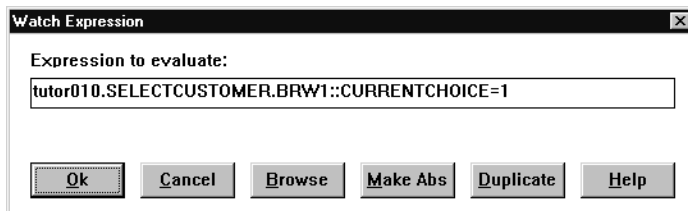
## Редактирование выражений наблюдения

Отладчик содержит диалоговое окно редактирования выражения, которое позволяет редактировать выражение наблюдения. Иногда вам нужно предпринять действие, зависящее от величины выражения, которое использует переменные из вашей программы. Например, вы можете захотеть остановить приложение и посмотреть на переменную, не является ли она отрицательной величиной, или продолжить до следующей точки прерывания, если она



положительна.

Чтобы отредактировать выражение наблюдения, выберите строку в диалоговом окне Watch Expressions (выражения наблюдения) и выберите Edit Edit. Появится диалоговое окно Watch Expression.



Введите выражение в поле Expression to Evaluate (оценить выражение), затем нажмите кнопку ОК. Когда отладчик будет выполнять программу, то по достижении точки прерывания он проверит значение этого выражения и остановит программу, если выражение дает оценку TRUE.

Диалоговое Окно Edit Expression (редактировать выражение) также содержит кнопку Browse (просмотр), предназначенную для того, чтобы создавать ваше выражение быстро и аккуратно. Нажмите кнопку Browse (просмотр), чтобы увидеть список переменных, используемых в отлаживаемой в данный момент процедуре.

Кнопка Make Abs автоматически предпосылает имена переменных их адресами памяти, именами модуля и именами процедуры.

Кнопка Duplicate (дублировать) создает дубликатное выражение наблюдения, которое появляется в диалоговом окне Выражения наблюдения.

Вы можете поставить префиксом для имени переменной имя процедуры и/или имя модуля. Это позволяет вам исследовать/использовать переменную, не находящуюся в поле зрения, например, переменную из другой процедуры, которая не будет видима для текущей процедуры.

□ Чтобы установить процедуру и переменную, используйте в качестве префикса переменной имя процедуры, отделенное от имени переменной точкой (“.”).

Например, “RoyalFlush.King” ссылается на переменную, называемую King в процедуре, называемой RoyalFlush.

□ Чтобы установить модульную и глобальную переменную, используйте в качестве префикса переменной имя модуля, отделенное от имени переменной точкой (“.”).

Например, “NewDeal.Shuffled” ссылается на глобальную переменную, называемую Shuffled в модуле, называемом NewDeal.

□ Чтобы установить локальную переменную в процедуре в другом модуле, скомбинируйте префиксы.

Например, “Poker.RoyalFlush:King” ссылается на переменную, называемую King, в процедуре, называемой RoyalFlush в модуле, называемом Poker.

- Вы можете использовать имена регистров (например, ax) в выражении наблюдения.
- Вы можете использовать унарный оператор (@ ) для указания адреса объекта памяти.

**Совет: Отладчик предположит правильный префикс, если переменная уникальна.**

Следующий список представляет синтаксис операторов и выражений для диалога Edit Expression. Операторы не зависят от языка, и наследуют свойства операторов C/C++ и Modula 2/ Pascal.

Клавиша	Функция
+	add (прибавить)
-	subtract (вычесть)
*	multiply (умножить)
/ или DIV	divide (разделить)
%или MOD	modulus (remainder) (модуль)(остаток)
	bitwise OR (поразрядный ИЛИ)
&	bitwise AND (поразрядный И)
<	less than (меньше, чем)
<=	less than or equal to (меньше, чем или равно)
>	greater than (больше, чем)
>=	greater than or equal to (больше, чем или равно)
=	equal (равно)
!=or <>	not equal (не равно)
! or NOT	logical NOT (логическое НЕТ)
& or AND	logical AND (логическое И)
or OR	logical OR (логическое ИЛИ)
*	indirection (when prefix) (косвенность(когда префикс))
^	indirection (when post-fix) (косвенность (когда пост-фикс))
->	point at member(указатель)
.	select member (record file)
::={e,d}	display expression “e” as if it was the same type “d” (показывает выражение “e”, как если бы оно было того же типа, как “d”)

## Редактирование переменных во время прогона

Используя отладчик, вы можете изменить величину, содержащуюся в переменной памяти, когда программа остановлена. Вы можете затем возобновить программу, чтобы испытать ее работу с переменной, содержащей новую величину.

Чтобы изменить содержание переменной:

1. Выберите переменную либо в окне Global Variables (глобальные переменные), либо в окне Active Procedures (Активные процедуры).
2. Нажмите F2 или выберите Edit Edit.



Появится диалоговое окно Edit Variable (отредактировать переменную) .

3. Введите новое значение переменной и нажмите кнопку ОК.

Когда вы выбираете Go!, GoCursor!, Setup!, или ProcStep!, программа возобновляет работу с новым значением отредактированной переменной.

## 32-битовый отладчик

### Запуск отладчика

32-битовый отладчик работает как отдельное приложение, но вы можете запустить его либо внутри интегральной Среды разработки или прямо из Windows. Если вы начинаете из Windows, с незагруженной средой разработки, это означает, что больше ресурсов системы будет доступно для вашего приложения и отладчика. 32-битовый отладчик может отлаживать одновременно много программ.

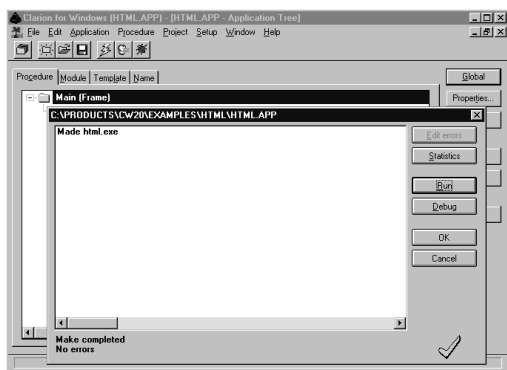
Чтобы запустить отладчик находясь внутри интегральной среды разработки, либо:

1. Выберите Project Debug или нажмите кнопку Debug (отладка) на панели инструментов.

Интегральная Среда разработки проверяет информацию проекта для того, чтобы определить, является ли ваше приложение 16-битовым или 32-битовым, и запускает соответствующий отладчик

либо...

2. Компилируйте и компоуйте ваше приложение, нажав кнопку Make (выполнить), затем, при еще открытом диалоговом окне результатов компилирования, нажмите кнопку Debug (отладка).



Два способа запуска отладчика.

□ Чтобы пустить отладчик из среды окон:

1. Переключитесь на диспетчер программ (Windows 3.x) или нажмите кнопку Start (старт) на линейке задач (Windows 95) и откройте программную группу Clarion.

2. Дважды щелкните мышью на 16-битовой пиктограмме (Windows 3.x) отладчика Clarion или выберите 32-битовый отладчик Clarion из меню программ (Windows 95).

3. С запущенным отладчиком выберите File File to Debug, затем выберите файл .EXE в диалоговом окне Open File (открыть файл).

**Совет:** При выполнении отладки выполняйте только отладчик и отлаживаемые программы. Поступая так, вы не потеряете данные в других приложениях в случае если крах случится во время процесса отладки.

## Загрузка исходных файлов

Исходная программа, связанная с программой отладки, автоматически загружается и становится доступной для вашего обследования. Однако вы можете назначить любые дополнительные файлы исходной программы, которые вы хотите видеть показанными на отладчике.

□ Чтобы назначить дополнительные файлы исходной программы:

1. Выберите Window Source.

Появится диалоговое окно Select Source (выбор исходной программы).

2. Выделите файл исходной программы и нажмите кнопку ОК.

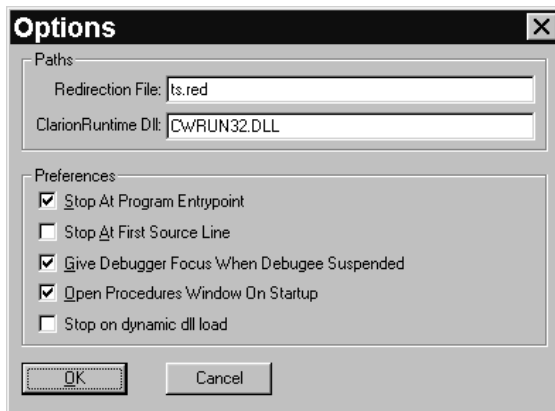
Повторите для каждого файла исходной программы, который вы хотите отладить

## Установка опций отладчика

Меню Options (параметры) отладчика предоставляет два выбора: Setup (установка) и Install as System Debugger (инсталлирование как системного отладчика). Используйте Setup для настройки отладчика.

### Setup

Выберите команду Options → Setup чтобы получить доступ к следующим опциям:



Redirection File

(файл переадресовки)

Отладчик использует файл переадресовки для нахождения компонент проекта. Файл переадресовки является необязательным и следует тем же условиям, что и файл переадресовки проекта. Смотрите главу Использование системы проекта, Редактирование файла переадресовки.

Clarion Runtime DLL

(динамическая библиотека

поддержки Clarion) Назначает динамическую библиотеку поддержки (DLL) Clarion, скомпонованную в отлаживаемый .EXE.

Stop At Program Entrypoint

(остановить в точке

ввода программы)

Сообщает отладчику о необходимости остановить программу отладки на ее точке ввода до загрузки начальной программы. Загрузка начальной программы (и старт) происходят тогда, когда вы выберете File to Debug и отберете файл .EXE из диалогового окна Open File (открыть файл).

Выбор этой опции позволяет вам наблюдать состояние вашей

программы на самой ранней стадии исполнения без явной установки точки прерывания.

Stop At First Source Line (остановиться на первой строке исходной программы)

Сообщает отладчику о необходимости остановить программу отладки на его первой строке исполняемой программы до загрузки начальной программы. Загрузка (и старт) начальной программы происходит тогда, когда вы выбираете File to Debug и отбираете файл .EXE из диалогового окна Open File.

Give Debugger Focus When Debuggee Suspended (дать фокус отладчику при остановке отлаживаемой программы)

Когда исполняемая программа приостановлена в точке прерывания, фокус немедленно возвращается к отладчику.

Open Procedure Window on Startup (открыть окно процедуры на старте)

Командует отладчику открыть окно Procedures In (процедуры в) на начальные действия отладчика. Смотрите ниже Окна отладчика.

Stop on dynamic DLL Load (остановиться при загрузке динамической библиотеки)

Сообщает отладчику о необходимости приостановить исполнение когда обнаруживается загрузка динамической библиотеки (DLL). Это дает вам возможность исследовать вновь загруженный код и установить точки прерывания прежде чем что-нибудь случится.

### **Установить как системный отладчик**

Инсталлирует 32-битовый отладчик как системный отладчик. В этой конфигурации отладчик автоматически вызывается, когда в программе случается крах.

### **Окна отладчика**

Отладчик состоит из набора дочерних окон, которые прослеживают для вас различную информацию об отлаживаемой программе. Этими окнами являются:

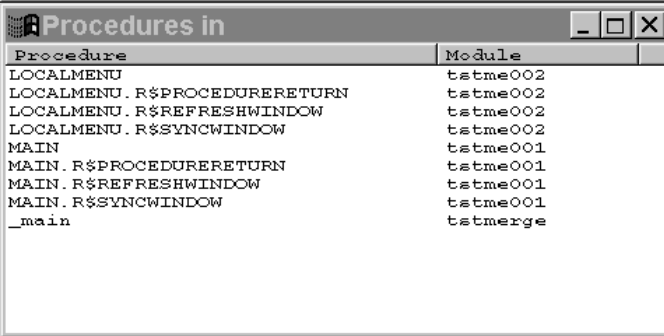
- ◆ Окно Procedures In (процедуры в)
- ◆ Окно Globals (глобальные)
- ◆ Окно Steck Trace (трасса стека)
- ◆ Окно source (исходная программа)
- ◆ Окно disassembly (дисассемблера)
- ◆ Окно memory (памяти)

После того, как вы запустили отладчик, остановитесь и настройте различные окна на удобный для вас формат. Расположите наиболее важные окна там, где вы можете быстро просматривать нужную вам информацию. Закройте или превратите в пиктограммы ненужные вам окна.

Используйте меню Window (окно) чтобы открыть окна, представляющие особый интерес. Сначала отладчик открывает три окна: окно Procedures In, окно Globals и окно Stack

Trace. Четвертое окно, окно исходной программы source, открывается как только вы щелкнете клавишей мыши на процедуре в окне Procedure In.

### Окно Procedure In

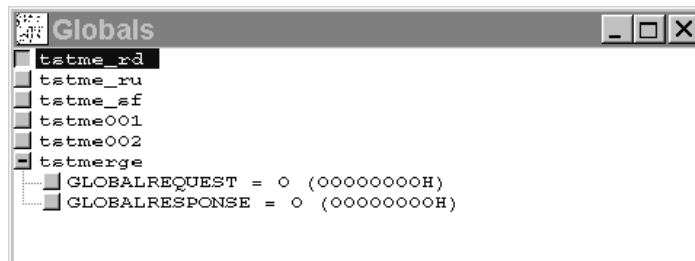


Procedure	Module
LOCALMENU	tstme002
LOCALMENU.R\$PROCEDURERETURN	tstme002
LOCALMENU.R\$REFRESHWINDOW	tstme002
LOCALMENU.R\$SYNCWINDOW	tstme002
MAIN	tstme001
MAIN.R\$PROCEDURERETURN	tstme001
MAIN.R\$REFRESHWINDOW	tstme001
MAIN.R\$SYNCWINDOW	tstme001
_main	tstmerge

Перечислите процедуры в отлаживаемой программе и их ассоциированные модули исходной программы. Щелкните клавишей мыши на имени процедуры для показа ее ассоциированной исходной программы или кода ассемблера.

**Совет: Используйте окно Procedure In для движения по вашей исходной программе.**

### Окно “Globals”



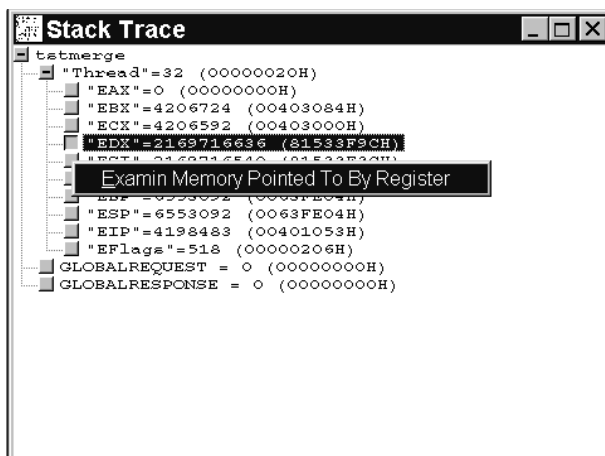
Показывает текущую величину каждой компоненты каждой глобальной переменной, а также состояние библиотеки. Щелкните правой клавишей мыши на переменной для изменения ее величины.

Окно Globals содержит элементы управления расширяемого дерева, так что вы можете спрятать переменные, которые вы не хотите видеть. Переменные с кнопкой (+) расширяются с помощью щелчка клавиши на них. Переменные с кнопкой (-) сокращаются при щелкании на них.

**Совет: Щелкните правой клавишей на переменной для изменения ее**

величины.

## Окно трассы стека “Stack Trace”



Показывает текущие величины регистров и величины локальных переменных. Имя переменной находится слева, а ее величина в десятичном формате, а затем в шестнадцатичном формате, - справа. Это информация только для текущей ветви процесса.

Окно Трасса стека содержит элементы управления расширяемого дерева, так что вы можете спрятать переменные, которые вы не хотите видеть. Переменные с кнопкой (+) расширяются с помощью щелчка клавиши на них. Переменные с кнопкой (-) сокращаются при щелкании на них.

**Совет:** Окно трассы стека имеет следующие специальные функциональные возможности:

Щелчок правой клавиши на переменной для изменения ее величины.

Щелчок правой клавиши на вызове для отыскания соответствующей строки исходной программы или строки ассемблера.

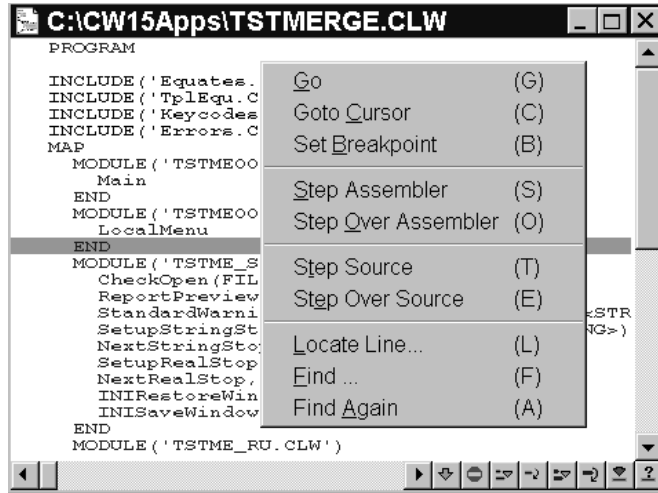
Щелчок правой клавиши на регистре для проверки памяти, назначенной для регистра.

## Окно Исходной программы

Показывается модуль исходной программы. Может быть много окон исходной



программы, показывающие различные модули. Линейка заголовка показывает имя модуля. Курсор - зеленый. Курсор просто отмечает для вас строку. Он может отмечать, а может и нет, текущую позицию программы. Строки прерывания - красные. Если текущая строка также является и строкой прерывания, она желтая.



Линейка задач окна исходной программы.

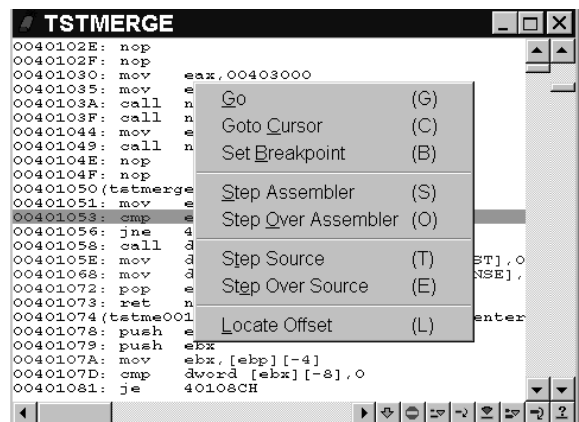
Используйте кнопки линейки задач в окне исходной программы для того, чтобы контролировать исполнение отладки и установить и удалить точки прерывания. Кнопки линейки задач соответствуют опциям на всплывающем меню, доступ к которому можно получить, щелкнув правой клавишей мыши в любом месте внутри окна. Описание каждой команды вы можете найти дальше в Исполнение программы.

**Совет:** Чтобы получить доступ к всплывающему меню, щелкните правой клавишей мыши в любом месте окна.

### Окно Дисассемблера

Линейка задач окна Дисассемблера.

Показывает код дисассемблера. Может много открытых окон Disassembly. Линейка заголовка показывает .EXE имя. Курсор - зеленый. Курсор просто отмечает для вас строку. Он может отмечать, а может и нет, текущую позицию программы. Строки прерывания - красные. Если текущая строка также является и строкой прерывания, она желтая.



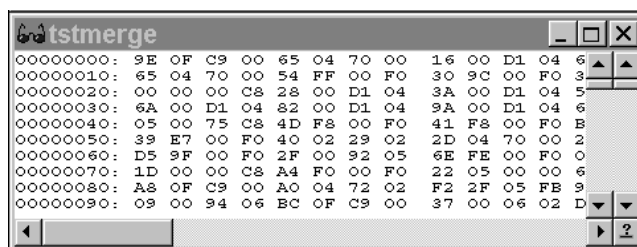
Синий текст имеет соответствующий оператор исходной программы, ассоциирующийся с ней. Перемещение курсора к строке с синим текстом перемещает курсор в окне исходной программы к соответствующей строке исходной программы.

Используйте кнопки линейки задач в окне Disassembly для того, чтобы контролировать исполнение отладки и установить и удалить точки прерывания. Кнопки линейки задач соответствуют опциям на всплывающем меню, доступ к которому можно получить, щелкнув правой клавишей мыши в любом месте внутри окна. Описание каждой команды вы можете найти дальше в Исполнение программы.

Окно Дисассемблера имеет две вертикальные линейки прокрутки. Левая линейка прокручивает за раз 64К кода, правая линейка - 1 строку показа за раз.

**Совет:** Чтобы получить доступ к всплывающему меню, щелкните правой клавишей мыши в любом месте окна.

### Окно памяти



Показывает память, распределенную для отлаживаемой программы. Линейка заголовка показывает .EXE имя. Окно Память имеет две вертикальные линейки прокрутки. Левая линейка прокручивает за раз 64К кода, правая линейка - 1 строку показа за раз.

## Назначение точек прерывания

Обыкновенно, при отладке вашей программы вы будете идентифицировать ту небольшую часть программы, которая дает неточный результат или вообще терпит крах. Процесс отладки для такой ситуации потребует, вероятно, прогона как раз этой части программы и остановки ее в одной или более точках для проверки ее состояния.

Точки прерывания позволяют автоматически останавливать исполнение на строке кода, в которой (или около которой), как вы думаете, возникла ваша проблема. Ваша программа идет до точки прерывания, затем останавливается и возвращает управление отладчику. Вы можете затем проверить содержание переменных для идентификации причины проблемы и пойти дальше от этой точки.

Когда вы устанавливаете точку прерывания, строка, где это точка возникает, появляется

в красном цвете в окнах исходной программы и дисассемблера.

**Совет: Точки прерывания появляются в желтом цвете когда вы впервые создаете их, так как уже имеются красный цвет точки прерывания и зеленый курсор.**

Чтобы назначить точку прерывания:

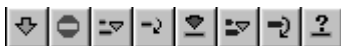
1. Пройдите к исходной программе или коду ассемблера, где вы хотите остановить отладчик.

Щелкните клавишей на имени процедуры в окне Procedures In и вы перепрыгните к этой процедуре. Или же щелкните правой клавишей в окне Исходная программа чтобы получить доступ к команде Find (найти) и найти текстовую строку символов.

2. Выделите строку кода для прерывания.

3. Нажмите кнопку прерывания.

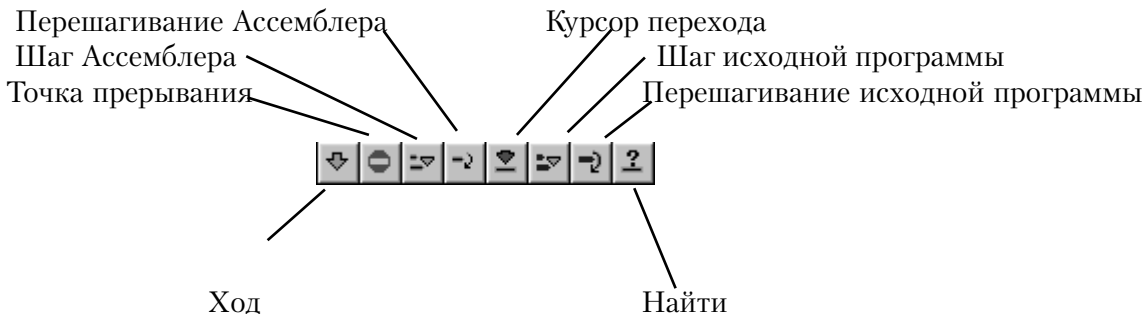
Появится кнопка точки прерывания на линейках задач окон Исходная программа и Дисассемблер, с красной круглой пиктограммой. Кнопка точки прерывания работает как переключатель. Нажатие ее второй раз устраняет точку прерывания.



Кнопка точки прерывания.

## Исполнение программы

Кнопки линейки задач на окнах Исходная программа и окна Дисассемблер контролируют исполнение вашей программы. Подобные линейки задач появляются на каждом окне Исходной программы и Дисассемблера. Не имеет значения, какую линейку задач использовать. Исполнение программы всегда продолжается от той точки, в которой она остановилась.



Другой способ: вы можете использовать всплывающие меню, доступные в каждом окне. Команды линейки задач дублированы на соответствующих всплывающих меню окон Исходной программы и Дисассемблера. Щелкните правой клавишей мыши в любом месте окна и вы получите доступ к всплывающему меню.

**Go! (выполнить)** Продвигает программу из ее текущего положения к следующей точке прерывания.

**Step Assembler (шаг ассемблера)** Продвигает программу из ее текущего положения, по одной строке кода ассемблера за раз.

**Step Over Assembler (Перешагивание ассемблера)** Продвигает программу из ее текущего положения к следующей точке прерывания ассемблера без исполнения каких-либо операторов между ними.

**Step Source (шаг исходной программы)** Продвигает программу из ее текущего положения, по одной строке исходного кода за раз.

**Step Over Source (Перешагивание исходной программы)** Продвигает программу из ее текущего положения к следующей точке прерывания исходной программы без исполнения каких-либо операторов между этими точками.

**GoCursor! (выполнить до курсора)** Продвигает программу из ее текущего положения к курсору. Это имеет своим эффектом то, что курсор становится на определенной на один раз временной точкой прерывания.

**Locate Line/Offset (найти строку/отступ)** Продвигает курсор (а не программу) к следующему установленному вами номеру строки (или отступу для ассемблера).

**Find (найти)** Продвигает курсор (а не программу) к назначенной вами строке символов исходной программы (только для окна исходной программы).

**Find Again (найти повторно)** Продвигает курсор (а не программу) к строке символов исходной программы, установленной для предыдущей команды Find (только

для окон исходной программы).

## Редактирование переменных во время работы

---

### Проверка переменных величин

Лучший способ проверки переменных величин во время работы - это посмотреть их либо в окне Глобальные, либо в окне Трасса стека. Глобальные переменные показаны в окне Глобальные, а локальные переменные - в окне Трасса стека, как в десятичном, так и в шестнадцатичном формате.

Оба окна содержат элементы управления дерева, поэтому вы можете расширить только те переменные, которые вы хотите проверить. Элементы управления, содержащие (+), расширяются с помощи щелчка клавишей по ним. Элементы управления, содержащие (-), сокращаются после щелчка клавиши по ним.

Окно Трасса стека также показывает величины машинного регистра и находит области памяти, на которые указывает регистр. Чтобы проверить точное размещение памяти в окне памяти, щелкните правой клавишей на регистре или выделите ее и нажмите ENTER.

### Изменение переменных величин

Щелкните правой клавишей на переменной или выделите ее и нажмите ENTER, либо в окне Глобальные, либо в окне Трасса стека, и вы можете менять ее величину.

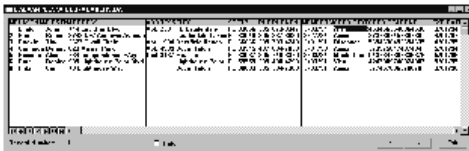
**Совет:** Для изменения величины переменной щелкните на ней правой клавишей.



## Глава 17 Менеджер базы данных



Менеджер базы данных был спроектирован для того, чтобы дать разработчикам приложений свободный доступ к их файлам данных. Единственное ограничение ввода - это шаблон показа, приписанный к столбцу. Элементы управления для обеспечения Целостности данных и Ссылочной целостности в вашем словаре данных и приложении не используются.



Используйте Менеджер базы данных для интерактивного просмотра ваших файлов данных. Вы можете добавить, модифицировать или удалить записи.



Используйте методику Query-Bu-Example (запрос-по-образцу) для нахождения записей, отвечающих некоторому специальному критерию.



Форматируйте список так, чтобы показывать только те поля, которые вы выбрали.



Используйте утилиту преобразования файла для того чтобы преобразовать существующие файлы данных в новый формат. Вы можете также генерировать исходную программу для получения исполняемой программы преобразования файлов данных для ваших конечных пользователей.

Менеджер базы данных - Обзор  
Просмотр файлов данных  
Закрытие файла данных  
Порядок сортировки  
Статистика файла  
Работа со столбцами  
Соккрытие столбцов  
Показ столбцов  
Использование команды Reformat  
Выравнивание столбца  
Ширина столбца  
Изменение шаблона показа столбца  
Изменение заголовка столбца  
Работа с файлами данных  
Движение по файлу  
Команда Locate (Key)  
Поиск и нахождение следующего  
Посылка строки символов драйверу  
Сохранение определения файла в виде текста  
Использование запроса-по-образцу  
Редактирование данных  
Редактирование записей  
Добавление записей  
Редактирование мемо-поля  
Показ удаленных записей  
Возвращение удаленных записей  
Удержание и освобождение записей  
Преобразование файла данных  
Немедленное преобразование  
Генерирование исходной программы для преобразования файла  
Редактирование исходной программы для выполнения присваиваний полей  
Преобразование законных данных  
Печать данных



## Обзор

Менеджер базы данных открывает вам прямой доступ к файлам данных без необходимости создания приложения. С помощью менеджера базы данных вы можете:

- ◆ Интерактивно просматривать свои файлы данных.
- ◆ Добавлять, удалять или изменять записи.
- ◆ Добавлять, удалять или изменять мемо-поля.
- ◆ Проверять файлы данных.
- ◆ Печатать данные.
- ◆ Сортировать данные.
- ◆ Использовать запрос-по-образцу для фильтрации данных.
- ◆ Искать данные.
- ◆ Преобразовывать файлы данных.

Нормально целостность данных обеспечивается в приложениях конечного пользователя проверками достоверности, установленными в словаре базы данных, позволяющими пользователю вводить любые достоверные данные в поле, к которому они относятся.

Ссылочная целостность обеспечивается в генерированных приложениях ограничениями на взаимоотношения, которые вы устанавливаете в словаре базы данных. Изменение величин в полях, которые связывают записи в двух файлах, или удаление родительской записи при существующих дочерних записях, могут подвергнуть опасности ссылочную целостность вашей базы данных. Это обсуждается ниже в разделе Использование редактора словаря.

## ***Просмотр файлов данных***

Имеется несколько способов просмотра файлов данных с помощью Менеджера базы данных.

- ◆ С помощью команды меню File > Browse FileLabel из редактора словаря.
- ◆ Выбирая File > Open (или нажимая кнопку Open button на списке Pick List).
- ◆ Выбирая File > Browse Database.

То, какой метод вы используете для вызова менеджера базы данных, влияет на его поведение. Если вы открываете файл через редактор словаря (при открытом соответствующем файле .DCT), менеджер базы данных использует всю информацию в словаре. Если вы открываете файл из какого-либо другого места, доступна только информация, сохраненная в самом файле. Это дает максимальную гибкость, позволяя вам сканировать файл без файла словаря данных (.DCT). Информация, хранящаяся в файле, зависит от типа файловой системы.

### **Из редактора словаря**

Это лучший метод вызова менеджера базы данных, так как он обеспечивает наибольшую информацию о файле.

1. Откройте подходящий файл словаря (.DCT).
2. В списке файлов Files выделите нужный файл.
3. Выберите File > Browse<FileLabel>.

<FileLabel> - это метка Clarion для поля, установленная в словаре. Меню File показывает этот выбор, основанный на выделенном файле.

Если этот файл не существует, появляется диалоговое окно, спрашивающее, не хотите ли вы его создать. С помощью Менеджера базы данных вы можете создать файл даже если у файла нет атрибута CREATE (создать) (поле флажков Разрешить создать файл в Свойствах файла).

Если файл существует, но не соответствует плану в словаре, появляется диалоговое окно, спрашивающее, не хотите ли вы преобразовать файл к текущему плану. Дополнительную информацию смотрите в Преобразование файлов данных.

Файл показан, он готов для любых операций Менеджера базы данных.

### **Из диалогового окна Открыть файл**

Чтобы открыть существующий файл данных:

1. Выберите File > Open (или нажмите кнопку Open (открыть) на списке Pick List (список указаний)).

Появляется диалоговое окно Open.

2. Выберите закладку Database (база данных).

3. Выделите файл, который вам нужно открыть, и нажмите кнопку Open.

Появится диалоговое окно с приглашением выбрать драйвер файла и файловой информации

4. Выберите драйвер базы данных из выпадающего списка.

5. По желанию установите имя Owner (владелец) и опции (Options).

Имя владельца является паролем для доступа к файлу. Для базы данных ODBC это Источник данных, идентификатор пользователя и пароль, отделенные запятыми.

Опции - это дополнительные инструкции для прохода к драйверу базы данных (строки символов драйвера). Смотрите приложение Драйверы базы данных, где вы можете найти дополнительную информацию о правильных строках символов драйверов для отдельных файловых систем.

6. Нажмите кнопку ОК.

Файл выбран и готов к любым операциям Менеджера базы данных.

### **Из меню команд окна просмотра базы данных**

Команда меню окна просмотра базы данных показывает специальный список указаний, показывающий недавно открытые файлы данных или логические таблицы в файлах базы данных, которые содержат многие файлы (ODBC и TopSpeed).

1. Выберите File > Browse Database.

Появится список указаний Менеджера базы данных, в котором приводятся недавно открытые файлы.

2. Выделите файл в списке и нажмите Select (выбрать) или нажмите кнопку Open чтобы выбрать файл из стандартного диалогового окна открывания файла.

3. Если вы открываете файл в первый раз, вы видите приглашение к определению драйвера файла. Выберите нужный вам драйвер, затем нажмите кнопку ОК.

4. Если вы просматриваете Источник данных ODBC или базу данных TopSpeed с многими таблицами в одном файле, появится диалоговое окно, которое позволит вам отобразить таблицу для просмотра.

Файл выбран и готов к любым операциям Менеджера базы данных.

## **Заккрытие файла данных**

Менеджер базы данных спрашивает вас, не хотите ли вы создать резервную копию перед тем как модифицировать какие-либо данные в файле. Создание резервного файла позволяет вам, если необходимо, отменить все изменения, которые вы произвели при сканировании файла. Однако некоторые драйверы файлов не поддерживают создание резервного файла. При использовании одной из таких файловых систем вы не получаете приглашение создать резервный файл.

**Внимание: Если вы не делаете резервной копии файла при модификации его, вы не сможете вернуть файл к его исходному состоянию.**

Чтобы закрыть файл:

1. Выберите File > Close.

Если вы модифицировали данные, появится диалоговое окно, спрашивающее, не хотите ли вы сохранить ваши изменения.

Yes (да) Сохраняет ваши изменения.

No (нет) Возвращает файл данных к его последнему сохраненному состоянию.

Cancel (отказ) Возвращает вас к менеджеру базы данных.

## **Порядок сортировки**

Когда файл открыт, вы можете изменить порядок сортировки, установив другой ключ.

1. Выберите Browse > Order (или нажмите CTRL+O).

Появится диалоговое окно Select File Order (выбор сортировки файла), дающий список имеющихся ключей и “Record Order” (порядок записей).

2. Выделите ключ, который соответствует желаемому порядку записей (или Порядок записей), затем нажмите кнопку Select (выбрать).

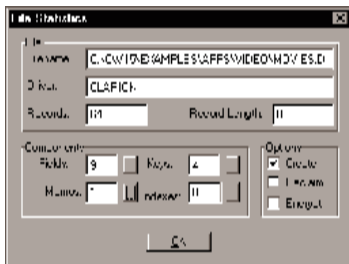
Появляется отсортированный в выбранном порядке файл, готовый для операций менеджера базы данных.

## Статистика файла

Команда File Statistics (статистика файла) позволяет вам исследовать информацию о файле, включая:

Filename (имя файла)	Имя файла DOS и PATH для файла данных.
Driver (драйвер)	Система файла, которую использует файл.
Records (записи)	Общее число записей в файле (включая удаленные записи).
Record Length (длина записи)	Размер записи.
Fields (поля)	Число полей в файле. Нажатие эллиптической (...) кнопки открывает схему полей.
Keys (ключи)	Число ключей в файле. Нажатие эллиптической (...) кнопки открывает компоненты ключей.
Memos (мемо)	Число мемо-полей (и BLOBs) в файле. Нажатие эллиптической (...) кнопки открывает схему поля мемо.
Indexes (индексы)	Число индексов в файле. Нажатие эллиптической (...) кнопки открывает компоненты индекса.
Options (опции)	Атрибуты Create, Reclaim и Encrypt.

Чтобы просмотреть статистику файлов:



1. Выберите File > File Statistics.

Появляется диалоговое окно File Statistics (статистика файла).

2. Для просмотра дополнительной информации о полях, ключах, мемо-полях или

индексах нажмите эллиптическую (...) кнопку рядом с соответствующим полем окна.

## ***Работа со столбцами***

Менеджер базы данных позволяет вам установить, какие столбцы (поля) вы хотите видеть на вашем экране, размер этих столбцов и порядок, в котором столбцы показываются.

### **Соккрытие столбцов**

---

Соккрытие столбцов делает выбранные столбцы невидимыми. Это никоим образом не влияет на файл данных. Пряча столбцы, вы можете просматривать только нужные вам столбцы данных.

1. Выделите столбец, который нужно спрятать.
2. Выберите Column > Hide (или нажмите CTRL+I).

Столбец исчезнет из поля зрения.

### **Показ столбцов**

---

Если столбцы спрятаны, вы можете восстановить их для просмотра или “открыть” их.

1. Выберите Column > Show.

Появится диалоговое окно Select Columns to Show (выделить столбцы для показа) со списком спрятанных полей.

2. Выделите нужное поле и нажмите кнопку ОК или нажмите кнопку All (все) для показа всех полей.

Снова появится диалоговое окно Show Fields (показать поля), позволяя вам выбрать другие поля для восстановления просмотра. Повторите последний шаг для любых других полей, которые вы хотите показать.

3. Когда вы показали все нужные вам файлы, нажмите кнопку Cancel.

### **Использование команды Reformat**

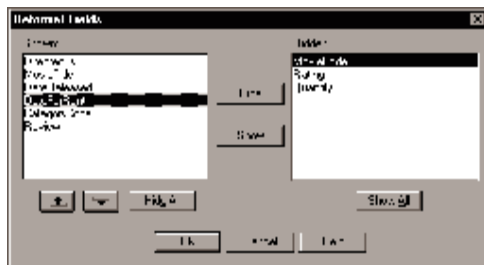
---

Команда Reformat позволяет быстро определить нужный формат просмотра. Вы можете установить поля, которые нужно спрятать или показать и одновременно установить порядок столбцов.

1. Выберите Column > Reformat.

Появится диалоговое окно Reformat Fields (изменить формат полей).

- ☐ Чтобы показать все поля, нажмите кнопку Show All (показать все).
- ☐ Чтобы спрятать все поля, нажмите кнопку Hide All (спрятать всё).
- ☐ Чтобы спрятать отдельные поля, выделите нужное поле в поле списка Shown Fields (показанные поля) и нажмите кнопку Hide (спрятать).
- ☐ Чтобы показать отдельные поля, выделите нужное поле в поле списка Hidden Fields (спрятанные поля) и нажмите кнопку Show (показать). Поле появится на своем первоначальном месте.



## Выравнивание столбца

Вы можете установить левое, правое, центральное или десятичное выравнивание для отдельных столбцов.

1. Выберите Column > Justify (или нажмите CTRL+J).

2. Выберите нужный тип выравнивания из выпадающего списка.

Left (левое)	Помещает начало показываемой величины на левом краю показываемого поля.
Center (центральное)	Центрирует показываемую величину в поле показа.
Right (правое)	Помещает конец показываемой величины на правый край поля показа.
Decimal (десятичное)	Выравнивает числовые данные по десятичной точке.

3. Нажмите кнопку OK.

## Ширина столбца

Вы можете настроить ширину показываемого столбца для каждого поля. Чтобы настроить ширину показа для одного поля:

1. Щелкните мышью и передвиньте линию сетки направо от столбца.

## Изменение шаблона показа столбца

Вы можете изменить шаблон показа любого столбца. Это дает вам возможность просмотреть данные в любом поддерживаемом формате.

Чтобы изменить показ столбца:

1. Выделите нужный столбец.
2. Выберите Column > Picture (или нажмите CTRL+P).
3. Наберите нужный шаблон в поле Picture (шаблон).

Данные показываются в установленном формате.

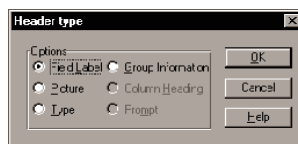
## Заголовки столбцов

Менеджер базы данных позволяет вам установить, что показывается в качестве заголовка столбца.

Чтобы установить заголовок столбца:

1. Выберите Column > Header.

Появляется подменю Header (заголовок), с действующими параметрами. Отметка около поля указывает, что данный параметр введен в действие.



Имеются следующие параметры заголовка:

Column Heading (заголовок столбца)	Показывает заголовок столбца по умолчанию из словаря базы данных.
Field Label (метка поля)	Показывает метку поля из словаря базы данных.
Picture (шаблон)	Показывает шаблон показа данных поля из словаря базы данных.
Type (тип)	Показывает тип данных поля из словаря базы данных.



Group Information (информация группы)	Показывает информацию GROUP поля из словаря базы данных.
Prompt (подсказка)	Показывает подсказку, принятую по умолчанию из словаря базы данных.

## Работа с файлами данных

Этот раздел описывает, как использовать менеджер базы данных для работы с файлами данных.

### Движение по файлу

Менеджер базы данных использует следующие соглашения об управлении с клавиатуры перемещением по файлу:

- ♦ В режиме просмотра ЛЕВАЯ СТРЕЛКА и ПРАВАЯ СТРЕЛКА перемещают между столбцами. В режиме редактирования ЛЕВАЯ СТРЕЛКА и ПРАВАЯ СТРЕЛКА перемещают между символами.
- ♦ СТРЕЛКА ВВЕРХ и СТРЕЛКА ВНИЗ, линейки прокрутки или VCR кнопки перемещают между записями.
- ♦ CTRL+ЛЕВАЯ СТРЕЛКА и CTRL+ПРАВАЯ СТРЕЛКА обменивают столбцы.
- ♦ Клавиши курсора HOME (первый) и END (последний) перемещают соответственно в первый и последний столбцы.
- ♦ СТРАНИЦА ВВЕРХ и СТРАНИЦА ВНИЗ прокручивают соответственно вверх и вниз по экранам записи.
- ♦ CTRL+СТРАНИЦА ВВЕРХ и CTRL+ СТРАНИЦА ВНИЗ перемещают к первой и последней записи.
- ♦ INSERT (вставить) дает вам возможность добавлять запись.
- ♦ DELETE (удалять) дает вам возможность удалить запись.
- ♦ В режиме просмотра ВВОД (ENTER) позволяет редактировать текущее выделенное поле на текущей записи. В режиме редактирования ВВОД принимает ваш ввод в текущее поле.

Менеджер базы данных обеспечивает также элементы управления VCR для перемещения по файлам:



## Команда Locate (Key)

Эта команда осуществляет поиск первой записи, содержащей величину, которую вы установили в ключевом поле (полях). Эта возможность доступна только тогда, когда файл данных показан в ключевой последовательности, а не в порядке номера записи. Эта команда ищет только те поля, которые являются компонентами выбранного ключа. Чтобы искать другие поля, используйте команду Search (поиск).

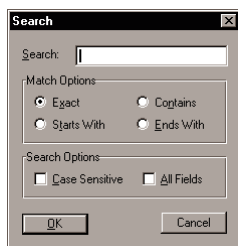
1. Выберите Edit > Locate.
2. Наберите нужную величину(ы) в поле (ях) Ключа .
3. Нажмите кнопку ОК.

Выделенная строка располагается на первом появлении установленной величины (величин) в поле (полях) ключа. Если введенная величина не существует, выделяется следующее верхнее совпадение.

## Поиск и нахождение следующего

Эта команда осуществляет поиск первой записи, содержащей величину, которую вы установили. Поиск может ограничиваться одним полем или всеми полями в записи. Вы можете искать:

- ◆ Точное совпадение.
- ◆ Запись с полем, начинающимся с установленной величины.
- ◆ Запись с полем, заканчивающимся установленной величиной.
- ◆ Запись с полем, содержащим величину где-либо внутри него.



Чтобы начать поиск:

1. Выберите Edit > Search (или нажмите кнопку ? VCR).
2. Введите искомую величину в поле Search For (поиск).
3. Выберите подходящие радиокнопки для нужного типа поиска.

Действующими вариантами поиска являются:

Exact match

(точное совпадение) Поиск величин, которые точно совпадают с заданной строкой символов.

Starts With

(начинается с) Поиск величин, которые начинаются с заданной строки символов.

Contains

(содержится) Поиск величин, которые содержат заданную строку символов.

Ends With

(заканчивается) Поиск величин, которые заканчиваются заданной строкой символов.

1. Если вы хотите вести поиск с учетом регистра, отметьте поле Case Sensitive (чувствительность к регистру).
2. Если вы хотите вести поиск с учетом всех полей, отметьте поле All Fields (все поля).
3. Нажмите кнопку ОК.

При поиске в больших файлах появляется диалоговое окно Search Status(состояние поиска), которое выводит информацию о протекании процесса поиска путем показа числа просмотренных записей и обеспечивает возможность прервать поиск. Когда поиск завершен, выделенная строка размещается на первой обнаруженной записи, которая соответствует критериям поиска.

Чтобы прервать идущий поиск:

1. Нажмите кнопку Cancel (отказ) на диалоговом окне Search Status(состояние поиска).

]Чтобы продолжить поиск:

1. Выберите Edit > Find Next (или нажмите CTRL+N).

Менеджер базы данных ищет в направлении вперед от текущей записи. Чтобы продолжить поиск, повторите последний шаг.

## **Посылка строки символов драйверу**

Менеджер базы данных позволяет вам общаться с драйвером файла, чтобы открыть файл. Это эквивалентно выдаче команды SEND (послать).

В Приложении Драйверы базы данных вы найдете реальные строки символов для каждого особого файла данных.

## Сохранение определения файла в виде текста

Менеджер базы данных позволяет экспортировать определение файла исходной программе Clarion. Этот код может быть впоследствии “вклеен” в другой кодовый сегмент или использован как часть процедуры исходной программы.

Чтобы экспортировать определение файла:

1. Выберите File > Save as Source.
2. Установите метку файла и имя файла для исходной программы.
3. Нажмите кнопку OK.

Создан программный текст содержащий описание файла.

## Использование запроса-по-образцу

Запрос по образцу (QBE) является мощным инструментом для поиска информации в файле данных. Это дает вам возможность задавать вопросы вашей базе данных, основанные на примерах желаемых результатов. Запрос по образцу фильтрует записи и показывает подмножество записей, основанное на определенном примере. Фильтр имеет форму выражения. Чаще всего это выражение будет сравнивать заданную величину с содержимым поля.

Вы устанавливаете свой запрос в поле списка QBE. Фильтр будет строится на базе выражения, введенного в это поле. Каждый столбец представляет поле, а каждая строка представляет логические группировки. Выражения, введенные в различные столбцы на одной и той же строке, оказывают влияние на оператор AND (И). Выражения в различных строках влияют на оператор OR (ИЛИ). Фильтрующее выражение показывается под полем списка при вводе выражения в поле списка.

Например, чтобы найти записи с числом ID между 10 и 100 и с фамилией Smith или Smythe, вы создаете запрос:

<u>ID</u> число	<u>Имя</u>	<u>Фамилия</u>
>10&<100		= 'Smith'
>10&<100		= 'Smythe'

Используйте символ (&) для представления оператора AND (И) и вертикальную линию (|) для представления оператора OR (ИЛИ) при использовании в одном и том же поле. Приведенные выше пример может быть представлен в таком стиле:

<u>IDчисло</u>	<u>Имя</u>	<u>Фамилия</u>
>10&<100		= 'Smith'   = 'Smythe'

Оба примера дают следующее фильтрующее выражение (IDчисло>10OR IDчисло<100)AND(Фамилия='Smith'ORФамилия='Smythe'). Выражение показывается под полем QBE.

**Совет:** Хотя выражение, созданное в запросе, неоптимизировано, блок вычисления программы QBE выполняет свою собственную оптимизацию. Таким образом, качество исполнения не страдает.

Запрос при выходе сохраняется в файле .INI. Следующий раз, когда вы откроете файл в менеджере базы данных, вы сможете отфильтровать записи с тем же фильтром QBE.

## Редактирование данных

Одна из первых функций менеджера базы данных - это способность обновлять записи без создания для этого процедур. Например, вы можете иметь необходимость создать файл из двадцати записей, например словарь, который вряд ли когда либо изменится. Вы можете использовать Менеджер базы данных для создания файла, ввести двадцать записей в файл данных и поставлять этот файл вместе с вашим приложением.

**Предупреждение:** Вы должны быть осторожны при введении изменений в файлы данных с помощью Менеджера базы данных. Это инструмент программирования для проверки и исправления файла данных, а не инструмент конечного пользователя для обращения с файлом данных. Нет механизмов, которые бы не позволили вам сделать изменения, которые могут подвергнуть риску целостность данных (неправильные величины данных) или ссылочную целостность (“сиротские” дочерние записи) вашей базы данных.

Менеджер базы данных спрашивает вас, не хотите ли вы создать резервную копию перед тем как модифицировать какие-либо данные в файле. Создание резервного файла дает вам возможность, если это необходимо, отменить все изменения, которые вы произвели при сканировании файла. Однако некоторые драйверы файлов не поддерживают создание резервного файла. При использовании одной из таких файловых систем вы не получаете приглашение создать резервный файл.

**Внимание:** Если вы не делаете резервной копии файла при модификации его, вы не сможете вернуть файл к его исходному состоянию.

## Редактирование записей

Вы можете легко отредактировать поле любой записи в менеджере базы данных.

1. Выделите нужное поле в нужной записи.

2. Выберите Edit > Change (или нажмите ENTER).

Вы можете теперь “редактировать по месту”. Новые данные вводятся в режиме вставки или в режиме замещения, в зависимости от последней использовавшейся установки.

3. Чтобы переместиться между полями, нажмите TAB, чтобы перейти к следующему полю или SHIFT+TAB, чтобы перейти к предыдущему полю.

Добавление записей

## **Менеджер базы данных дает вам возможность вводить новые записи в файл.**

---

1. Выберите Edit > Insert (или нажмите INSERT)

Внизу поля списка добавляется новая запись. Курсор находится в позиции Режим редактирования в первом поле .

2. Введите нужную вам информацию для занесения ее в поле файла, затем нажмите TAB для перехода к следующему полю (или SHIFT+TAB для перехода к предыдущему полю). Повторите для всех полей, в которые вы хотите ввести данные.

3. Когда вы закончили вводить все поля записи, нажмите ESC.

Запись будет добавлена к файлу, а ключи будут обновлены.

## **Редактирование мемо-полей**

---

Вы можете отредактировать мемо-поле в формате ASCII текста или в шеснадцатиричном формате (Hex Mode). Редактирование в Hex Mode полезно для мемо-полей, которые содержат двоичные данные.

Чтобы отредактировать мемо-поле в текстовом режиме:

1. Выделите запись.

2. Выберите Edit > Edit Memo (или нажмите CTRL+E).

3. Если файл имеет более одного поля мемо, появится поле списка. Выберите подходящее поле мемо.

4. Отредактируйте мемо.

Чтобы отредактировать мемо в режиме HEX:

1. Выделите запись.

2. Выберите Edit>Hex Edit Memo (или нажмите CTRL+X).

3. Если файл имеет более одного поля мемо, появится поле списка. Выберите подходящее поле мемо.

4. Отредактируйте мемо-поле.

## Показ удаленных записей

По умолчанию показываются только активные записи; однако вы можете показать и удаленные записи. Вы можете использовать эту возможность для просмотра недавно удаленных записей или для восстановления удаленных записей.

Вы можете посмотреть удаленные записи в любом файле, который не имеет атрибута RECLAIM. Если у файла есть атрибут RECLAIM, вы еще можете посмотреть удаленные записи, если новая запись не была добавлена вместо нее.

1. Выберите Window > Show Deleted.

Появляется отметка рядом с данной позицией меню, которая сигнализирует, что удаленные записи показаны.

Когда удаленная запись выделена, внизу окна показывается слово Deleted (удалено).



## Возвращение удаленных записей

Вы можете восстановить удаленную ранее запись в любом файле - если файловая система поддерживает это и если файл не имеет атрибута RECLAIM. Если файл имеет атрибут RECLAIM, вы еще можете восстановить удаленную запись, если к этому моменту это место в файле не заняла добавленная новая запись.

1. Убедитесь, что ваше окно просмотра включает удаленные записи (выберите Window > Show Deleted).

2. Выберите Edit > Undelete (или нажмите CTRL+DELETE).

## Удержание и освобождение записей

---

Удержание записи блокирует доступ к записи в среде многих пользователей. В общем случае исключается только обновление записи другими пользователями, для чтения запись остается доступной. Конкретное выполнение операции HOLD зависит от драйвера файла.

Чтобы удержать выделенную запись:

1. Выберите Edit > Hold (или нажмите CTRL+H).

Чтобы освободить выделенную задержанную запись:

1. Выберите Edit > Release (или нажмите CTRL+R).

Когда задержанная запись выделена, поле Held под полем списка отмечено.

Вы можете также удержать или освободить выделенную запись путем установки или опускания флажка в поле Held.

## Преобразование файла данных

Утилита преобразования файла дает вам возможность преобразовать записи в существующем файле данных в файл нового формата. Например, если вы модифицируете словарь данных и приложение, вы можете использовать утилиту преобразования, чтобы преобразовать ваши существующие данные к новому формату.

Утилита преобразования файла предоставляет вам два метода преобразования:

- ◆ Немедленное преобразование, а также
- ◆ Генерирование исходной программы для преобразования файла..

**Совет: Всегда хорошо иметь резервные копии ваших файлов перед исполнением любого процесса преобразования.**

## Немедленное преобразование

---

Немедленное преобразование обслуживает простые преобразования (тот же драйвер файла, те же имена поля, новые или измененные поля и ключи) между файлами с одной и той же файловой системой. Преобразование выполняется с помощью Менеджера базы данных Clarion for Windows. Редактор словаря должен быть тоже активным.

**Внимание: Если вы измените имя поля или если вы меняете драйверы файла, вы должны генерировать исходную программу и редактировать эту исходную программу для того, чтобы выполнить присвоение для полей.**

1. Откройте словарь, описывающий файл, который вы хотите преобразовать.



2. Модифицируйте определение файла данных по вашему желанию (добавьте или измените поля или ключи).

3. При выделенном модифицированном файле выберите File > Browse Filelabel для того, чтобы загрузить файл данных в Менеджер базы данных.

Появится сообщение, предупреждающее, что физическая структура файла не соответствует объявлению файла.

4. Чтобы преобразовать файл, нажмите кнопку Yes(да).

Процесс преобразования теперь закончен и Менеджер базы данных показывает содержимое файла.

### **Генерирование исходной программы для преобразования файла**

Генерирование исходной программы для преобразования файлов создает файл исходной программы и файл проекта, так что вы можете выполнять более сложные преобразования, вы также можете прогонять программу преобразования так часто, как вам это нужно. Генерирование и компилирование исходной программы также создает исполняемый файл, который вы можете поставить для конечных пользователей, чтобы они могли преобразовать свои файлы данных в новый формат. Исходная программа генерируется Менеджером базы Clarion for Windows.

1. Откройте словарь данных, описывающий файл, который вы хотите преобразовать.

2. Скопируйте определение файла данных на новое имя. Чтобы скопировать определение файла, выделите файл, который нужно скопировать, в списке файлов Files List и нажмите CTRL+C, затем нажмите CTRL+V чтобы вставить его из буфера. Вам поступит приглашение дать новое имя и префикс. (Например - копировать Покупатель в СтарыйПокупатель).

Альтернативным методом будет копирование всего словаря данных на новое имя. Вы можете использовать этот метод если имеется несколько файлов, которые нужно преобразовать в одном сеансе.

3. Выполните любые нужные изменения (добавьте поля, измените тип драйвера файла и т.д.) для определения файла с первоначальным именем. В приведенном выше примере файл Покупатель - это тот файл, который должен быть модифицирован.

4. Закройте словарь и сохраните его.

**Внимание: Файл словаря должен быть закрыт, если его необходимо использовать для преобразования. Загрузите файл в менеджер базы данных**

5. Выберите из меню File > Open.

6. Выберите закладку Database (база данных).

7. Перейдите к файлу, подлежащему преобразованию и затем выберите его. Вам поступит приглашение назначить драйвер файла.

8. Отберите драйвер файла из ниспадающего списка.

Менеджер базы данных открывает файл и показывает его содержимое.

9. Выберите File > Convert (или нажмите CTRL+V).

Появится диалоговое окно File Convert (преобразовать файл) , приглашающее к следующей информации:

10. В поле Target Filename (имя целевого файла) назначьте имя нового файла.

По умолчанию это имя текущего файла. Назначьте иное имя, если вы не создали резервной копии вашего файла данных.

11. В поле Target Dictionary (целевой словарь) назначьте словарь, который содержит определение файла, в который должно быть выполнено преобразование.

12. В поле Target Structure (целевая структура) назначьте структуру в словаре целевого файла, которая определяет целевой файл.

13. В поле Generated Source (генерированная исходная программа) назначьте имя файла для генерированной исходной программы или примите по умолчанию CONVERT.CLW.

**Внимание: Утилита преобразования пишет CONVERT.CLW в подкаталог активного приложения или файла проекта, а не обязательно в подкаталог, содержащий данные или словарь данных.**

14. Нажмите кнопку ОК.

Это генерирует файл исходной программы и файл проекта (.PRJ). Этот файл будет теперь скомпилирован и скомпонован в исполняемую программу, которая выполнит преобразование файла.

15. Нажмите кнопку Exit (выход) для того, чтобы закрыть файл данных в Менеджере базы данных.

**Совет: Перед исполнением сгенерированной программы преобразования вы должны закрыть файл данных, открытый в Менеджере базы данных.**

16. Загрузите программу преобразования, выбрав для этого File > Open и отберите закладку Source (исходная программа).


17. Выберите CONVERT.CLW (или установленное вами имя файла) В диалоговом окне File Open (открыть файл).

18. Отредактируйте исходную программу в соответствии с требованиями присваивания полей.

Смотрите Редактирование исходной программы для выполнения присваиваний полей.

19. Выберите Project > Set для загрузки файла проекта.

Перейдите к файлу проекта и выберите его. По умолчанию это CONVERT.PRJ.

20. Нажмите  для выполнения и прогона программы преобразования.

После того, как программа преобразования сработает:

21. Проверьте файл, который только что был преобразован, загрузив его в Менеджер базы данных.

После просмотра преобразованного файла все, что вам остается сделать, это выполнить некоторую приборку:

22. Удалите “старое” определение файла из активного словаря или сархивируйте его в резервный файл словаря .

23. Если преобразованный файл помещен в другой каталог, вы можете теперь скопировать его в каталог работающей программы. Если вы переименовали файл, вы теперь можете вернуть ему первоначальное имя.

Процесс преобразования на этом завершен. Этот пример создает CONVERT.EXE, который вы можете быть направите конечным пользователям для преобразования их файлов.

## ***Редактирование исходной программы для выполнения присваиваний полей***

Утилита преобразования файла создает исходную программу для преобразования файла к другой спецификации. Преобразование выполняется автоматически, за исключением следующих случаев:

- ◆ Если изменена метка поля
- ◆ Если поле расщеплено на два самостоятельных поля.
- ◆ Если скомбинированы два поля или более.
- ◆ Если данные преобразованы в стандарт данных Clarion из некоторого другого формата данных файловой системы.

В этих случаях вы должны модифицировать исходную программу для того, чтобы выполнить присваивания поля. Часть исходной программы, которую вам следует исследовать - это AssignRecord ROUTINE. Именно здесь выполняется присваивание поля.

Вот пример:

```
AssignRecord      ROUTINE
  CLEAR(CUS:Record)
  CUS:NUMBER = IN::NUMBER
  CUS:FIRSTNAME = IN::FIRSTNAME
  CUS:LASTNAME = IN::LASTNAME
  CUS:ADDRESS = IN::ADDRESS
  CUS:CITY = IN::CITY
  CUS:STATE = IN::STATE
  US:ZIP = IN::ZIP
  CUS:PHONENUMBER = IN::PHONENUMBER
  CUS:ENTRYDATE = IN::ENTRYDATE
```

Если вы проверите исходную программу, вы увидите, что первая строка в подпрограмме очищает буфер записи. Каждое следующее поле в выходном файле присваивает величину из соответствующего поля во входном файле.

### **Изменение метки поля**

Однако если метки поля не совпадают, не делается никакого присвоения. Например, если вы меняете поле Последнее Имя (LastName) на Фамилию (Surname), утилита преобразования файла генерирует оператор комментария, предупреждающий вас о присвоении, которое вам нужно выполнить:

```
AssignRecord      ROUTINE
  CLEAR(CUS:Record)
  CUS:NUMBER = IN::NUMBER
  CUS:FIRSTNAME = IN::FIRSTNAME
  ! CUS:SURNAME = ‘ ‘
  CUS:ADDRESS = IN::ADDRESS
  CUS:CITY = IN::CITY
  CUS:STATE = IN::STATE
  CUS:ZIP = IN::ZIP
  CUS:PHONENUMBER = IN::PHONENUMBER
  CUS:ENTRYDATE = IN::ENTRYDATE
```

Чтобы присвоить величины из первоначального файла, отредактируйте строку, содержащую присвоение для того, чтобы присвоить величину LastName к полю SurName как показано ниже:

```
CUS:SURNAME = IN::LASTNAME
```

### **Расщепление поля на два**

Написание операторов присваивания для расщепления содержания поля на два поля требует еще некоторой работы, но вырезание строки символов минимизирует усилия. Для данного примера предположим, что у вас есть одно поле в первоначальном файле для номера телефона и кода территории. Теперь вы хотите хранить код территории в одном

поле, а телефонный номер - в другом. Предполагая, что эти поля являются полями цифровых данных, вам нужно присвоить величину строке символов, затем “вырезать” строку символов для присвоения желаемой части к каждому новому полю.

В этом примере первоначальное поле НомерТелефона - это десятиразрядный номер, код территории - это трехразрядное число, а новое поле НомерТелефона - имеет семиразрядный номер. AssignRecord ROUTINE в генерированной исходной программе преобразования исходной программы будет выглядеть вроде этого:

```
AssignRecord      ROUTINE
  CLEAR(CUS:Record)
  CUS:NUMBER = IN::NUMBER
  CUS:FIRSTNAME = IN::FIRSTNAME
  CUS:LASTNAME = IN::LASTNAME
  CUS:ADDRESS = IN::ADDRESS
  CUS:CITY = IN::CITY
  CUS:STATE = IN::STATE
  US:ZIP = IN::ZIP
  ! CUS:AREACODE =
  CUS:PHONENUMBER = IN::PHONENUMBER
  CUS:ENTRYDATE = IN::ENTRYDATE
```

Обратите внимание, что это присвоение из первоначального поля НомерТелефона в новое поле НомерТелефона. Однако, так как новое поле хранит только семь разрядов, вы должны изменить его. Чтобы управлять присвоением полей, создайте неявную переменную в виде строки символов, присвойте ей величину первоначального поля НомерТелефона, затем используете вырезание строки символов для присвоения желаемых частей к новым полям, как показано ниже:

```
AssignRecord      ROUTINE
  CLEAR(CUS:Record)
  CUS:NUMBER = IN::NUMBER
  CUS:FIRSTNAME = IN::FIRSTNAME
  CUS:LASTNAME = IN::LASTNAME
  CUS:ADDRESS = IN::ADDRESS
  CUS:CITY = IN::CITY
  CUS:STATE = IN::STATE
  US:ZIP = IN::ZIP
  TempPhoneNumber = IN::PHONENUMBER
  CUS:AREACODE = TempPhoneNumber[1:3]
  CUS:PHONENUMBER = TempPhoneNumber[4:10]
  CUS:ENTRYDATE = IN::ENTRYDATE
```

Дополнительную информацию о нарезании строки символов можно получить в Неявные массивы и вырезание строк символов в Справочнике языка.

### **Комбинирование двух или более полей**

В этом примере первоначальное поле PhoneNumber (номер телефона) является числом из семи цифр, AreaCode (код района) - это число из трех цифр, а новое поле PhoneNumber - число из десяти цифр. Подпрограмма присвоения записи (AssignRecord ROUTINE) в сгенерированной исходной программе преобразования файла выглядит следующим образом:

```
AssignRecord      ROUTINE
  CLEAR(CUS:Record)
  CUS:NUMBER = IN::NUMBER
  CUS:FIRSTNAME = IN::FIRSTNAME
  CUS:LASTNAME = IN:: LASTNAME
  CUS:ADDRESS = IN::ADDRESS
  CUS:CITY = IN::CITY
  CUS:STATE = IN::STATE
  US:ZIP = IN::ZIP
  ! CUS:AREACODE =
  CUS:PHONENUMBER = IN::PHONENUMBER
  CUS:ENTRYDATE = IN::ENTRYDATE
```

Обратите внимание на то, что это присвоение из первоначального поля номера телефона к новому полю номера телефона, а первоначальный код района опущен. Чтобы управлять этими присваиваниями полей, создайте две неявные строки символов и присвойте им первоначальные величины кода района и номера телефона, затем конкатенируйте строки символов и присвойте результат новому номеру телефона, как это показано ниже:

```
AssignRecord      ROUTINE
  CLEAR(CUS:Record)
  CUS:NUMBER = IN::NUMBER
  CUS:FIRSTNAME = IN::FIRSTNAME
  CUS:LASTNAME = IN:: LASTNAME
  CUS:ADDRESS = IN::ADDRESS
  CUS:CITY = IN::CITY
  CUS:STATE = IN::STATE
  US:ZIP = IN::ZIP
  TempAreaCode" = IN::AREACODE
  TempPhoneNumber = CLIP(TempAreaCode") & TempPhoneNumber"
  ! CUS:AREACODE =
  CUS:PHONENUMBER = IN::PHONENUMBER
  CUS:ENTRYDATE = IN::ENTRYDATE
```

Более полную информацию о вырезании строки вы можете найти в Неявные массивы и в Вырезание строки в Справочнике языка.

### **Преобразование даты**

Предположим, что поле ENTRYDATE (ввод даты) содержит строчную дату в формате ММ/DD/YY. Мы хотим преобразовать эту строчную дату в стандартную дату Clarion, так чтобы мы могли выполнять вычисления с ней и показывать ее в различных других форматах.

Мы воспользуемся для выполнения преобразования, как это показано ниже, функцией DEFORMAT. Здесь @D1 представляет шаблон даты Clarion'a, который соответствует формату DD/YY. Смотрите Шаблоны даты в Справочнике языка, где имеется более полная информация.

```
AssignRecord      ROUTINE
  CLEAR(CUS:Record)
  CUS:NUMBER = IN::NUMBER
  CUS:FIRSTNAME = IN::FIRSTNAME
  CUS:LASTNAME = IN::LASTNAME
  CUS:ADDRESS = IN::ADDRESS
  CUS:CITY = IN::CITY
  CUS:STATE = IN::STATE
  US:ZIP = IN::ZIP
  ! CUS:AREACODE =
  CUS:PHONENUMBER = IN::PHONENUMBER
  CUS:ENTRYDATE = DEFORMAT( IN::ENTRYDATE, @D1)
```

### **Преобразование существующих (законных) данных**

К вам, как к разработчику, может быть часто обращаются с просьбой написать программу для поддержания существующих или “законных” данных. Хотя вначале у вас нет словаря данных, описывающего эти данные, вы можете легко преобразовать эти данные путем создания словаря данных. а затем повторяя шаги, описанные в Генерировании исходного файла для преобразования.

1. Импортируйте данный файл в словарь данных, выполняя для этого шаги, описанные в Редакторе словаря - Импортирование определений файла.

2. Выполните шаги, описанные в Генерировании исходного файла для преобразования и в Редактировании исходной программы для выполнения присваиваний полей.

## Печать данных

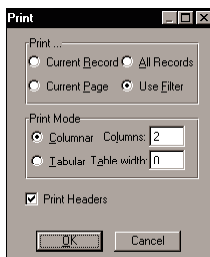
Вы можете напечатать данные из менеджера базы данных. Вы можете использовать эти простые отчеты, чтобы просмотреть данные в ваших файлах.

Чтобы напечатать данные:

1. Выберите File > Print.

Появится диалоговое окно Print (печать), позволяющее вам выбрать формат отчета.

2. Выберите подходящие радиокнопки из группы Print... (печатать...), чтобы установить записи для печати.



Current Record (текущая запись)

Печатает только текущую выделенную запись

Current Page (текущая страница)

Печатает только те записи, которые находятся в данное время на экране.

All Records (все записи)

Показывает все записи в файле.

Use Filter (использование фильтра)

Печатает только те записи, которые соответствуют фильтру, созданному в диалоговом окне QBE.

3. Выберите подходящие радиокнопки из группы Print Mode (режим печати) для установки формата печати.

Columnar (режим “по колонкам”)

Печатает записи в формате типа “крупноформатной таблицы”, в котором каждое поле в записи является отдельным столбцом.

Tabular (табличный режим)

Печатает записи в формате типа “форма”, в которой каждое поле в записи находится на своей собственной отдельной строке печати.

4. В поле Columns (столбцы), если вы выбрали Columns (столбцы), установите число записей для печати рядом .

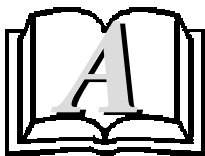
5. Если вы выбрали режим Табличный (Tabular), в поле Table Width (ширина таблицы) установите общее число символов для печати для одной записи .

6. Если вы хотите печатать заголовки столбцов в режиме “по колонкам” или метки поле в режиме бланка (Form Mode), отметьте поле Print Headers (печатать заголовки).

7. Нажмите кнопку Print (печать) для печати выбранных записей.



## **Приложение А Вопросы конструирования Окон**



В этой главе дается введение в:

- ◆ Принципы конструирования интерфейса приложений, основанных на Microsoft Windows.
- ◆ Управляемое событиями программирование и его влияние на процесс конструирования вашего приложения.
- ◆ Типы окон, элементов управления, курсоров и других объектов, общих для приложений Windows.
- ◆ Стандартные меню и команды меню, рекомендуемые для приложений Windows.
- ◆ Использование цвета и какое это имеет отношение к пользователю.

Обзор: вопросы конструирования окон  
Принципы конструирования  
Программирование, управляемое событиями  
Фоновая обработка  
Windows и элементы окна  
Окно приложения  
MDI  
Диалоговые поля  
Кнопки  
Поля флажков (Check Box)  
Радиокнопки (Radio Button)  
Поля списков ( List Boxes)  
Комбинированное поле списка (ComboBox)  
Выпадающее поле списка (Drop-Down List Boxes)  
Текстовые поля (Text)  
Спин поля (Spin Boxes)  
Статический текст  
Групповые поля  
Лист и закладки  
Wizards (мастера)  
Метки элементов управления  
Курсоры  
Меню  
Файловое меню  
Меню редактирования  
Меню просмотра  
Меню окна  
Меню помощи  
Ускоряющие клавиши  
Цвет

## Обзор

Если вы сделаете свою программу так, что она будет выглядеть и действовать подобно другим программам Windows, ваш пользователь освоит ее намного быстрее и будет чувствовать себя более уверенно, используя ее для выполнения тех задач, на которые вы ее рассчитывали при конструировании. Среда GUI(графический интерфейс пользователя) требует, чтобы вы общались со своими пользователями на их условиях. Конструирование программы должно отразить то обстоятельство, что пользователь контролирует свою работу с приложением, как визуально, так и на основе того, что происходит в процессе выполнения программы.

Microsoft выдает определенные рекомендации относительно того, как должно выглядеть приложение Windows, и как должны выглядеть и действовать элементы управления в окне. Имеется целый набор рекомендаций, касающихся стандартных меню и команд.

В данной главе также обсуждается вопрос применения цвета в вашем приложении.

## Принципы конструирования

Фирма Apple в своем Human Interface Guidelines (Руководство по интерфейсу человека), фирма IBM в своем Common User Access: Advanced Interface Design Guide (Для широкого пользователя: Современное руководство по конструированию интерфейса), и Microsoft в (Windows Interface: an Application Design Guide (Интерфейс Windows: Руководство по конструированию приложения) подробно рассмотрели принципы конструирования пользовательского интерфейса для проектировщиков матобеспечения, работающих в популярной среде GUI. Создание стандарта конструирования программ обещает следующие преимущества:

- ◆ При наличии согласованности между приложениями пользователи осваивают новые приложения намного быстрее и легче, сводя до минимума необходимость обучения.
- ◆ Согласованность между приложениями повышает уровень доверия у пользователя, что в результате приводит к повышению производительности.

Существуют два особенно важных принципа конструирования Windows, которые должны иметь в виду программисты Clarion при конструировании интерфейса пользователя. Первое - это контроль со стороны пользователя - обеспечение программой интерфейса, характеризуемого метафорой "реальный мир" и поддержка согласованного вида и ощущения всех частей программы - формируют доверие пользователя. Второе - это необходимость помнить, что программирование Windows является процессом, управляемым событиями - пользователь сам решает, какое действие предпринять следующим. Программист ответственен за то, чтобы обеспечить визуальный список параметров, на основании которых будет действовать пользователь.

## Управление, осуществляемое пользователем

---

У вашего пользователя может не быть многолетнего опыта использования разнообразных программ. Обеспечение метафоры реального мира поможет обеспечить “декорации” или группу ожиданий в отношении вашей программы. Программа, обрабатывающая слова, например, использует “бумажную” метафору - документ подобен бумаге, на которой нужно писать, стирать с нее символы и т.д. Многие программы баз данных используют метафору “картотека”. Устанавливая связь с реальным миром, вы тем самым повышаете уровень комфорта для ваших пользователей и активно вовлекаете их в работу программы.

Очень важна визуальная согласованность. По мере возможности приложение должно использовать единый способ выполнения действий. Пользователь, изучающий новую процедуру в вашей программе, строит свое понимание на предыдущем опыте работы с другими процедурами в той же программе. Создание стандартного вида ваших диалоговых полей и изготовление экранов для различных задач так, чтобы они были похожи один на другой, также снижает время, необходимое вам для конструирования различных экранов, требуемых вашей программой.

Непосредственность заставляет пользователя чувствовать себя так, как будто он командует. Перемещение документа из одной стопки в другую или перемещение от одной пиктограммы к другой, такой, как пиктограмма мусорной корзины, могут казаться пользователю настоящим действием. Clarion обеспечивает поддержку механизма drag-and-drop, который позволяет вам создать, например, пиктограмму с картинкой некоего лица на ней, представляющую запись служащего. Пользователь может перетащить эту пиктограмму к другой, представляющей деньги, чтобы таким образом открыть диалоговое поле, показывающее записи выплат служащему.

Простота обычно является признаком наилучшей конструкции. Беспорядочное заполнение экрана слишком большим числом окон, кнопок, пиктограмм, списков и других объектов может смутить пользователя. Диалоговые поля должны вставляться в малое пространство, их сообщения должны быть просты. То же самое относится к цветам. Ограничьте использование цвета только теми областями, где он нужен для подчеркивания, как например красный цвет для важных сообщений.

Пусть пользователь знает, что происходит - обеспечьте обратную связь. Когда пользователь выбирает команду или начинает операцию, визуальная обратная связь должна подтвердить, что она выполняется. Подтверждение может быть графическим, таким, как изменение курсора, или просто как линейка прогресса или сообщение на линейке статуса. Если будет иметь место пауза на время, пока программа выполняет другую операцию, информируйте пользователя и подскажите ему, если это возможно, сколько это займет времени.

Ваше приложение должно использовать понятный язык. При конструировании

приложения для корпоративной среды часто существенными оказываются технические термины. Будьте внимательны относительно тех случаев, когда лучше будет простой человеческий язык. Особенно это важно для программистов новичков в Windows программировании - берегитесь “Window-измов”. Очень просто сделать две кнопки с надписями “ОК” и “Cancel”. Однако “Да” и “Нет” намного лучше в качестве меток кнопок, где задается вопрос типа: “Хотите ли вы удалить данную запись?”.

## ***Программирование, управляемое событиями***

В связи с проблемой управления со стороны пользователя и наиболее важной концепцией для новых программистов Windows является то, что вы должны конструировать управляемые событиями программы Windows. В многоцелевой среде программа просто не может управлять последовательностью действий программы - пользователь управляет ею.

Базовая структура Windows такова, что операционная система информирует программу через сообщения о том, что хочет от программы пользователь. Это прямо противоположно программированию DOS.

Сообщения Windows информируют программу, когда выбрано меню, когда выбрано окно, когда пользователь хочет закрыть работающую систему - набор документации Windows 3.1 по разработке матобеспечения посвящает более 200 страниц только перечислению и объяснению различных сообщений. Большинство языков программирования требуют тщательно разработанных процедур обращения с сообщениями для разветвления потока программы при получении различных типов сообщений Windows.

Clarion автоматически управляет “кухней” обработки сообщений. Цикл АСЦЕРТ освобождает программиста Clarion от беспокойства о сообщениях системы. Однако вы все-таки должны иметь ввиду модель управления через события при конструировании вашей программы.

Программа Windows должна постоянно ожидать ввода - например, в форме ввода данных в поле редактирования. В окне с несколькими полями пользователь может осуществлять табуляцию или щелкать клавишей мыши, чтобы активировать какое-либо из них. Соответственно, вы должны планировать ваши диалоги ввода данных.

Кроме того, ваши пользователи будут ожидать наличия полного набора элементов интерфейса пользователя, включая такие элементы, как меню, множественные окна и графические элементы управления, причем все доступные одновременно.

Вы можете создать приложение, которое дает возможность пользователю открыть два окна с резко различными функциями. Clarion будет автоматически управлять событиями, генерированными в пределах каждого окна. Однако, что можно сказать о командах меню или линейке инструментов? Вы должны планировать их так, чтобы команды действовали

на любое активное окно. Clarion помогает вам сделать это автоматически, создавая приложение, использующее Интерфейс Многих Документов (MDI). Могут, однако, быть случаи, когда вам может понадобиться вручную сделать недоступными или восстановить доступность команд меню.

И наконец, модель управления через события должна повлиять на проектируемые для вашего приложения окна. Планируйте использование вашего главного окна приложения в качестве спинного хребта для вашей программы. Генерируйте новое окно только после некоторого действия пользователя в главном окне.

## **Фоновая обработка**

Фоновая обработка является еще одной важной для программистов Windows концепцией. В среде многих задач ваша программа всегда должна оставлять управление в руках пользователя, особенно когда она выполняет продолжительный процесс, такой как чтение или записывание большого файла.

На уровне языка Clarion поддерживает фоновую обработку через вызов и обнаружение событий TIMER (таймер). Это значит, что назначение TIMER (таймера) для WINDOW (окна) указывает операционной системе о необходимости генерировать событие таймера для окна через равные интервалы времени. Затем обнаруживается событие TIMER:Events и обрабатывается в пределах цикла АСЦЕРТ ОКНА. Ваша программа выполняет небольшую долю более крупного процесса для каждого события таймера. В результате этого эффект сводится к тому, что ваша программа может работать конкурентно с другими программами, которые используют подобную технику.

**Внимание: Оператор YIELD (возвращать) достигает того же результата для программ, которые не содержат цикла АСЦЕРТ.**

Более полная информация по этим вопросам имеется в Справочнике языка.

На уровне шаблонов все шаблоны Clarion for Windows автоматически поддерживают фоновую обработку путем использования описанной выше методики.

Для процессов, кодированных вручную, вы можете контролировать распределение ресурсов машины путем регулирования интервала времени между событиями таймера и объемом обработки, которая производится для каждого события таймера. Например, малый интервал таймера с большим объемом обработки, приходящимся на этот интервал, поглощает ресурсы, в то время как большой интервал таймера и малый объем обработки, приходящийся на интервал, приводит к увеличению быстродействия программы.

Разумный интервал таймера располагается между 1/100 и 30/100 секунды в зависимости от типа выполняемого процесса и типа устройства, выполняющего обработку.

```
MyWindow    WINDOW('With a Timer'),TIMER(1)    !1/100 секунды
...
CODE
...
MyWindow{PROP:Timer} = 30                        !30/100 секунды
```

Разумное число итераций или циклов LOOP, приходящихся на интервал, меняется от 3 до 1000, опять в зависимости от типа выполняемого процесса и типа устройства, выполняющего обработку. Смотрите переменную RecordsPerCycle, генерированную шаблоном процесса.

Ваш выбор 16-битового или 32-битового приложения, или 16-битовой или 32-битовой операционной системы не оказывает никакого влияния на метод, который вы используете для реализации фоновой обработки.

## **Windows и элементы окна**

В данном разделе описываются обычные элементы Windows, включая типы окон и элементы управления. Выбор подходящего элемента для подходящей процедуры помогут вашему пользователю извлечь максимум из вашей программы.

Существуют три типа окон: окна приложения, окна документа и окна диалоговых полей.

### **Окно приложения**

---

Окно приложения должно, как правило, содержать все другие окна, которые может генерировать ваша программа. Если вы открываете окно просмотра, оно должно появиться внутри окна приложения. Если пользователь изменяет размер окна приложения на меньший, вы можете позволить, чтобы другие ваши окна появлялись частично вне его. Кроме того, вы можете дать возможность пользователям переместить передвигаемые диалоговые поля из окна приложения наружу.

Окна приложения должны всегда иметь строку заголовка, в которой находится имя приложения. Microsoft рекомендует, чтобы окна приложения также имели рамки с возможностью изменения размера, поле меню управления и кнопки минимизации и восстановления. В Clarion'е вы можете добавить эти атрибуты к окну через диалоговое окно Window Properties (свойства окна).

## MDI

---

Clarion делает доступным Интерфейс многих документов (MDI). Он управляет множественными документами или множественными просмотрами одного и того же документа, каждый в отдельном окне, появляющимся в главном окне приложения. Если пользователь меняет размер окна приложения, чтобы сделать его меньше документного окна, документное окно появится урезанным.

В программировании базы данных документный “файл” часто является базой данных. Используйте документные окна для показа окон сканирования, просмотров и запросов. Так как Clarion предоставляет инструментарий, позволяющий делать в MDI-окне практически все, будьте консервативными в использовании документных окон. Обязательно используйте документное окно в качестве графического Viewer-а для просмотра, например, содержания текстового файла или для просмотра отчета. НЕ используйте документное окно для процедур, которые относятся к таким диалоговым полям, как, например, поля для ввода записи.

Microsoft рекомендует, чтобы каждое документное окно содержало строку заголовка с именем документа, поле с меню управления и кнопку максимизации. Если необходимо, вы можете включить кнопку минимизации.

## Диалоговые поля

---

Используйте диалоговое поле тогда, когда вам для выполнения задачи требуется дополнительная информация от пользователя. Добавление записи к базе данных является примером такой задачи - ввод данных для новой записи будет процессом получения дополнительной информации.

Руководящие принципы Windows в вопросе конструирования диалоговых полей очень гибки. Поля могут быть перемещаемыми или сохранять фиксированную позицию. Они могут иметь один размер или кнопку More>> (больше), чтобы развернуть их и предложить дополнительные параметры. Они могут быть модальными или внеэкранными. Они могут представлять краткое сообщение с единственной опцией ‘ОК’ или обеспечивать сложные выборы, элементы управления и входные параметры. Так как руководящие принципы допускают широкие вариации, данный раздел представит только несколько указателей, которые помогут вам сконструировать более легкие в использовании диалоги.

☐ Если вы используете строку заголовка диалогового поля, или статический текст в поле, кратко объясните функцию диалога или укажите, какая команда вызвала его появление.

☐ Поставьте как можно больше команд в состояние по умолчанию, так, чтобы вы требовали от пользователя наименьшего количества ввода.

☐ Поместите наиболее важную информацию на самый верх, налево, наименее важную - направо, в самом низу. Ваши пользователи читают слева направо и сверху вниз - это



естественный способ, в котором они ожидают увидеть ввод информации.

□ Установите фокус на первом поле редактирования (ввода) текста. Это позволит пользователю легко набрать слово или слова с клавиатуры без необходимости устанавливать положение курсора.

□ Поместите кнопки по умолчанию поля диалога - наиболее вероятный выбор пользователя - справа. Это приводит пользователя к возможности “читать” выборы слева перед принятием “решения”.

□ Представляя краткое сообщение, воспользуйтесь имеющимися пиктограммами по умолчанию, используя для этого функцию MESSAGE. Пиктограммы Стоп, Информация и Вопрос знакомы пользователю по другим приложениям.

## Кнопки

---

Кнопка дает начало действию. Когда пользователь нажимает кнопку, кнопка видна, как нажатая. Когда действие кнопки недоступно, метка кнопки должна быть стерта.

Clarion дает вам возможность использовать либо текстовые, либо графические метки (кнопки-картинки) или и то и другое. Если вы используете кнопки с изображениями, вы должны включить подсказки инструмента, чтобы дать возможность пользователю увидеть, на словах, какое действие вызовет кнопка. Будьте осторожны в использовании кнопок-картинок. Придерживайтесь стандартных пиктограмм (таких, как пиктограммы, которые многими самыми ходовыми приложениями используются для File > Open, File > Save и т.д.). Слишком много кнопок-картинок в приложении могут смутить пользователя. Некоторые обозреватели осудили программы “iconitis” - программы, которые имеют так много графических кнопок на экране одновременно, что для пользователя становится невозможным запомнить, какая кнопка какое действие исполняет.

Кнопки могут выполнять несколько типов действий.

- ◆ Кнопка может дать начало действию.
- ◆ Кнопка может закрыть активное диалоговое поле, затем открыть другой, связанный диалог.
- ◆ Кнопка может открыть другой диалог поверх текущего, не закрывая при этом текущий.
- ◆ Кнопка может “раскрыть” или изменить размер диалогового окна для включения большего числа параметров.
- ◆ Кнопка может перевернуть “страницу” на диалоговом окне Мастера.

Всегда назначайте одну кнопку, как кнопку по умолчанию. Когда пользователь нажимает ENTER, это будет инициировать действие кнопки по умолчанию. Microsoft также предлагает, чтобы вы не назначали мнемонику - например, “&OK”, которая выглядит на экране как ОК, - для кнопки по умолчанию.

## **Поля флажков**

---

Поля флажков управляют истина/ложь, да/нет, вкл/выкл или логическими переменными. Пользователь переключает состояние, щелкая мышью на поле или нажимая клавишу пробела. Если параметр поля флажков недоступен, метка должна быть сделана невидимой.

## **Радиокнопки**

---

Радиокнопки, также называемые кнопками с зависимой фиксацией, предоставляют пользователю единственный выбор в наборе взаимоисключающих вариантов. Пользователь может выбрать только одну кнопку за раз. Если пространство в окне дорого, а число вариантов больше четырех, рассмотрите вариант выпадающего поля списка, которое занимает меньше места.

## **Поля списка**

---

Поля списка показывают пользователю возможные варианты выбора. Если какой-либо вариант выбора недоступен, в общем случае вы должны исключить его из списка. Если вариант достаточно важен для того, чтобы пользователь должен был знать о его недоступности, Microsoft рекомендует, чтобы он появился в поле списка как затененный выбор.

Всегда предоставляйте своим полям списка достаточно места. Постарайтесь выделить по вертикали место для трех - восьми выборов; по горизонтали - для средней длины выбранного текста и еще несколько пробелов.

Помните, что поле списка может предоставить пользователю много информации - поэтому оставляйте его простым!

## **Комбинированные поля**

---

Комбинированное поле - это сочетание текстового вводного поля и поля списка. Такие поля удобны там, где данные для ответов предоставляются системой в виде списка, но пользователю вместе с тем дается возможность ввести ответ, которого нет в данном списке, вручную.

Принципы проектирования для полей списков применимы к комбинированным полям.

## **Выпадающие списковые поля**

---

Выпадающие списки с простым выбором выполняют те же функции, что и поля списка, но занимают меньше места. Будучи закрытым, выпадающий список имеет место только для одного варианта выбора. В открытом состоянии список покажет больше позиций, как стандартное поле списка.

Начинающие пользователи часто намного больше затрудняются с выбором позиции из выпадающего поля, чем из нормального поля списка. Там, где место позволяет, используйте

радиокнопки (для четырех или менее выборов) или нормальные поля.

## **Текстовые поля**

---

Текстовые поля позволяют пользователю вводить информацию. Они могут иметь одну строку или много строк. Многострочные поля редактирования обычно должны иметь вертикальную линейку прокручивания.

Стандартные ускоряющие клавиши Windows для копирования, вставки и т.д. активны по умолчанию. Это весьма полезно, так как позволяет пользователю копировать, например, параграф из другого приложения, затем вставить его прямо в многострочное текстовое поле в вашем приложении. По этой причине мы рекомендуем вам не переназначать клавиши редактирования окон, принятые по умолчанию - такие, как CTRL+C для копирования или CTRL+V для вставки из буфера- для различных команд в вашем приложении Clarion.

Текстовые поля с фиксированной длиной и автоматическим выходом могут ускорить ввод данных. Как только пользователь заполняет текстовое поле (набирая максимально допустимое количество символов), фокус перемещается к следующему элементу управления. Microsoft рекомендует, чтобы приложения использовали этот тип текстового поля умеренно, так как сдвиг фокуса может приводить пользователя в замешательство, если он застает его врасплох. Мы рекомендуем использование этого типа текстового поля, когда в диалоге имеется много вводных полей. Для диалогов с небольшим числом полей программист должен попытаться предугадать, что будет ожидать конечный пользователь и сделать свой выбор в соответствии с этим.

Clarion позволяет вам выбрать шрифт для текстовых полей. Мы предлагаем использовать шрифт системы по умолчанию. Microsoft специально выбирает этот шрифт для меню и других позиций системы, так как он особенно легко читается на мониторе.

## **Spin-Поля**

---

Spin-поля - это специальные текстовые поля с парой стрелок (кнопки прокрутки) у правого края текстового поля. Spin-поля содержат ограниченный набор величин, которые пользователь может ввести, или, используя стрелки, увеличивать или уменьшать на некоторую, заранее установленную величину. Spin-поля могут обеспечить альтернативу выпадающим спискам, когда набор величин ограничен в количестве и подпадает под естественную прогрессию; например, "Весна, Лето, Осень, Зима".

Кроме элементов управления увеличения или уменьшения для простых чисел или выборов вы можете использовать Spin-поля для манипуляции величинами, которые состоят из нескольких компонент. Вы можете, например, показать время в часах, минутах и секундах. Непременнo визуальнo отделите каждую компоненту с помощью соответствующего символа разделителя, такого, как двоеточие.

## Статическое поле

---

Статическое поле должно представлять информацию только для чтения, такую, как указания для ввода данных в других элементах в диалоговом поле. Вы должны также использовать статический текст для пометки элементов управления, не помеченных автоматически форматером окна, таких, как в текстовом поле.

Clarion дает вам возможность выбрать шрифт для статического поля. Мы предлагаем использовать шрифт системы по умолчанию. Microsoft специально выбирает этот шрифт для меню и других позиций системы, так как он особенно легко читается на мониторе. Конечно, вы вправе сделать текст крупнее, как, например, при создании “заголовка” для “формы” диалогового поля.

Мы также советуем вам побороть желание использовать необычные цвета или много разнообразных цветов для статического текста. Вы никогда не сможете сказать, каков был цвет фона.

## Групповые поля

---

Групповые поля обеспечивают визуальное группирование для связанных элементов управления. Они состоят из прямоугольной рамки с меткой слева наверху.

Групповое поле может направить пользователя прямо к элементам управления, которые наиболее важны для выполняемой в данный момент задачи. Если ваше приложение требует диалога с десятью или более элементами управления или полями, мы очень рекомендуем остановиться и подумать, попадают ли некоторые из элементов управления в логические группы.

## Листы и закладки

---

Лист с закладками обеспечивает другой метод группирования связанных элементов управления, он позволяет вам размещать элементы управления с подобными или связанными функциями на отдельные “страницы”.

Закладочные диалоговые окна могут сделать “более плоским” ваше приложение за счет уменьшения числа видимых элементов управления и показа только тех, которые наиболее важны для данной задачи. Если вашему приложению требуется диалоговое окно с более чем десятью элементами управления или полями, вы должны рассмотреть возможность “многостраничного” подхода.

Имейте в виду, что поля Требуемого ввода должны находиться на первой видимой закладке.

## Мастера

---

Атрибут WIZARD (мастер) на элементе управления Лист свойств дает вам возможность контролировать движения пользователя по “страницам” закладок. Это дает вам возможность

представлять набор диалоговых окон в линейной манере. По желанию вы можете контролировать следующую “страницу” на основе ответов, которые пользователь представит на предыдущих страницах.

Мастера стали все более популярными из-за того, что они дают возможность пользователю отвечать только на один вопрос за раз, уменьшая тем самым шансы путаницы или ошибки.

## Метки элементов управления

---

Форматер окна автоматически поставяет метки для многих, но не для всех элементов управления. Вы можете снабдить метками другие элементы управления, используя статический текст. Это не только идентифицирует элементы управления для пользователя, но также позволит пользователям клавиатуры быстро направить фокус на данный элемент управления.

Когда пользователь набивает клавишами мнемонику (такую как “S” в ‘Daily Sales’), Windows автоматически направляет фокус на следующий элемент управления после метки статического элемента управления. Таким образом, вы можете поместить ‘Daily &Sales’ налево от выпадающего поля. Когда пользователь нажимает ALT+S, фокус получает комбинированное поле. Пользователь клавиатуры должен только нажать клавишу стрелка вниз, чтобы просмотреть варианты выбора в списке.

Microsoft предлагает следующие принципы построения текста меток:

□ Сделайте заглавной первую букву каждого слова, за исключением артиклей (например, a, an, the), союзов (например, and, or, for), предлогов (например, by, with) или слова “to” в инфинитивах.

□ Постарайтесь использовать первую букву первого слова в качестве мнемоники. Так как, однако, мнемоника должны быть уникальна, это не всегда возможно. Или используйте другую букву, если эта позволит получить более сильную мнемоническую связь: такую как “x” в “Exit”. Если первое слово в метке управления менее важно, чем следующее, используйте другое (например, “by Ascending Order”).

□ Отдавайте предпочтение согласным перед гласными: они более различимы и легче запоминаются.

Microsoft также предлагает устанавливать следующие позиции для меток управления следующих элементов управления:

- ◆ Командные кнопки: внутри кнопки.
- ◆ Поля флажков, радиокнопки: справа от управления.
- ◆ Текстовые поля, поля прокрутки, списки, комбинированные поля: над элементом управления или слева от него. Поместите после последнего слова метки- двоеточия. Выводите метку слева в разделе диалогового поля, в котором она появляется.

## Курсоры

В графической среде, такой как Windows, курсор мыши или экранный указатель являются средствами, с помощью которых пользователь показывает приложению, что дальше делать. Например, полоска |, или точка вставки, может подсказать приложению, обрабатываемому слово, где должен появиться следующий, набираемый пользователем символ. Это ключевая часть “управляемого событиями программирования”, о котором говорилось в начале главы.

Хотя Microsoft не располагает особыми руководствами по использованию курсора каждой системы, следующие использования превратились в стандарты в платформах GUI:

Стрелка: выбирает элементы управления и команды меню.

I-луч: выбирает и вставляет текст.

Перекрестие: чертит и манипулирует графикой.

Знак плюс: выбирает поля в массиве.

Песочные часы: показывает что продолжается длительная операция.

## Меню

Меню показывает пользователю набор доступных для исполнения команд. Пользователи Windows привычны к стандартным меню и командам, которые появляются во многих различных приложениях. Если вы будете использовать те же самые меню, новые пользователи легче освоят ваши приложения, а чувство знакомого повысит уверенность всех пользователей и уровень производительности.

При конструировании дополнительных, заказных меню и команд учтите, что моделью для GUI конструирования является принцип “Существительное - Глагол”. Apple образно называет этот принцип так: “Эй вы - делайте это!”.

В модели “существительное - глагол” пользователь указывает на что-то - например, на об

ъект на экране, такой, как текст. Это существительное. Модель предполагает, что следующим командным действием, которое выберет пользователь, будет приказ приложению, что делать с существительным. Действие - это глагол. Если вы выберете такие слова для ваших меню и команд, чтобы команды были короткими фразами “сделать то-то”, такими, как “Insert > Record,” (вставить запись) “View > Transaction,” (показать транзакцию) “List > Activity,” (пролистать активные) или “Select > Current Group” (выбрать текущую группу), ваши меню будут иметь выигрышную ясность.

Этот принцип не должен сильно ограничивать вас. Бывают случаи, когда более подходящим является использование позиции меню для имитации сложных инструкций, таких, как привлечение диалогового поля со многими различными вариантами и параметрами, которые должен установить пользователь. При выполнении этого добавьте эллипс (...) после последнего слова команды меню.

Далее обсуждаются стандартные реализации меню, рекомендуемые Microsoft.

## Меню файлов

---

Для многих приложений базы данных не является естественным разрешить пользователю открывать и закрывать файлы внешних документов. В простейшем случае база или базы данных открываются автоматически вместе с приложениями. Редактирование со стороны пользователя ограничивается редактированием отдельных записей. Программисты Clarion могут захотеть ограничить команды в меню файла только теми, которые влияют на глобальную операцию в приложении - Printer Setup (установка принтера) и Exit (выход) в самом крайнем случае. Ваши Clarion приложения должны иметь как минимум команду File > Exit: в таком виде ожидают пользователи Windows увидеть команду, разрешающую им покинуть программу.

New or New...

(новый или новый...)

Создает новый файл с именем по умолчанию, таким как “Без заглавия”. Это иногда бывает проблематичной позицией меню при создании приложений базы данных. Если ваше приложение не позволяет пользователю создать новую базу данных или создает отдельные редактируемые внешние файлы (такие как текстовые файлы), программисты Clarion могут выбросить эту команду.

Open (открыть)

Показывает диалоговое окно File Open (открыть файл) из обычной библиотеки диалогов Windows. Дает пользователю возможность открыть внешние файлы.

Save (сохранить)

Сохраняет активный документ. Для вновь создаваемого файла вызовите диалог Save As.

Save As (сохранить как)

Показывает диалоговое окно Save As из обычной библиотеки диалогов Windows.

Print or Print...

(печатать или печатать...)

Печатает активный документ или ведет к диалогу, позволяющему пользователю установить параметры печати.

Команда Print (печатать) может быть в приложении базы данных интересной командой. Во многих случаях приложение

базы данных разрешает пользователю печатать только предварительно отформатированные отчеты.

Другие “документно-центричные” приложения Windows могут просто идти и печатать текущий документ во всей его полноте - но от приложения базы данных вряд ли можно ожидать печати базы данных в 30,000 записей, как варианта печати, принятого по умолчанию.

Одним из решений, которое используют некоторые популярные приложения, является полное отбрасывание команды печати и обеспечение отдельного меню Report (отчет). Это хорошее решение для приложений с ограниченным числом отчетов. Другой вариант, приложение с ограниченным числом отчетов, может также использовать каскадное меню, приложенное к команде File > Print.

Для приложения с большим числом предформатированных отчетов одним из решений может быть представление поля списка в диалоговом окне, когда пользователь выбирает команду File > Print.

#### Print Setup

(установка принтера)

Показывает диалоговое окно Printer Setup из обычной библиотеки диалогов Windows.

Этот диалог дает возможность пользователю изменить активный принтер и/или определить установки для выбранного принтера.

#### Exit (выйти)

Закрывает все окна приложения и заканчивает приложение. Если в вашем приложении нет меню File (файл), поместите вашу команду Exit (выйти) на самое левое меню приложения, как последнюю команду меню.

### **Меню редактирования Edit**

---

Меню редактирования Edit обычно обеспечивает команды для отмены последнего действия пользователя, а также буферные команды редактирования: вырезать, скопировать, вставить.

Undo (отменить сделанное) Команда Undo должна отменять последнее действие пользователя. Она должна быть всегда первой командой в меню Edit, если ваше приложение поддерживает эту команду. Очевидно, что программы базы данных представляют особые проблемы для операции Undo. В общем случае, приложения Windows делают команду Undo недоступной после файловых



операций, так, как тогда, когда команда File > Save сохраняет отредактированный документ на диске. Приложение базы данных может легко создать ситуацию, в которой оно пишет данные на диск каждые несколько секунд, когда, например, пользователь вводит группу новых записей.

Cut (вырезать)	Передает выделенный объект в буфер обмена и удаляет его из поля.
Copy (копировать)	Помещает копию выделенного объекта в буфер обмена.
Paste (вставить)	Помещает копию объекта, ранее помещенную в буфер обмена, в текущее поле.

Когда приложение находится в поле редактирования, Clarion автоматически делает доступным поддержку буфера обмена для Cut, Copy и Paste. Ускоряющие клавиши для этих действий по умолчанию - соответственно CTRL+X, CTRL+C и CTRL+V.

## Меню помощи

---

Меню помощи Help обеспечивает пользователю доступ к системе помощи Windows. Это всегда должно быть последнее меню справа. Меню помощи обычно содержит следующие команды:

Contents (содержание)	Загружает систему Windows Help, затем открывает внешний файл помощи на странице главного оглавления.
-----------------------	--

Search for Help On (поиск помощи)	Загружает внешний файл помощи, затем автоматически открывается диалоговое окно Search (поиск). Это дает возможность пользователю набрать слово; если слово появляется как файл темы, система помощи делает переход к заголовку данной темы.
-----------------------------------	---

How to Use Help (как использовать помощь)	Эта позиция открывает систему помощи Windows и показывает инструкции по ее использованию. Файл "WINHELP.HLP," который Windows автоматически устанавливает, содержит инструкции.
---	---

## Меню просмотра

---

Microsoft определяет меню просмотра View, как необязательное и заявляет, что оно включает команды для изменения того, как программа представляет данные для пользователя, без изменения каких-либо элементов данных. Как таковое, оно представляет естественные средства приложения базы данных, чтобы сделать возможными различные варианты просмотра данных в базе данных.

Меню просмотра может представить параметры показа различных элементов интерфейса, таких, как панели инструментов, строка статуса и другие специальные элементы управления, которые являются частью окна приложения. Особого командного текста для меню просмотра не предлагается.

## Меню окна

---

Это необязательное меню. Если вы поддерживаете Интерфейс многих документов (MDI) в вашем приложении, меню окна даст возможность пользователю манипулировать всеми дочерними окнами.

Команды для этого меню гибкие. Обычные команды включают:

Tile (черепица)	Располагает дочерние окна одно к другому, так что видны все окна.
Cascade (каскад)	Располагает дочерние окна в перекрывающемся виде, так что видны строки заголовка каждого окна.

Меню окна может содержать перенумерованный список до девяти открытых дочерних окон. Номер окна предшествует имени окна. Когда пользователь выбирает какое-либо окно из списка, это окно получает фокус.

## Ускоряющие клавиши

---

Ряд команд приобрели стандартные ускоряющие клавиши (или горячие, быстрые клавиши). Рекомендуем вам использовать следующие комбинации клавиш при создании своего приложения:

<i>Команда</i>	<i>Ускоритель</i>
<b>File &gt; New</b>	CTRL+N
<b>File &gt; Open</b>	CTRL+O
<b>File &gt; Save</b>	CTRL+S или SHIFT+F12
<b>File &gt; Exit</b>	ALT+F4
<b>Edit &gt; Undo</b>	CTRL+Z
<b>Edit &gt; Cut</b>	CTRL+X
<b>Edit &gt; Copy</b>	CTRL+C
<b>Edit &gt; Paste</b>	CTRL+V

## Цвет

Цвет может сильно повлиять на то, как ваш пользователь будет работать с вашим приложением. Microsoft не публикует стандартных принципов применения цвета - пока. При конструировании вашего приложения для вас могут быть полезными следующие принципы:

- Windows дают возможность пользователям выбрать цвета по умолчанию для текста и фона окна. Лучше принять эти цвета по умолчанию для тех частей программы, которые требуют наибольшего ввода данных: пользователи выразили свое предпочтение и вы должны это уважать!
- Не забывая сказанное в первом пункте, вы можете выбрать цвет для акцентирования элементов окон и экрана. Цвет может выделить особые площадки в окне - это может быть более эффективно, чем групповое поле.
- Используйте цвет для разделения отдельных частей вашей программы. Например, вы можете ассоциировать один цвет фона окна с диалоговыми полями, связанными с вводом данных о получаемых счетах, а другой цвет использовать для выплат.
- Используйте цвет для того, чтобы связать визуально подобные части программы. Например, вы можете ассоциировать цвет фона одного окна с данными номера телефона.
- Используйте стандартные ассоциации для специальных сигналов тревоги. В западной культуре наиболее “значащими” цветами являются, по-видимому, цвета на транспортных светофорах: красный, желтый и зеленый. Вы можете использовать красный для сигнализации об остановке в процедуре. Желтый вы можете использовать в качестве сигнала предупреждения. Зеленый конечно означает, что все в порядке.
- При добавлении цвета к текстовым элементам помните, что большинство цветов выглядят лучше на фоне нейтрального серого. Если вы не используете серый цвет, убедитесь, что имеется высокий контраст между текстом и фоном. При слабом освещении цвета имеют тенденцию к смыванию.
- Помните, что у 8% мужчин в Европе и Америке имеется некоторая степень цветовой слепоты. Наиболее распространенный вид цветовой слепоты снижает способность отличать красное и зеленое от серого. При менее распространенном типе цветовой слепоты пользователь не может различить желтый, синий и серый цвета.
- Помните, что на монохромных экранах на жидких кристаллах голубой цвет очень трудно отличить от серого и белого.



## Приложение Б Драйверы базы данных



Clarion for Windows достигает независимости базы данных с помощью технологии встроенного драйвера, дающего возможность получить доступ к данным практически из любой файловой системы. Имеется много драйверов файлов и многие драйверы добавляются. Все драйверы файлов читают и пишут данные в формате файловой системы используемого драйвера без применения временных файлов или процедур импорта/экспорта.

Часто главной задачей вашего приложения является доступ к существующим данным в их оригинальном формате. Для этих случаев вам следует только включить подходящий драйвер файла. Для тех случаев, когда вы не “заперты” в конкретной файловой системе, это приложение предлагает советы относительно наиболее подходящих для разных видов работ драйверов файлов. Вы можете выбрать правильный инструмент в зависимости от типа, размера и природы файлов данных, необходимых для вашего приложения.

Команды для доступа к данным из различных систем одинаковы; просто выберите правильный драйвер файла из выпадающего списка в вашем словаре данных и больше не беспокойтесь об этом.

Имеются переключатели, которые вы можете установить для оптимизации того, каким образом ваше приложение создает, читает и пишет файлы с определенным драйвером. Переключатели устанавливаются с помощью “строк драйвера”. Строки драйвера - это просто характерные для данного драйвера сообщения или параметры, которые могут быть посланы драйверу файла во время работы для управления его поведением. Строки драйвера направляются двумя способами: с оператором OPEN (открыть) и с функцией SEND (отослать).

Оператор OPEN автоматически отправляет любые строки драйвера, назначенные в атрибуте DRIVER файла FILE. Атрибут DRIVER (ДРАЙВЕР) назначается в словаре данных или при кодировании вручную в объявлении ФАЙЛА. Чтобы назначить строки драйвера с помощью словаря данных, наберите их в поле Driver Options (опции драйвера) в диалоговом окне File properties (свойства файла), как это описано в главе Редактор словаря. Для того, чтобы назначить строки драйверов кодированием вручную смотрите атрибут DRIVER для ФАЙЛА в Справочнике языка, а также смотрите раздел о разных драйверах в настоящем приложении.

Функция SEND отсылает одну строку драйвера к драйверу файла в любое время, включая момент перед тем, как файл открыт (OPENed). Команды функции SEND принимают две формы - со знаком равенства для изменения величины переключателя, и без знака равенства, для возврата величины переключателя. Некоторые строки драйвера не оказывают никакого влияния на процесс обработки, если они выданы после открытия файла, поэтому для этих строк синтаксис функции SEND не приводится.

#### ASCII Files

Стандартные файлы ASCII без разграничителей полей. Записи разграничиваются символом возврата каретки/переводом строки.

#### BASIC Files

Файлы ASCII, разграниченные запятой - двойными кавычками. Записи разграничены символом возврата каретки/переводом строки.

#### Btrieve Files

Файлы Btrieve, использующие прямой низкоуровневый доступ.

#### Clarion Files

Файлы, созданные с помощью собственной Clarion Professional Developer 2.1 и Clarion 3.0 файловой системой.

#### Clipper Files

Файлы данных и индексные файлы, совместимые с Clipper 5.0 и выше.

#### dBase III Files

Файлы данных и индексов, совместимые с dBase III.

#### dBase IV Files

Файлы данных и индексов, совместимые с dBase IV.

#### DOS Files

Любой двоичный байт-адресуемый файл.

#### FoxPro and FoxBase Files

Файлы данных и индексов, совместимые с FoxBase и FoxPro.

#### ODBC - Open Data Base Connectivity

Любые файлы DBMS, которые поддерживают стандарт открытого интерфейса доступа к базам данных фирмы Microsoft

#### TopSpeed Database Files

Новый патентованный формат Clarion, быстрый, надежный, хранит множество файлов

данных и ключей в одном DOS файле.

The the TopSpeed Database Recovery Utility

Использование утилиты восстановления базы данных TopSpeed.

The TopSpeed Database Copy Utility

Использование утилиты копирования базы данных TopSpeed.

Имеются также другие драйверы файлов для Clarion for Windows . Более полную информацию вы получите, если обратитесь в Отдел обслуживания покупателей фирмы TopSpeed или к местным распространителям фирмы.

## Файлы ASCII

Драйвер ASCII читает и пишет стандартные ASCII файлы без разграничителей полей. Это часто используется для импорта/экспорта данных с больших машин через ASCII двухмерный файл. Записи разграничиваются символом возврата каретки/переводом строки. Драйвер ASCII не поддерживает ключи.

<b>Файлы:</b>	<b>CW2ASC16.LIB</b>	Библиотека экспорта Windows (16-битовая)
	<b>CW2ASC32.LIB</b>	Библиотека экспорта Windows (32-битовая)
	<b>CL2ASC16.LIB</b>	Статически вызываемая библиотека Windows (16-битовая)
	<b>CL2ASC32.LIB</b>	Статически вызываемая библиотека Windows (32-битовая)
	<b>CW2ASC16.DLL</b>	Динамически вызываемая библиотека Windows (16-битовая)
	<b>CW2ASC32.DLL</b>	Динамически вызываемая библиотека Windows (32-битовая)

**Совет:** Из-за отсутствия реляционных характеристик и ненадежности (любой может посмотреть и изменить файл ASCII с помощью Notepad), вы вряд ли будете использовать драйвер ASCII для хранения больших файлов данных. Но он может помочь вам при управлении полем списка или при создании программы для просмотра текстовых файлов - используйте его, чтобы открыть файл и читать его в многострочном редакторе.

## Поддерживаемые типы данных

---

STRING

GROUP

## Характеристики файла/максимумы

---

Размер файла:

4,294,967,295 байт

Записей на файл:	4,294,967,295 байт
Размер записи:	65,520 байт
Размер поля:	65,520 байт
Полей на запись:	65,520 байт
Ключей/Индексов на файл:	не имеется
Полей мемо на файл:	не имеется
Размер поля мемо:	не имеется
Открытие файла данных:	зависит от операционной системы.

## Разное

---

Тип данных возвращаемый функцией POSITION (POSITION Return Data Type)

- ★ POSITION (файл) возвращает строку символов STRING(4).

## Строки символов драйвера и функции SEND

---

Строкам символов драйвера, назначенным в объявлении FILE (второй параметр атрибута DRIVER (драйвер)) всегда предшествует косая черта (/). Они отсылаются драйверу файла когда приложение выдает команду OPEN (открыть). Нет возвращаемой величины, связанной с этими строками драйвера.

Функция SEND отсылает строки драйвера в любое время. Некоторые строки символов драйвера не действуют после открытия файла, поэтому синтаксис функции SEND для модификации установки не перечислен. Однако синтаксис функции SEND для возврата величины переключателя приводится в перечне для всех строк символов драйвера. Строки драйвера функции SEND принимают два формата - один со знаком равенства модифицирует установку переключателя и возвращает величину предыдущей установки ; другой формат (без знака равенства ) возвращает величину переключателя.

### /FILEBUFFERS=n

Устанавливает величину для числа буферов, используемых для чтения и записи в файл. Драйвер ASCII размещает внутренние буферы по 512 байт или по размеру записи, что больше. Число буферов по умолчанию 2 для файлов, открытых с запретом доступа по записи для других пользователей и 1 для всех других открытых режимов. Используйте необязательную строку символов драйвера, чтобы увеличить буферы, если вы найдете, что доступ к записям слишком медленен.

### SEND (file, 'FILEBUFFERS')

Возвращает СТРОКУ СИМВОЛОВ, содержащую число байт в буферах.



**/TAB= n**

Устанавливает расширение TAB/SPACE. Драйвер ASCII заменяет символы табуляции (ASCII символ 9) при чтении пробелами. Величина указывает число пробелов, которыми необходимо заменить символ табуляции в соответствии с указаниями, даваемыми ниже. Величина по умолчанию равна 8.

Если  $n > 0$ , пробелы заменяют каждый символ табуляции, до тех пор, пока указатель символа движется до следующего, кратного числу  $n$ . Например, с величиной по умолчанию 8, если символ TAB является третьим символом в записи, символ табуляции заменяется на 6 пробелов.

Если  $n = 0$ , драйвер перемещает символ табуляции без замены.

Если  $n < 0$ , драйвер перемещает символ табуляции с положительной величиной  $n$  пробелов. Например, "TAB=-4" вызывает 4 пробела для замены каждого символа табуляции, независимо от позиции символа в записи.

Если  $n = -100$ , символы табуляции остаются как есть; драйвер не заменяет их пробелами.

**SEND(file, 'TAB')**

Возвращает число пробелов, которые заменяют символ tab в форме строки символов STRING.

**/ENDOFRECORD=n,<m>**

Устанавливает конец разграничителя записи.

$n$  представляет число символов, которые образуют разделитель конца записи.

$m$  представляет ASCII код(ы) для символов конца записи, отделенный запятыми. По умолчанию 2,13,10, показывая тем самым, что 2 символа отмечают конец записи, возврат каретки (13) и перевод строки (10).

**SEND(file, 'ENDOFRECORD')**

Возвращает разграничитель конца записи в форме STRING.

**Совет:** Большие машины часто используют возврат каретки для разграничения записей. Вы можете использовать /ENDOFRECORD для

чтения этих файлов.

/QUICKSCAN=on|off

SEND (file, 'QUICKSCAN=on|off')

Устанавливает способ взаимодействия с буфером. Драйвер ASCII читает буфер за раз (не запись), давая тем самым возможность быстрого доступа. В многопользовательской среде этим буферам нельзя доверять на 100% для последующего доступа, так как другой пользователь может изменить файл между отдельными обращениями к данным. В качестве меры безопасности драйвер повторно читает данные в буфер перед доступом к каждой записи. Чтобы отключить повторное чтение, установите QUICKSCAN на ON. По умолчанию стоит ON для файлов, открытых с запретом доступа по записи для других пользователей, и OFF для всех других открытых режимов.

SEND(file, 'QUICKSCAN')

Возвращает установку QUICKSCAN (ON или OFF) в форме строки символов STRING(3).

/CLIP=on|off

Драйвер автоматически удаляет пробелы в конце записи перед записыванием ее в файл. Для отключения этой характеристики установите CLIP на OFF. По умолчанию стоит ON.

SEND(file, 'CLIP')

Возвращает установку CLIP (ON или OFF) в форме STRING(3).

## **Поддерживаемые атрибуты файла, команды и функции**

Атрибуты файла	Поддерживаемые
CREATE	Да
DRIVER (filetype,[driver string])	Да
NAME	Да
ENCRYPT	Нет
OWNER (password)	Нет
RECLAIM	Нет
PRE (prefix)	Да

BINDABLE	Да
THREAD	Да
EXTERNAL (member)	Да
DLL ([flag])	Да
OEM	Да
Файловые структуры	Поддерживаемые
INDEX	Нет
KEY	Нет
MEMO	Нет
BLOB	Нет
RECORD	Да
Атрибуты индекса, ключа, мемо	Поддерживаемые
BINARY	Нет
DUP	Нет
NOCASE	Нет
OPT	Нет
PRIMARY	Нет
NAME	Нет
Ascending Components	Нет
Descending Components	Нет
Mixed Components	Нет
Файловые команды	Поддерживаемые
BUILD(file)	Нет
BUILD (key)	Нет
BUILD (index)	Нет
BUILD (index,components)	Нет
BUILD (index,components,filter)	Нет
CLOSE (file)	Да
COPY (file,new file)	Да
CREATE (file)	Да
EMPTY (file)	Да
FLUSH (file)	Нет
LOCK (file)	Да
OPEN (file,access mode)	Да
PACK (file)	Нет
REMOVE (file)	Да
RENAME (file,new file)	Да
SHARE (file,access mode)	Да
STATUS (file)	Да
STREAM (file)	Нет

UNLOCK (file)	Да
Доступ к записям	Поддерживаемые
ADD (file)	Да
ADD (file,length)	Нет
APPEND (file)	Да
APPEND (file,length)	Нет
DELETE (file)	Нет
GET (file,key)	Нет
GET (file,filepointer)	Да
GET (file,filepointer,length)	Нет
GET (file,keypointer)	Нет
HOLD (file)	Нет
NEXT (file)	Да
NOMEMO (file)	Нет
PREVIOUS (file)	Нет
PUT (file)	Да
PUT (file,filepointer)	Да
PUT (file,filepointer,length)	Нет
RELEASE (file)	Нет
REGET ( file,string)	Да
REGET ( key,string)	Нет
RESET ( file,string)	Да
RESET ( key,string)	Нет
SET (file)	Да
SET (file,key)	Нет
SET (file,filepointer)	Да
SET (key)	Нет
SET (key,key)	Нет
SET (key,keypointer)	Нет
SET (key,key,filepointer)	Нет
SKIP (file,count)	Нет
WATCH (file)	Нет
Файловые функции	Поддерживаемые
BOF (file)	Нет
BYTES (file)	Да
DUPLICATE (file)	Нет
DUPLICATE (key)	Нет
EOF (file)	Да
NAME (label)	Да
POINTER (file)	Да
POINTER (key)	Нет

POSITION (file)	Да
POSITION (key)	Нет
RECORDS (file)	Нет
RECORDS (key)	Нет
SEND (file,message)	Да

Диалоговая обработка	Поддерживаемые
LOGOUT (timeout,file,...file)	Нет
COMMIT	Нет
ROLLBACK	Нет

Обработка пустых данных	Поддерживаемые
NULL (field)	Нет
SETNULL (field)	Нет
SETNONULL (field)	Нет

## Basic файлы

Драйвер файла BASIC читает и записывает разграниченные запятыми ASCII файлы. Кавычки окружают строки символов, запятая разграничивает поля, а возврат каретки/перевод строки разграничивает записи. Впервые этот формат файла был использован в языке программирования BASIC. Драйвер Basic не поддерживает ключи.

**Совет: Формат файла Basic обеспечивает хороший выбор для обычного формата файла с той целью, чтобы иметь общие данные с программами электронных таблиц. Обычное расширение файла, используемое для этих файлов - это \*.CSV, что означает величины, разделенные запятой.**

<b>Файлы:</b>	<b>CW2BAS16.LIB</b>	Библиотека экспорта Windows (16-битовая)
	<b>CW2BAS32.LIB</b>	Библиотека экспорта Windows (32-битовая)
	<b>CL2BAS16.LIB</b>	Статически вызываемая библиотека Windows (16-битовая)
	<b>CL2BAS32.LIB</b>	Статически вызываемая библиотека Windows (32-битовая)
	<b>CW2BAS16.DLL</b>	Динамически вызываемая библиотека Windows (16-битовая)
	<b>CW2BAS32.DLL</b>	Динамически вызываемая библиотека Windows (32-битовая)

## Поддерживаемые типы данных

BYTE	DECIMAL
SHORT	PDECIMAL
USHORT	STRING
LONG	CSTRING

ULONG	PSTRING
SREAL	DATE
REAL	TIME
BFLOAT4	GROUP
BFLOAT8	

**Характеристики файла/максимумы**

Размер файла:	4,294,967,295 байт
Записей на файл:	4,294,967,295 байт
Размер записи:	65,520 байт
Размер поля:	65,520 байт
Полей на запись:	65,520 байт
Ключей/Индексов на файл:	не имеется
Полей мемо на файл:	0
Размер поля мемо:	не имеется
Открытие файла данных:	зависит от операционной системы.

**Строки символов драйвера и функции SEND**

Строкам символов драйвера, назначенным в объявлении FILE (второй параметр атрибута DRIVER (драйвер)) всегда предшествует косая черта (/). Они отсылаются драйверу файла, когда приложение выдает команду OPEN (открыть). Нет возвращаемой величины , связанной с этими строками драйвера.

Функция SEND отсылает строки драйвера в любое время. Некоторые строки символов драйвера не действуют после открытия файла, поэтому синтаксис функции SEND для модификации установки не перечислен. Однако синтаксис функции SEND для возврата величины переключателя приводится в перечне для всех строк символов драйвера. Строки драйвера функции SEND принимают два формата - один со знаком равенства модифицирует установки переключателя и возвращает величину предыдущей установки переключателя; другой формат (без знака равенства ) возвращает величину переключателя.

**/FILEBUFFERS=n**

Устанавливает величину для числа буферов, используемых для чтения и записи в файл.

Драйвер Basic размещает внутренние буферы по 512 байт или по размеру записи, что больше, для хранения отысканных данных. Число буферов по умолчанию 2 для файлов, открытых с запретом доступа по записи для других пользователей и 1 для всех других открытых режимов. Используйте необязательную строку символов драйвера, чтобы увеличить буферы, если вы найдете, что доступ к записям слишком медленен.

**SEND (file, 'FILEBUFFERS')**

Возвращает число буферов в форме STRING.

**/ENDOFRECORD=n,<m>**

Устанавливает конец разделителя записи.

n представляет число символов разделителя конца записи.

t представляет ASCII код(ы) для символов конца записи, отделенных запятой. По умолчанию 2,13,10.

2 символа отмечают конец записи, а именно, возврат каретки (13) и перевод строки (10).

**SEND(file, 'ENDOFRECORD')**

Возвращает конец разграничителя записи в форме STRING.

**Совет: Большие машины часто используют возврат каретки для разграничения записей. Вы можете использовать /ENDOFRECORD для чтения этих записей.**

**/QUICKSCAN=on|off****SEND(file, 'QUICKSCAN=ON|OFF')**

Устанавливает способ взаимодействия с буфером. Драйвер Basic читает буфер за раз (не запись), давая тем самым возможность быстрого доступа. В многопользовательской среде этим буферам нельзя доверять на 100% для последующего доступа, так как другой пользователь может изменить файл между отдельными обращениями к данным. В качестве меры безопасности драйвер повторно читает данные в буфер перед доступом к каждой записи. Чтобы отключить повторное чтение, установите QUICKSCAN на ON. По умолчанию стоит ON для файлов, открытых с запретом доступа по записи для других пользователей, и OFF для всех других открытых режимов.

**SEND(file, 'QUICKSCAN')**

Возвращает QUICKSCAN установку (ON или OFF) в форме STRING(3).

**/FILEDELIMITER=n,<m>**

Устанавливает разделитель конца поля.

n представляет число символов, которые составляют разделитель конца поля.

m представляет ASCII код(ы) для символов конца поля, разделенных запятыми. По умолчанию 1,44, что указывает на символ “запятая”.

**SEND(file, 'FIELDDELIMITER')**

Возвращает величину разделителя поля в форме STRING.

**/COMMA=n**

Устанавливает символ разделителя конца поля. n представляет ASCII код для символа конца поля. По умолчанию 1,44, что эквивалентно “/FIELDDELIMITER=1,44”.

**SEND(file, 'COMMA')**

Возвращает ASCII код для символа разделителя конца поля в форме STRING.

**Совет: Величины, разграниченные TAB, представляют обычный формат, совместимый с буфером обмена Windows. Использование строки символов драйвера файла BASIC /COMMA=9 позволит вам читать файлы буфера обмена Windows.**

**/QUOTE=n**

Устанавливает символ разделителя строки символов.

n представляет ASCII код. По умолчанию 34, ASCII величина для символа двойной кавычки.

**SEND(file, 'QUOTE')**

Возвращает ASCII код величины символа разделителя строки символов в форме STRING.

**/ALWAYSQUOTE=on|off**

Для совместимости с файлами данных в формате Basic, созданными продуктами, которые не помещают строки символов в кавычки, поставьте ALWAYSQUOTE на OFF.

Когда содержание поля строки символов включает символ(ы) запятую или кавычку, а ALWAYSQUOTE выключено, драйвер Basic автоматически помещает кавычки вокруг строки символов при записывании в файл. Это также относится к символам-разделителям, установленным с FIELDDELIMITER или COMMA. Например, при использовании по умолчанию и при ALWAYSQUOTE выключенном, поле STRING содержащее величину 1313 Mockingbird Lane, Apt.33 автоматически сохраняется как: “1313 Mockingbird Lane, Apt.33”

**SEND(file, 'ALWAYSQUOTE')**

Возвращает ALWAYSQUOTE установку (ON или OFF) в форме STRING(3).



## Разное

---

### Популярные форматы файла

Следующие примеры демонстрируют, как использовать строки символов драйвера для создания двух популярных форматов файла mailmerge:

- ◆ Microsoft Word для Windows Mail Merge:

```
/ALWAYSQUOTE=OFF
/FILEDELIMITER=2,13,7
/ENDOFRECORD=4,13,7,13,7
```

- ◆ TAB-разделительный формат:
- ```
/COMMA=9
```

### Тип данных, возвращаемых функцией POSITION (POSITION Return Data Type)

- \* POSITION(file) (ПОЗИЦИЯ(файл)) возвращает строку символов STRING(4)

## Поддерживаемые атрибуты файла, команды и функции

---

### Атрибуты файла

CREATE .  
 DRIVER (filetype,[driver string])  
 NAME  
 ENCRYPT  
 OWNER (password)  
 RECLAIM  
 PRE (prefix)  
 BINDABLE  
 THREAD  
 EXTERNAL (member)  
 DLL ([flag])  
 OEM

### Поддерживаемые

Да  
 Да  
 Да  
 Нет  
 Нет  
 Нет  
 Да  
 Да  
 Да  
 Да  
 Да  
 Да

### Файловые структуры

INDEX  
 KEY  
 MEMO  
 BLOB  
 RECORD

### Поддерживаемые

Нет  
 Нет  
 Нет  
 Нет  
 Да

**Атрибуты индекса, ключа, мемо**

BINARY  
DUP  
NOCASE  
OPT  
PRIMARY  
NAME  
Ascending Components  
Descending Components  
Mixed Components

**Поддерживаемые**

Нет  
Нет  
Нет  
Нет  
Нет  
Нет  
Нет  
Нет  
Нет

**Файловые команды**

BUILD(file)  
BUILD (key)  
BUILD (index)  
BUILD (index,components)  
BUILD (index,components,filter)  
CLOSE (file)  
COPY (file,new file)  
CREATE (file)  
EMPTY (file)  
FLUSH (file)  
LOCK (file)  
OPEN (file,access mode)  
PACK (file)  
REMOVE (file)  
RENAME (file,new file)  
SHARE (file,access mode)  
STATUS (file)  
STREAM (file)  
UNLOCK (file)

**Поддерживаемые**

Нет  
Нет  
Нет  
Нет  
Нет  
Да  
Да  
Да  
Да  
Нет  
Да  
Да  
Нет  
Да  
Да  
Да  
Да  
Нет  
Да

**Доступ к записям**

ADD (file)  
ADD (file,length)  
APPEND (file)  
APPEND (file,length)  
DELETE (file)  
GET (file,key)  
GET (file,filepointer)  
GET (file,filepointer,length)  
GET (file,keypointer)

**Поддерживаемые**

Да  
Нет  
Да  
Нет  
Нет  
Нет  
Да  
Нет  
Нет

|                               |     |
|-------------------------------|-----|
| HOLD (file)                   | Нет |
| NEXT (file)                   | Да  |
| NOMEMO (file)                 | Нет |
| PREVIOUS (file)               | Нет |
| PUT (file)                    | Да  |
| PUT (file,filepointer)        | Да  |
| PUT (file,filepointer,length) | Нет |
| RELEASE (file)                | Нет |
| REGET ( file,string)          | Да  |
| REGET ( key,string)           | Нет |
| RESET ( file,string)          | Да  |
| RESET ( key,string)           | Нет |
| SET (file)                    | Да  |
| SET (file,key)                | Нет |
| SET (file,filepointer)        | Да  |
| SET (key)                     | Нет |
| SET (key,key)                 | Нет |
| SET (key,keypointer)          | Нет |
| SET (key,key,filepointer)     | Нет |
| SKIP (file,count)             | Нет |
| WATCH (file)                  | Нет |

**Файловые функции****Поддерживаемые**

|                     |     |
|---------------------|-----|
| BOF (file)          | Нет |
| BYTES (file)        | Да  |
| DUPLICATE (file)    | Нет |
| DUPLICATE (key)     | Нет |
| EOF (file)          | Да  |
| NAME (label)        | Да  |
| POINTER (file)      | Да  |
| POINTER (key)       | Нет |
| POSITION (file)     | Да  |
| POSITION (key)      | Нет |
| RECORDS (file)      | Нет |
| RECORDS (key)       | Нет |
| SEND (file,message) | Да  |

**Диалоговая обработка****Поддерживаемые**

|                               |     |
|-------------------------------|-----|
| LOGOUT (timeout,file,...file) | Нет |
| COMMIT                        | Нет |
| ROLLBACK                      | Нет |

**Обработка пустых данных**

NULL (field)  
SETNULL (field)  
SETNONULL (field)

**Поддерживаемые**

Нет  
Нет  
Нет

**Файлы Btrieve**

Этот драйвер файла читает и записывает файлы Btrieve, используя прямой низкоуровневый доступ.

В рамках Clarion for Windows драйвер файла Btrieve выполнен с использованием .DLL и .EXE, поставленного Btrieve Technology Inc.(BTI). Для приложения, которое должно использовать драйвер файлов Btrieve, исполняемая программа должна сопровождаться следующими BTI файлами:

**16-битовые**

**WBTR32.EXE**  
**WBTRLOCL.DLL**  
**WBTRCALL.DLL**  
**WBTRVRES.DLL**

**32-битовые**

Пожалуйста, купите 32-битовый процессор Btrieve у BTI и обратитесь к документации BTI для получения дополнительной информации о вашей конкретной 32-битовой платформе.

**ЛИЦЕНЗИОННОЕ ПРЕДУПРЕЖДЕНИЕ:** Зарегистрированный владелец Clarion for Windows не может распространять вышеупомянутые файлы BTI за пределами своей организации без лицензии от BTI. Для того, чтобы получить лицензию, пожалуйста обратитесь в:

**Btrieve Technologies, Inc.**

**5918 West Courtyard Drive, Suite 400**

**Austin, Texas 78730**

**Phone: (800)287-4383**

Для Btrieve на основе Клиент/Сервер Netware Btrieve является Сервер-ориентированной версией Btrieve, которая работает на сервере Novell. Программа запрашивающей стороны Btrieve BREQUEST.EXE должна быть загружена на каждой рабочей станции до запуска Windows.

Одиночный файл обычно хранит и данные и все ключи. Имена файлов данных по умолчанию получают расширение файла \*.DAT. По умолчанию драйвер сохраняет мемо-

поля в отдельном файле или в самом файле данных, если на это имеется указание в соответствующей строке символов драйвера.

Ключи KEYS - динамические и автоматически обновляются при изменении файла данных.

Индексы INDEXs сохраняются отдельно от файлов данных. Файлы INDEX получают временное имя файла и удаляются, когда программа заканчивается нормально. INDEXs - статичны - они не обновляются автоматически при изменении файлов данных. Оператор BUILD создает или обновляет индексные файлы.

Файл формата Btrieve сохраняет минимальную информацию о структуре файла в файле. Драйвер проверяет достоверность вашего описания относительно информации в файле. Возможно успешное открытие файла Btrieve, который имеет определения ключей, которые не совпадают точно с вашим определением. Вы должны быть уверены, что ваш файл и определения ключей точно соответствуют файлу Btrieve.

|               |                     |                                                         |
|---------------|---------------------|---------------------------------------------------------|
| <b>Файлы:</b> | <b>CW2BTR16.LIB</b> | Экспортная библиотека Windows (16-битовая)              |
|               | <b>CW2BTR32.LIB</b> | Экспортная библиотека Windows (32-битовая)              |
|               | <b>CL2BTR16.LIB</b> | Статически загружаемая библиотека Windows (16-битовая)  |
|               | <b>CL2BTR32.LIB</b> | Статически загружаемая библиотека Windows (32-битовая)  |
|               | <b>CW2BTR16.DLL</b> | Динамически загружаемая библиотека Windows (16-битовая) |
|               | <b>CW2BTR32.DLL</b> | Динамически загружаемая библиотека Windows (32-битовая) |

Имя Владельца подобно паролю. Шифрованный файл Btrieve использует имя владельца как шифровальный ключ.

## Типы данных

---

### Типы данных Clarion

BYTE  
SHORT  
LONG  
SREAL  
REAL  
BFLOAT4  
BFLOAT8  
PDECIMAL  
STRING

### Типы данных Btrieve

STRING (1 byte)  
INTEGER(2 bytes)  
INTEGER(4 bytes)  
FLOAT(4 bytes)  
FLOAT(8 bytes)  
BFLOAT(4 bytes)  
BFLOAT(8 bytes)  
DECIMAL  
STRING

|                                         |                                 |
|-----------------------------------------|---------------------------------|
| CSTRING                                 | ZSTRING                         |
| PSTRING                                 | LSTRING                         |
| DATE                                    | DATE                            |
| TIME                                    | TIME                            |
| USHORT                                  | UNSIGNED BINARY(2 bytes)        |
| ULONG                                   | UNSIGNED BINARY(4 bytes)        |
| MEMO                                    | STRING, IVAR или NOTE (см ниже) |
| BYTE, NAME('LOGICAL')                   | LOGICAL*                        |
| USHORT, NAME('LOGICAL')                 | LOGICAL*                        |
| PDECIMAL, NAME('MONEY')                 | MONEY*                          |
| STRING(@NO <sub>n</sub> -), NAME('STS') | SIGNED TRAILING SEPERATE*       |
| DECIMAL*                                |                                 |

### Замечания:

★ Вы можете сохранить типы Clarion DECIMAL в файле Btrieve. Однако, вы не можете построить ключ или индекс, используя это поле.

★ Если вы хотите создать файл с типами полей LOGICAL или MONEY, вы должны установить LOGICAL или MONEY в атрибуте NAME поля, соответственно. Если вы осуществляете доступ к существующему файлу, атрибут NAME не требуется.

Поле LOGICAL может быть объявлено как BYTE или USHORT, в зависимости от того, является ли оно одно- или двухбайтным LOGICAL:

```
LogicalField1 BYTE !One byte LOGICAL
LogicalField2 USHORT !Two byte LOGICAL
```

MONEY может быть объявлено как PDECIMAL(x,2), где x - это общее число разрядов для хранения:

```
MoneyField PDECIMAL(7,2), NAME('MONEY') !Store up to 99999,99
```

★ Поля Btrieve NUMERIC не полностью поддерживаются драйвером. Btrieve NUMERIC сохраняется, как строка символов с последним символом, содержащим разрядность и знак. Возможные величины для этого последнего символа:

```
1 2 3 4 5 6 7 8 9 0
Positive: A B C D E F G H I {
Negative: J K L M N O P Q R }
```

Чтобы получить доступ к полю NUMERIC, вы должны определить STRING(@NO<sub>x</sub>), где x на единицу меньше чем число разрядов в NUMERIC, и STRING (1) для индикатора

знака. Драйвер Btrieve не управляет этим знаковым полем, приложение должно быть написано так, чтобы непосредственно обслуживать его.

Например, для того, чтобы получить доступ к NUMERIC(7):

|              |              |                          |
|--------------|--------------|--------------------------|
| NumericGroup | GROUP        | !Store -999999 to 999999 |
| Number       | STRING(@NO6) | !Numbers                 |
| Sign         | STRING(1)    | !Sign indicator          |
| END          |              |                          |

### Характеристики файла/максимумы:

|                          |                              |
|--------------------------|------------------------------|
| Размер файла:            | 4,000,000,000 байт           |
| Записей на файл:         | Ограничено размером файла    |
| Размер записи клиента:   | 65,520 байт переменной длины |
| сервера                  | 54К переменной длины         |
| Размер поля:             | 65,520 байт                  |
| Полей на запись:         | 65,520 байт                  |
| Ключей/Индексов на файл: | 24 с NLM5                    |
| 256 с NLM6               |                              |
| Клиент Btrieve v.6.15    |                              |

#### Размеры

512  
1,024  
1,536  
2,048  
4,096

#### Макс. сегментов ключей

8  
23  
24  
54  
119

Это общее число компонент.  
Если у вас многокомпонентный ключ, построенный из трех полей, это считается, как три индекса при счете числа разрешенных индексов.

|                    |                                 |
|--------------------|---------------------------------|
| Размер ключа       | 255 байт                        |
| Полей мемо н файл: | Зависит от памяти системы       |
| Размер поля мемо:  | 65,520 байт                     |
| Открывание файлов: | Зависит от операционной системы |

## Строки символов драйвера и функции SEND

Строкам символов драйвера, назначенным в объявлении FILE (второй параметр атрибута DRIVER (драйвер)) всегда предшествует косая черта (/). Они отсылаются драйверу файла когда приложение выдает команду OPEN (открыть). Нет возвращаемой величины, связанной с этими строками драйвера.

Функция SEND отсылает строки драйвера в любое время. Некоторые строки символов драйвера не действуют после открытия файла, поэтому синтаксис функции SEND для модификации установки не перечислен. Однако синтаксис функции SEND для возврата величины переключателя приводится в перечне для всех строк символов драйвера. Строки драйвера функции SEND принимают два формата - один со знаком равенства модифицирует установки переключателя и возвращает величину предыдущей установки переключателя; другой формат (без знака равенства) возвращает величину переключателя.

### /MEMO=SINGLE

Чтобы получить доступ к существующим файлам Btrieve, созданным с помощью Btrieve LEM из Clarion 2.1, или файлам с переменной длиной записей, установите MEMO на SINGLE.

### /MEMO=LVAR

Чтобы получить доступ к файлу с переменной длиной записей, используйте MEMO в стиле SINGLE, чей размер равняется максимальному размеру компоненты записи переменной длины. Чтобы добавить/поместить записи к файлу этого стиля с бинарными данными, сохраненными в секции переменной длины, используйте ADD(файл,длина), APPEND(файл,длина) и PUT(файл,позиция,длина) функции. Драйвер игнорирует параметр позиция в функции PUT, но иницирует его в 0(нуль) для будущей совместимости. Функции ADD, APPEND или PUT удалят все концевые пробелы для текстовых мемо и NULL символы для бинарных мемо перед сохранением записи.

### /MEMO=NOTE,<delimiter>

Чтобы получить доступ к файлам данных Xtrieve, которые имеют тип данных Note или LVar.

При типе данных NOTE установите разграничитель конца поля. Установите величину ASCII для разграничителя. Мемо NOTE и LVAR не требуют использования размерных вариантов ADD, APPEND и PUT при сохранении записей. Маркер конца записи не обязателен для мемо стиля NOTE. Драйвер автоматически добавляет маркер конца записи перед сохранением записи и удаляет его перед помещением данных мемо в буфер мемо.

В качестве примера “/MEMO=NOTES,141” указывает файл с полем Xtrieve Notes, использующим CR/LF в качестве разграничителя. Более подробную информацию о типах данных Xtrieve обратитесь к документации, поставляемой Novell.



**SEND(file,'MEMO')**

Возвращает установку MEMO: NORMAL, NOTE, LVAR или SINGLE.

**/PAGESIZE=<size>**

В качестве необязательного параметра устанавливает размер Btrieve Page во время создания файла. Ключевое слово должно быть в верхнем регистре. Оно должно быть всегда кратным 512 с максимумом 4096. Большой размер страницы обычно приводит к более эффективному хранению на диске. Не прибавляйте пробелов перед или после знака равенства.

**SEND(file,'PAGESIZE')**

Возвращает размер страницы в форме STRING.

**/ALLOWREAD=[ON|OFF]**

По умолчанию к файлу Btrieve, созданному с именем владельца, можно получить доступ только в режиме “только-для-чтения” если имя владельца неизвестно. Чтобы предупредить любой доступ к файлу без имени владельца, установите ALLOWREAD на OFF.

**SEND(file,'ALLOWREAD')**

Возвращает ALLOWREAD установку (ON или OFF) в форме STRING(3).

**/COMPRESS=[ON|OFF]**

Btrieve дает вам возможность сжать данные перед хранением. Это позволяет снизить требования к хранению, но одновременно уменьшает эффективность. Когда COMPRESS в позиции ON, CREATE создает сжатый файл Btrieve.

**SEND(file,'COMPRESS')**

Возвращает COMPRESS установку (ON или OFF) в форме STRING(3).

**/PREALLOCATE=n**

При создании файла Btrieve вы можете предусмотреть n страниц пространства диска для размещения файла. Установка по умолчанию нуль.

**SEND(file,'PREALLOCATE')**

Возвращает число страниц предусмотренного пространства диска в форме STRING.

**/FREESPACE=[0|10|20|30]**

Устанавливает процент свободного пространства для поддержания страниц переменной длины. Величина по умолчанию нуль.

**SEND(file, 'FREESPACE')**

Возвращает процент свободного пространства для поддержания страниц переменной длины в форме STRING.

**/ACS=file\_name**

При создании файла Btrieve вы можете установить альтернативную сортирующую последовательность, в которой будут отсортированы ключи STRING. Эта сортирующая последовательность нормально получается из определяемой в файле INI вашей программы. Однако Btrieve поставляет файлы для сортировок, нечувствительных к регистру. Чтобы создать ваш файл, используя эти сортирующие последовательности, установите имя сортирующего файла в строке символов драйвера.

Например. Чтобы использовать альтернативный файл сортирующей последовательности UPPER.ALT, вы должны установить:

```
AFile FILE, DRIVER('BTRIEVE', '/ACS=UPPER.ALT'), CREATE
```

**/APPENDBUFFER=size****SEND(file, 'APPENDBUFFER=size')**

По умолчанию APPEND добавляет все записи к файлу за один раз. Чтобы получить лучшую эффективность в сети, вы можете приказать драйверу построить буфер записей, затем отправьте их все сразу к Btrieve. Это делается с помощью SEND(file, 'APPENDBUFFER=size'), где size - число записей, которые вы хотите разместить в буфер.

**SEND(file, 'APPENDBUFFER')**

Возвращает число записей, которые будут готовы в буфере.

**/BALANCEKEYS=[ON|OFF]')**

При создании файла Btrieve вы можете использовать эту строку символов драйвера, чтобы сказать Btrieve, что все ключи, связанные с файлом, должны быть сохранены в виде сбалансированного двоичного дерева. Это экономит пространство диска, но замедлит выполнение операций добавления, удаления и обновления там, где изменяются величины ключей.

**SEND(file, 'BALANCEKEYS')**

Возвращает установки BALANCEKEYS (ON или OFF) в форме STRING(3).

**SEND(file, 'FREEAPPENDBUFFER')**

Освобождает память, используемую буфером присоединения, размещенным при вызове

SEND(file,APPENDBUFFER=size).Возвращает число записей, которые помещены в старый буфер.

### /LACS=

В версии Btrieve 6.15 Btrieve добавлена характеристика Local Alternate Collating Sequences (локальные переменные сортирующие последовательности). Это позволяет вашему ключу в виде строки символов сортировать на основе кода страны для машины, выполняющей вашу программу. Чтобы использовать эту характеристику, поместите '/LACS=' в строку символов вашего драйвера.

### /LACS=country\_ID,code\_page

В версии Btrieve 6.15 Btrieve добавлена характеристика Определяемые пользователем Alternate Collating Sequences (переменные сортирующие последовательности). Это позволяет вашему ключу в виде строки символов сортировать на основе DOS кода страны и кода страницы для конкретной страны. Чтобы использовать эту характеристику, поместите '/LACS=country\_id,codepage' в строку символов вашего драйвера. Запомните, что не должно быть пробелов вокруг запятой.

### SEND(file,'LACS')

Возвращает country\_ID,code\_page или строку символов ',' (если используется зависящее от машины LACS).

### /TRUNCATE=[ON|OFF]

При создании файла Btrieve вы можете использовать эту строку символов драйвера, чтобы сообщить Btrieve о необходимости отсечения остаточных пробелов. Это принуждает к тому, чтобы запись хранилась, как хранятся записи переменной длины.

### SEND(file,'TRUNCATE')

Возвращает установку (ON или OFF) для TRUNCATE в форме STRING(3).

## **Поддерживаемые атрибуты файла, команды и функции**

### Атрибуты файла

CREATE .  
DRIVER (filetype,[driver string])  
NAME  
ENCRYPT  
OWNER (password)  
RECLAIM  
PRE (prefix)  
BINDABLE

### Поддерживаемые

Да  
Да  
Да  
Да  
Да  
Да  
Да  
Да

|                   |    |
|-------------------|----|
| THREAD            | Да |
| EXTERNAL (member) | Да |
| DLL ([flag])      | Да |
| OEM               | Да |

**Файловые структуры****Поддерживаемые**

|        |     |
|--------|-----|
| INDEX  | Да  |
| KEY    | Да  |
| MEMO   | Да+ |
| BLOB   | Нет |
| RECORD | Да  |

**Атрибуты индекса, ключа, мемо****Поддерживаемые**

|                       |     |
|-----------------------|-----|
| BINARY                | Да  |
| DUP                   | Да  |
| NOCASE                | Да  |
| OPT                   | Да  |
| PRIMARY               | Да  |
| NAME                  | Да+ |
| Ascending Components  | Да  |
| Descending Components | Да  |
| Mixed Components      | Да  |

**Файловые команды****Поддерживаемые**

|                                 |      |
|---------------------------------|------|
| BUILD(file)                     | Да1  |
| BUILD (key)                     | Да1  |
| BUILD (index)                   | Да1  |
| BUILD (index,components)        | Да1  |
| BUILD (index,components,filter) | Нет  |
| CLOSE (file)                    | Да   |
| COPY (file,new file)            | Да   |
| CREATE (file)                   | Да   |
| EMPTY (file)                    | Да   |
| FLUSH (file)                    | Да   |
| LOCK (file)                     | Нет2 |
| OPEN (file,access mode)         | Да   |
| PACK (file)                     | Да   |
| REMOVE (file)                   | Да   |
| RENAME (file,new file)          | Да   |
| SHARE (file,access mode)        | Да   |
| STATUS (file)                   | Да   |
| STREAM (file)                   | Да   |

UNLOCK (file)

Да

**Доступ к записям****Поддерживаемые**

ADD (file)

Да4

ADD (file,length)

Да4

APPEND (file)

Да5

APPEND (file,length)

Да4,5

DELETE (file)

Да6

GET (file,key)

Да

GET (file,filepointer)

Да

GET (file,filepointer,length)

Нет

GET (file,keypointer)

Да

HOLD (file)

Да

NEXT (file)

Да

NOMEMO (file)

Да

PREVIOUS (file)

Да

PUT (file)

Да4

PUT (file,filepointer)

Нет

PUT (file,filepointer,length)

Да

RELEASE (file)

Да

REGET ( file,string)

Да

REGET ( key,string)

Да

RESET ( file,string)

Да

RESET ( key,string)

Да

SET (file)

Да

SET (file,key)

Да

SET (file,filepointer)

Да7

SET (key)

Да

SET (key,key)

Да

SET (key,keypointer)

Да7

SET (key,key,filepointer)

Да8

SKIP (file,count)

Да

WATCH (file)

Да

**Файловые функции****Поддерживаемые**

BOF (file)

Да9

BYTES (file)

Нет

DUPLICATE (file)

Да

DUPLICATE (key)

Да

EOF (file)

Да9

NAME (label)

Да

POINTER (file)

Да

POINTER (key)

Да

|                     |    |
|---------------------|----|
| POSITION (file)     | Да |
| POSITION (key)      | Да |
| RECORDS (file)      | Да |
| RECORDS (key)       | Да |
| SEND (file,message) | Да |

**Диалоговая обработка****Поддерживаемые**

|                               |     |
|-------------------------------|-----|
| LOGOUT (timeout,file,...file) | Да3 |
| COMMIT                        | Да  |
| ROLLBACK                      | Да  |

**Обработка пустых данных****Поддерживаемые**

|                   |     |
|-------------------|-----|
| NULL (field)      | Нет |
| SETNULL (field)   | Нет |
| SETNONULL (field) | Нет |

**Замечания**

+ Драйвер игнорирует любой атрибут NAME поля MEMO. Поля MEMO могут находиться либо в отдельном файле, либо в файле данных, если строка драйвера MEMO установлена на SINGLE, LVAR или NOTE. Если строка драйвера MEMO не установлена, имя отдельного файла MEMO - "MEM", предваряемое первыми пятью символами метки файла, плюс расширение файла "DAT". Установка строки драйвера MEMO ограничивает вас одним мемо полем на файл.

1 Если используется после APPEND(), но прежде, чем файл закрыт, это добавляет ключи, опущенные APPEND(). Во всех других случаях BUILD() перестраивает файл и ключи. Если вы хотите только перестроить ключи, выполнить BUILD(ключ) для каждого ключа быстрее, чем BUILD(файл).

2 Btrieve прямо не поддерживает захват файла, Если вам требуется захват файла, используйте LOGOUT.

3 Когда вы выдаете вызов LOGOUT(), все файлы Btrieve, к которым был доступ во время транзакции, выводятся из системы. Это означает, что следующий фрагмент программы некорректен (так как вы не можете закрыть выведенный из системы файл:

```
LOGOUT(1,file1)
OPEN(file2)
CLOSE(file2)
```

4 При использовании мемо типа LVAR или NOTE, убедитесь, что мемо имеет подходящую структуру. Если структура неправильная, а драйвер рассчитывает длину большую, чем

максимальный размер мемо, определенный для данного файла, эти функции отказывают и поэтому устанавливайте код ошибки на 57 - Недопустимый файл мемо.

5 Btrieve не поддерживает бесключевые изменения. Чтобы эмулировать поведение APPEND(), драйвер опускает все возможные индексы при первом вызове. Вызов BUILD() немедленно после дополнения записей перестраивает опущенные ключевые поля.

6 DELETE Btrieve разрушает информацию позиционирования при обработке в порядке записей файлов. Драйвер пытается изменить позицию к соответствующей записи. Это не всегда возможно и может потребовать от драйвера читать с начала файла. Использование ключевого порядка обработки позволяет избежать этого возможного замедления.

7 Если указатель файла или указатель ключа имеют величину нуль, драйвер игнорирует параметр указателя. Обработка устанавливается или на файловый, либо на ключевой порядок, указатель записи устанавливается на первый элемент.

8 Если указатель файла имеет величину нуль, обработка начинается на величине первого ключа, чья позиция больше, чем (или меньше чем для PREVIOUS) указатель файла.

9 Эти функции поддерживаются, но они не рекомендуются. Они вызывают большее количество вводов/выводов с диска, чем ERRORCODE(). Btrieve возвращает EOF при чтении после последней записи. Следовательно, драйвер должен читать следующую запись, а затем следующую, чтобы увидеть, конец ли это файла, а затем вернуться к записи, которая вам нужна.

## Разное

---

### Длины записей

★ Драйвер сохраняет записи менее 4К как записи фиксированной длины. Записи более 4К он сохраняет как записи переменной длины. Минимальная длина записи 4 байта. Одна запись может удерживаться в каждом открытом файле каждым пользователем.

### Размер страницы

★ Чтобы определить физическую длину записи, прибавьте 8 байт для каждого KEY, который разрешает дублирование. Прибавьте 4 байта, если файл допускает переменную длину записи. Наконец, прибавьте 6 байт для overhead на страницу.

Например: Если размер записи равен 300 байт, а файл имеет три ключа KEY, которые допускают дублирование, полный размер записи равен:

|   |     |                                       |
|---|-----|---------------------------------------|
|   | 300 | размер записи                         |
| x | 24  | overhead для трех KEY с атрибутом DUP |
| = | 324 | физическая длина записи               |

Размер страницы 512 содержит только одну такую запись, а 182 байта на страницу останутся неиспользованными (512-6-324). Если бы размер страницы был 1024, на страницу пришлось бы три сохраненных записи и только 46 байт остались бы неиспользованными (1024-6-(324\*3)).

Вы должны загрузить BTRIEVE.EXE с размером страницы, равным или большим, чем самый большой размер страницы любого файла, к которому вы получите доступ.

### **Совместное использование файла**

★ Btrieve позволяет вам открыть файл в пяти различных форматах: NORMAL, ACCELERATED, READ-ONLY, VERIFY или EXCLUSIVE. Эквивалентные состояния Clarion OPEN:

#### **Btrieve State**

ACCELERATED  
READ-ONLY  
VERIFY  
EXCLUSIVE

#### **Clarion режим доступа OPEN/SHARE**

Читать/записывать с FCB режимом совместимости (2H)  
Только читать (0H,10H,20H,30H,40H)  
Только записывать с FCB совместимостью (1H)  
Только писать с любым флажком отрицания Deny  
(11H,21H,31H,41H);  
Читать/писать с Deny All, читать и писать (12H,22H,32H)  
Читать/писать с Deny None (не отрицать никого)(42H)

NORMAL

★ Btrieve позволяет файлу иметь особого владельца. Смотрите строку символов драйвера /READONLY относительно деталей установки этого флажка. Файл может быть также зашифрован. Это устанавливается с помощью атрибута ENCRYPT. Файл может быть зашифрован только тогда, когда дано имя владельца.

### **Указатель записи**

★ Btrieve использует незаконную длинную переменную для своего внутреннего указателя записи; отрицательные величины лишаются своего знака. Мы рекомендуем тип данных ULONG для вашего указателя записи.

### **Последовательность сортировки**

★ Атрибут ключа: NOCASE

NLM 5 не поддерживает нечувствительное к регистру индексирования. При необходимости вы должны иметь переменную, сортирующую последовательность, которая нечувствительна к регистру сортировки.

Btrieve поддерживает переменную, сортирующую последовательность. Однако NLM 6 не поддерживает ни NOCASE, ни переменную, сортирующую последовательность. Если вы установите и то и другое, преимущество имеет атрибут NOCASE. Из функции SEND не



возвращается никакая ошибка.

### **Определения КЛЮЧА**

★ При определении файла, определение ключа не требует точного совпадения глубинного строения файла. Например, вы можете иметь физический файл с единственной компонентой `STRING (20)`. Вы можете определить это как ключ с двумя строчными компонентами общей длиной 20. Правило таково, что типы данных должны совпадать и должен совпадать общий размер. Однако, если Clariion определение не точно совпадает с глубинным файлом, драйвер не может оптимизировать операторы `APPEND()` или `BUILD()`.

★ Атрибут Ключа `NAME` может прибавить дополнительные функциональные возможности.

#### **KEY,NAME( 'MODIFIABLE=true|false' )**

`Btrieve` дает вам возможность создать ключ, который не может быть изменен, будучи создан. Чтобы использовать эту характеристику, вы можете воспользоваться атрибутом имени при ключе для установки `MODIFIABLE` на `FALSE`. По умолчанию это установится на `TRUE`.

#### **KEY,NAME( 'ANYNULL' )**

`Btrieve` позволяет вам создать ключ, который не будет включать запись, если одна какая-то компонента его нулевая. Чтобы создать такой ключ, установите `ANYNULL` в имени ключа.

Например, чтобы создать ключ, который неизменим и исключить ключи, если какая-либо компонента равна нулю:

```
Key1 KEY(+pre:field,-pre:field2),NAME('ANYNULL MODIFIABLE=FALSE')
```

#### **KEY,NAME('REPEATINGDUPLICATE')**

По умолчанию `Btrieve 6` сохраняет ссылку только для первой записи в серии повторяющихся записей в ключе. Другие появления дубликата величины ключа получены следованием за списком связи, сохраненном на записи. Чтобы создать индекс, где все дублирующие записи сохранены в ключе, используйте `NAME('REPEATINGDUPLICATE')`. Это дает более крупные ключи, но случайный доступ к дубликатным записям быстрее. (Эта характеристика имеется только для файлов версии 6).

### Тип данных возвращаемый функцией POSITION (POSITION Return Data Type)

- ★ POSITION(File) возвращает STRING(4).
- ★ POSITION(Key) возвращает STRING, содержащую размер поля ключей + 4 байта.

## Файлы Clarion

Драйвер файла Clarion совместим с файловой системой, используемой Clarion for DOS 3.1 и Clarion Professional Developer 2.1.

Ключи и индексы существуют как файлы, отдельные от файла данных. Ключи динамичны - они автоматически обновляются при изменении файлов данных. Расширение файла ключа по умолчанию - \*.К##. Индексы статичны - они не подвержены автоматическому обновлению и обновляются при помощи оператора BUILD.

Драйвер сохраняет записи, как записи фиксированной длины. МЕМО поля хранятся в отдельном файле. По умолчанию имя файла мемо - первые восемь символов метки файла и расширение .MEM.

|               |                     |                                                         |
|---------------|---------------------|---------------------------------------------------------|
| <b>Файлы:</b> | <b>CW2CLA16.LIB</b> | Экспортная библиотека Windows (16-битовая)              |
|               | <b>CW2CLA32.LIB</b> | Экспортная библиотека Windows (32-битовая)              |
|               | <b>CL2CLA16.LIB</b> | Статически загружаемая библиотека Windows (16-битовая)  |
|               | <b>CL2CLA32.LIB</b> | Статически загружаемая библиотека Windows (32-битовая)  |
|               | <b>CW2CLA16.DLL</b> | Динамически загружаемая библиотека Windows (16-битовая) |
|               | <b>CW2CLA32.DLL</b> | Динамически загружаемая библиотека Windows (32-битовая) |

**Совет:** За исключением только некоторых ASCII форматов некоторых популярных PC систем разработки приложений баз данных, формат файла Clarion обеспечивает наиболее надежные средства хранения данных.

### Типы данных

|       |                            |
|-------|----------------------------|
| BYTE  | DECIMAL                    |
| SHORT | STRING (255 байт максимум) |
| LONG  | MEMO                       |
| REAL  | GROUP                      |

### Максимальные характеристики файла

|                  |                                 |       |
|------------------|---------------------------------|-------|
| Размер файла:    | ограничивается только местом на | диске |
| Записей на файл: | 4,294,967,295                   |       |

|                                       |                                 |
|---------------------------------------|---------------------------------|
| Размер записи:                        | 65,520 байт                     |
| Размер поля:                          | 65,520 байт                     |
| Полей на запись:                      | 65,520 байт                     |
| Ключей/индексов<br>на файл:           | 251                             |
| Размер ключа:                         | 245 байт                        |
| Полей мемо на файл:                   | 1                               |
| Размер поля мемо:                     | 65,520 байт                     |
| Количество открытых<br>файлов данных: | зависит от операционной системы |

### **Строки символов драйвера и функции SEND**

Строкам символов драйвера, назначенным в объявлении FILE (второй параметр атрибута DRIVER (драйвер)) всегда предшествует косая черта (/). Они отсылаются драйверу файла когда приложение выдает команду OPEN (открыть). Нет возвращаемой величины, связанной с этими строками драйвера.

Функция SEND отсылает строки драйвера в любое время. Некоторые строки символов драйвера не действуют после открытия файла, поэтому синтаксис функции SEND для модификации установки не перечислен для них. Однако синтаксис функции SEND для возврата величины переключения приводится в перечне для всех строк символов драйвера. Строки драйвера функции SEND принимают два формата - один со знаком равенства модифицирует установки переключателя и возвращает величину предыдущей установки переключателя; другой формат (без знака равенства) возвращает величину переключателя.

#### **SEND(file, 'RECOVER=n')**

Строка символов RECOVER (восстановление), когда n больше 0, разблокирует (UNLOCK) файл данных или освобождает (RELEASE), удерживаемую запись для того, чтобы восстановиться от разрушения системы.

n представляет число или секунды ожидания перед вызовом процесса восстановления. Когда n равно нулю, процесс восстановления отключен.

Функция SEND возвращает пустую строку символов.

Чтобы освободить (RELEASE), удерживаемую запись, вы должны прочитать эту запись в памяти. Если удерживаемых записей много, необходимо последовательно прочитать весь файл после отсылки SEND драйверу сообщения=RECOVER.

RECOVER не может быть использовано в качестве строки драйвера - вы можете его использовать только с функций SEND.

#### **SEND(file, 'IGNORESTATUS=on|off')**

**/IGNORESTATUS=on|off**

Будучи установлен ON, драйвер не пропускает удаленные записи при доступе к файлу с помощью GET(),NEXT() и PREVIOUS() в порядке файла. Эта позиция также позволяет поместить PUT() на удаленную или удержанную запись. /IGNORESTATUS требует открыть файл в эксклюзивном режиме.

**SEND(file,'IGNORESTATUS')**

Возвращает установку (ON или OFF) IGNORESTATUS в форме строки символов STRING(3).

**SEND(file,'DELETED')**

Для использования только с командой SEND, когда включен (ON) /IGNORESTATUS. Сообщает состояние загруженной записи. Если удалена, возвращаемая строка символов - "ON", если нет - "OFF".

**SEND(file,'HELD')**

Для использования только с командой SEND, когда включен (ON) /IGNORESTATUS. Сообщает состояние загруженной записи. Если удержана, возвращенная строка символов - "ON", если нет - "OFF".

**Разное**

Тип данных возвращаемый функцией POSITION (POSITION Return Data Type)

- ★ POSITION(File) возвращает строку символов STRING(4).
- ★ POSITION(Key) возвращает строку символов STRING, содержащую размер полей ключа + 4 байта.

**Поддерживаемые атрибуты файла, команды и функции****Атрибуты файла****Поддерживаемые**

|                                   |    |
|-----------------------------------|----|
| CREATE .                          | Да |
| DRIVER (filetype,[driver string]) | Да |
| NAME                              | Да |
| ENCRYPT                           | Да |
| OWNER (password)                  | Да |
| RECLAIM                           | Да |
| PRE (prefix)                      | Да |
| BINDABLE                          | Да |
| THREAD                            | Да |
| EXTERNAL (member)                 | Да |
| DLL ([flag])                      | Да |
| OEM                               | Да |

## Файловые структуры

|        |     |
|--------|-----|
| INDEX  | Да  |
| KEY    | Да  |
| MEMO   | Да  |
| BLOB   | Нет |
| RECORD | Да  |

## Поддерживаемые

Да  
Да  
Да  
Нет  
Да

### Атрибуты индекса, ключа, мемо

|                       |     |
|-----------------------|-----|
| BINARY                | Да  |
| DUP                   | Да  |
| NOCASE                | Да  |
| OPT                   | Да  |
| PRIMARY               | Да  |
| NAME                  | Да  |
| Ascending Components  | Да  |
| Descending Components | Нет |
| Mixed Components      | Нет |

## Поддерживаемые

Да  
Да  
Да  
Да  
Да  
Да  
Нет  
Нет

## Файловые команды

|                                 |     |
|---------------------------------|-----|
| BUILD(file)                     | Да  |
| BUILD (key)                     | Да  |
| BUILD (index)                   | Да  |
| BUILD (index,components)        | Да  |
| BUILD (index,components,filter) | Нет |
| CLOSE (file)                    | Да  |
| COPY (file,new file)            | Да  |
| CREATE (file)                   | Да  |
| EMPTY (file)                    | Да  |
| FLUSH (file)                    | Да  |
| LOCK (file)                     | Да  |
| OPEN (file,access mode)         | Да  |
| PACK (file)                     | Да  |
| REMOVE (file)                   | Да  |
| RENAME (file,new file)          | Да  |
| SHARE (file,access mode)        | Да  |
| STATUS (file)                   | Да  |
| STREAM (file)                   | Да  |
| UNLOCK (file)                   | Да  |

## Поддерживаемые

[illegible]

**Доступ к записям**

ADD (file)  
ADD (file,length)  
APPEND (file)  
APPEND (file,length)  
DELETE (file)  
GET (file,key)  
GET (file,filepointer)  
GET (file,filepointer,length)  
GET (file,keypointer)  
HOLD (file)  
NEXT (file)  
NOMEMO (file)  
PREVIOUS (file)  
PUT (file)  
PUT (file,filepointer)  
PUT (file,filepointer,length)  
RELEASE (file)  
REGET ( file,string)  
REGET ( key,string)  
RESET ( file,string)  
RESET ( key,string)  
SET (file)  
SET (file,key)  
SET (file,filepointer)  
SET (key)  
SET (key,key)  
SET (key,keypointer)  
SET (key,key,filepointer)  
SKIP (file,count)  
WATCH (file)

**Поддерживаемые**

Да  
Нет  
Да  
Нет  
Да  
Да  
Нет  
Да  
Да  
Да  
Да  
Да  
Нет  
Да  
Да  
Да  
Да  
Да  
Да  
Да  
Да  
Да  
Да  
Да  
Да  
Да  
Да  
Да  
Да

**Файловые функции**

BOF (file)  
BYTES (file)  
DUPLICATE (file)  
DUPLICATE (key)  
EOF (file)  
NAME (label)  
POINTER (file)  
POINTER (key)  
POSITION (file)

**Поддерживаемые**

Да1  
Да  
Да  
Да  
Да1  
Да  
Да  
Да  
Да

|                     |    |    |
|---------------------|----|----|
| POSITION (key)      | Да |    |
| RECORDS (file)      | Да |    |
| RECORDS (key)       | Да |    |
| SEND (file,message) |    | Да |

**Диалоговая обработка****Поддерживаемые**

|                               |    |
|-------------------------------|----|
| LOGOUT (timeout,file,...file) | Да |
| COMMIT                        | Да |
| ROLLBACK                      | Да |

**Обработка пустых данных****Поддерживаемые**

|                   |     |
|-------------------|-----|
| NULL (field)      | Нет |
| SETNULL (field)   | Нет |
| SETNONULL (field) | Нет |

**Замечания**

1 Эти функции поддерживаются, но не рекомендуются из-за отсутствия поддержки в других файловых системах. NEXT и PREVIOUS посылают Error 33 если вы попытаетесь читать за концом файла.

**Файлы Clipper**

Драйвер файла Clipper совместим с Clipper Summer '87 и Clipper 5.0. Расширение файла данных по умолчанию \*.DBF.

Ключи и индексы существуют, как файлы, отдельные от файла данных. Ключи динамические - они автоматически обновляются при изменении файла данных. Индексы статические - они не обновляются автоматически, а вместо этого требуют для обновления выполнения оператора BUILD. Расширение индексного файла по умолчанию \*.NTX .

Драйвер сохраняет записи, как записи фиксированной длины. Поля мемо хранятся в отдельном файле. Имя мемо файла - первые восемь символов метки файла с расширением .DBT.

|               |                     |                                                         |
|---------------|---------------------|---------------------------------------------------------|
| <b>Файлы:</b> | <b>CW2CLP16.LIB</b> | Экспортная библиотека Windows (16-битовая)              |
|               | <b>CW2CLP32.LIB</b> | Экспортная библиотека Windows (32-битовая)              |
|               | <b>CL2CLP16.LIB</b> | Статически загружаемая библиотека Windows (16-битовая)  |
|               | <b>CL2CLP32.LIB</b> | Статически загружаемая библиотека Windows (32-битовая)  |
|               | <b>CW2CLP16.DLL</b> | Динамически загружаемая библиотека Windows (16-битовая) |
|               | <b>CW2CLP32.DLL</b> | Динамически загружаемая библиотека Windows (32-битовая) |

**Совет:** Будучи популярной системой разработки приложений баз данных xBase, Clipper обеспечивает обычный формат файла для многих установленных деловых приложений и их файлов данных. Используйте драйвер Clipper, чтобы получить доступ к этим файлам в их родном формате.

## Типы данных

Формат файла xBase сохраняет все данные в виде строк символов ASCII. Вы можете установить типы строк символов STRING с помощью символьных шаблонов для каждого поля, либо установить типы данных Clarion, которые драйвер преобразует автоматически.

| <u>Тип данных Clipper</u> | <u>Тип данных Clarion</u> | <u>символьный шаблон</u> |
|---------------------------|---------------------------|--------------------------|
| Date                      | DATE                      | STRING(@D12)             |
| *Numeric                  | REAL                      | STRING(@N-_p.d)          |
| *Logical                  | BYTE                      | STRING(1)                |
| Character                 | STRING                    | STRING                   |
| *Memo                     | MEMO                      | MEMO                     |

Если ваше приложение читает и записывает в существующие файлы, достаточен будет символьный шаблон. Однако, если ваше приложение создает Clipper файл, вам может потребоваться дополнительная информация для этих типов данных Clipper:

★ Чтобы создать цифровое поле в словаре данных, выберите тип данных REAL (действительный). В поле Внешнее имя на закладке Атрибуты установите 'NumericFieldName=N(Precision,DecimalPlaces)' где NumericFieldName - имя поля, Precision - точность поля, а DecimalPlaces - число десятичных мест. С данными типа REAL вы не можете получить доступ к полям символа или мест в определении поля, вам следует назначить новые атрибуты с помощью выражения во Внешнем имени поля на закладке Атрибуты.

Например, если вы хотите создать поле, называемое Number (номер) с девятью значащими цифрами и двумя десятичными знаками, введите 'Number=N(9,2)' во внешнем имени поля на закладке Атрибуты свойств поля в словаре данных.

Если вы кодируете вручную присущие Clarion типы данных, добавьте атрибут NAME используя тот же самый синтаксис. Если вы кодируете вручную символьный шаблон, STRING@N-\_9.2),NAME('Number'),где Number - имя поля.

Чтобы создать логическое поле, используя словарь данных, выберите тип данных BYTE. Специальных шагов нет; однако, смотрите в разных разделах советы о чтении данных из поля.



★ Если вы вручную кодируете символьный шаблон, добавьте атрибут NAME: STRING(1),NAME('LogFld=L').

★ Чтобы создать поле данных, используя словарь данных, выберите тип данных DATE, а не LONG, который вы обычно используете для форматов файлов TopSpeed или Clarion.

★ Объявления поля MEMO требуют указателя поля в файловой структуре записи. Объявите указатель поля как STRING(10) или LONG. Это поле будет сохранено в файле .DBF, содержащем смещение мемо в файле .DBT. Объявление MEMO должно иметь атрибут NAME(), называющий указатель поля. Пример объявления поля:

```
File FILE, DRIVER('Clipper')
Memo1 MEMO(200), NAME('Notes')
Memo2 MEMO(200), NAME('Text')
Rec RECORD
Mem1Ptr LONG, NAME('Notes')
Mem2Ptr STRING(10), NAME('Text')
END
END
```

**Совет:** Когда это возможно, пользуйтесь утилитой импорта файла (File Import Utility) в редакторе словаря Clarion for Windows для того, чтобы создать описание своих файлов.

## Характеристики файла/максимумы

|                          |                                       |
|--------------------------|---------------------------------------|
| Размер файла:            | 2,000,000,000 байт                    |
| Записей на файл:         | 1,000,000,000 байт                    |
| Размер записи:           | 4,000 байт (Clipper '87)              |
|                          | 8,192 байт (Clipper 5.0)              |
| Размер поля              |                                       |
| Символы:                 | 254 байт (Clipper '87)                |
|                          | 2048 байт (Clipper 5.0)               |
| Данные:                  | 8 байт                                |
| Логическое:              | 1 байт                                |
| Цифровое:                | 20 байт включая десятичную точку      |
| Мемо:                    | 65,520 байт (см. замечание)           |
| Полей на запись:         | 1024                                  |
| Ключей/индексов на файл: | неограниченно                         |
| Размер ключа             |                                       |
| Символ:                  | 100 байт                              |
| Цифры, данные:           | 8 байт                                |
| Мемо полей на файл:      | в зависимости от имеющейся памяти     |
| Открывание файла:        | в зависимости от операционной системы |

## Строки символов драйвера и функции SEND

---

Строкам символов драйвера, назначенным в объявлении FILE (второй параметр атрибута DRIVER (драйвер)) всегда предшествует косая черта (/). Они отсылаются драйверу файла, когда приложение выдает команду OPEN (открыть). Нет величины возврата, связанной с этими строками драйвера.

Функция SEND отсылает строки драйвера в любое время. Некоторые строки символов драйвера не действуют после открытия файла, поэтому синтаксис функции SEND для модификации установки не перечислен. Однако синтаксис функции SEND для возврата величины переключения приводится в перечне для всех строк символов драйвера. Строки драйвера функции SEND принимают два формата - один со знаком равенства модифицирует установки переключателя и возвращает величину предыдущей установки переключателя; другой формат (без знака равенства) возвращает величину переключателя.

### /BUFFERS=n

Устанавливает величину для числа буферов, используемых для чтения и записи в файл.

Драйвер Clipper использует систему буферов DOS. По умолчанию имеется три буфера по 1024 байта каждый. Увеличение числа буферов не улучшает эффективность работы, если доступ к файлу делится между многими пользователями.

### SEND(file, 'BUFFERS')

Возвращает число буферов в форме строки символов STRING.

### /RECOVER

### SEND(file, 'RECOVER')

Эквивалентно команде Xbase RECALL, которая восстанавливает записи, маркированные для удаления. При использовании драйвера Clipper оператор DELETE отмечает флажком запись как “неактивную”. Драйвер не удаляет запись до тех пор, пока не выполнены команды PACK (упаковать) или BUILD (построить).

/RECOVER оценивается каждый раз при открывании вами файла, если вы добавите строку символов драйвера к словарю данных. Когда драйвер восстанавливает записи, ранее отмаркированные для удаления, вы должны вручную перестроить ключи и индексы с помощью оператора BUILD.

### SEND(file, 'IGNORESTATUS=on|off')

### /IGNORESTATUS=on|off

Когда установка ON, драйвер не перескакивает через удаленные записи во время

выполнения операторов GET, NEXT и PREVIOUS при доступе к файлу, открытому в физической последовательности. Это также делает доступным выполнение оператора PUT для удаленной или задержанной записи. /IGNORESTATUS требует открытия файла в эксклюзивном режиме.

### **SEND(file, 'IGNORESTATUS')**

Возвращает установку (ON или OFF) IGNORESTATUS в форме STRING(3).

### **SEND(file, 'DELETED')**

Для использования только с командой SEND, когда включен IGNORESTATUS. Сообщает состояние текущей записи. Если она удалена, возвращаемая строка символов - "ON", если нет - "OFF".

## **Поддерживаемые атрибуты файла, команды и функции**

### **Атрибуты файла**

CREATE .  
 DRIVER (filetype,[driver string])  
 NAME  
 ENCRYPT  
 OWNER (password)  
 RECLAIM  
 PRE (prefix)  
 BINDABLE  
 THREAD  
 EXTERNAL (member)  
 DLL ([flag])  
 OEM

### **Поддерживаемые**

Да  
 Да  
 Да  
 Нет  
 Нет  
 Нет5  
 Да  
 Да  
 Да  
 Да  
 Да  
 Да

### **Файловые структуры**

INDEX  
 KEY  
 MEMO  
 BLOB  
 RECORD

### **Поддерживаемые**

Да  
 Да  
 Да  
 Нет  
 Да

### **Атрибуты индекса, ключа, мемо**

BINARY  
 DUP  
 NOCASE  
 OPT

### **Поддерживаемые**

Нет  
 Да10  
 Да  
 Нет

|                       |    |
|-----------------------|----|
| PRIMARY               | Да |
| NAME                  | Да |
| Ascending Components  | Да |
| Descending Components | Да |
| Mixed Components      | Да |

**Файловые команды**

|                                 |     |
|---------------------------------|-----|
| BUILD(file)                     | Да  |
| BUILD (key)                     | Да  |
| BUILD (index)                   | Да  |
| BUILD (index,components)        | Да1 |
| BUILD (index,components,filter) | Нет |
| CLOSE (file)                    | Да  |
| COPY (file,new file)            | Да2 |
| CREATE (file)                   | Да  |
| EMPTY (file)                    | Да  |
| FLUSH (file)                    | Да  |
| LOCK (file)                     | Нет |
| OPEN (file,access mode)         | Да  |
| PACK (file)                     | Да  |
| REMOVE (file)                   | Да  |
| RENAME (file,new file)          | Да3 |
| SHARE (file,access mode)        | Да  |
| STATUS (file)                   | Да  |
| STREAM (file)                   | Да  |
| UNLOCK (file)                   | Да  |

**Поддерживаемые****Доступ к записям**

|                               |     |
|-------------------------------|-----|
| ADD (file)                    | Да4 |
| ADD (file,length)             | Нет |
| APPEND (file)                 | Да4 |
| APPEND (file,length)          | Нет |
| DELETE (file)                 | Да5 |
| GET (file,key)                | Да  |
| GET (file,filepointer)        | Да  |
| GET (file,filepointer,length) | Нет |
| GET (file,keypointer)         | Да  |
| HOLD (file)                   | Да6 |
| NEXT (file)                   | Да  |
| NOMEMO (file)                 | Да  |
| PREVIOUS (file)               | Да  |
| PUT (file)                    | Да  |
| PUT (file,filepointer)        | Да  |

**Поддерживаемые**

|                               |     |
|-------------------------------|-----|
| PUT (file,filepointer,length) | Нет |
| RELEASE (file)                | Да  |
| REGET ( file,string)          | Да  |
| REGET ( key,string)           | Да  |
| RESET ( file,string)          | Да  |
| RESET ( key,string)           | Да  |
| SET (file)                    | Да  |
| SET (file,key)                | Да  |
| SET (file,filepointer)        | Да  |
| SET (key)                     | Да  |
| SET (key,key)                 | Да  |
| SET (key,keypointer)          | Да  |
| SET (key,key,filepointer)     | Да  |
| SKIP (file,count)             | Да  |
| WATCH (file)                  | Да  |

**Файловые функции**

|                     |     |
|---------------------|-----|
| BOF (file)          | Да7 |
| BYTES (file)        | Нет |
| DUPLICATE (file)    | Да  |
| DUPLICATE (key)     | Да  |
| EOF (file)          | Да7 |
| NAME (label)        | Да  |
| POINTER (file)      | Да8 |
| POINTER (key)       | Да8 |
| POSITION (file)     | Да  |
| POSITION (key)      | Да  |
| RECORDS (file)      | Да9 |
| RECORDS (key)       | Да9 |
| SEND (file,message) | Да  |

**Диалоговая обработка**

|                               |     |
|-------------------------------|-----|
| LOGOUT (timeout,file,...file) | Нет |
| COMMIT                        | Нет |
| ROLLBACK                      | Нет |

**Обработка пустых данных**

|                   |     |
|-------------------|-----|
| NULL (field)      | Нет |
| SETNULL (field)   | Нет |
| SETNONULL (field) | Нет |

**Поддерживаемые****Поддерживаемые****Поддерживаемые**

## Замечания

---

1 При построении динамических индексов, компоненты могут принять одну из двух следующих форм:

```
BUILD(DynNdx, '+Pre:FLD1, -Pre:FLD2' )
```

Эта форма устанавливает имена полей, на которых строится индекс. Имена полей должны появиться в атрибуте поля NAME, если он используется, или это должны быть имена меток. Для совместимости с Clarion может быть использован префикс, но он игнорируется.

```
BUILD(DynNdx, 'T[Expression]')
```

Эта форма устанавливает тип и выражение, используемые для построения индекса, о чем смотрите ниже - в разделе “Разное” - Определение ключа..

2 Команда COPY() копирует файлы данных и мемо файлы, используя newfile (новый файл), при помощи которого вы можете установить новое имя файла или каталога. Файлы ключей или индексов копируются, если newfile является характеристикой подкаталога. Для копирования индексного файла в новый файл используйте специальную форму команды копирования:

```
COPY(file, '<index>|< newfile >')
```

Этот оператор выдает сообщение File Not Found (файл не найден), если используется ссылка на недействительный индекс. Команда COPY предполагает расширение по усмотрению “.NTX” для имен файлов источника и целевого файла, если ни одно из них специально не определено. Если вы требуете имя файла без расширения, закончите имя точкой.

Дана структура файла:

```
Clar2 FILE,CREATE,DRIVER('Clipper')
NumKey KEY(Num),DUP
StrKey KEY(Str1)
StrKey2 KEY(Str2)
AMemo MEMO(100),NAME('mem')
Record RECORD
Num STRING(@n-_9.2)
STR1 STRING(2)
STR2 STRING(2)
Mem STRING(10)
```

Следующие команды копируют это определение файла в A:

```
COPY(Clar2,'A:\CLAR2')
```

```
COPY(Clar2,'StrKey|A:\STRKEY')  
COPY(Clar2,'StrKey2|A:\STRKEY2')  
COPY(Clar2,'NumKey|A:\NUMKEY')
```

После этих вызовов на накопителе А будут существовать следующие файлы: CLAR2.DBE, CLAR2.DBT, STRKEY.NTX, STRLEY2.NTX и NUMKEY.NTX.

3 Команда RENAME копирует файлы данных и файлы мемо с помощью newfile, определяющем новое имя файла или путь. Файлы ключей и индексов должны быть переименованы с помощью того же синтаксиса, что и команда COPY (см. выше).

4 Оператор ADD проверяет на дублирование ключи перед модификацией файла данных или связанных с ним ключевых KEY файлов. В результате оператор ADD выполняется медленнее, чем APPEND, который не выполняет никаких проверок и не обновляет ключи KEY. При добавлении больших количеств данных к базе данных, используйте APPEND...BUILD вместо ADD.

5 Когда драйвер удаляет запись из базы данных Clipper, запись физически не удаляется, а вместо этого драйвер маркирует ее, как неактивную. Поля мемо не удаляются физически из мемо файла, однако они не могут быть отысканы, если они относятся к неактивной записи. Записи ключа удаляются из индексного файла. Чтобы удалить записи и мемо файлы навсегда, выполните оператор PACK(file).

**Совет: Для программистов, знакомых с Clipper, этот драйвер обрабатывает удаленные записи в соответствии с тем, как Clipper обрабатывает их после того, как дана команда SET DELETED ON (установите удаленное). Записи, отмеченные для удаления, игнорируются исполняемыми программными операторами, но остаются в файле данных.**

6 Clipper выполняет блокирование записи путем блокирования всей записи в файле данных. Это предотвращает доступ для чтения для других процессов. Следовательно, мы рекомендуем уменьшить до предела количество времени, на которое удерживается запись.

7 Хотя драйвер поддерживает эти функции, мы не рекомендуем их использовать. Эти функции для своего выполнения требуют физического доступа к файлам и значительных ресурсов. Вместо этого проверьте величину, возвращаемую функцией ERRORCODE() после каждого последовательного доступа. NEXT() или PREVIOUS() посылают код ошибки 33 (запись недоступна), если сделана попытка получить доступ к записи за концом или началом файла.

8 Нет различия между указателями файла и указателями ключа; и тот и другой содержат ту же самую величину для любой данной записи.

9 В рамках Clipper функция RECORDS() сообщает одно и то же число записей для файла данных и его ключей и индексов. Обычно не бывает различия в числе записей, если

INDEX не устарел. Так как оператор DELETE не удаляет записи физически, число записей, о которых сообщает функция RECORDS(), включает неактивные записи. Будьте осторожны при использовании этой функции.

10 В Clipper допустимым является ввод многих записей с дубликатами уникальных ключевых компонент. Однако только первая из этих записей индексируется. Таким образом обработка в ключевом порядке показывает только эту первую запись. Если вы удаляете запись, затем вводите новую запись с той же самой величиной ключа, ключевой файл продолжает указывать на удаленную запись, а не на новую запись. В этой ситуации файловый драйвер Clipper Clarion for Windows изменяет ключевой файл, чтобы указать на активную запись, а не на удаленную запись. Это означает, что если вы используете программу Clipper для удаления уникальной записи, а затем вставите дубликат этой записи, новая запись невидима при обработке в ключевом порядке до тех пор, пока не выполнена упаковка. Если вы выполняете тот же самый процесс в программе Clarion for Windows 2.0, новая запись оказывается видимой при обработке в ключевом порядке.

## Разное

---

### Булева оценка

★ Clipper позволяет логическому полю принять одну из девяти возможных величин (y,Y,n,N,t,T,f,F или символ пробела). Символ пробела - это ни истина и ни ложь. При использовании логического поля из заранее существующей базы данных в логическом выражении учтите все эти возможности. Помните, что когда поле STRING используется как выражение, оно истинно, если содержит какие-либо данные, и оно ложно, если равно нулю или пусто. Следовательно, чтобы оценить истинность логического поля, выражение должно быть истина, если поле содержит какую-либо из “истинных” характеристик (T,t,Y или y). Например, если логическое поле было использовано, чтобы установить продукт как налогооблагаемый или необлагаемый, выражение для оценки его истинности будет:

(If Condition):

Taxable='T' OR Taxable='t' OR Txable='Y' OR  
Taxable='y'

### Большие MEMO поля

★ Clarion for Windows поддерживает поля MEMO размером до 64К. Если у вас существует файл, включающий мемо, намного большее чем 64К, вы можете использовать файл, но не модифицировать большие MEMO.

★ Вы можете определить, когда ваше приложение встретит большое MEMO путем определения, что переменная указателя мемо не пуста, но поле мемо при этом выглядит пустым. Выдается код ошибки 47 (неправильное объявление записи). Если вы попытаетесь обновить такую запись, любая модификация MEMO поля будет игнорироваться.

### Длинные имена поля



★ Clipper поддерживает максимум 10 символов в имени поля. Если вам нужно больше, используйте External Name (внешнее имя) с 10 или менее символами.

### **Последовательность сортировки**

★ Драйвер Clipper Clarion for Windows поддерживает международные порядки сортировки, однако, для соблюдения совместимости с международным порядком сортировки Clipper, удалите строку CLADIGRAPH= из файла ..\CW20\BIN\CW20.ENV.

### **Тип данных возвращаемый функцией POSITION (POSITION Return Data Type)**

- ★ POSITION(file) возвращает STRING(12)
- ★ POSITION(key) возвращает STRING размер полей ключа + 4 байта.

### **Определение ключа**

★ Clipper поддерживает использование выражений для определения ключей. В редакторе словаря вы можете поместить выражение во внешнее имя поля в диалоговом окне Key Properties(свойства ключа). Общий формат внешнего имени:

'FileName=T[Expression]'

Где FileName представляет имя индексного файла (который может содержать путь и расширение файла), а T представляет тип индекса. Действительными типами являются: C = символ, D = данные, N = цифра. Если типом является D или N, тогда Expression (выражение) может назвать только одно поле.

Выражения в виде строк символов могут использовать оператор “+” для конкатенации многих строк символов аргументов. Цифровые выражения используют операторы ‘+’ или ‘-’ в их основном смысле. Максимальная длина Clipper выражения - 250 символов.

Выражение может относиться к множественным полям в записи и содержать функции xBase. Квадратные скобки должны ставиться вокруг выражения. Текущие поддерживаемые функции даются ниже. Если драйвер встречает неподдерживаемую xBase функцию в заранее существовавшем файле, он посылает код ошибки 76 “Недействительная строка символов индекса”, когда файл открыт для ключей и статических индексов.

### **Поддерживаемые xBase функции определения ключа**

|                 |                                                                                                                                                                                                                                |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ALLTRIN(string) | Удаляет начальные и конечные пробелы.                                                                                                                                                                                          |
| CTOD(string)    | Преобразует ключ в виде строки символов в дату. Формат строки string mm/dd/yy; результат имеет форму 'ууууmmdd'. Элемент уууу по умолчанию означает двадцатое столетие. Недействительные данные дают в результате пустой ключ. |

|                                          |                                                                                                                                                                                                                          |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DELETED()                                | Возвращает TRUE если запись удалена.                                                                                                                                                                                     |
| DESCEND(string date numeric)             | Инвертирует аргумент и создает нисходящие Clipper индексы.                                                                                                                                                               |
| DTOC(date)                               | Преобразует ключ данных в строку символов формата 'mm/dd/yy'.                                                                                                                                                            |
| DTOS(date)                               | Преобразует ключ данных в строку символов формата 'ууууmmdd'.                                                                                                                                                            |
| FIXED(float)                             | Преобразует ключ из представления с плавающей точкой в представление с фиксированной точкой.                                                                                                                             |
| FLOAT(numeric)                           | Преобразует ключ из представления с фиксированной точкой в представление с плавающей точкой.                                                                                                                             |
| IIF(bool, val1, val2)                    | Возвращает величину val1 если первый параметр TRUE, иначе возвращает величину val2.                                                                                                                                      |
| LEFT(string, n)                          | Возвращает самые левые n символов строки символов ключа в виде строки символов длиной n.                                                                                                                                 |
| LOWER (string, n)                        | Преобразует строку символов ключа в нижний регистр.                                                                                                                                                                      |
| LTRIM (string)                           | Удаляет пробелы с левой стороны в строке символов.                                                                                                                                                                       |
| RECNO( )                                 | Возвращает номер текущей записи.                                                                                                                                                                                         |
| RIGHT(string, n)                         | Возвращает самые правые n символов строки символов ключа в виде строки символов длиной n.                                                                                                                                |
| RTRIM(string)                            | Удаляет пробелы с правой стороны строки символов.                                                                                                                                                                        |
| STR(numeric [,length[, decimal places]]) | Преобразует цифровые данные в строку символов. Длина строки символов и число десятичных знаков после запятой - параметры необязательные. По умолчанию длина строки символов 10, а число знаков после десятичной точки 0. |
| SUBSTR(string,offset,n)                  | Возвращает подстроку символов ключа в виде строки символов string, начиная с позиции offset и длиной в n символов.                                                                                                       |
| TRIM (string)                            | Удаляет пробелы с правой стороны от строки символов (идентично RTRIM).                                                                                                                                                   |
| UPPER(string)                            | Преобразует строку символов ключа в верхний регистр.                                                                                                                                                                     |
| VAL(string)                              | Преобразует строку символов ключа в цифру.                                                                                                                                                                               |

## Файлы dBase III

Драйвер dBase 3 совместим с dBaseIII. Расширение файла по умолчанию \*.DBF.

Ключи и индексы существуют как файлы, отдельные от файла данных. Ключи динамические - они автоматически обновляются по мере изменения файла данных. Индексы статические - они не обновляются автоматически, но вместо этого требуют для обновления выполнения оператора BUILD. Расширение файла по умолчанию для индексного файла - \*.NDX.

Драйвер сохраняет записи в виде записей фиксированной длины. Поля мемо хранятся в отдельном файле. Имя файла мемо состоит из первых восьми символов имени метки с расширением .DBT.

|               |                     |                                                         |
|---------------|---------------------|---------------------------------------------------------|
| <b>Файлы:</b> | <b>CL2DB316.LIB</b> | Экспортная библиотека Windows (16-битовая)              |
|               | <b>CL2DB332.LIB</b> | Экспортная библиотека Windows (32-битовая)              |
|               | <b>CW2DB316.LIB</b> | Статически загружаемая библиотека Windows (16-битовая)  |
|               | <b>CW2DB332.LIB</b> | Статически загружаемая библиотека Windows (32-битовая)  |
|               | <b>CW2DB316.DLL</b> | Динамически загружаемая библиотека Windows (16-битовая) |
|               | <b>CW2DB332.DLL</b> | Динамически загружаемая библиотека Windows (32-битовая) |

**Совет:** dBaseIII является, по-видимому, наиболее распространенным форматом файла для PC приложений баз данных. В настоящее время даже издательские программы могут импортировать dBaseIII совместимые .DBF файлы. Если главная задача вашего приложения - экспортировать файлы данных для других приложений, о которых вы не знаете ничего, вы должны рассмотреть прежде всего этот формат.

## Типы данных

Формат xBase сохраняет все данные, как строки символов ASCII. Вы можете установить типы строк STRING с объявленными шаблонами для каждого поля или установить типы Clarion, которые драйвер преобразует автоматически.

| <u>тип данных dBase</u> | <u>тип данных Clarion</u> | <u>STRING с шаблоном</u> |
|-------------------------|---------------------------|--------------------------|
| DATE                    | DATE                      | STRING(@D12)             |
| *Numeric                | REAL                      | STRING(@N-_p.d)          |
| *Logical                | BYTE                      | STRING(1)                |

Character  
\*Memo

STRING  
MEMO

STRING  
MEMO

Если ваше приложение читает и записывает в существующие файлы, достаточно будет STRING с шаблоном. Однако, если ваше приложение создает dBaseIII файл, вам может потребоваться дополнительная информация для этих типов dBaseIII :

★ Чтобы создать цифровое поле в словаре данных, выберите тип данных REAL (действительный). В атрибуте Внешнее имя (External Name) установите 'NumericFieldName=N(Precision,DecimalPlaces)' где NumericFieldName - имя поля, Precision - точность поля, а DecimalPlaces - число десятичных мест.

Если вы кодируете вручную присущие Clarion типы данных, добавьте атрибут NAME используя тот же самый синтаксис. Если вы кодируете вручную символьный шаблон, STRING@N-9.2),NAME('Number'),где Number - имя поля.

★ Чтобы создать логическое поле, используя словарь данных, выберите тип данных BYTE. Специальных шагов нет; однако, смотрите в разных разделах советы о чтении данных из поля.

Если вы вручную кодируете STRING с шаблоном, добавьте атрибут NAME: STRING(1),NAME('LogFld=L').

★ Чтобы создать поле данных, используя словарь данных, выберите тип данных DATE, а не LONG, который вы обычно используете для форматов файлов TopSpeed или Clarion.

★ Объявления поля MEMO требуют указателя поля в файловой структуре записи. Объявите указатель поля как STRING(10) или LONG. Это поле будет сохранено в файле .DBF, содержащем смещение мемо поля в файле .DBT. Объявление MEMO должно иметь атрибут NAME(), содержащем указатель поля. Пример объявления поля:

```
File FILE, DRIVER('dBase 3')
Memo1 MEMO(200), NAME('Notes')
Memo2 MEMO(200), NAME('Text')
Rec RECORD
Mem1Ptr LONG, NAME('Notes')
Mem2Ptr STRING(10), NAME('Text')
END
```

END

**Совет: Используйте Утилиту Импорта Файла в редакторе словаря в Clarion for Windows для определения ваших файлов.**

## Характеристики файла/максимумы

|                          |                                       |
|--------------------------|---------------------------------------|
| Размер файла:            | 2,000,000,000 байт                    |
| Записей на файл:         | 1,000,000,000 байт                    |
| Размер записи:           | 4,000 байт                            |
| Размер поля              |                                       |
| Символы:                 | 254 байт                              |
| Данные:                  | 8 байт                                |
| Логическое:              | 1 байт                                |
| Цифровое:                | 20 байт включая десятичную точку      |
| Мемо:                    | 64K (см. замечание)                   |
| Полей на запись:         | 255                                   |
| Ключей/индексов на файл: | неограниченно                         |
| Размер ключа             |                                       |
| Символ:                  | 100 байт                              |
| Цифры, данные:           | 8 байт                                |
| Мемо полей на файл:      | в зависимости от имеющейся памяти     |
| Открытие файла:          | в зависимости от операционной системы |

## Строки символов драйвера и функции SEND

Строкам символов драйвера (второй параметр атрибута DRIVER (драйвер)) предшествует символ наклонной черты (/). Команды функции SEND могут принять два формата - один со знаком равенства- модифицирует установку переключения и возвращает величину предыдущей установки переключения; другой формат (без знака равенства) возвращает величину переключения.

Строки символов драйвера отсылаются к драйверу файла когда файл открыт. Функция SEND посылает команду на изменение установки после того, как файл открыт. Некоторые строки символов драйвера не оказывают никакого влияния после того, как файл открыт, так что синтаксис функции SEND модификации установки не перечисляется. Однако, синтаксис функции SEND для возвращения величины переключения перечисляется для всех строк символов драйвера.

### /BUFFERS=n

Устанавливает величину для числа буферов, используемых для чтения и записи в файл.

Драйвер dBase III использует систему буферов DOS. По умолчанию имеется три буфера по 1024 байта каждый. Увеличение числа буферов не улучшает эффективность работы, если доступ к файлу делится между многими пользователями.

### SEND(file, 'BUFFERS')

Возвращает число буферов в форме строки символов STRING.

**/RECOVER****SEND(file, 'RECOVER')**

Эквивалентно команде Xbase RECALL, которая восстанавливает записи, маркированные для удаления. При использовании драйвера dBase III оператор DELETE отмечает флажком запись, как “неактивную”. Драйвер не удаляет запись до тех пор, пока не выполнены команды PACK (упаковать) или BUILD (построить).

/RECOVER оценивается каждый раз при открывании вами файла, если вы добавите строку символов драйвера к словарю данных. Когда драйвер восстанавливает записи, ранее отмаркированные для удаления, вы должны вручную перестроить ключи и индексы с помощью оператора BUILD.

**SEND(file, 'IGNORESTATUS=on|off')****/IGNORESTATUS=on|off**

Когда установка ON, драйвер не перескакивает через удаленные записи во время выполнения операторов GET, NEXT и PREVIOUS при доступе к файлу, открытому в физической последовательности. Это также делает доступным выполнение оператора PUT для удаленной или задержанной записи. /IGNORESTATUS требует открытия файла в эксклюзивном режиме.

**SEND(file, 'IGNORESTATUS')**

Возвращает установку (ON или OFF) IGNORESTATUS в форме STRING(3)

**SEND(file, 'DELETED')**

Для использования только с командой SEND, когда включен IGNORESTATUS. Сообщает состояние текущей записи. Если она удалена, возвращаемая строка символов - “ON”, если нет - “OFF”.

**/OMNIS**

Устанавливает совместимость заголовка файла и разграничителя файла OMNIS.

## **Поддерживаемые атрибуты файла, команды и функции**

**Атрибуты файла**

CREATE .  
DRIVER (filetype,[driver string])  
NAME

**Поддерживаемые**

Да  
Да  
Да

|                   |      |
|-------------------|------|
| ENCRYPT           | Нет  |
| OWNER (password)  | Нет  |
| RECLAIM           | Нет6 |
| PRE (prefix)      | Да   |
| BINDABLE          | Да   |
| THREAD            | Да   |
| EXTERNAL (member) | Да   |
| DLL ([flag])      | Да   |
| OEM               | Да   |

**Файловые структуры**

|        |     |
|--------|-----|
| INDEX  | Да  |
| KEY    | Да  |
| MEMO   | Да  |
| BLOB   | Нет |
| RECORD | Да  |

**Поддерживаемые****Атрибуты индекса, ключа, мемо**

|                       |     |
|-----------------------|-----|
| BINARY                | Нет |
| DUP                   | Да1 |
| NOCASE                | Да  |
| OPT                   | Нет |
| PRIMARY               | Да  |
| NAME                  | Да  |
| Ascending Components  | Да  |
| Descending Components | Да  |
| Mixed Components      | Нет |

**Поддерживаемые****Файловые команды**

|                                 |     |
|---------------------------------|-----|
| BUILD(file)                     | Да  |
| BUILD (key)                     | Да  |
| BUILD (index)                   | Да  |
| BUILD (index,components)        | Да2 |
| BUILD (index,components,filter) | Нет |
| CLOSE (file)                    | Да  |
| COPY (file,new file)            | Да3 |
| CREATE (file)                   | Да  |
| EMPTY (file)                    | Да  |
| FLUSH (file)                    | Да  |
| LOCK (file)                     | Нет |
| OPEN (file,access mode)         | Да  |
| PACK (file)                     | Да  |

**Поддерживаемые**

|                          |     |
|--------------------------|-----|
| REMOVE (file)            | Да  |
| RENAME (file,new file)   | Да4 |
| SHARE (file,access mode) | Да  |
| STATUS (file)            | Да  |
| STREAM (file)            | Да  |
| UNLOCK (file)            | Да  |

**Доступ к записям**

|                               |     |
|-------------------------------|-----|
| ADD (file)                    | Да5 |
| ADD (file,length)             | Нет |
| APPEND (file)                 | Да5 |
| APPEND (file,length)          | Нет |
| DELETE (file)                 | Да6 |
| GET (file,key)                | Да  |
| GET (file,filepointer)        | Да  |
| GET (file,filepointer,length) | Нет |
| GET (file,keypointer)         | Да  |
| HOLD (file)                   | Да7 |
| NEXT (file)                   | Да  |
| NOMEMO (file)                 | Да  |
| PREVIOUS (file)               | Да  |
| PUT (file)                    | Да  |
| PUT (file,filepointer)        | Да  |
| PUT (file,filepointer,length) | Нет |
| RELEASE (file)                | Да  |
| REGET ( file,string)          | Да  |
| REGET ( key,string)           | Да  |
| RESET ( file,string)          | Да  |
| RESET ( key,string)           | Да  |
| SET (file)                    | Да  |
| SET (file,key)                | Да  |
| SET (file,filepointer)        | Да  |
| SET (key)                     | Да  |
| SET (key,key)                 | Да  |
| SET (key,keypointer)          | Да  |
| SET (key,key,filepointer)     | Да  |
| SKIP (file,count)             | Да  |
| WATCH (file)                  | Да  |

**Поддерживаемые****Файловые функции**

|                  |     |
|------------------|-----|
| BOF (file)       | Да8 |
| BYTES (file)     | Нет |
| DUPLICATE (file) | Да  |

**Поддерживаемые**



|                     |      |    |
|---------------------|------|----|
| DUPLICATE (key)     | Да   |    |
| EOF (file)          | Да8  |    |
| NAME (label)        | Да   |    |
| POINTER (file)      | Да9  |    |
| POINTER (key)       | Да9  |    |
| POSITION (file)     | Да   |    |
| POSITION (key)      | Да   |    |
| RECORDS (file)      | Да10 |    |
| RECORDS (key)       | Да10 |    |
| SEND (file,message) |      | Да |

**Диалоговая обработка****Поддерживаемые**

|                               |     |
|-------------------------------|-----|
| LOGOUT (timeout,file,...file) | Нет |
| COMMIT                        | Нет |
| ROLLBACK                      | Нет |

**Обработка пустых данных****Поддерживаемые**

|                   |     |
|-------------------|-----|
| NULL (field)      | Нет |
| SETNULL (field)   | Нет |
| SETNONULL (field) | Нет |

**Замечания**

1 dBase III не поддерживает никакую форму уникального индекса. Поэтому атрибут DUP должен быть у всех ключей.

2 При построении динамических индексов, компоненты могут принять одну из двух следующих форм:

BUILD(DynNdx, '+Pre:FLD1, -Pre:FLD2' )

Эта форма устанавливает имена полей, на которых строится индекс. Имена полей должны появиться в атрибуте поля NAME, если он используется, или это должны быть имена меток. Для совместимости с Clarion может быть использован префикс, но он игнорируется.

BUILD(DynNdx, 'T[Expression]')

Эта форма устанавливает тип и выражение, используемые для построения индекса, о чем смотрите ниже - в разделе "Разное" - Определение ключа..

3 Команда COPY() копирует файлы данных и мемо файлы используя newfile (новый файл), при помощи которого вы можете установить новое имя файла или каталога. Файлы

ключей или индексов копируются, если newfile является характеристикой подкаталога. Для копирования индексного файла в новый файл используйте специальную форму команды копирования:

```
COPY(file, '<index>|< newfile >')
```

Этот оператор выдает сообщение File Not Found (файл не найден), если используется ссылка на недействительный индекс. Команда COPY предполагает расширение по усмотрению ".NDX" для имен файлов источника и целевого файла, если ни одно из них специально не определено. Если вы требуете имя файла без расширения, закончите имя точкой.

Дана структура файла:

```
Clar2 FILE,CREATE,DRIVER('dBase3')
NumKey KEY(Num),DUP
StrKey KEY(Str1)
StrKey2 KEY(Str2)
AMemo MEMO(100),NAME('mem')
Record RECORD
Num STRING(@n-_9.2)
STR1 STRING(2)
STR2 STRING(2)
Mem STRING(10)
```

Следующие команды копируют это определение файла в A:

```
COPY(Clar2,'A:\CLAR2')
COPY(Clar2,'StrKey|A:\STRKEY')
COPY(Clar2,'StrKey2|A:\STRKEY2')
COPY(Clar2,'NumKey|A:\NUMKEY')
```

После этих вызовов на накопителе A будут существовать следующие файлы:

CLAR2.DBE,CLAR2.DBT,STRKEY.NTX,STRLEY2.NTX и NUMKEY.NTX.

4 Команда RENAME копирует файлы данных и файлы мемо с помощью newfile, определяющим новое имя файла или путь. Файлы ключей и индексов должны быть переименованы с помощью того же синтаксиса, что и команда COPY (см. выше).

5 Оператор ADD проверяет на дублирование ключи перед модификацией файла данных или связанных с ним ключевых KEY файлов. В результате оператор ADD выполняется медленнее, чем APPEND, который не выполняет никаких проверок и не обновляет ключи KEY. При добавлении больших количеств данных к базе данных, используйте APPEND...BUILD вместо ADD.

6 Когда драйвер удаляет запись из базы данных dBase III, запись физически не удаляется, а вместо этого драйвер маркирует ее, как неактивную. Поля мемо не удаляются физически из мемо файла, однако они не могут быть отысканы, если они относятся к неактивной записи. Величины ключа удалены из индексных файлов. Чтобы удалить записи и мемо файлы навсегда, выполните оператор PASC(file).

**Совет: Для программистов, знакомых с dBase III, этот драйвер обрабатывает удаленные записи в соответствии с тем, как dBase III обрабатывает их после того, как дана команда SET DELETED ON (установите удаленное). Записи, отмеченные для удаления, игнорируются исполняемыми программными операторами, но остаются в файле данных.**

7 dBase III выполняет блокирование записи путем блокирования всей записи в файле данных. Это предотвращает доступ для чтения для других процессов. Следовательно, мы рекомендуем уменьшить до предела количество времени, на которое удерживается запись.

8 Хотя драйвер поддерживает эти функции, мы не рекомендуем их использовать. Эти функции для своего выполнения требуют физического доступа к файлам и значительных ресурсов. Вместо этого проверьте величину, возвращаемую функцией ERRORCODE() после каждого последовательного доступа. NEXT() или PREVIOUS() посылают код ошибки 33 (запись недоступна), если сделана попытка получить доступ к записи за концом или началом файла.

9 Нет различия между указателями файла и указателями ключа; и тот и другой содержат ту же самую величину для любой данной записи.

10 В рамках dBase III функция RECORDS() сообщает одно и то же число записей для файла данных и его ключей и индексов. Обычно не бывает различия в числе записей, если INDEX не устарел. Так как оператор DELETE не удаляет записи физически, число записей, о которых сообщает функция RECORDS(), включает неактивные записи. Будьте осторожны при использовании этой функции.

## Разное

---

### Булева оценка

★ dBase III позволяет логическому полю принять одну из девяти возможных величин (y,Y,n,N,t,T,f,F или символ пробела). Символ пробела - это ни истина и ни ложь. При использовании логического поля из заранее существующей базы данных в логическом выражении учтите все эти возможности. Помните, что когда поле STRING используется как выражение, оно истинно, если содержит какие-либо данные, и оно ложно, если равно нулю или пусто. Следовательно, чтобы оценить истинность логического поля, выражение должно быть истина, если поле содержит какую-либо из “истинных” характеристик (T,t,Y или y). Например, если логическое поле было использовано, чтобы установить продукт как налогооблагаемый или необлагаемый, выражение для оценки его истинности будет:

(If Condition):

Taxable='T' OR Taxable='t' OR Txable='Y' OR  
Taxable='y'

### **Большие MEMO поля**

★ Clarion for Windows поддерживает поля MEMO размером до 64К. Если у вас существует файл, включающий мемо, намного большее чем 64К, вы можете использовать файл, но не модифицировать большие MEMO.

★ Вы можете определить, когда ваше приложение встретит большое MEMO путем определения, что переменная указателя мемо не пуста, но поле мемо при этом выглядит пустым. Выдается код ошибки 47 (неправильное объявление записи), любая модификация MEMO поля будет игнорироваться.

### **Длинные имена полей**

★ dBase III поддерживает максимум 10 символов в имени поля. Если вам нужно больше, используйте External Name (внешнее имя) с 10 или менее символами.

### **Международная последовательность сортировки**

★ Драйвер dBase III Clarion for Windows поддерживает международные порядки сортировки, однако, для соблюдения совместимости с международным порядком сортировки dBase III, удалите строку CLADIGRAPH= из файла ..\CW20\BIN\CW20.ENV.

### **Тип данных возвращаемый функцией POSITION (POSITION Return Data Type)**

- ★ POSITION(file) возвращает STRING(12)
- ★ POSITION(key) возвращает STRING размер полей ключа + 4 байта.

### **Определение ключа**

★ dBase III поддерживает использование выражений для определения ключей. В редакторе словаря вы можете поместить выражение во внешнее имя поля в диалоговом окне Key Properties(свойства ключа). Общий формат внешнего имени:

'FileName=T[Expression]'

Где FileName представляет имя индексного файла (который может содержать путь и расширение файла), а T представляет тип индекса. Действительными типами являются: C = символ, D = данные, N = цифра. Если типом является D или N, тогда Expression (выражение) может назвать только одно поле.

Выражения в виде строк символов могут использовать оператор "+" для конкатенации многих строк символов аргументов. Цифровые выражения используют операторы '+' или '-' в их основном смысле. Максимальная длина Clipper выражения - 250 символов.

Выражение может относиться к множественным полям в записи и содержать функции xBase. Квадратные скобки должны ставиться вокруг выражения. Текущие поддерживаемые функции даются ниже. Если драйвер встречает неподдерживаемую xBase функцию в заранее существовавшем файле, он посылает код ошибки 76 “Недействительная строка символов индекса”, когда файл открыт для ключей и статических индексов.

### **Поддерживаемые xBase функции определения ключа**

|                                                                      |                                                                                                                                                                                                                                  |
|----------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ALLTRIN(string)                                                      | Удаляет начальные и конечные пробелы.                                                                                                                                                                                            |
| CTOD(string)                                                         | Преобразует ключ в виде строки символов в дату. Формат строки string mm/dd/yy; результат имеет форму ‘ууууmmdd’. Элемент уууу по умолчанию означает двадцатое столетие. Недействительные данные дают в результате пустой ключ.   |
| DELETED()<br>DTOC(date)                                              | Возвращает TRUE если запись удалена.<br>Преобразует ключ данных в строку символов формата ‘mm/dd/yy’.                                                                                                                            |
| DTOS(date)                                                           | Преобразует ключ данных в строку символов формата ‘ууууmmdd’.                                                                                                                                                                    |
| FIXED(float)                                                         | Преобразует ключ из представления с плавающей точкой в представление с фиксированной точкой.                                                                                                                                     |
| FLOAT(numeric)                                                       | Преобразует ключ из представления с фиксированной точкой в представление с плавающей точкой.                                                                                                                                     |
| IF(bool, val1, val2)                                                 | Возвращает величину val1 если первый параметр TRUE, иначе возвращает величину val2.                                                                                                                                              |
| LEFT(string, n)                                                      | Возвращает самые левые n символов строки символов ключа в виде строки символов длиной n.                                                                                                                                         |
| LOWER (string, n)<br>LTRIM (string)<br>RECNO ( )<br>RIGHT(string, n) | Преобразует строку символов ключа в нижний регистр.<br>Удаляет пробелы слева в строке символов.<br>Возвращает номер текущей записи.<br>Возвращает самые правые n символов строки символов ключа в виде строки символов длиной n. |
| RTRIM(string)<br>STR(numeric [,length[, decimal places]])            | Удаляет пробелы с правой стороны строки символов.<br>Преобразует цифровые данные в строку символов. Длина строки символов и число десятичных знаков после запятой - параметры необязательные. По умолчанию длина строки          |

символов 10, а число знаков после десятичной точки 0.

|                         |                                                                                                                     |
|-------------------------|---------------------------------------------------------------------------------------------------------------------|
| SUBSTR(string,offset,n) | Возвращает подстроку символов ключа в виде строки символов string , начиная с позиции offset и длиной в n символов. |
| TRIM (string)           | Удаляет пробелы с правой стороны от строки символов (идентично RTRIM).                                              |
| UPPER(string)           | Преобразует строку символов ключа в верхний регистр.                                                                |
| VAL(string)             | Преобразует строку символов ключа в цифру.                                                                          |

## Файлы dBase IV

Драйвер dBase 4 совместим с dBaseIV . Расширение файла по умолчанию \*.DBF.

Ключи и индексы существуют как файлы, отдельные от файла данных. Ключи динамические - они автоматически обновляются по мере изменения файла данных. Индексы статические - они не обновляются автоматически, но вместо этого требуют для обновления выполнения оператора BUILD. Расширение для индексного файла по умолчанию \*.NDX.

dBase IV поддерживает многоиндексные файлы, чье расширение - \*.MDX. В разделе “Разное” описываются процедуры использования файлов .MDX.

Драйвер сохраняет записи в виде записей фиксированной длины. Поля мемо хранятся в отдельном файле. Имя файла мемо состоит из первых восьми символов имени метки с расширением .DBT.

|               |                     |                                                         |
|---------------|---------------------|---------------------------------------------------------|
| <b>Файлы:</b> | <b>CL2DB416.LIB</b> | Экспортная библиотека Windows (16-битовая)              |
|               | <b>CL2DB432.LIB</b> | Экспортная библиотека Windows (32-битовая)              |
|               | <b>CW2DB416.LIB</b> | Статически загружаемая библиотека Windows (16-битовая)  |
|               | <b>CW2DB432.LIB</b> | Статически загружаемая библиотека Windows (32-битовая)  |
|               | <b>CW2DB416.DLL</b> | Динамически загружаемая библиотека Windows (16-битовая) |
|               | <b>CW2DB432.DLL</b> | Динамически загружаемая библиотека Windows (32-битовая) |

**Совет: dBaseIV никогда не была так широко распространена, как dBase III. Выбирайте этот драйвер только тогда, когда вы должны обмениваться данными с другим конечным пользователем, использующим dBase IV.**

## Типы данных

Формат xBase сохраняет все данные как строки символов ASCII. Вы можете либо установить типы строк STRING с объявленными шаблонами для каждого поля, либо установить типы Clarion, которые драйвер преобразует автоматически.

| <u>тип данных dBase</u> | <u>тип данных Clarion</u> | <u>STRING с картинкой</u> |
|-------------------------|---------------------------|---------------------------|
| DATE                    | DATE                      | STRING(@D12)              |
| *Numeric                | REAL                      | STRING(@N-_p.d)           |
| *Logical                | BYTE                      | STRING(1)                 |
| Character               | STRING                    | STRING                    |
| *Memo                   | MEMO                      | MEMO                      |

★ Если ваше приложение читает и записывает в существующие файлы, достаточно будет STRING с шаблонами. Однако, если ваше приложение создает dBaseIV файл, вам может потребоваться дополнительная информация для этих типов dBaseIV :

Чтобы создать цифровое поле в словаре данных, выберите тип данных REAL (действительный). В атрибуте Внешнее имя (External Name) установите 'NumericFieldName=N(Precision,DecimalPlaces)' где NumericFieldName - имя поля, Precision - точность поля, а DecimalPlaces - число десятичных мест.

Если вы кодируете вручную присущие Clarion типы данных, добавьте атрибут NAME используя тот же самый синтаксис. Если вы кодируете вручную символьный шаблон, STRING@N-\_9.2),NAME('Number'),где Number - имя поля.

★ Чтобы создать логическое поле, используя словарь данных, выберите тип данных BYTE. Специальных шагов нет; однако, смотрите в разных разделах советы о чтении данных из поля.

Если вы вручную кодируете STRING с шаблоном, добавьте атрибут NAME: STRING(1),NAME('LogFld=L').

★ Чтобы создать поле даты, используя словарь данных, выберите тип данных DATE, а не LONG, который вы обычно используете для форматов файлов TopSpeed или Clarion.

★ Объявления поля MEMO требуют указателя поля в файловой структуре записи. Объявите указатель поля как STRING(10) или LONG. Это поле будет сохранено в файле .DBF, содержащем смещение поля-мемо в файле .DBT. Объявление MEMO должно иметь атрибут NAME(), называющий указатель поля. Пример объявления поля:

```
File FILE, DRIVER('dBase IV')
Memo1 MEMO(200), NAME('Notes')
Memo2 MEMO(200), NAME('Text')
Rec    RECORD
Mem1Ptr LONG, NAME('Notes')
Mem2Ptr STRING(10), NAME('Text')
END
```

END

## Характеристики файла/максимумы

---

|                         |                                       |
|-------------------------|---------------------------------------|
| Размер файла:           | 2,000,000,000 байт                    |
| Записей на файл:        | 1,000,000,000 байт                    |
| Размер записи:          | 4,000 байт                            |
| Размер поля             |                                       |
| Символы:                | 254 байт                              |
| Данные:                 | 8 байт                                |
| Логическое:             | 1 байт                                |
| Цифровое:               | 20 байт включая десятичную точку      |
| Плавающее:              | 20 байт включая десятичную очку       |
| Мемо:                   | 64K (см. замечание)                   |
| Полей на запись:        | 255                                   |
| Ключей/индексов на файл |                                       |
| .NDX:                   | неограниченно                         |
| .MDX                    | 47 признаков на .MDX файлы            |
| Размер ключа            |                                       |
| Символ:                 | 100 байт                              |
| Цифры, данные:          | 8 байт                                |
| Мемо полей на файл:     | в зависимости от имеющейся памяти     |
| Открывание файла:       | в зависимости от операционной системы |

## Строки символов драйвера и функции SEND

---

Строкам символов драйвера (второй параметр атрибута DRIVER (драйвер)) всегда предшествует символ наклонной черты (/). Команды функции SEND могут принять два формата - один со знаком равенства - модифицирует установку переключения и возвращает величину предыдущей установки переключения; другой формат (без знака равенства) возвращает величину переключения.

Строки символов драйвера отсылаются к драйверу файла, когда файл открыт. Функция SEND посылает команду на изменение установки после того, как файл открыт. Некоторые строки символов драйвера не оказывают никакого влияния после того, как файл открыт, так что синтаксис функции SEND модификации установки не перечисляется. Однако, синтаксис функции SEND для возвращения величины переключения перечисляется для всех строк символов драйвера.

### /BUFFERS=n

Устанавливает величину для числа буферов, используемых для чтения и записи в файл.

Драйвер dBase IV использует систему буфера DOS. По умолчанию имеется три буфера по 1024 байта каждый. Увеличение числа буферов не улучшает эффективность работы, если доступ к файлу делится между многими пользователями.



**SEND(file, 'BUFFERS')**

Возвращает число буферов в форме строки символов STRING.

**/RECOVER****SEND(file, 'RECOVER')**

Эквивалентно команде Xbase RECALL, которая восстанавливает записи, маркированные для удаления. При использовании драйвера dBase III оператор DELETE отмечает флажком запись как “неактивную”. Драйвер не удаляет запись до тех пор, пока не выполнены команды PACK (упаковать) или BUILD (построить).

/RECOVER оценивается каждый раз при открывании вами файла, если вы добавите строку символов драйвера к словарю данных. Когда драйвер восстанавливает записи, ранее отмаркированные для удаления, вы должны вручную перестроить ключи и индексы с помощью оператора BUILD.

**SEND(file, 'IGNORESTATUS=on|off')****/IGNORESTATUS=on|off**

Когда установка ON, драйвер не перескакивает через удаленные записи во время выполнения операторов GET, NEXT и PREVIOUS, при доступе к файлу, открытому в физической последовательности. Это также делает доступным выполнение оператора PUT для удаленной или задержанной записи. /IGNORESTATUS требует открывания файла в эксклюзивном режиме.

**SEND(file, 'IGNORESTATUS')**

Возвращает установку (ON или OFF) IGNORESTATUS в форме STRING(3).

**SEND(file, 'DELETED')**

Для использования только с командой SEND, когда включен IGNORESTATUS. Сообщает состояние текущей записи. Если она удалена, возвратная строка символов - “ON”, если нет - “OFF”.

## **Поддерживаемые атрибуты файла, команды и функции**

---

### **Атрибуты файла**

### **Поддерживаемые**

|                                   |      |
|-----------------------------------|------|
| CREATE                            | Да   |
| DRIVER (filetype,[driver string]) | Да   |
| NAME                              | Да   |
| ENCRYPT                           | Нет  |
| OWNER (password)                  | Нет  |
| RECLAIM                           | Нет6 |
| PRE (prefix)                      | Да   |
| BINDABLE                          | Да   |
| THREAD                            | Да   |
| EXTERNAL (member)                 | Да   |
| DLL ([flag])                      | Да   |
| OEM                               | Да   |

### **Файловые структуры**

### **Поддерживаемые**

|        |     |
|--------|-----|
| INDEX  | Да  |
| KEY    | Да  |
| MEMO   | Да  |
| BLOB   | Нет |
| RECORD | Да  |

### **Атрибуты индекса, ключа, мемо** **Поддерживаемые**

|                       |      |
|-----------------------|------|
| BINARY                | Нет  |
| DUP                   | Да11 |
| NOCASE                | Да   |
| OPT                   | Нет  |
| PRIMARY               | Да   |
| NAME                  | Да   |
| Ascending Components  | Да   |
| Descending Components | Да   |
| Mixed Components      | Нет  |

### **Файловые команды**

### **Поддерживаемые**

|                                 |     |
|---------------------------------|-----|
| BUILD(file)                     | Да  |
| BUILD (key)                     | Да  |
| BUILD (index)                   | Да  |
| BUILD (index,components)        | Да1 |
| BUILD (index,components,filter) | Нет |
| CLOSE (file)                    | Да  |

|                          |     |
|--------------------------|-----|
| COPY (file,new file)     | Да2 |
| CREATE (file)            | Да  |
| EMPTY (file)             | Да  |
| FLUSH (file)             | Да  |
| LOCK (file)              | Нет |
| OPEN (file,access mode)  | Да  |
| PACK (file)              | Да  |
| REMOVE (file)            | Да  |
| RENAME (file,new file)   | Да3 |
| SHARE (file,access mode) | Да  |
| STATUS (file)            | Да  |
| STREAM (file)            | Да  |
| UNLOCK (file)            | Да  |

**Доступ к записям****Поддерживаемые**

|                               |     |
|-------------------------------|-----|
| ADD (file)                    | Да5 |
| ADD (file,length)             | Нет |
| APPEND (file)                 | Да5 |
| APPEND (file,length)          | Нет |
| DELETE (file)                 | Да6 |
| GET (file,key)                | Да  |
| GET (file,filepointer)        | Да  |
| GET (file,filepointer,length) | Нет |
| GET (file,keypointer)         | Да  |
| HOLD (file)                   | Да7 |
| NEXT (file)                   | Да  |
| NOMEMO (file)                 | Да  |
| PREVIOUS (file)               | Да  |
| PUT (file)                    | Да  |
| PUT (file,filepointer)        | Да  |
| PUT (file,filepointer,length) | Нет |
| RELEASE (file)                | Да  |
| REGET ( file,string)          | Да  |
| REGET ( key,string)           | Да  |
| RESET ( file,string)          | Да  |
| RESET ( key,string)           | Да  |
| SET (file)                    | Да  |
| SET (file,key)                | Да  |
| SET (file,filepointer)        | Да  |
| SET (key)                     | Да  |
| SET (key,key)                 | Да  |
| SET (key,keypointer)          | Да  |
| SET (key,key,filepointer)     | Да  |

|                   |    |
|-------------------|----|
| SKIP (file,count) | Да |
| WATCH (file)      | Да |

**Файловые функции****Поддерживаемые**

|                     |      |
|---------------------|------|
| BOF (file)          | Да8  |
| BYTES (file)        | Нет  |
| DUPLICATE (file)    | Да   |
| DUPLICATE (key)     | Да   |
| EOF (file)          | Да8  |
| NAME (label)        | Да   |
| POINTER (file)      | Да9  |
| POINTER (key)       | Да9  |
| POSITION (file)     | Да   |
| POSITION (key)      | Да   |
| RECORDS (file)      | Да10 |
| RECORDS (key)       | Да10 |
| SEND (file,message) | Да   |

**Диалоговая обработка****Поддерживаемые**

|                               |     |
|-------------------------------|-----|
| LOGOUT (timeout,file,...file) | Нет |
| COMMIT                        | Нет |
| ROLLBACK                      | Нет |

**Обработка пустых данных****Поддерживаемые**

|                    |     |
|--------------------|-----|
| NULL (field)       | Нет |
| SETNULL (field)    | Нет |
| SETNONNULL (field) | Нет |

**Замечания**

1 При построении динамических индексов, компоненты могут принять одну из двух следующих форм:

BUILD(DynNdx, '+Pre:FLD1, -Pre:FLD2' )

Эта форма устанавливает имена полей, на которых строится индекс. Имена полей должны появиться в атрибуте поля NAME, если он используется, или это должны быть имена меток. Для совместимости с Clarion может быть использован префикс, но он игнорируется.

BUILD(DynNdx, 'T[Expression]')

Эта форма устанавливает тип и выражение, используемые для построения индекса, о чем смотрите ниже - в разделе “Разное” - Определение ключа..

2 Команда COPY() копирует файлы данных и мемо файлы, используя newfile (новый файл), при помощи которого вы можете установить новое имя файла или каталога. Файлы ключей или индексов копируются, если newfile является характеристикой подкаталога. Для копирования индексного файла в новый файл используйте специальную форму команды копирования:

```
COPY(file, '<index>|< newfile >')
```

Этот оператор выдает сообщение File Not Found (файл не найден), если используется ссылка на недействительный индекс. Команда COPY предполагает расширение по усмотрению “.NDX” для имен файлов источника и целевого файла, если ни одно из них специально не определено. Если вы требуете имя файла без расширения, закончите имя точкой.

Дана структура файла:

```
Clar2 FILE,CREATE,DRIVER('dBase4')
NumKey KEY(Num),DUP
StrKey KEY(Str1)
StrKey2 KEY(Str2)
AMemo MEMO(100),NAME('mem')
Record RECORD
Num STRING(@n-_9.2)
STR1 STRING(2)
STR2 STRING(2)
Mem STRING(10)
```

Следующие команды копируют это определение файла в A:

```
COPY(Clar2,'A:\CLAR2')
COPY(Clar2,'StrKey|A:\STRKEY')
COPY(Clar2,'StrKey2|A:\STRKEY2')
COPY(Clar2,'NumKey|A:\NUMKEY')
```

После этих вызовов на накопителе A будут существовать следующие файлы:

CLAR2.DBE,CLAR2.DBT,STRKEY.NTX,STRLEY2.NTX и NUMKEY.NTX.

3 Команда RENAME копирует файлы данных и файлы мемо с помощью newfile, определяющем новое имя файла или путь. Файлы ключей и индексов должны быть переименованы с помощью того же синтаксиса, что и команда COPY (см. выше).

5 Оператор ADD проверяет на дублирование ключи перед модификацией файла данных

или связанных с ним ключевых KEY файлов. В результате оператор ADD выполняется медленнее, чем APPEND, который не выполняет никаких проверок и не обновляет ключи KEY. При добавлении больших количеств данных к базе данных, используйте APPEND...BUILD вместо ADD.

6 Когда драйвер удаляет запись из базы данных dBase IV запись физически не удаляется, а вместо этого драйвер маркирует ее, как неактивную. Поля мемо не удаляются физически из мемо файла, однако они не могут быть отысканы, если они относятся к неактивной записи. Величины ключа удалены из индексных файлов. Чтобы удалить записи и мемо файлы навсегда, выполните оператор PACK(file).

**Совет: Для программистов, знакомых с dBase IV, этот драйвер обрабатывает удаленные записи в соответствии с тем, как dBase IV обрабатывает их после того, как дана команда SET DELETED ON (установите удаленное). Записи, отмеченные для удаления, игнорируются исполняемыми программными операторами, но остаются в файле данных.**

7 dBase IV выполняет блокирование записи путем блокирования всей записи в файле данных. Это предотвращает доступ для чтения для других процессов. Следовательно, мы рекомендуем уменьшить до предела количество времени, на которое удерживается запись.

8 Хотя драйвер поддерживает эти функции, мы не рекомендуем их использовать. Эти функции для своего выполнения требуют физического доступа к файлам и значительных ресурсов. Вместо этого проверьте величину, возвращаемую функцией ERRORCODE() после каждого последовательного доступа. NEXT() или PREVIOUS() посылают код ошибки 33 (запись недоступна), если сделана попытка получить доступ к записи за концом или началом файла.

9 Нет различия между указателями файла и указателями ключа; и тот и другой содержат ту же самую величину для любой данной записи.

10 В рамках dBase IV функция RECORDS() сообщает одно и то же число записей для файла данных и его ключей и индексов. Обычно не бывает различия в числе записей, если INDEX не устарел. Так как оператор DELETE не удаляет записи физически, число записей, о которых сообщает функция RECORDS(), включает неактивные записи. Будьте осторожны при использовании этой функции. Имена полей должны появиться в соответствии с назначением в атрибуте полей NAME(), если он обеспечен, или должно быть имя метки. Может быть использован префикс для совместимости с условиями Clarion'a, но игнорируется.

11 В dBase IV законным является ввод многих записей с дубликатами уникальных ключевых компонент. Однако только первая из этих записей индексируется. Таким образом обработка в ключевом порядке показывает только эту первую запись. Если вы удаляете запись, затем вводите новую запись с той же самой величиной ключа, ключевой файл

продолжает указывать на удаленную запись, а не на новую запись. В этой ситуации файловый драйвер dBase IV Clarion for Windows изменяет ключевой файл, чтобы указать на активную запись, а не на удаленную запись. Это означает, что если вы используете программу dBaseIV для удаления уникальной записи, а затем вставите дубликат этой записи, новая запись невидима при обработке в ключевом порядке до тех пор, пока не выполнена упаковка. Если вы выполняете тот же самый процесс в программе Clarion for Windows 2.0, новая запись оказывается видимой при обработке в ключевом порядке.

## Разное

---

### Международная последовательность сортировки

★ Драйвер dBase IV сортирует так, как будто бы в поле нет диакритических знаков, поэтому А сортируется так же, как Д . Если два слова идентичны за исключением диакритических знаков, то эти слова сортируются так, как будто диакритический символ больше, чем нормальный символ. Например, Д а < А b < Д b , в то время как CLADIGRAPH Д АЕ будет сортировать как А b < Д а < Д b. Решение - если тот же самый файл использован в Clarion for Windows и в dBase IV, выполните опертор BUILD чтобы перестроить ключи перед обновлением файла (чтение файла не вызывает проблем).

### Булева оценка

★ dBase IV позволяет логическому полю принять одну из девяти возможных величин (1,0,y,Y,n,N,t,T,f,F или символ пробела). Символ пробела - это ни истина и ни ложь. При использовании логического поля из заранее существующей базы данных в логическом выражении учтите все эти возможности. Помните, что когда поле STRING используется как выражение, оно истинно, если содержит какие-либо данные, и оно ложно, если равно нулю или пусто. Следовательно, чтобы оценить истинность логического поля, выражение должно быть истина, если поле содержит какую-либо из "истинных" характеристик (T,t,Y или y). Например, если логическое поле было использовано, чтобы установить продукт как налогооблагаемый или необлагаемый, выражение для оценки его истинности будет:

(If Condition):

Taxable='1' OR Taxable='T' OR Taxable='t' OR Taxable='Y' OR  
Taxable='y'

### Большие MEMO поля

★ Clarion for Windows поддерживает поля MEMO размером до 64К. Если у вас существует файл, включающий мемо, намного большее чем 64К, вы можете использовать файл, но не модифицировать большие MEMO.

★ Вы можете определить, когда ваше приложение встретит большое MEMO путем определения, что переменная указателя мемо не пуста, но поле мемо при этом выглядит пустым. Выдается код ошибки 47 (неправильное объявление записи), любая модификация MEMO поля будет игнорироваться.

### Тип данных возвращаемый функцией POSITION (POSITION Return Data Type)

- ★ POSITION(file) возвращает STRING(12)
- ★ POSITION(key) возвращает STRING размер полей ключа + 4 байта.

### Длинные имена поля

★ dBase IV поддерживает максимум 10 символов в имени поля. Если вам нужно больше, используйте External Name (внешнее имя) с 10 или менее символами.

### Определение ключа

★ dBase IV поддерживает использование выражений для определения ключей. В редакторе словаря вы можете поместить выражение во внешнее имя поля в диалоговом окне Key Properties(свойства ключа). Общий формат внешнего имени:

‘FileName=T[Expression]’

Где FileName представляет имя индексного файла (который может содержать путь и расширение файла), а T представляет тип индекса. Действительными типами являются: C = символ, D = данные, N = цифра. Если типом является D или N, тогда Expression (выражение) может назвать только одно поле.

★ Многоиндексные файлы (.MDX) требуют атрибута NAME() для KEY и INDEX, чтобы установить тип хранения ключа и любое выражение, которое используется для генерации величин ключа. Общий формат атрибута NAME() для KEY или INDEX:

Name(‘TagName|FileName[ PageSize]=T[Expression],FOR[Expression]’)

Дальше приводятся параметры для атрибута NAME():

|          |                                                                                                                                                                                                                                                                                                                                                                                          |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TagName  | Устанавливает имя индексного признака в многоиндексном файле. Если он опущен, драйвер создает dBase IV стиля .NDX файл с помощью имени, установленного в FileName.                                                                                                                                                                                                                       |
| FileName | Устанавливает имя индексного файла, который может содержать путь и расширение.                                                                                                                                                                                                                                                                                                           |
| PageSize | Устанавливает, что при создании .MDX файла (если установлен TagName) число в интервале 2-32 определяет число 512-байтовых блоков в каждой индексной странице. Эта величина используется только при создании файла. Если вы установите кратные величины через объявления для различных признаков в одном и том же .MDX файле, выбрана будет наибольшая величина. Величина по умолчанию 2. |



|            |                                                                                                                                                                                                                                |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| T          | Устанавливает тип индекса. Законные типы - это C = символ, D = данные, N = цифра. Если типом являются D или N, выражение Expression может называть только одно поле.                                                           |
| Expression | Определяет выражение для генерации индекса. Оно может относиться к множественным полям и вызывать различные xBase функции. Текущие поддерживаемые функции приведены ниже. Выражение должно быть заключено в квадратные скобки. |

★ Элементы атрибута NAME() могут быть опущены справа. При определении выражения вы должны также установить тип и имя. Если выражение опущено, драйвер определяет выражение из ключевых полей, когда файл создан, или из индексного файла, когда открыт.

Если опущен тип, драйвер определяет тип индекса из первого компонента ключа, когда файл создан, или из индексного файла, когда открыт.

Если атрибут NAME() опущен целиком, имя индексного файла определяется из метки ключа. По умолчанию путь установлен на то же местоположение, что и .DBF.

Имена признаков ограничены по длине 9 символами. Если представленное имя слишком велико, оно автоматически усекается.

Установите все имена поля в атрибуте NAME() без префикса.

★ dBase IV автоматически поддерживает использование оператора Xbase FOR в выражении для определения ключей. Выражения, поддерживаемые в условии FOR, должны быть простым условием вида:

expression comparison\_or expression

comparison\_or может быть одно из следующих: <,<=,<,>,<=>,>= или >.

Выражение может относиться к множественным полям в записи и содержать функции xBase. Выражение должно быть заключено в квадратные скобки. Текущие поддерживаемые функции даются ниже. Если драйвер встречает неподдерживаемую xBase функцию в заранее существовавшем файле, он посылает код ошибки 76 “Недействительная строка символов индекса”, когда файл открыт для ключей и статических индексов.

Выражения в виде строк символов могут использовать оператор “+” для конкатенации многих строк символов аргументов. Цифровые выражения используют операторы ‘+’ или ‘-’ в их основном смысле. Максимальная длина Clipper выражения - 250 символов.

### Поддерживаемые функции определения ключа xBase

|                 |                                                               |
|-----------------|---------------------------------------------------------------|
| ALLTRIN(string) | Удаляет начальные и конечные пробелы.                         |
| CTOD(string)    | Преобразует ключ в виде строки символов в дату. Формат строки |

string mm/dd/yy; результат имеет форму 'ууууммдд'. Элемент уууу по умолчанию означает двадцатое столетие. Недействительные данные дают в результате пустой ключ.

|                                          |                                                                                                                                                                                                                          |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DELETED()                                | Возвращает TRUE (истинно) если запись удалена.                                                                                                                                                                           |
| DTOC(date)                               | Преобразует ключ данных в строку символов формата 'mm/dd/yy'.                                                                                                                                                            |
| DTOS(date)                               | Преобразует ключ данных в строку символов формата 'ууууммдд'.                                                                                                                                                            |
| FIXED(float)                             | Преобразует ключ из представления с плавающей точкой в представление с фиксированной точкой.                                                                                                                             |
| FLOAT(numeric)                           | Преобразует ключ из представления с фиксированной точкой в представление с плавающей точкой.                                                                                                                             |
| IF(bool, val1, val2)                     | Возвращает величину val1 если первый параметр -TRUE, в противном случае возвращает величину val2.                                                                                                                        |
| LEFT(string, n)                          | Возвращает самые левые n символов строки символов ключа в виде строки символов длиной n.                                                                                                                                 |
| LOWER (string, n)                        | Преобразует строку символов ключа в нижний регистр.                                                                                                                                                                      |
| LTRIM (string)                           | Удаляет пробелы слева в строке символов.                                                                                                                                                                                 |
| RECNO( )                                 | Возвращает номер текущей записи.                                                                                                                                                                                         |
| RIGHT(string, n)                         | Возвращает самые правые n символов строки символов ключа в виде строки символов длиной n.                                                                                                                                |
| RTRIM(string)                            | Удаляет пробелы с правой стороны строки символов.                                                                                                                                                                        |
| STR(numeric [,length[, decimal places]]) | Преобразует цифровые данные в строку символов. Длина строки символов и число десятичных знаков после запятой - параметры необязательные. По умолчанию длина строки символов 10, а число знаков после десятичной точки 0. |
| SUBSTR(string,offset,n)                  | Возвращает подстроку символов ключа в виде строки символов string, начиная со позиции offset и длиной в n символов.                                                                                                      |
| TRIM(string)                             | Удаляет пробелы с правой стороны строки символов (идентично RTRIM).                                                                                                                                                      |
| UPPER(string)                            | Преобразует строку символов ключа в верхний регистр.                                                                                                                                                                     |
| VAL(string)                              | Преобразует строку символов ключа в цифру.                                                                                                                                                                               |

Файлы DOS

Драйвер файла DOS читает и записывает любые бинарные, байт-адресуемые файлы. Ни поля, ни записи не имеют разграничителей. При чтении записи драйвер читает число байт, определенных в структуре файла RECORD, если параметр длины не установлен в операторе GET.

Драйвер DOS поддерживает параметр длины для операторов ADD, APPEND, GET и PUT; это позволяет вести в файле DOS записи переменной длины.

Функция POINTER (указатель) возвращает относительную байтовую позицию в пределах файла для начала последней записи, к которой имели доступ операторы ADD, APPEND, GET или NEXT.

Этот драйвер файла выполняет только последовательную обработку вперед. Не поддерживаются никакие ключи или функции обработки транзакции, не поддерживается и оператор PREVIOUS.

**Совет: В связи с его ограничениями, главная функция данного драйвера - это функция дискового редактора для бинарных файлов.**

|       |              |                                                         |
|-------|--------------|---------------------------------------------------------|
| Файлы | CL2DOS16.LIB | Экспортная библиотека Windows (16-битовая)              |
|       | CL2DOS32.LIB | Экспортная библиотека Windows (32-битовая)              |
|       | CW2DOS16.LIB | Статически загружаемая библиотека Windows (16-битовая)  |
|       | CW2DOS32.LIB | Статически загружаемая библиотека Windows (32-битовая)  |
|       | CW2DOS16.DLL | Динамически загружаемая библиотека Windows (16-битовая) |
|       | CW2DOS32.DLL | Динамически загружаемая библиотека Windows (32-битовая) |

Типы данных

|         |          |
|---------|----------|
| BYTE    | DECIMAL  |
| SHORT   | PDECIMAL |
| USHORT  | STRING   |
| LONG    | CSTRING  |
| ULONG   | PSTRING  |
| SREAL   | DATE     |
| REAL    | TIME     |
| BFLOAT4 | GROUP    |
| BFLOAT4 |          |

## Характеристики файла/максимумы

|                         |                                       |
|-------------------------|---------------------------------------|
| Размер файла:           | 4,294,967,295 байт                    |
| Записей на файл:        | 4,294,967,295 байт                    |
| Размер записи:          | 64К                                   |
| Размер поля             | 64К                                   |
| Полей на запись:        | 64К                                   |
| Ключей/индексов на файл | неограниченно                         |
| Размер ключа            | неограничен                           |
| Мемо полей на файл:     | неограниченно                         |
| Размер поля мемо        | неограничен                           |
| Открывание файла:       | в зависимости от операционной системы |

## Строки символов драйвера и функции SEND

Строкам символов драйвера (второй параметр атрибута DRIVER (драйвер)) всегда предшествует символ наклонной черты (/). Команды функции SEND могут принять два формата - один со знаком равенства - модифицирует установку переключения и возвращает величину предыдущей установки переключения; другой формат (без знака равенства) возвращает величину переключения.

Строки символов драйвера отсылаются к драйверу файла, когда файл открыт. Функция SEND посылает команду на изменение установки после того, как файл открыт. Некоторые строки символов драйвера не оказывают никакого влияния после того, как файл открыт, так что синтаксис функции SEND модификации установки не перечисляется. Однако, синтаксис функции SEND для возвращения величины переключения перечисляется для всех строк символов драйвера.

### /FILEBUFFERS=n

Устанавливает величину для числа буферов, используемых для чтения и записи в файл.

Драйвер DOS размещает внутренние буферы по 512 байт или размером вашей записи, что больше. Число буферов по умолчанию 2 для файлов, открытых с запрещением записи для других пользователей и 1 для всех других режимов открытия. Используйте необязательную строку символов драйвера, для увеличения числа буферов если вы найдете, что доступ к записям слишком медленен.

### SEND(file, 'FILEBUFFERS')

Возвращает величину числа буферов в форме строки символов STRING.

/QUICKSCAN=on|off SEND(file, 'QUICKSCAN=on|off')

Драйвер DOS читает буфер за раз (не запись), что обеспечивает быстрый доступ. В многопользовательской среде эти буферы не имеют 100% - ой надежности для последовательного доступа, так как другой пользователь может изменить базу данных отдельными обращениями к данным. В качестве меры защиты драйвер повторно считывает буферы перед каждым доступом к записи. Чтобы устранить повторное чтение, установите QUICKSCAN на ON. Позиция по умолчанию ON для файлов, открытых с запрещением доступа по записи для других пользователей и OFF для всех других режимов открытия.

### **SEND(file,'QUICKSCAN')**

Возвращает установку (ON или OFF) QUICKSCAN в форме STRING.

## **Разное**

---

- ◆ POSITION(File) возвращает STRING(4).

## **Поддерживаемые атрибуты файла, команды и функции**

---

### **Атрибуты файла**

CREATE .  
 DRIVER (filetype,[driver string])  
 NAME  
 ENCRYPT  
 OWNER (password)  
 RECLAIM  
 PRE (prefix)  
 BINDABLE  
 THREAD  
 EXTERNAL (member)  
 DLL ([flag])  
 OEM

### **Поддерживаемые**

Да  
 Да  
 Да  
 Нет  
 Нет  
 Нет  
 Да  
 Да  
 Да  
 Да  
 Да  
 Да

### **Файловые структуры**

INDEX  
 KEY  
 MEMO  
 BLOB  
 RECORD

### **Поддерживаемые**

Нет  
 Нет  
 Нет  
 Нет  
 Да

### **Атрибуты индекса, ключа, мемо**

BINARY

### **Поддерживаемые**

Нет

|                       |     |
|-----------------------|-----|
| DUP                   | Нет |
| NOCASE                | Нет |
| OPT                   | Нет |
| PRIMARY               | Нет |
| NAME                  | Нет |
| Ascending Components  | Нет |
| Descending Components | Нет |
| Mixed Components      | Нет |

### **Файловые команды**

|                                 |     |
|---------------------------------|-----|
| BUILD(file)                     | Нет |
| BUILD (key)                     | Нет |
| BUILD (index)                   | Нет |
| BUILD (index,components)        | Нет |
| BUILD (index,components,filter) | Нет |
| CLOSE (file)                    | Да  |
| COPY (file,new file)            | Да  |
| CREATE (file)                   | Да  |
| EMPTY (file)                    | Да  |
| FLUSH (file)                    | Нет |
| LOCK (file)                     | Да  |
| OPEN (file,access mode)         | Да  |
| PACK (file)                     | Нет |
| REMOVE (file)                   | Да  |
| RENAME (file,new file)          | Да  |
| SHARE (file,access mode)        | Да  |
| STATUS (file)                   | Да  |
| STREAM (file)                   | Нет |
| UNLOCK (file)                   | Да  |

### **Доступ к записям**

|                               |     |
|-------------------------------|-----|
| ADD (file)                    | Да  |
| ADD (file,length)             | Да  |
| APPEND (file)                 | Да  |
| APPEND (file,length)          | Да  |
| DELETE (file)                 | Нет |
| GET (file,key)                | Нет |
| GET (file,filepointer)        | Да  |
| GET (file,filepointer,length) | Да  |
| GET (file,keypointer)         | Нет |
| HOLD (file)                   | Нет |
| NEXT (file)                   | Да  |
| NOMEMO (file)                 | Нет |

### **Поддерживаемые**

|     |
|-----|
| Нет |
| Нет |
| Нет |
| Нет |
| Нет |
| Да  |
| Да  |
| Да  |
| Да  |
| Нет |
| Да  |
| Да  |
| Нет |
| Да  |
| Да  |
| Да  |
| Нет |
| Да  |

### **Поддерживаемые**

|     |
|-----|
| Да  |
| Да  |
| Да  |
| Да  |
| Нет |
| Нет |
| Да  |
| Да  |
| Нет |
| Нет |
| Да  |
| Нет |

|                               |     |
|-------------------------------|-----|
| PREVIOUS (file)               | Нет |
| PUT (file)                    | Да  |
| PUT (file,filepointer)        | Да  |
| PUT (file,filepointer,length) | Да  |
| RELEASE (file)                | Нет |
| REGET ( file,string)          | Да  |
| REGET ( key,string)           | Нет |
| RESET ( file,string)          | Да  |
| RESET ( key,string)           | Нет |
| SET (file)                    | Да  |
| SET (file,key)                | Нет |
| SET (file,filepointer)        | Да  |
| SET (key)                     | Нет |
| SET (key,key)                 | Нет |
| SET (key,keypointer)          | Нет |
| SET (key,key,filepointer)     | Нет |
| SKIP (file,count)             | Нет |
| WATCH (file)                  | Нет |

**Файловые функции**

|                     |     |
|---------------------|-----|
| BOF (file)          | Нет |
| BYTES (file)        | Да  |
| DUPLICATE (file)    | Нет |
| DUPLICATE (key)     | Нет |
| EOF (file)          | Да  |
| NAME (label)        | Да  |
| POINTER (file)      | Да  |
| POINTER (key)       | Нет |
| POSITION (file)     | Да  |
| POSITION (key)      | Нет |
| RECORDS (file)      | Нет |
| RECORDS (key)       | Нет |
| SEND (file,message) | Да  |

**Диалоговая обработка**

|                               |     |
|-------------------------------|-----|
| LOGOUT (timeout,file,...file) | Нет |
| COMMIT                        | Нет |
| ROLLBACK                      | Нет |

**Обработка пустых данных**

|                 |     |
|-----------------|-----|
| NULL (field)    | Нет |
| SETNULL (field) | Нет |

**Поддерживаемые****Поддерживаемые****Поддерживаемые**

SETNONULL (field)

Нет

**Файлы FoxPro и FoxBased**

Драйвер файла FoxPro совместим с FoxPro FoxBase. Расширение файла по умолчанию \*.DBF.

Расширение индексного файла по умолчанию \*.IDX. Расширение по умолчанию файла Мемо .FBT. FoxPro также поддерживает файлы множественных индексов ( расширение по умолчанию - \*.CDX). В различных разделах описываются процедуры использования файлов .CDX.

|        |              |                                                         |
|--------|--------------|---------------------------------------------------------|
| Файлы: | CL2FOX16.LIB | Экспортная библиотека Windows (16-битовая)              |
|        | CL2FOX32.LIB | Экспортная библиотека Windows (32-битовая)              |
|        | CW2FOX16.LIB | Статически загружаемая библиотека Windows (16-битовая)  |
|        | CW2FOX32.LIB | Статически загружаемая библиотека Windows (32-битовая)  |
|        | CW2FOX16.DLL | Динамически загружаемая библиотека Windows (16-битовая) |
|        | CW2FOX32.DLL | Динамически загружаемая библиотека Windows (32-битовая) |

Совет: Формат индексного файла FoxPro является становым хребтом его перевозносимой технологии “Rushmore”. Старая поговорка “There’s no free lunch” применима в данном случае. Добавление и присоединение записей к большой базе данных является более медленным процессом, чем в других форматах xBase, что связано со временем, необходимым для обновления индексного файла.

**Типы данных**

Формат xBase сохраняет все данные, как строки символов ASCII. Вы можете либо установить типы строк STRING с шаблоном для каждого поля, либо установить типы Clarion, которые драйвер преобразует автоматически.

| <u>тип данных</u> | <u>FoxPro тип данных</u> | <u>Clarion</u> | <u>STRING с картинкой</u> |
|-------------------|--------------------------|----------------|---------------------------|
|                   | DATE                     | DATE           | STRING(@D12)              |
|                   | *Numeric                 | REAL           | STRING(@N-_p.d)           |
|                   | *Logical                 | BYTE           | STRING(1)                 |
|                   | Character                | STRING         | STRING                    |
|                   | *Memo                    | MEMO           | MEMO                      |

Если ваше приложение читает и записывает в существующие файлы, достаточно будет STRING с шаблоном. Однако, если ваше приложение создает FoxPro или FoxBase файл, вам может потребоваться дополнительная информация для этих типов FoxPro и FoxBase :



♦ Чтобы создать цифровое поле в словаре данных, выберите тип данных REAL. В атрибуте External Name (внешнее имя) установите 'NumericFieldName=N(Precision,DecimalPlaces)' где NumericFieldName - имя поля, Precision - точность поля, а DecimalPlaces - число десятичных мест.

Если вы вручную кодируете тип данных, родной для Clarion, добавьте атрибут с помощью того же самого индекса. Если вы кодируете вручную STRING с шаблоном, STRING@N\_9.2),NAME('Number'),где Number - имя поля.

♦ Чтобы создать логическое поле, используя словарь данных, выберите тип данных BYTE. Специальных шагов нет; однако, смотрите в разных разделах советы о чтении данных из поля.

Если вы вручную кодируете STRING с шаблоном, добавьте атрибут NAME: STRING(1),NAME('LogFld=L').

♦ Чтобы создать поле даты, используя словарь данных, выберите тип данных DATE, а не LONG, который вы обычно используете для форматов файлов TopSpeed или Clarion.

♦ Объявления поля MEMO требуют указателя поля в файловой структуре записи. Объявите указатель поля как STRING(10) или LONG. Это поле будет сохранено в файле .DBF, содержащем смещение поля-мемо в файле .DBT. Объявление MEMO должно иметь атрибут NAME(), называющий указатель поля. Пример объявления поля:

```
File FILE, DRIVER('FoxPro')
Memo1 MEMO(200), NAME('Notes')
Memo2 MEMO(200), NAME('Text')
Rec RECORD
Mem1Ptr LONG, NAME('Notes')
Mem2Ptr STRING(10), NAME('Text')
END
END
```

### Характеристики файла/максимумы

---

|                  |                                  |
|------------------|----------------------------------|
| Размер файла:    | 2,000,000,000 байт               |
| Записей на файл: | 1,000,000,000 байт               |
| Размер записи:   | 4,000 байт                       |
| Размер поля      |                                  |
| Символы:         | 254 байт                         |
| Данные:          | 8 байт                           |
| Логическое:      | 1 байт                           |
| Цифровое:        | 20 байт включая десятичную точку |

|                          |                                       |
|--------------------------|---------------------------------------|
| Плавающее:               | 20 байт включая десятичную точку      |
| Мемо:                    | 64,520 (см. замечание)                |
| Полей на запись:         | 255                                   |
| Ключей/индексов на файл: | неограниченно                         |
| Размер ключа             |                                       |
| Символ:                  | 100 байт (.IDX)<br>254 байта (.CDX)   |
| Цифры, данные:           | 8 байт                                |
| Мемо полей на файл:      | в зависимости от имеющейся памяти     |
| Открывание файла:        | в зависимости от операционной системы |

## Строки символов драйвера и функции SEND

Строкам символов драйвера (второй параметр атрибута DRIVER (драйвер)) всегда предшествует символ наклонной черты (/). Команды функции SEND могут принять два формата - один со знаком равенства модифицирует установку переключения и возвращает величину предыдущей установки переключения; другой формат (без знака равенства) возвращает величину переключения.

Строки символов драйвера отсылаются к драйверу файла, когда файл открыт. Функция SEND посылает команду на изменение установки после того, как файл открыт. Некоторые строки символов драйвера не оказывают никакого влияния после того, как файл открыт, так что синтаксис функции SEND модификации установки не перечисляется. Однако, синтаксис функции SEND для возвращения величины переключения перечисляется для всех строк символов драйвера.

### /BUFFERS=n

Устанавливает величину для числа буферов, используемых для чтения и записи в файл.

Драйвер FoxPro использует систему буфера DOS. По умолчанию имеется три буфера по 1024 байта каждый. Увеличение числа буферов не улучшает эффективность работы, если доступ к файлу делится между многими пользователями.

### SEND(file, 'BUFFERS')

Возвращает число буферов в форме строки символов STRING.

### /RECOVER

### SEND(file, 'RECOVER')

Эквивалентно команде Xbase RECALL, которая восстанавливает записи, маркированные

для удаления. При использовании драйвера FoxPro оператор DELETE отмечает флажком запись как “неактивную”. Драйвер не удаляет запись до тех пор, пока не выполнены команды PACK (упаковать) или BUILD (построить).

/RECOVER оценивается каждый раз при открывании вами файла, если вы добавите строку символов драйвера к словарю данных. Когда драйвер восстанавливает записи, ранее отмаркированные для удаления, вы должны вручную перестроить ключи и индексы с помощью оператора BUILD.

**SEND(file, 'IGNORESTATUS=on|off')**

**/IGNORESTATUS=on|off**

Когда установка ON, драйвер не перескакивает через удаленные записи во время выполнения операторов GET, NEXT и PREVIOUS, при доступе к файлу, открытому в физической последовательности. Это также делает доступным выполнение оператора PUT для удаленной или задержанной записи. /IGNORESTATUS требует открывания файла в эксклюзивном режиме.

**SEND(file, 'IGNORESTATUS')**

Возвращает установку (ON или OFF) IGNORESTATUS в форме STRING(3).

**SEND(file, 'DELETED')**

Для использования только с командой SEND, когда включен IGNORESTATUS. Сообщает состояние текущей записи. Если она удалена, возвращаемая строка символов - “ON”, если нет - “OFF”.

## **Поддерживаемые атрибуты файла, команды и функции**

### **Атрибуты файла**

CREATE .  
 DRIVER (filetype,[driver string])  
 NAME  
 ENCRYPT  
 OWNER (password)  
 RECLAIM  
 PRE (prefix)  
 BINDABLE  
 THREAD  
 EXTERNAL (member)  
 DLL ([flag])  
 OEM

### **Поддерживаемые**

Да  
 Да  
 Да  
 Нет  
 Нет  
 Нет6  
 Да  
 Да  
 Да  
 Да  
 Да  
 Да

**Файловые структуры**

INDEX  
KEY  
MEMO  
BLOB  
RECORD

**Поддерживаемые**

Да  
Да  
Да  
Нет  
Да

**Атрибуты индекса, ключа, мемо**

BINARY  
DUP  
NOCASE  
OPT  
PRIMARY  
NAME  
Ascending Components  
Descending Components  
Mixed Components

**Поддерживаемые**

Нет  
Да11  
Да  
Нет  
Да  
Да  
Да  
Нет

**Файловые команды**

BUILD(file)  
BUILD (key)  
BUILD (index)  
BUILD (index,components)  
BUILD (index,components,filter)  
CLOSE (file)  
COPY (file,new file)  
CREATE (file)  
EMPTY (file)  
FLUSH (file)  
LOCK (file)  
OPEN (file,access mode)  
PACK (file)  
REMOVE (file)  
RENAME (file,new file)  
SHARE (file,access mode)  
STATUS (file)  
STREAM (file)  
UNLOCK (file)

**Поддерживаемые**

Да  
Да  
Да  
Да1  
Нет  
Да  
Да2  
Да  
Да  
Да  
Нет  
Да  
Да  
Да  
Да3  
Да  
Да  
Да  
Да

**Доступ к записям**

ADD (file)  
ADD (file,length)

**Поддерживаемые**

Да5  
Нет

|                               |     |
|-------------------------------|-----|
| APPEND (file)                 | Да5 |
| APPEND (file,length)          | Нет |
| DELETE (file)                 | Да6 |
| GET (file,key)                | Да  |
| GET (file,filepointer)        | Да  |
| GET (file,filepointer,length) | Нет |
| GET (file,keypointer)         | Да  |
| HOLD (file)                   | Да7 |
| NEXT (file)                   | Да  |
| NOMEMO (file)                 | Да  |
| PREVIOUS (file)               | Да  |
| PUT (file)                    | Да  |
| PUT (file,filepointer)        | Да  |
| PUT (file,filepointer,length) | Нет |
| RELEASE (file)                | Да  |
| REGET ( file,string)          | Да  |
| REGET ( key,string)           | Да  |
| RESET ( file,string)          | Да  |
| RESET ( key,string)           | Да  |
| SET (file)                    | Да  |
| SET (file,key)                | Да  |
| SET (file,filepointer)        | Да  |
| SET (key)                     | Да  |
| SET (key,key)                 | Да  |
| SET (key,keypointer)          | Да  |
| SET (key,key,filepointer)     | Да  |
| SKIP (file,count)             | Да  |
| WATCH (file)                  | Да  |

**Файловые функции**

|                     |      |
|---------------------|------|
| BOF (file)          | Да8  |
| BYTES (file)        | Нет  |
| DUPLICATE (file)    | Да   |
| DUPLICATE (key)     | Да   |
| EOF (file)          | Да8  |
| NAME (label)        | Да   |
| POINTER (file)      | Да9  |
| POINTER (key)       | Да9  |
| POSITION (file)     | Да   |
| POSITION (key)      | Да   |
| RECORDS (file)      | Да10 |
| RECORDS (key)       | Да10 |
| SEND (file,message) | Да   |

**Поддерживаемые**

**Диалоговая обработка**

LOGOUT (timeout,file,...file)  
 COMMIT  
 ROLLBACK

**Поддерживаемые**

Нет  
 Нет  
 Нет

**Обработка пустых данных**

NULL (field)  
 SETNULL (field)  
 SETNONNULL (field)

**Поддерживаемые**

Нет  
 Нет  
 Нет

**Замечания**

1 При построении динамических индексов компоненты могут принять одну из двух следующих форм:

BUILD(DynNdx, '+Pre:FLD1, -Pre:FLD2' )

Эта форма устанавливает имена полей, на которых строится индекс. Имена полей должны появиться в атрибуте поля NAME, если он используется, или это должны быть имена меток. Для совместимости с Clarion может быть использован префикс, но он игнорируется.

BUILD(DynNdx, 'T[Expression]')

Эта форма устанавливает тип и выражение, используемые для построения индекса, о чем смотрите ниже - в разделе "Разное" - Определение ключа..

2 Команда COPY() копирует файлы данных и мемо файлы, используя newfile (новый файл), при помощи которого вы можете установить новое имя файла или каталога. Файлы ключей или индексов копируются, если newfile является характеристикой подкаталога. Для копирования индексного файла в новый файл используйте специальную форму команды копирования:

COPY(file, '<index>|< newfile >')

Этот оператор выдает сообщение File Not Found (файл не найден), если используется ссылка на недействительный индекс. Команда COPY предполагает расширение по усмотрению ".NDX" для имен файлов источника и целевого файла, если ни одно из них специально не определено. Если вы требуете имя файла без расширения, закончите имя точкой.

Дана структура файла:

Clar2 FILE,CREATE,DRIVER('FoxPro')

```
NumKey KEY(Num),DUP
StrKey KEY(Str1)
StrKey2 KEY(Str2)
AMemo MEMO(100),NAME('mem')
Record RECORD
Num STRING(@n-_9.2)
STR1 STRING(2)
STR2 STRING(2)
Mem STRING(10)
```

Следующие команды копируют это определение файла в A:

```
COPY(Clar2,'A:\CLAR2')
COPY(Clar2,'StrKey|A:\STRKEY')
COPY(Clar2,'StrKey2|A:\STRKEY2')
COPY(Clar2,'NumKey|A:\NUMKEY')
```

После этих вызовов на накопителе A будут существовать следующие файлы: CLAR2.DBE, CLAR2.DBT, STRKEY.NTX, STRKEY2.NTX и NUMKEY.NTX.

3 Команда RENAME копирует файлы данных и файлы мемо с помощью newfile, определяющем новое имя файла или путь. Файлы ключей и индексов должны быть переименованы с помощью того же синтаксиса, что и команда COPY (см. выше).

5 Оператор ADD проверяет на дублирование ключи перед модификацией файла данных или связанных с ним ключевых KEY файлов. В результате оператор ADD выполняется медленнее, чем APPEND, который не выполняет никаких проверок и не обновляет ключи KEY. При добавлении больших количеств данных к базе данных, используйте APPEND...BUILD вместо ADD.

6 Когда драйвер удаляет запись из базы данных FoxPro, запись физически не удаляется, а вместо этого драйвер маркирует ее, как неактивную. Поля мемо не удаляются физически из мемо файла, однако они не могут быть отысканы, если они относятся к неактивной записи. Величины ключа удалены из индексных файлов. Чтобы удалить записи и мемо файлы навсегда, выполните оператор PACK(file).

**Совет: Для программистов, знакомых с FoxPro, этот драйвер обрабатывает удаленные записи в соответствии с тем, как FoxPro обрабатывает их после того, как дана команда SET DELETED ON (установите удаленное). Записи, отмеченные для удаления, игнорируются исполняемыми программными операторами, но остаются в файле данных.**

7 FoxPro выполняет блокирование записи путем блокирования всей записи в файле данных. Это предотвращает доступ для чтения для других процессов. Следовательно, мы рекомендуем уменьшить до предела количество времени, на которое удерживается запись.

8 Хотя драйвер поддерживает эти функции, мы не рекомендуем их использовать. Эти функции для своего выполнения требуют физического доступа к файлам и значительных ресурсов. Вместо этого проверьте величину, возвращаемую функцией `ERRORCODE()` после каждого последовательного доступа. `NEXT()` или `PREVIOUS()` посылают код ошибки 33 (запись недоступна), если сделана попытка получить доступ к записи за концом или началом файла.

9 Нет различия между указателями файла и указателями ключа; и тот и другой содержат ту же самую величину для любой данной записи.

10 В рамках FoxPro функция `RECORDS()` сообщает одно и то же число записей для файла данных и его ключей и индексов. Обычно не бывает различия в числе записей, если `INDEX` не устарел. Так как оператор `DELETE` не удаляет записи физически, число записей, о которых сообщает функция `RECORDS()`, включает неактивные записи. Будьте осторожны при использовании этой функции. Имена полей должны появиться в соответствии с назначением в атрибуте полей `NAME()`, если он обеспечен, или должно быть имя метки. Может быть использован префикс для совместимости с условиями Clarion'a, но игнорируется.

11 В FoxPro законным является ввод многих записей с дубликатами уникальных ключевых компонент. Однако только первая из этих записей индексируется. Таким образом обработка в ключевом порядке показывает только эту первую запись. Если вы удаляете запись, затем вводите новую запись с той же самой величиной ключа, ключевой файл продолжает указывать на удаленную запись, а не на новую запись. В этой ситуации файловый драйвер FoxPro Clarion for Windows изменяет ключевой файл, чтобы указать на активную запись, а не на удаленную запись. Это означает, что если вы используете программу FoxPro для удаления уникальной записи, а затем вставите дубликат этой записи, новая запись невидима при обработке в ключевом порядке до тех пор, пока не выполнена упаковка. Если вы выполняете тот же самый процесс в программе Clarion for Windows 2.0, новая запись оказывается видимой при обработке в ключевом порядке.

## Разное

---

### Булева оценка

◆ FoxPro и FoxBase позволяют логическому полю принять одну из девяти возможных величин (`y`, `Y`, `n`, `N`, `t`, `T`, `f`, `F` или символ пробела). Символ пробела - это ни истина и ни ложь. При использовании логического поля из заранее существующей базы данных в логическом выражении учтите все эти возможности. Помните, что когда поле `STRING` используется как выражение, оно истинно, если оно содержит какие-либо данные, и оно ложь, если равно нулю или пусто. Следовательно, чтобы оценить истинность логического поля, выражение должно быть истина, если поле содержит какую-либо из "истинных" характеристик (`T`, `t`, `Y` или `y`). Например, если логическое поле было использовано, чтобы установить продукт,



как налогооблагаемый или необлагаемый, выражение для оценки его истинности будет:

(If Condition):

Taxable='1' OR Taxable='T' OR Taxable='t' OR Taxable='Y' OR  
Taxable='y'

### **Большие MEMO поля**

◆ Clarion for Windows поддерживает поля MEMO размером до 64К. Если у вас существует файл, включающий мемо, намного большее чем 64К, вы можете использовать файл, но не модифицировать его.

◆ Вы можете определить, когда ваше приложение встретит большое MEMO путем определения, что переменная указателя мемо не пуста, но поле мемо при этом выглядит пустым. Выдается код ошибки 47 (неправильное объявление записи). Если вы попытаетесь обновить такую запись, любая модификация MEMO поля будет игнорироваться.

### **Тип данных возвращаемый функцией POSITION (POSITION Return Data Type)**

- ◆ POSITION(file) возвращает строку символов STRING(12)
- ◆ POSITION(key) возвращает строке символов STRING размер полей ключа + 4 байта.

### **Длинные имена полей**

◆ FoxPro и FoxBase поддерживает максимум 10 символов в имени поля. Если вам нужно больше, используйте External Name (внешнее имя) с 10 или менее символами.

### **Определение ключа**

◆ FoxPro и FoxBase поддерживают использование выражений для определения ключей. В редакторе словаря вы можете поместить выражение во внешнее имя поля в диалоговом окне Key Properties(свойства ключа). Общий формат внешнего имени:

‘FileName=T[Expression]’

Где FileName представляет имя индексного файла (который может содержать путь и расширение файла), а T представляет тип индекса. Действительными типами являются: C = символ, D = данные, N = цифра. Если типом является D или N, тогда Expression (выражение) может назвать только одно поле.

◆ Многоиндексные файлы (.CDX) требуют атрибута NAME() для KEY и INDEX, чтобы установить тип хранения ключа и любое выражение, которое используется для генерации величин ключа. Общий формат атрибута NAME() для KEY или INDEX:

Name('TagName|FileName[PageSize]=T[Expression],COMPRESSED')

Далше приводятся параметры для атрибута NAME():

|            |                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TagName    | Устанавливает имя индексного признака в многоиндексном файле. Если TagName опущен, драйвер создает .IDX файл с помощью имени, установленного в FileName.                                                                                                                                                                                                                                                           |
| FileName   | Устанавливает имя индексного файла, который может содержать путь и расширение.                                                                                                                                                                                                                                                                                                                                     |
| PageSize   | Может быть установлен только при создании .CDX файла (если установлен TagName). Это число в интервале 2-32 определяет количество 512-байтовых блоков в каждой индексной странице. Эта величина используется только при создании файла. Если вы через объявления установите кратные величины для различных признаков в одном и том же .MDX файле, будет выбрана будет наибольшая величина. Величина по умолчанию 2. |
| T          | Устанавливает тип индекса. Действительные типы - это C = символ, D = данные, N = цифра. Если типом являются D или N, выражение Expression может называть только одно поле.                                                                                                                                                                                                                                         |
| Expression | Определяет выражение для генерации индекса. Оно может относиться к множественным полям и использовать xBase функции. Текущие поддерживаемые функции приведены ниже. Выражение должно быть заключено в квадратные скобки.                                                                                                                                                                                           |
| COMPRESSED | Когда установлена эта позиция, драйвер FoxPro создает FoxPro 2 совместимый сжатый файл .IDX.                                                                                                                                                                                                                                                                                                                       |

Элементы атрибута NAME() могут быть опущены справа. При определении выражения вы должны также установить тип и имя. Если выражение опущено, драйвер определяет выражение из ключевых полей, когда файл создан, или из индексного файла, когда открыт.

Если опущен тип, драйвер определяет тип индекса из первого компонента ключа, когда файл создан, или из индексного файла, когда открыт.

Если атрибут NAME() опущен целиком, имя индексного файла определяется из метки ключа. По умолчанию путь установлен на то

же местоположение, что и .DBF.

Имена признаков ограничены по длине 9 символами. Если представленное имя слишком велико, оно автоматически усекается.

Установите все имена поля в атрибуте NAME() без префикса.

◆ FoxPro автоматически поддерживает использование оператора Xbase FOR в выражении для определения ключей. Выражения, поддержанные в условии FOR, должны быть простым условием вида:

expression comparison\_or expression

comparison\_or может быть одно из следующих: <,<=,<=<,<>,<=>,>= или >.

Выражение может относиться к множественным полям в записи и содержать функции xBase. Выражение должно быть заключено в квадратные скобки. Поддерживаемые функции даются ниже. Если драйвер встречает неподдерживаемую xBase функцию в заранее существовавшем файле, он посылает код ошибки 76 “Недействительная строка символов индекса”, когда файл открыт для ключей и статических индексов.

Выражения в виде строк символов могут использовать оператор “+” для конкатенации многих строк символов аргументов. Цифровые выражения используют операторы ‘+’ или ‘-’ в их основном смысле. Максимальная длина Clipper выражения - 250 символов.

### Поддерживаемые xBase функции определения ключа

|                      |                                                                                                                                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ALLTRIM(string)      | Удаляет начальные и конечные пробелы.                                                                                                                                                                                                      |
| CTOD(string)         | Преобразует ключ в виде строки символов в дату. Формат строки string должен быть mm/dd/yy; результат имеет форму ‘ууууmmdd’. Элемент уууу по умолчанию означает двадцатое столетие. Недействительные данные дают в результате пустой ключ. |
| DELETED ()           | Возвращает TRUE (истинно) если запись удалена.                                                                                                                                                                                             |
| DTOD(date)           | Преобразует ключ данных в строку символов формата ‘mm/dd/yy’.                                                                                                                                                                              |
| DTOS(date)           | Преобразует ключ данных в строку символов формата ‘ууууmmdd’.                                                                                                                                                                              |
| FIXED(float)         | Преобразует ключ из представления с плавающей точкой в представление с фиксированной точкой.                                                                                                                                               |
| FLOAT(numeric)       | Преобразует ключ из представления с фиксированной точкой в представление с плавающей точкой.                                                                                                                                               |
| IF(bool, val1, val2) | Возвращает величину val1 если первый параметр - TRUE, в противном случае возвращается величина val2).                                                                                                                                      |

|                                          |                                                                                                                                                                                                                          |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LEFT(string, n)                          | Возвращает самые левые n символов строки символов ключа в виде строки символов длиной n.                                                                                                                                 |
| LOWER (string, n)                        | Преобразует строку символов ключа в нижний регистр.                                                                                                                                                                      |
| LTRIM (string)                           | Удаляет пробелы слева в строке символов.                                                                                                                                                                                 |
| RECNO( )                                 | Возвращает номер текущей записи.                                                                                                                                                                                         |
| RIGHT(string, n)                         | Возвращает самые правые n символов строки символов ключа в виде строки символов длиной n.                                                                                                                                |
| RTRIM(string)                            | Удаляет пробелы с правой стороны строки символов.                                                                                                                                                                        |
| STR(numeric [,length[, decimal places]]) | Преобразует цифровые данные в строку символов. Длина строки символов и число десятичных знаков после запятой - параметры необязательные. По умолчанию длина строки символов 10, а число знаков после десятичной точки 0. |
| SUBSTR(string,offset,n)                  | Возвращает подстроку символов ключа в виде строки символов string , начиная со позиции offset и длиной в n символов.                                                                                                     |
| TRIM(string)                             | Удаляет пробелы с правой стороны от строки символов (идентично RTRIM).                                                                                                                                                   |
| UPPER(string)                            | Преобразует строку символов ключа в верхний регистр.                                                                                                                                                                     |
| VAL(string)                              | Преобразует строку символов ключа в цифру.                                                                                                                                                                               |

## **ODBC- Открытый интерфейс базы данных**

ODBC (Открытый интерфейс связи баз данных) является “стратегическим интерфейсом” Windows, предназначенным для доступа к данным из разнообразных систем управления базами данных, среди разнообразных сетей и платформ. ODBC предлагает те же конечные результаты, что и библиотеки драйверов файлов, которые поставляются с Clarion for Windows - независимость драйвера файла - хотя в несколько иной форме.

Стандарт ODBC был разработан и поддерживается Microsoft, который публикует ODBC Software Development Kit (SDK), и приспособлен для использования с Visual C++ продуктом. ODBC - это еще один способ, которым Clarion for Windows обеспечивает для вас расширяемую платформу для создания ваших приложений.

## ODBC, за и против

---

Использование ODBC дает следующие преимущества:

- ◆ ODBC является прекрасным выбором в среде клиент - сервер, особенно если сервер сродни Структурированному языку запросов (SQL) DBMS. Это дает вам возможность добавить поддержку архитектуры клиент - сервер к вашему приложению без необходимости делать что-либо помимо выбора драйвера файла. ODBC был специально спроектирован для создания непродолаваемого специфического метода связи приложений с сервером. Через посредство ODBC сервер может выполнить большую часть работы, особенно для операций SQL JOIN и PROJECT, ускоряя тем самым работу вашей программы.

- ◆ Существующие драйверы ODBC учитывают многие типы баз данных. Однако также существуют драйверы ODBC для баз данных, для которых Clarion может не иметь своего драйвера - например, для файлов Microsoft Excel и Lotus Notes.

- ◆ ODBC уже получил широкое распространение. Большие прикладные пакеты, такие, как Microsoft Office, например, устанавливают драйверы ODBC для таких форматов файлов, как dBase и Microsoft Access. Помните, что многие внутренние ODBC драйверы усовершенствованы и вам следует получить самые последние выпуски.

- ◆ ODBC не зависит от платформы. Одной из первейших целей Microsoft при установлении ODBC было поддержать более легкий доступ к системам наследства или корпоративным средам, где данные находятся на различных платформах или многочисленных DBMS. Когда имеются драйвер ODBC, не имеет значения, используете ли вы продукты Microsoft NetBEUL, SPX/IPX, DECNet или иные; ваше приложение может связаться с DBMS и получить доступ к данным.

Если имеется много драйверов, а стандарт был разработан компанией, которая разработала Windows, вы можете рассмотреть использование ODBC, как выбранного драйвера для всех ваших приложений Windows. Тем не менее, при выборе использования драйвера ODBC вместо собственного драйвера базы данных Clarion for Windows, вы должны также учесть возможные невыгодные моменты:

- ◆ ODBC добавляет свой собственный дополнительный слой - диспетчер драйвера ODBC - между вашим приложением и базой данных. Когда вы получаете доступ к файлам на локальном жестком диске, это обычно приводит к более медленной работе. Диспетчер драйвера должен транслировать обращение ODBC API приложения к оператору SQL прежде чем осуществить доступ к данным.

ODBC использует SQL для связи с внутренней базой данных. Хотя это может быть очень эффективно при связи с процессорами базы данных Клиент/сервер, это обычно менее эффективно, чем прямой доступ к записи при использовании файловой системы, сконструированной для доступа к одиночной записи, такого как xBase или Btrieve.

- ◆ Информация, которая требуется диспетчеру базы данных ODBC для связывания с источником данных, изменяется при переходе от одного драйвера ODBC к другому. В отличие от выбора драйверов файла Clarion, где операции фактически прозрачны, вам может потребоваться выполнить некоторую работу, чтобы собрать информацию, требуемую для использования конкретного драйвера ODBC. В этой главе приводится ряд советов, которые позволят сделать это проще. Многие драйверы ODBC приходят с файлом .HLP,

который документирует специальные установки (обычно хранящиеся в ODBC.INI); но на вас возлагается определенная ответственность при решении ваших проблем с помощью драйверов ODBC, поставляемых третьими сторонами.

- ◆ Драйверы ODBC не включены в Windows. При распространении вашего приложения вам нужно установить в систему конечного пользователя драйверы ODBC и диспетчер драйвера ODBC, если, конечно, он их еще не имеет. Для этого нужно получить от Microsoft ODBC SDK. В некоторых случаях на сервере уже может быть установлен комплект распределения, который устанавливает драйвер ODBC на рабочую станцию.

- ◆ Нормальная программа запуска Microsoft, которая устанавливает диспетчер драйвера ODBC, добавляет applet к окну панели управления конечного пользователя для управления ODBC. Конечному пользователю очень удобно использовать этот инструмент для изменения установок в файле ODBC.INI. Конечный пользователь может непреднамеренно удалить конечный драйвер ODBC, что сделает невозможным для вашего приложения связаться с файлом данных. Кроме того, так как большинство драйверов ODBC хранят директорию данных в ODBC.INI, конечному пользователю очень легко изменить его, что опять создаст возможные проблемы для вашего приложения.

Учитывая все за и против, мы рекомендуем использование собственных драйверов файлов Clarion for Windows в тех случаях, когда для одного и того же формата файла существуют как собственный драйвер, так и драйвер ODBC.

## Как работает ODBC

Когда вы используете ODBC, чтобы получить доступ к данным, для успешной работы требуется взаимодействие четырех компонент:

- ◆ Ваше приложение вызывает диспетчер драйвера ODBC и посылает ему соответствующие запросы данных через ODBC.API.

Clarion for Windows делает это для вас открыто, используя либо CW2ODBC16.DLL (16-битовая) либо CW2ODBC32.DLL расширения приложения. При ручном кодировании не забудьте включить эту библиотеку в проект. При распределении вашего приложения не забудьте включить этот файл вместе с вашим файлом .EXE (если вы не используете полностью укомплектованный .EXE).

- ◆ Диспетчер драйвера ODBC получает вызов API, проверяет ODBC.INI относительно информации об источнике данных, затем загружает “внутренний драйвер” драйвер ODBC.

Реальным “интерфейсом” для диспетчера драйвера является файл, называемый ODBCADM.EXE, который программа установки Microsoft помещает в каталог \Windows\System. Это администратор ODBC, который затем загружает другие библиотеки для работы.

- ◆ “Выходной” драйвер ODBC - это другая библиотека (.DLL), которая содержит исполняемый код, осуществляющий доступ к данным.

“Внутренние” драйверы поставляются различными источниками. Например, корпорация Lotus Development предоставляет драйвер ODBC для Lotus Notes. Microsoft Office распространяет ODBC SDK, содержащий драйверы для большинства их продуктов

базы данных.

♦ Источник данных - это файл данных (обычно ODBC используется для доступ к локальным данным), или удаленная DBMS, такая как база данных Oracle 7.

Источник данных имеет описательное имя; например, "Microsoft Access Databases." Это имя служит именем раздела в файле ODBC.INI.

Диспетчер драйвера ODBC должен знать точное имя источника данных, чтобы он мог загрузить правильный драйвер для доступа к данным. Следовательно, жизненно важно, чтобы вы знали точное имя драйвера.

## **Добавление поддержки ODBC к вашему приложению - Основы**

Для добавления поддержки ODBC к вашему приложению, требуется только выбор Clarion драйвера ODBC и обеспечение параметров для передачи диспетчеру драйвера ODBC. Параметры вы предоставляете в атрибуты NAME и OWNER объявления файла FILE.

**Совет: При создании словаря данных для таблиц ODBC импортрование определений файла обеспечивает эту информацию в соответствующих полях.**

Нижеследующее является базой при к таблицам ODBC из редактора словаря данных. Конечно, вы должны также быть уверены, что типы полей в вашем словаре совпадают с форматами, поддерживаемыми DBMS, к которой вы присоединяетесь.

1. Создайте новый файл словаря.

2. Выберите File > Import File .

Появляется диалоговое окно Выбор драйвера файла.

3. Выберите ODBC из выпадающего списка, затем нажмите кнопку ОК.

Появляется диалоговое окно Источники данных. Оно подобно интерфейсу Администратора ODBC. Если источник данных еще не был определен, вы можете добавить его путем нажатия кнопки New(новый).

4. Выделите источник данных, затем нажмите кнопку Next (следующий).

5. Если у источника данных есть защита в виде пароля, появляется диалоговое окно Logon. Представьте идентификатор пользователя UserID и пароль. затем нажмите кнопку ОК.

Если файл содержит много таблиц, появится диалоговое окно Tables for...(таблицы для).

6. Выделите таблицу, затем нажмите кнопку Finish (окончание).

Определение файла импортируется и появляется диалоговое окно File Properties (свойства файла), позволяющее вам модифицировать атрибуты, если вы это выберете.

Обратите внимание на поля в диалоговом окне Свойства файла, которые он заполнил во время импорта:

|                            |                                                                                                                                                                                                                                                           |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name: (имя)                | Имя извлекается из имени таблицы, определенного в базе данных ODBC. Вы можете его модифицировать. Оно используется в качестве метки Clarion в вашей исходной программе.                                                                                   |
| Prefix (префикс)           | По умолчанию ограничивается первыми тремя символами имени таблицы, вы можете модифицировать его тоже.                                                                                                                                                     |
| Owner name (имя владельца) | Имя источника данных ODBC и, по желанию, идентификатор пользователя и пароль, разделенные запятыми. Некоторые базы данных требуют дополнительной информации о связи. Эта информация следует за паролем и отделена точками с запятой, используя синтаксис: |

keyword=value;keyword=value.

Например, получая доступ к базе данных Sybase, это появится в виде:

Data Source, UserID,PassWord,DATABASE=DatabaseName; APP=APPName

Имя источника данных является именем раздела в файле ODBC.INI, который хранит всю информацию, необходимую менеджеру ODBC для того, чтобы загрузить драйвер и получить доступ к данным. Генератор приложения добавит информацию к атрибуту OWNER объявления файла:

OWNER(DataSourceName, UserID, Password)

Full Pathname (полное имя пути):

Процесс импорта помещает только имя таблицы в это поле. Драйвер ODBC находит физическое имя файла из ODBC.INI. Это помещает имя файла или таблицы в атрибут NAME объявления файла:

NAME(DataFileName) или NAME( TableName)

Остальная часть атрибута зависит от ваших предпочтений и вашего приложения.

7. Повторите шаги 2 - 6 для каждой таблицы в базе данных.



## Использование SQL-вставок

Вы можете использовать патентованный синтаксис Clarion для вставки операторов SQL в код вашей программы с помощью PROP:SQL, называющего файл целью. Это подходит только при использовании SQL драйвера файла, такого как драйвер ODBC.

Вы можете вставить любые операторы SQL, поддерживаемые сервером SQL. Если вы выдадите оператор SQL, в результате выполнения которого возвращается некоторое множество (такой как оператор SQL SELECT), вы можете использовать NEXT(file) для получения результата по одной строке за раз в буфере записи файла. Функции FILEERRORCODE() или FILEERROR() вернут любой ошибочный код или ошибочный набор строки символов внутренним сервером SQL.

Вы можете также запросить содержание PROP:SQL чтобы получить последний оператор SQL, выданный драйвером файла.

Например:

```
SQLFile{PROP:SQL} = 'SELECT field1,field2 FROM table1'
                  & 'WHERE field1 > (SELECT max(field1)
                  & 'FROM table2'
                  ! Возвращает набор результата, который
                  ! вы получаете по одной строке за раз
                  ! используя NEXT(SQLFile)
```

```
SQLFile{PROP:SQL} = 'CALL GetRowsBetween(2,8)!' Вызывает хранящуюся
процедуруSQLFile{PROP:SQL} = 'CREATE INDEX ON table1 (field1 DESC)'
```

! Нет результирующего множества

SQLString = SQLFile{PROP:SQL} ! Получите последний оператор SQL, который выдал драйвер

### **PROP:SQLFilter (Свойство :SQL фильтр)**

Вы можете использовать PROP:SQLFilter для фильтрации ваших VIEWS с помощью собственного SQL кода, а не кода Clarion.

Когда вы используете PROP:SQLFilter, фильтр SQL передается прямо к серверу. Как таковой он не может содержать имен переменных или функций, о которых сервер не знает.

Например:

```
View{PROP:SQLFilter} = 'Date = TO_DATE('01-MAY-1996' , 'DD-MON-YYYY ' )'
или
```

```
View{PROP:SQLFilter} = 'StrField LIKE 'AD%' '
```

Типы данных

| Тип данных Clarion |                  |        |         |      |       |      |       |
|--------------------|------------------|--------|---------|------|-------|------|-------|
| Тип данных ODBC    |                  | STRING | CSTRING | BYTE | SHORT | LONG | ULONG |
| MIN                | CHAR             | C      | *       | *    | *     | *    | *     |
|                    | VARCHAR          | *      | C       | *    | *     | *    | *     |
|                    | LONG             |        |         |      |       |      |       |
|                    | VARCHAR          |        |         |      |       |      |       |
| CORE               | DECIMAL          |        | *       | 1    | 1     | 1    |       |
|                    | NUMERIC          |        | *       | 1    | 1     | 1    |       |
|                    | SMALLINT         |        | *       | 3    | C     | 3    | 3     |
|                    | INTEGER          |        | *       | 3    | 3     | C    | 3     |
|                    | REAL             |        | *       |      |       |      |       |
|                    | FLOAT            |        | *       |      |       |      |       |
|                    | DOUBLE PRECISION |        | *       |      |       |      |       |
| EXTENDED           | BIT              |        | *       | *    | 3     | 3    | 3     |
|                    | TINYINT          |        | *       | C    | 3     | 3    | 3     |
|                    | BIGINT           |        |         |      | 3     | 3    | 3     |
|                    | BINARY           | *      |         |      |       |      |       |
|                    | VARBINARY        | *      |         |      |       |      |       |
|                    | LONG             | *      |         |      |       |      |       |
|                    | VARBUNARY        |        |         |      |       |      |       |
|                    | DATE             |        |         |      |       |      |       |
|                    | TIME             |        |         |      |       |      |       |
|                    | TIMESTAMP        | 2      |         |      |       |      |       |

Замечания

|   |                                                                                                                                             |
|---|---------------------------------------------------------------------------------------------------------------------------------------------|
| C | Тип данных Clarion может быть использован для управления типом данных ODBC.CREATE создает тип данных ODBC.                                  |
| * | Тип данных Clarion может быть использован для управления типом данных ODBC, однако CREATE НЕ создает тип данных ODBC.                       |
| 1 | LONG, SHORT и BYTE из Clarion'а могут быть использованы с типами данных ODBC DECIMAL и NUMERIC, если поле ODBC не имеет десятичных мест.    |
| 2 | Полями ODBC TIMESTAMP можно управлять с помощью STRING(8), с последующей GROUP поверх этого, которая содержит только поле DATE в поле TIME. |

SREAL REAL DECIMAL PDECIMAL DATE TIME

\* \* \* \*

4 4 \*

C 3 4 4  
3 \* 4 4  
3 C 4 4

3 3 \* \*

C  
C

Пример:

```
TimeStampField STRING(8),NAME( 'TimeStampField')
TimeStampGroup GROUP ,OVER(TimeStampField)
TimeStampDate DATE
TimeStampTime TIME
END
```

CREATE создает поле TIMESTAM, если вы используете подобную структуру.

- 3 Может возникнуть некоторая потеря точности.
- 4 Могут произойти ошибки округления.

Внимание: Ваша внутренняя база данных может содержать типы данных, которые не перечислены здесь. Эти типы данных преобразуются к данным типа ODBC внутренней базой данных. Просмотрите вашу документацию по внутренней базе данных для того, чтобы определить, какой тип данных ODBC используется.

## Посылка оператора SQL

С помощью драйвера ODBC вы можете использовать Clarion функцию SEND для отсылки команды SQL во внешнюю DBMS.

Используйте функцию SEND только для операций, которые не возвращают множеств, Нормальные файловые операции, которые требуют возврата множеств, нуждаются в операторах языка Clarion или во вставных SQL (смотрите предыдущий раздел).

Например, вы можете поддерживать функции обслуживания сети через команду SEND:  
ReturnValue=SEND(FileLabel, 'GRANT SELECT ON mytable TO fred' )

## Посылка строки драйвера /WHERE

Драйвер ODBC автоматически строит предложения WHERE, когда ваш код содержит SET, за которым следует NEXT или PREVIOUS. Драйвер также дает вам возможность добавить свои собственные ограничения с помощью переключателя /WHERE:

```
SendReturn = SEND (file,'/WHERE where-clause')
```

/WHERE SEND должно быть исполнено после оператора SET, но прежде первого оператора NEXT или PREVIOUS. Так как генерированный драйвером оператор SELECT не компилируется до оператора NEXT или PREVIOUS, никакой ошибочный код с помощью SEND не отправляется. Нет возврата результата. Например:

```
Ord   FILE,PRE(Ord),DRIVER('ODBC'),NAME('ord')
NameDate KEY(+Ord:NameID,-Ord:Date)
Record   RECORD
Name     STRING(12),NAME('NameID')
Date     DATE,NAME('OrderDate')
Type     STRING(1),NAME('OrderDetails')

. .
CODE
Ord:Name = 'SMITH'
SET(Ord:NameDate,Ord:NameDate)
SendReturn = SEND(Orders,'/WHERE OrderType = "M"')
LOOP
    NEXT(Ord)
    !....некоторая обработка
END
```

Это генерирует оператор SELECT подобный:

```
SELECT NameID,OrderDate,OrderType,OrderDetails FROM Ord
WHERE (NameID >= 'SMITH') AND (OrderType = 'M')
```

## **Проверка вашего ODBC приложения**

---

Диспетчер драйвера ODBC может создать регистрационный файл, документирующий все вызовы ODBC. Это включает действующие операторы SQL, выполненные драйвером по отношению к источнику данных и включает любые присланные сообщения об ошибках.

Кроме того, вы можете использовать функцию FILEERROR() для прерывания сообщений об ошибках “внутреннего” драйвера ODBC, который он передает обратно в ODBC драйвер Clarion for Windows. Последующие разделы расскажут вам, как с пользой воспользоваться этими советами.

### **Файл регистрации событий ODBC**

Вы можете приготовить различные регистрационные файлы. Один из них производится драйвером ODBC Clarion, другой - через Менеджера драйвера ODBC.

Регистрация Менеджера драйвера ODBC записывает каждый вызов ODBC и операторы SQL, которые они генерируют для диска после вызовов. Драйвер Clarion ODBC регистрирует только случающиеся ошибки. Это позволяет вам сравнивать вызовы к SQLError в регистраторе менеджера ODBC с реальными сообщениями об ошибках. Это значительно замедляет процесс, поэтому это следует активировать только во время испытания. Кроме того, регистрационный файл может вырасти до больших размеров очень быстро, поэтому вы должны выключить его и удалить файл после использования.

Кроме “сования носа” в действующие операторы SQL, генерируемые драйвером, вы можете установить на нуль любые ошибки. Если приложение не могло связаться с внешней базой, вы можете открыть регистрационный файл, используя Write или WordPad (файл обычно слишком велик для Notepad). Прокрутите его до тех пор, пока вы не обнаружите слово “Error” (ошибка). Заметим, что после текста ошибки часть появляются ненужные символы; игнорируйте их.

### **Регистрация событий ODBC**

Вы можете запустить регистрацию на системной основе, на файловой основе или используя по требованию команду SEND().

**Для системной регистрации** добавьте нижеследующее к вашему файлу WIN.INI:

[CWODBC]

Trace=1

TraceFile=[имя файла трассировки]

**Для файловой регистрации** в диалоговом окне Свойства файла, в редакторе словаря, добавьте в поле ввода Опции драйвера следующее :

/LOGFILE=filename.ext

где filename.ext - имя регистрационного файла, который вы хотите создать.

**Для регистрации по требованию** используйте команду SEND() в соответствующей точке в вашем коде со следующим синтаксисом:

```
SEND( file,'/LOGFILE=filename.ext')
```

где file - это метка файла данных, а filename.ext - имя регистрационного файла, который вы хотите создать.

**Чтобы выключить регистрацию** используйте команду SEND() в соответствующей точке в вашем коде со следующим синтаксисом:

```
SEND( file,'/LOGFILE')
```

где file - это метка файла данных.

### **Регистрирование администратора ODBC**

1. Запустите администратор ODBC.

Вы можете сделать это, выполняя программу ODBCADM.EXE в каталоге \Windows\System, либо дважды щелкая мышью на пиктограмме ODBC в панели управления.

2. Нажмите кнопку Options (параметры) в диалоговом окне Data Sources (источники данных).

3. Отметьте поле Trace ODBC Calls (проследить вызовы ODBC) .

4. Если необходимо, уберите отметку в поле Stop Tracing Automatically (остановить автоматическое прослеживание) , если вы думаете, что вам нужно испытать соединение более одного раза.

Обычным является проверка в течение нескольких раз перед фиксацией ошибки.

5. Нажмите кнопку Select File (выбрать файл) и назовите файл для регистрации.

По умолчанию называется SQL.LOG.

6. Переключитесь на вашу программу и начните испытание.

После того, как возникнут ошибки, откройте регистрационный файл и исследуйте его. Не забудьте выключить поле Trace ODBC Calls (ODBC :проследить вызовы ODBC) при проведении испытаний.

### **Поддерживаемые атрибуты файла, команды и функции**

#### **Атрибуты файла**

CREATE .  
DRIVER (filetype,[driver string]) Да  
NAME Да

#### **Поддерживаемые**

Да  
Да  
Да

|                   |      |
|-------------------|------|
| ENCRYPT           | Нет  |
| OWNER (password)  | Да   |
| RECLAIM           | Нет  |
| PRE (prefix)      | Да   |
| BINDABLE          | Да   |
| THREAD            | Да   |
| EXTERNAL (member) | Да   |
| DLL ([flag])      | Да   |
| OEM               | Нет2 |

**Файловые структуры****Поддерживаемые**

|        |     |
|--------|-----|
| INDEX  | Да2 |
| KEY    | Да2 |
| MEMO   | Нет |
| BLOB   | Нет |
| RECORD | Да  |

**Атрибуты индекса, ключа, мемоПоддерживаемые**

|                       |     |
|-----------------------|-----|
| BINARY                | Нет |
| DUP                   | Да  |
| NOCASE                | Да  |
| OPT                   | Нет |
| PRIMARY               | Да  |
| NAME                  | Да  |
| Ascending Components  | Да  |
| Descending Components | Да  |
| Mixed Components      | Да  |

**Файловые команды****Поддерживаемые**

|                                 |     |
|---------------------------------|-----|
| BUILD(file)                     | Да  |
| BUILD (key)                     | Да  |
| BUILD (index)                   | Да  |
| BUILD (index,components)        | Да  |
| BUILD (index,components,filter) | Нет |
| CLOSE (file)                    | Да  |
| COPY (file,new file)            | Нет |
| CREATE (file)                   | Да  |
| EMPTY (file)                    | Да  |
| FLUSH (file)                    | Нет |
| LOCK (file)                     | Нет |
| OPEN (file,access mode)         | Да  |
| PACK (file)                     | Нет |

|                          |     |
|--------------------------|-----|
| REMOVE (file)            | Да  |
| RENAME (file,new file)   | Нет |
| SHARE (file,access mode) | Да  |
| STATUS (file)            | Да  |
| STREAM (file)            | Нет |
| UNLOCK (file)            | Нет |

**Доступ к записям**

|                               |
|-------------------------------|
| ADD (file)                    |
| ADD (file,length)             |
| APPEND (file)                 |
| APPEND (file,length)          |
| DELETE (file)                 |
| GET (file,key)                |
| GET (file,filepointer)        |
| GET (file,filepointer,length) |
| GET (file,keypointer)         |
| HOLD (file)                   |
| NEXT (file)                   |
| NOMEMO (file)                 |
| PREVIOUS (file)               |
| PUT (file)                    |
| PUT (file,filepointer)        |
| PUT (file,filepointer,length) |
| RELEASE (file)                |
| REGET ( file,string)          |
| REGET ( key,string)           |
| RESET ( file,string)          |
| RESET ( key,string)           |
| SET (file)                    |
| SET (file,key)                |
| SET (file,filepointer)        |
| SET (key)                     |
| SET (key,key)                 |
| SET (key,keypointer)          |
| SET (key,key,filepointer)     |
| SKIP (file,count)             |
| WATCH (file)                  |

**Файловые функции**

|                  |     |
|------------------|-----|
| BOF (file)       | Нет |
| BYTES (file)     | Да  |
| DUPLICATE (file) | Да  |

**Поддерживаемые**

|     |
|-----|
| Да  |
| Нет |
| Да  |
| Нет |
| Да  |
| Да  |
| Нет |
| Нет |
| Нет |
| Да  |
| Нет |
| Да3 |
| Да  |
| Нет |
| Нет |
| Нет |
| Да  |
| Нет |
| Да3 |
| Нет |
| Нет |
| Да  |
| Нет |
| Да  |
| Нет |
| Нет |
| Да  |
| Нет |

**Поддерживаемые**

|     |
|-----|
| Нет |
| Да  |
| Да  |



|                     |     |
|---------------------|-----|
| DUPLICATE (key)     | Да  |
| EOF (file)          | Нет |
| NAME (label)        | Да  |
| POINTER (file)      | Нет |
| POINTER (key)       | Нет |
| POSITION (file)     | Нет |
| POSITION (key)      | Да  |
| RECORDS (file)      | Да  |
| RECORDS (key)       | Да  |
| SEND (file,message) | Да  |

**Диалоговая обработка****Поддерживаемые**

|                               |    |
|-------------------------------|----|
| LOGOUT (timeout,file,...file) | Да |
| COMMIT                        | Да |
| ROLLBACK                      | Да |

**Обработка пустых данных****Поддерживаемые**

|                    |    |
|--------------------|----|
| NULL (field)       | Да |
| SETNULL (field)    | Да |
| SETNONNULL (field) | Да |

**Замечания**

1. Драйвер файлов ODBC Clarion for Windows поддерживает перечисленные позиции, однако лежащая в основании файловая система может поддержать не все эти позиции.
2. Предполагается, что международная сортировка делается внутренней файловой системой. Как таковые атрибут OEM и файл .ENV игнорируются.
3. PREVIOUS не поддерживается в файловом порядке.

**Разное****Connection/Login (Соединение/начало сеанса)**

Вы можете использовать синтаксис свойств Clarion для сохранения информации о соединении источников данных для переменной с помощью {PROP:ConnectionString} используя метку файла как цель.

Пример:

```
FileOwner STRING(256)
Afile FILE,DRIVER('ODBC'),OWNER(AFileOwner)
  AFileOwner='DataSource'
OPEN(AFile)
IF NOT ERRORCODE() THEN
```

AFileOwner=AFile{PROP:ConnectionString}

Вы можете также использовать синтаксис свойств Clarion, чтобы установить временной предел (TimeOut) для экрана регистрации базы данных ODBC. Если пользователь не отвечает в отведенное время, соединение заканчивается и регистрация прекращается. По умолчанию - ждать неопределенно долго ввода пользователя. Не все back ends поддерживают эту характеристику и могут игнорировать вашу величину.

Пример:

AFile{PROP:LoginTimeOut}=60 ! разрешает 1 минуту на регистрацию

### **Поля только для чтения (ReadOnly)**

Добавление переключателя READONLY к внешнему имени NAME поля командует драйверу не вставлять это поле, когда запись добавляется. Это необходимо для некоторых внутренних баз (таких как Watcom), которые не позволяют устанавливать на ноль автонкрементальные поля.

### **PROP:OrderAllTables**

Этот переключатель может потребоваться вам, если вы используете структуру Clarion VIEW, который соединяет многие таблицы. По умолчанию (View{PROP:OrderAllTables}=FALSE) драйвер ODBC включает в ORDER BY предложение, посылаемое серверу ODBC, только компоненты ключа первичного файла.

Путем установки View{PROP:OrderAllTables}=TRUE вы можете заставить драйвер использовать в предложении ORDER BY связанные поля и компоненты ключа вторичного файла, а также компоненты ключа первичного файла.

Пример:

```
BRW1 :: View:Browse      View(Customer)
                        PROJECT(CUST:CustNo)
                        PROJECT(CUST:Name)
                        PROJECT(CUST:Zip)
                        PROJECT(CUST:CustNo)
                        JOIN(ORD:ByCustomer,CUST:CustNo)
                        PROJECT(ORD:OrderNo)
                        PROJECT(ORD:OrderDate)
                        END
                        END
```

CODE

```
?BRW1 :: View:Browse{PROP:OrderAllTables} = TRUE
```

Доступ к этому VIEW генерирует оператор SELECT подобный:

```
SELECT CustNo,Name,Zip,OrderNo,OrderDate FROM Customer,Ord
WHERE (Customer,CustNo = Ord.CustNo)
```

ORDER BY CustNo,OrderNo

## **MS доступ и ODBC**

---

Имеется много различных драйверов ODBC, поставляемых Microsoft и предназначенных для доступа к файлам MDB. Каждый из них требует внимания при его использовании.

Драйвер ODBC, поставляемый вместе с Access 2.0

Этот драйвер работает только с MS Office applets. Чтобы получить драйвер общего назначения, работающий с Clarion for Windows, вам нужно приобрести у Microsoft драйвер ODBC Kit v2.

### **Драйвер ODBC, поставляемый вместе с Access 7.0**

Этот драйвер не работает с 16-битовыми приложениями в Windows 95, вы также не можете импортировать файлы в ваш словарь данных с помощью этого драйвера в Windows 95. Решением будет работа с Windows NT. Чтобы получить доступ к 32-битовым ODBC драйверам, таким как драйвер Access 7.0, из 16-битового приложения, вам следует использовать ODBC.DLL, версию 2.10 или более позднюю.

## ***Файлы TopSpeed***

Файловая система базы данных TopSpeed является высокоэффективным, высоконадежным, запатентованным драйвером файла для Clarion инструментов разработки. Она не совместима по файлу с данными драйвера файлов Clarion.

Таблицы данных, ключи, индексы и все мемо могут быть сохранены вместе в одном DOS файле. Расширение файла по умолчанию \*.TPS. Отдельный “Файл контроля транзакции” принимает расширение \*.TCE.

Драйвер TopSpeed может, если это необходимо, сохранять многочисленные таблицы в одном файле DOS. Это дает вам возможность открыть столько файлов данных, ключей и индексов, сколько необходимо при использовании одного дескриптора файла DOS. Эта характеристика может быть особенно полезна, когда есть большое число малых таблиц или когда группа связанных файлов обычно используется совместно. Все ключи, индексы и мемо сохраняются внутри.

Кроме того, файловая система TopSpeed поддерживает тип данных BLOB (Binary Large Object - бинарный большой объект), поле, которое полностью переменной длины и может быть размером более 64 К (как в 16-битовых, так и в 32-битовых приложениях). BLOB должен быть объявлен перед структурой RECORD. Память для BLOB выделяется и

освобождается динамически по мере необходимости. Более полную информацию можно найти в разделе BLOB в Справочнике языка.

|        |              |                                                         |
|--------|--------------|---------------------------------------------------------|
| Файлы: | CL2TPS16.LIB | Экспортная библиотека Windows (16-битовая)              |
|        | CL2TPS32.LIB | Экспортная библиотека Windows (32-битовая)              |
|        | CW2TPS16.LIB | Статически загружаемая библиотека Windows (16-битовая)  |
|        | CW2TPS32.LIB | Статически загружаемая библиотека Windows (32-битовая)  |
|        | CW2TPS16.DLL | Динамически загружаемая библиотека Windows (16-битовая) |
|        | CW2TPS32.DLL | Динамически загружаемая библиотека Windows (32-битовая) |

**Совет:** Этот новый драйвер предлагает скорость, безопасность и требует меньших ресурсов от системы конечного пользователя.

Типы данных

|        |         |
|--------|---------|
| BYTE   | DECIMAL |
| SHORT  | STRING  |
| USHORT | CSTRING |
| LONG   | PSTRING |
| ULONG  | MEMO    |
| SREAL  | GROUP   |
| REAL   |         |

Характеристики файла/максимумы

|                          |                                       |
|--------------------------|---------------------------------------|
| Размер файла:            | Ограничен только местом на диске      |
| Записей на файл:         | 4,294,967,295                         |
| Размер записи:           | 15K                                   |
| Размер поля:             | 15K                                   |
| Полей на запись:         | 15K                                   |
| Ключей/индексов на файл: | 240                                   |
| Размер ключа:            | 15K                                   |
| Мемо полей на файл:      | 255                                   |
| Размер поля мемо:        | 64K                                   |
| Полей BLOB на файл:      | 255                                   |
| Размер BLOB:             | зависит от компьютера                 |
| Открывание файла:        | в зависимости от операционной системы |

Строки символов драйвера и функции SEND

Строкам символов драйвера, назначенным в объявлении FILE (второй параметр атрибута DRIVER (драйвер)) всегда предшествует косая черта (/). Они отсылаются драйверу файла когда приложение выдает команду OPEN (открыть). Нет возвращаемой величины, связанной с этими строками драйвера.

Функция SEND отсылает строки драйвера в любое время. Некоторые строки символов драйвера не действуют после открытия файла, поэтому синтаксис функции SEND для модификации установки не перечислен. Однако синтаксис функции SEND для возврата величины переключателя приводится в перечне для всех строк символов драйвера. Строки драйвера функции SEND принимают два формата - один со знаком равенства модифицирует установки переключателя и возвращает величину предыдущей установки переключателя; другой формат (без знака равенства) возвращает величину переключателя.

### **SEND(file, 'TCF=file name')**

#### **/TCF=file name**

Устанавливает файл контроля транзакции, иной, чем принятый по умолчанию \TOPSPEED.TCF. Файл сохраняет все многофайловые законченные транзакции до окончания программы или до исполнения SEND(TCF=file name).

### **SEND(file, 'PNM=name')**

#### **/PNM=name**

Отыскивает имена таблиц в многотабличном файле .TPS.

Чтобы отыскать первое имя, выдайте команду: SEND(file, 'PNM='). Это возвращает имя первой таблицы. Последующие вызовы передают полученное имя и возвращают следующее имя.

Например, дан файл с тремя таблицами - Supp, Part и Ship, пример ниже показывает алфавитное перечисление:

```
Code
name" = ' '
LOOP
name"=(SEND(Supp,'PNM=' &
name")
If name"
DISPLAY name"
ELSE
BREAK
END
END . . .
```

**SEND(file,'FULLBUILD[=on|off]')****/'FULLBUILD[=on|off]**

Драйвер TopSpeed имеет оптимизированный механизм добавления, с помощью которого вы можете добавлять большое число записей к существующей таблице с помощью оператора APPEND. Выполнение последующего BUILD обновляет только информацию добавленного ключа, что делает пакетное обновление очень быстрым. Это поведение по умолчанию. Используйте строку символов драйвера FULLBUILD для изменения этого назначаемого по умолчанию поведения.

FULLBUILD=ON говорит следующему оператору BUILD полностью перестроить ключи. FULLBUILD=OFF восстанавливает BUILD до его оптимизированного состояния. Обе версии команды SEND возвращают текущее состояние build в виде строки символов 'ON' или 'OFF'. Выдайте SEND(file,' FULLBUILD) для возврата текущего состояния build без изменения его.

**Поддерживаемые атрибуты файла, команды и функции****Атрибуты файла****Поддерживаемые**

|                                   |      |
|-----------------------------------|------|
| CREATE .                          | Да   |
| DRIVER (filetype,[driver string]) | Да   |
| NAME                              | Да   |
| ENCRYPT                           | Да   |
| OWNER (password)                  | Да   |
| RECLAIM                           | Нет1 |
| PRE (prefix)                      | Да   |
| BINDABLE                          | Да   |
| THREAD                            | Да   |
| EXTERNAL (member)                 | Да   |
| DLL ([flag])                      | Да   |
| OEM                               | Да   |

**Файловые структуры****Поддерживаемые**

|        |    |
|--------|----|
| INDEX  | Да |
| KEY    | Да |
| MEMO   | Да |
| BLOB   | Да |
| RECORD | Да |

**Атрибуты индекса, ключа, мемо** **Поддерживаемые**

|        |    |
|--------|----|
| BINARY | Да |
| DUP    | Да |

|                       |     |
|-----------------------|-----|
| NOCASE                | Да  |
| OPT                   | Да  |
| PRIMARY               | Да  |
| NAME                  | Да2 |
| Ascending Components  | Да  |
| Descending Components | Да  |
| Mixed Components      | Да  |

**Файловые команды****Поддерживаемые**

|                                 |    |
|---------------------------------|----|
| BUILD(file)                     | Да |
| BUILD (key)                     | Да |
| BUILD (index)                   | Да |
| BUILD (index,components)        | Да |
| BUILD (index,components,filter) | Да |
| CLOSE (file)                    | Да |
| COPY (file,new file)            | Да |
| CREATE (file)                   | Да |
| EMPTY (file)                    | Да |
| FLUSH (file)                    | Да |
| LOCK (file)                     | Да |
| OPEN (file,access mode)         | Да |
| PACK (file)                     | Да |
| REMOVE (file)                   | Да |
| RENAME (file,new file)          | Да |
| SHARE (file,access mode)        | Да |
| STATUS (file)                   | Да |
| STREAM (file)                   | Да |
| UNLOCK (file)                   | Да |

**Доступ к записям****Поддерживаемые**

|                               |     |
|-------------------------------|-----|
| ADD (file)                    | Да  |
| ADD (file,length)             | Нет |
| APPEND (file)                 | Да  |
| APPEND (file,length)          | Нет |
| DELETE (file)                 | Да  |
| GET (file,key)                | Да  |
| GET (file,filepointer)        | Да3 |
| GET (file,filepointer,length) | Нет |
| GET (file,keypointer)         | Да  |
| HOLD (file)                   | Да  |
| NEXT (file)                   | Да  |
| NOMEMO (file)                 | Да  |
| PREVIOUS (file)               | Да  |

|                               |     |
|-------------------------------|-----|
| PUT (file)                    | Да  |
| PUT (file,filepointer)        | Да  |
| PUT (file,filepointer,length) | Нет |
| RELEASE (file)                | Да  |
| REGET ( file,string)          | Да  |
| REGET ( key,string)           | Да  |
| RESET ( file,string)          | Да  |
| RESET ( key,string)           | Да  |
| SET (file)                    | Да  |
| SET (file,key)                | Да  |
| SET (file,filepointer)        | Да  |
| SET (key)                     | Да  |
| SET (key,key)                 | Да  |
| SET (key,keypointer)          | Да  |
| SET (key,key,filepointer)     | Да  |
| SKIP (file,count)             | Да  |
| WATCH (file)                  | Да  |

**Файловые функции****Поддерживаемые**

|                     |    |
|---------------------|----|
| BOF (file)          | Да |
| BYTES (file)        | Да |
| DUPLICATE (file)    | Да |
| DUPLICATE (key)     | Да |
| EOF (file)          | Да |
| NAME (label)        | Да |
| POINTER (file)      | Да |
| POINTER (key)       | Да |
| POSITION (file)     | Да |
| POSITION (key)      | Да |
| RECORDS (file)      | Да |
| RECORDS (key)       | Да |
| SEND (file,message) | Да |

**Диалоговая обработка****Поддерживаемые**

|                               |    |
|-------------------------------|----|
| LOGOUT (timeout,file,...file) | Да |
| COMMIT                        | Да |
| ROLLBACK                      | Да |

**Обработка пустых данных****Поддерживаемые**

|                    |    |
|--------------------|----|
| NULL (field)       | Да |
| SETNULL (field)    | Да |
| SETNONNULL (field) | Да |



## Замечания

1 Драйвер TopSpeed автоматически использует пространство, освобожденное удаленными записями и ключами.

2 Драйвер TopSpeed не поддерживает внешние имена для ключей, так как все ключи хранятся внутри.

3 GET(file, filepointer) требует указателя, возвращаемого функцией POINTER. Следовательно, вы не можете использовать

GET(file,l)

для поиска первой записи в файле, так как l - это не указатель.

## Разное

### SHARE и режимы открытого доступа:

| Поддерживаются следующие режимы открытого доступа                           | Требование Share |
|-----------------------------------------------------------------------------|------------------|
| 34 (12h) Читать/записывать, запрет записи<br>(умолчание для OPEN(открыто)   | Да               |
| 66 (42h) Читать/записывать, запретов никаких<br>(умолчание для SHARE(общий) | Да               |
| 64 (40h) Только читать, запретов никаких                                    | Да               |
| 18 (12h) Читать/записывать, запрет на все                                   | Нет              |
| 16 (10h) Только читать, запрет на все                                       | Нет              |
| 32 (20h) Только читать, запрет записывать                                   | Нет              |

Для указанных режимов SHARE.EXE должно быть загружено в AUTOEXEC.BAT или CONFIG.SYS. Следующий пример загружает SHARE в AUTOEXEC.BAT, обеспечивая максимум 500 файлов заблокированными и по умолчанию 2048 байт для зоны хранения.

C:\DOS\SHARE.EXE /L:500

Если SHARE требуется, но не загружено, программа генерирует рабочую ошибку, когда вызваны OPEN или SHARE (нет запрета на режимы) или, когда сделана попытка обновления (запрет на режимы записывания).

#### ◆ LOCK(file)

LOCK() только влияет на другие LOCK() вызовы. Единственным результатом успешного обращения к LOCK() является то, что другие процессы получают ошибку FLALLK, когда они вызовут LOCK().

#### ◆ LOGOUT(), COMMIT(), ROLLBACK()

Файл контроля транзакции используется, для обеспечения автономности завершения, транзакции, которая обновляет более одного файла DOS. По умолчанию

файл контроля транзакций (TCF) имеет имя “\TOPSPEED.TCF.” Команда SEND() позволяет вам изменить это.

Файл TCF должен быть доступен, когда любые файлы, контролируемые им, доступны. Если транзакция включает обновление более, чем одного разделяемого сетевого файла, вы должны установить файл контроля транзакций в сети. Не является необходимым использовать тот же TCF файл для всех транзакций; однако, он должен находиться там, где он может быть прочитан каждым, осуществляющим доступ к файлу. Если это не так, после разрушения системы/отсутствия питания во время COMMIT() некоторые файлы могут быть обновлены, а другие нет. (Файлы не будут разрушены - они могут быть просто несовместимы один с другим).

Файл .TCF может быть удален, если все файлы, контролируемые им, могут быть открыты (для записывания) после разрушения системы/отсутствия питания.

### **Большие ключи (или малая оперативная память RAM)**

#### ◆ APPEND( )

Если общий размер ключей превышает количество имеющейся памяти с произвольной выборкой, если имеется более одного ключа или при добавлении большого числа записей, APPEND( ) рекомендуется более, чем ADD( ). Размер ключа (для этой цели) - это количество операций ввода (сумма полей ключа + 10 байт). Если добавляемые записи уже находятся в приблизительном порядке ключей, тогда вы можете учесть этот ключ для целей вышеупомянутого расчета.

В качестве примера, если файл имеет два 40-байтных ключа и 2 Мегабайта памяти, ADD( ) становится (относительно) медленным, когда размер базы данных превышает примерно  $2,000,000 / (40 + 10 + 40 + 10) = 20,000$  записей.

### **Инкрементное Key/Index Build (построение ключей/индексов)**

#### ◆ BUILD(file), BUILD(key)

Драйвер TopSpeed выполняет инкрементное построение; это означает, что при построении ключа читаются только записи, начиная с первой записи, добавленной после того, как ключ был построен. Драйвер сливает новые ключи с существующим ключом. Таким образом, построение большого ключа, где только несколько недавно добавленных записей было модифицировано, должно быть быстрым. Смотрите выше строку драйвера BUILD.

Построение индекса аналогично, но должно начинаться при минимальной физической записи, чья позиция в индексе изменилась с тех пор, как был построен последний индекс.

Динамические индексы не поддерживаются и поэтому не могут быть построены

инкрементно.

### **Выполнение пакетной обработки**

#### ◆ **STREAM(), FLUSH()**

При чтении большого числа записей используйте STREAM() или откройте файл в режиме запрета записи, например OPEN(file), а не SHARE(file). После того, как записи были прочитаны, вызовите FLUSH(), чтобы разрешить доступ другим пользователям.

Очень важно использовать STREAM() при добавлении/присоединении/помещении большого числа записей. STREAM() обычно делает обработку в 20 раз быстрее. Например, добавление 1000 записей может потребовать около 2 минут без STREAM() и только 5 секунд при использовании STREAM(). Нет необходимости использовать STREAM() или FLUSH() на выведенном из системы файле (работа на выведенных файлах всегда успешна).

**Совет:** При использовании STREAM() для обновления огромного числа записей драйвер сохраняет незавершенные (uncommitted) или незанесенные в системный каталог (unflushed) страницы в памяти и есть возможность недостатка памяти. Вызывание COMMIT(), FLUSH() или LOGOUT() периодически предупреждает это. Чтобы вычислить максимально число “обновлений” между каждым COMMIT(), разделите имеющуюся память на размер обновления. При присоединении размер обновления равен приблизительно размеру записи в байтах. При добавлении размер обновления приблизительно равен размеру записей и полей компонент ключа в байтах. При обновлении записей с помощью PUT() теоретически возможно, что размер обновления достигнет 7К. На практике мы рекомендуем завершать транзакцию (committing) через каждые 100 или примерно столько обновлений.

### **Тип данных возвращаемый функцией POSITION (POSITION Return Data Type)**

- ◆ POSITION(File) возвращает STRING(4).
- ◆ POSITION(Key) возвращает STRING, содержащую размер полей ключа и + 4 байта.

### **Указатели (POINTERS) и удаленные записи**

#### ◆ **POINTER(key)**

Величина, возвращенная POINTER(key), соответствует физическому порядку записей данных. Поэтому, когда запись DELETED (удалена), указатель становится недействительным. Любой последующий доступ с помощью указателя терпит неудачу. Если вам нужно точное совпадение, по которому возвращается ближайшая запись, используйте функцию POSITION().

## Сохранение многих таблиц (файлов данных) в одном файле .TPS

Используя специальную управляющую последовательность '\!' в атрибуте NAME() декларации файла TopSpeed, вы можете установить, что один .TPS файл будет хранить более одной таблицы. Например, чтобы объявить один TPS файл 's&p.tps', который должен содержать 3 логических таблицы, вызываем `supp`, `part`, `ship`:

```
Supp FILE, DRIVER('TopSpeed'), PRE(Supp), CREATE, NAME('S&P\!Supp')
```

```
...
```

```
Part FILE, DRIVER('TopSpeed'), PRE(Part), CREATE, NAME('S&P\!Part')
```

```
...
```

```
Ship FILE, DRIVE('TopSpeed'), PRE(Ship), CREATE, NAME('S&P\!Ship')
```

```
...
```

Файлы данных имеют общий дескриптор одного DOS файла, открытый, когда открыт первый файл, и закрытый, когда закрыт последний файл. Первый открытый режим-определяет открытый режим для всех других файлов. Если первый открытый режим это режим только для чтения, тогда не может быть выполнено успешно никакое обновление любого типа (ACCDNID будет возвращено).

Если один файл в группе выведен(`logged out`), тогда и все файлы в группе эффективно выведены(`logged out`).

Если для одного из файлов в группе откорректирован каталог (`Flushed`), тогда и для всех остальных файлов в группе корректируются каталоги.

Эта возможность особенно полезна тогда, когда имеется большое число малых таблиц, или когда приложение должно иметь одновременный доступ к группе связанных файлов.

Команда SEND() позволяет программисту определить имена файлов в группе. Чтобы найти имя, выдайте:

SEND(file,'PNM='). Это возвращает имя первой таблицы. Чтобы найти второе имя, выдайте: SEND(file,'PNM=FirstTableName'). Это возвращает имя второй таблицы, и т.д.

Файлы могут быть переименованы внутри группы; например, для вышеприведенных объявлений следующая команда переименует файл, названный `Supp` в `Old_Supp`:

```
RENAME(Supp,'S&P\!Old_Supp')
```

Переименование в другую существующую группу нормально включает копирование/удаление, поэтому менее эффективно.

Если вы используете атрибут владельца OWNER на многочисленных таблицах в файле базы данных TopSpeed, все таблицы должны иметь тот же самый атрибут OWNER.

Если не установлена никакая управляющая последовательность, тогда дается имя таблицы по умолчанию 'unnamed' (без названия), так что все следующее эквивалентно:

```
foo FILE,DRIVER('TopSpeed')
foo FILE,DRIVER('TopSpeed'),NAME('foo')
foo FILE,DRIVER('TopSpeed'),NAME('foo\!unnamed')
```

Файлы могут быть переименованы в группе;

## **Утилита восстановления базы данных TopSpeed**

Система файлов TopSpeed предназначена для автоматического исправления большинства ошибок. Если файл данных физически поврежден во время неправильной работы системы, Утилита восстановления базы данных TopSpeed может восстановить поврежденные части ваших данных.

**Внимание: Утилита восстановления базы данных TopSpeed является аварийным инструментом ремонта и не может использоваться в обычных условиях. Пользуйтесь ею только тогда, когда файл был поврежден.**

Утилита восстановления базы данных TopSpeed читает поврежденное поле и пишет восстановленные записи в новый файл. Она использует информацию, хранящуюся в заголовке файла или сканирует файл в поисках неповрежденных частей. По желанию вы можете обеспечить образцовый файл, содержащий таблицу (индивидуальный файл) и план ключа.

Утилита восстановления базы данных TopSpeed является свободно распределяемой утилитой, предназначенной для того, чтобы дать конечному пользователю возможность восстановить поврежденные файлы.

Утилита восстановления предназначена для работы интерактивно или явно через параметры командной строки. Интерактивно вы обеспечиваете параметры через два диалоговых окна мастера. С помощью параметров командной строки вы можете включить ее в ваше приложение с помощью параметра RUN(). Или же вы можете создать переключку (в Windows 95) или пиктограмму менеджера базы данных ( в Windows 3.1x) с этими параметрами

**Совет: Лицензионное соглашение Clarion for Windows применимо к TSPFIX.EXE; распределение ограничено. Вы можете распределять утилиту своим пользователям, но они этого делать не имеют права.**

## **Использование утилиты восстановления в интерактивном режиме**

1. Запустите утилиту, щелкнув для этого дважды клавишей мыши на Пиктограмме утилиты восстановления базы данных TopSpeed в Программной группе Clarion for Windows 1.5.

Появляется диалоговое окно TopSpeed Database Recovery Utility (утилита восстановления базы данных TopSpeed) .

2. В разделе Source (файл, который нужно восстановить) установите имя файла или нажмите кнопку Browse (просмотр) для выбора ее из стандартного диалогового окна открывания файла.

3. Если у файла есть пароль, наберите его в поле ввода Password (пароль).

Если файл базы данных содержит много таблиц (файлы данных), каждая таблица должна иметь тот же самый пароль.

4. По желанию в разделе Destination (назначение - файл результата) установите имя файла для целевого файла или нажмите кнопку Browse (просмотр) для выбора его в стандартном диалоговом окне открывания файла.

По умолчанию расширение .TPR добавляется к имени исходного файла. Это параметр, выбираемый по желанию. Если он опущен, первоначальный (исходный) файл переписывается и создается резервный файл. Исходный файл переименовывается в filename.TPX (имя файла), где x автоматически меняется от 1 до 9 каждый раз как создается новый файл. Если использованы все девять номеров, любой последующий созданный файл получает расширение .TP\$ и переписывается.

5. Если файл результата должен иметь другой пароль, наберите его в поле ввода Password (пароль). Если он опущен, пароль удаляется.

6. Нажмите кнопку Next (следующий).

Появится второе мастер-диалоговое окно для Утилиты восстановления базы данных TopSpeed.

7. По желанию установите имя файла Example File (образцовый файл) или нажмите кнопку Browse (просмотр) для выбора его из стандартного диалогового окна открывания файла.

Утилита использует образцовый файл для определения плана таблицы и определения ключей в том случае, если эти области исходного файла повреждены. расширение по умолчанию - .TPR, но если вы будете выбирать, вы можете использовать расширение DOS.

**Совет: Мы рекомендуем отгрузить образцовый файл когда вы будете разворачивать ваше приложение. Это улучшает восстановление данных из поврежденного файла.**

8. Если у образцового файла есть пароль, наберите его в поле ввода Password (пароль).

9. Если вы хотите, чтобы утилита перестроила Ключи, отметьте поле флажков Build Keys (построить ключи).

Если вы это не делаете, ключи перестраиваются первоначальным приложением, когда оно пытается открыть.

10. Если вы хотите использовать информацию Header (заголовка) в исходном файле, отметить поле Use Header (использовать заголовок).

Использование информации заголовка оптимизирует работу утилиты. но оно не должно применяться, если заголовок файла испорчен. Если это опущено, утилита просматривает весь файл данных и восстанавливает все неповрежденные страницы.

11. Если приложение использует Локальный (Locate .ENV) файл для альтернативной сортирующей последовательности, установите файл .ENV или нажмите кнопку Browse (просмотр) для выбора его из стандартного диалогового окна открывания файла.

12. Если файл использует атрибут OEM для управления сортирующей последовательностью, отметьте поле Use OEM (используйте OEM).  
Это сделает возможным преобразование OEMTOANSI и ANSITOOEM.

13. Нажмите кнопку Start (пуск) для того, чтобы начать процесс восстановления.  
Если утилита не находит никаких ошибок, появляется сообщение, информирующее вас о том, что “No Errors Detected in <filename.ext>” (в ... никаких ошибок не найдено) и спрашивающее, хотите ли вы продолжать восстановление.

**Параметры командной строки**

Утилита может также принять параметры командной строки, что дает вам возможность исполнить ее из приложения или пиктограммы Менеджера программы (или Shortcut в Windows 95).

```
TPSFIX sourcepath[?password] [destpath[?password]]  
[/E:examplepath[?password]] [/L:Localepath] [/H] [/K] [/P] [/O]
```

|                |                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TPSFIX         | Исполняемый (TPSFIX.EXE)                                                                                                                                   |
| sourcepath     | Имя файла и путь исходного (поврежденного) файла базы данных.                                                                                              |
| ?password      | Пароль файла.                                                                                                                                              |
| destpath       | Имя файла и путь восстановленного файла базы данных.                                                                                                       |
| /E:examplepath | Имя файла и путь образцового файла базы данных.                                                                                                            |
| /L:localepath  | Имя и путь Locate (.ENV) файла, используемого для установки альтернативной сортировочной последовательности.                                               |
| /H             | Если установлено, утилита использует информацию заголовка в исходном файле.                                                                                |
| /K             | Если установлено, утилита перестраивает все ключи для базы данных.                                                                                         |
| /P             | Если установлено, пользователь получает приглашение для каждого параметра даже если они имеются на командной строке.                                       |
| /O             | Если установлено, файл использует OEMTOANS и ANSITOOEM для определения сортировочной последовательности. Смотрите Интернационализация в Справочнике языка. |

## Использование утилиты в вашем приложении

---

Есть несколько моментов, которые нужно учесть, прежде чем использовать утилиту.

- ◆ Файл базы данных НЕ должен быть открытым при работе утилиты. Обеспечьте, чтобы файл был закрыт прежде, чем разрешить пользователю запускать утилиту.
- ◆ Чтобы предотвратить доступ во время процесса восстановления до его завершения, утилита запирает файл автоматически.
- ◆ Будет более эффективно и безопасно, если ваше приложение перестроит ключи KEYS (опуская параметр /K в восстановлении). Это также хороший способ проверить статус восстановления.

### Работа с утилитой восстановления базы данных TopSpeed

---

Имеются два основных метода, которые вы можете использовать из оператора RUN(): Используя первый метод, вы опускаете параметр destpath, так что первоначальный (исходный) файл переписан. Это требует образцового файла.

В генераторе приложений:

1. В диалоговом окне Actions (действия) для кнопки или позиции меню выберите из выпадающего списка Run a Program (прогнать программу).
2. В поле ввода Program Name (имя программы) установите TPSFIX.EXE.
3. В поле ввода параметров установите параметры (смотрите выше Параметры командной строки).

Например:

TPSFIX.EXE Datafile.TPS /E:Example.TPE /H

Во вставном исходном тексте:

RUN('TPSFIX.EXE Datafile.TPS /E:Example.TPE /H')

Это восстанавливает файл "datafile.TPS" с помощью образцового файла "Example.TPE" как образца для планов таблицы и ключа, не перестраивает ключи и использует информацию заголовка в исходном файле. Первоначальный файл сохранен в резервный файл с расширением от TP1 до TP9. Каждый раз, когда выполняется утилита, цифровая часть расширения изменяется.

Второй метод требует двух строк вставной исходной программы, но дает вам контроль над процессом переименования. Вы вставляете исходную программу в точку вставки из позиции меню или кнопкой.

Например:

COPY(datafilelabel, 'Datafile.OLD')

!копирует



первоначальный файл

! в Datafile.OLD

RUN(TPSFIX Datafile.OLD Datafile.tps /H) ! прогоняет утилиту, используя переименованный файл как исходный, а первоначальное имя как цель

Это копирует файл datafilelabel в DATAFILE.OLD, восстанавливает файл и пишет его в DATAFAIL.TPS , используя для этого информацию заголовка в исходном файле.

## **Утилита копирования базы данных TopSpeed**

Утилита копирования базы данных TopSpeed дает вам возможность копировать в файлы (и из файлов) базы данных TopSpeed, содержащие много таблиц (файлы данных) в одном файле DOS.

**Совет: Лицензионное соглашение Clarion for Windows применимо к TPSFIX.EXE; распределение утилиты ограничено. Вы можете распределять своим пользователям, но они этого делать не имеют права.**

### **Пример использования**

---

Используя данную утилиту, вы можете:

- ◆ Извлечь файлы из мультитабличной базы данных TopSpeed. Это дает вам возможность извлечь файл, использовать программу преобразования, а затем вставить файл обратно.
- ◆ Вставить файлы в мультитабличную базу данных TopSpeed. Это дает вам возможность создать каждый файл по-отдельности, а затем вставить их все в один .TPS файл.
- ◆ Заменить файлы из мультитабличной базы данных TopSpeed.
- ◆ Удалить файлы из мультитабличной базы данных TopSpeed.
- ◆ Добавить новую таблицу к мультитабличной базе данных TopSpeed.
- ◆ Добавить новый ключ к существующему файлу в мультитабличной базе данных TopSpeed.
- ◆ Добавить новое поле или модифицировать существующее поле в таблице внутри мультитабличной базы данных TopSpeed.
- ◆ Заменить или модифицировать поле или ключ в мультитабличной базе данных TopSpeed.
- ◆ Комбинировать однотабличные файлы базы данных TopSpeed в один мультитабличный файл базы данных TopSpeed.
- ◆ Копировать файл данных в файл мультитабличной базы данных TopSpeed в другую базу данных.

## Рассмотрение совместного использования файлов

---

**Внимание:** Перед тем, как применять эту или любую другую утилиту на “живых” файлах данных, вы должны создать резервные файлы.

Файл базы данных НЕ следует открывать во время использования утилиты. Прежде, чем запускать утилиту, убедитесь, что файл закрыт. Если исключительный доступ не разрешается, утилита показывает предупреждение “Отказ доступа” и деактивирует кнопку Copy. Чтобы предотвратить доступ до момента, когда процесс копирования будет завершен, утилита автоматически блокирует файл.

## Интерфейс утилиты копирования

---

### Source (copy from) Источник (откуда копировать)

Database Name (имя базы данных)

Имя файла DOS для файла-источника базы данных TopSpeed, из которого копируются таблицы. Наберите имя файла, или же отберите существующий файл путем нажатия кнопки Browse (просмотр).

Password (пароль)

Пароль (атрибут OWNER (владелец)) для файла-источника. Если файл имеет пароль, он должен быть представлен. Если база данных источника не имеет пароля, это поле ввода деактивируется.

File to Copy (файл, который нужно копировать)

Таблица в базе данных TopSpeed, из которой нужно копировать. Отберите из ниспадающего списка. Если база данных содержит только одну таблицу, она обычно называется UNNAMED (без имени).

Remove File from Source Database (удалить файл из базы данных)

Отметьте это поле если вы хотите удалить таблицу из базы данных источника после ее копирования. Если это поле не отмечено, таблица остается в базе данных источника. При использовании этой опции, когда последняя таблица перемещена из базы данных источника, вы имеете опцию удаления физического файла с диска.

Rebuild Keys (построить ключи заново)

Выполняет полное построение ключей базы данных источника.

### Destination (copy to) (назначение (куда копировать))

Database Name (имя базы данных)

Имя файла DOS для файла-мишени базы данных TopSpeed, в который копируются таблицы. Наберите имя файла, или же отберите существующий файл путем нажатия кнопки Browse (просмотр).

Password (пароль)

Пароль (атрибут OWNER (владелец)) для файла-мишени. Если файл места назначения имеет пароль, он должен быть представлен. Если база данных источника не имеет пароля, а пароль представлен, он добавляется. Если создается новая база данных, введите здесь новый

|                       |                                                                                                                                                                                                                                                    |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       | пароль.                                                                                                                                                                                                                                            |
| New File (новый файл) | Имя таблицы в базе данных TopSpeed, в которую нужно копировать файл- источник. По умолчанию это дает имя таблицы из базы данных источника. Если вы хотите создать базу данных с одной таблицей, новый файл должен быть назван UNNAMED (без имени). |
| Copy (копировать)     | Нажмите эту кнопку для того, чтобы скопировать таблицу источника в место ее назначения.                                                                                                                                                            |
| Exit (выход)          | Нажмите эту кнопку, чтобы закрыть утилиту.                                                                                                                                                                                                         |

### Параметры командной строки утилиты копирования

**TPSCOPY** *sourcefile*[?*password*] [*tablename* *destinationfile*][?*password*]  
*!tablename* [/R] [/D]

|                                         |                                                                                                                                                                        |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TPSCOPY                                 | Запускает исполняемую программу(TPSCOPY.EXE)<br>sourcefile (файл- источник)<br>Имя файла DOS для файла-источника базы данных TopSpeed, из которого копируются таблицы. |
| ?password (пароль)                      | Пароль файла (атрибут OWNER (владелец)). Если у файла имеется пароль, он должен быть представлен. Если у базы данных нет пароля, это можно опустить.                   |
| !tablename (имя таблицы)                | Имя конкретной таблицы в базе данных TopSpeed. Если база данных содержит только одну таблицу, она называется UNNAMED (без имени).                                      |
| destinationfile (файл места назначения) | Имя файла DOS для файла-мишени базы данных TopSpeed, в который копируется таблица. Наберите имя файла (включая расширение) для того, чтобы создать новый файл.         |
| /R                                      | Переместить данную таблицу из базы данных источника после того, как она скопирована.                                                                                   |
| /D                                      | Удалить базу данных источника, не имеющую оставшихся таблиц (удалить физический файл с диска).                                                                         |
| /K                                      | Выполняет полную постройку ключей базы данных источника перед копированием.                                                                                            |

Примеры:

TPSCOPY orders.tps?acme!CUSTOMER customer.tps?acme96!UNNAMED

Копирует таблицу заказчика из OREDRS.TPS (используя пароль acme) в одно-табличный файл CUSTOMER.TPS, который имеет пароль acme96.

TPSCOPY orders.tps?fred!CUSTOMER customer.tps?barney!UNNAMED

Копирует таблицу заказчика из OREDRS.TPS (используя пароль fred) в одно-табличный файл CUSTOMER.TPS, который имеет пароль barney.

TPSCOPY orders.tps!UNNAMED orders.tps?acme96!UNNAMED

Копирует таблицу UNNAMED (единственную таблицу в файле) из OREDRS.TPS (который не имеет пароля) в таблицу CUSTOMER.TPS, который имеет пароль acme96.

***Приложение В Стратегии разработки и развертывания - EXE, .LIB и .DLL***



## Обзор

Разработка с участием многих программистов

Приведение в действие и организация командных проектов

Процедурно-ориентированный подход

Модульно-ориентированный подход

Суб-приложение

.EXE, состоящий из одного куска

Когда осуществлять

Как осуществлять

.EXE плюс .DLLs

Когда осуществлять

Как осуществлять

Размещение библиотеки поддержки Clarion for Windows

## Обзор

То, как вы организуете файлы своего приложения, может оказать значительное влияние на эффективность ваших рабочих процессов в течение всего жизненного цикла вашего приложения. Например, разработка всех ваших процедур в пределах одного файла .APP позволяет держать все под одной крышей. Это может быть выгодно для приложений небольшого размера - содержание всех файлов в одном каталоге делает работу по поиску и резервированию файлов достаточно простой в случае, если файлов достаточно мало. Эта выгода может превратиться в проблему для более крупных приложений - изготовление резервных копий или компиляция сразу сотен файлов может оказаться делом сложным и требующим больших затрат времени. Отсюда вы можете видеть, как организация влияет на фазу разработки вашего приложения.

Предельный размер и количество исполняемых файлов вашего приложения влияет на вашу способность быстро и легко распределить вашим конечным пользователем обновления и исправления ошибок. Это один из путей влияния организации приложения на фазу обслуживания вашего приложения.

Данное приложение обсуждает некоторые факторы, которые вам следует учесть, решая, как структурировать файлы своего приложения, причем это относится как к фазе разработки, так и к фазе обслуживания. Сюда относятся и инструкции относительно того, как применить организацию, максимально подходящую вашим задачам.

В целом, выгоды от деления больших программных проектов на более мелкие логически связанные части сводятся к следующему:

### Фаза разработки

- ◆ Чем меньше, тем легче решаются проблемы.
- ◆ Возможность проверять и отлаживать более мелкие куски кода. Чем меньше код, тем легче изолировать проблемы.
- ◆ Меньше время, требуемое на компиляцию и компоновку.
- ◆ Возможность делать код родовым и заново его использовать.
- ◆ Возможность делегировать программные задачи многим программистам.

### Фаза обслуживания

- ◆ Возможность продавать и распределять отдельные компоненты приложения.
- ◆ Возможность развертывать исправления ошибок и обновления с малыми файлами.
- ◆ Для повторно используемого кода - возможность влиять на многие процедуры путем изменения одного файла исходной программы.

Эти преимущества наиболее просто реализуются путем создания многих .APP файлов, которые используются для получения отдельных .DLL и .EXE, разрабатываемых и

испытываемых независимо. По мере продвижения проекта к своему завершению исполняемые файлы компоуются вместе и испытываются как одно целое.

К числу проблем, связанных с делением больших программных проектов на меньшие части, относятся:

### **Фаза разработки**

- ◆ Управление большим числом файлов: резервирование, условности присвоения названий и синхронизация файлов превращаются в большое дело.
- ◆ Точность компоновки вместе связанных частей исполняемой программы.

### **Фаза обслуживания**

- ◆ Управление большим числом файлов: контроль версии затруднен. Конечный пользователь должен иметь полный набор совместимых файлов.
- ◆ Для повторно используемого кода - ненамеренное влияние на многие процедуры при изменении одного файла исходной программы.
- ◆ Случайное упущение требуемого файла во время развертывания.

## **Разработка с участием многих программистов**

---

Модульный подход Clarion for Windows к работе с исходной программой, его процедурно-ориентированный язык и его способность производить файлы .DLL и .LIB дают возможность вашей команде программистов разделить работу над большим программным проектом.

Наши рекомендуемые подходы к групповой разработке предполагают, что команда связана сетью, которая поддерживает возможность распределения привилегий “только-читать” или “читать-записывать” между отдельными разработчиками. Не имеет значения, является ли ваша сеть системой одноранговой или более традиционной сетевой операционной системой. Мы также предполагаем, что проект должен иметь руководителя команды или менеджера проекта, чтобы координировать общие усилия команды. Наконец, мы предполагаем, что в соответствии с нашим лицензионным соглашением, каждый программист Clarion for Windows имеет лицензионную копию Clarion for Windows.

Первый шаг, который надо предпринять для проекта с командой разработчиков - это создать словарь данных, доступный каждому разработчику, но который может редактировать только лидер команды. Опция генератора приложений (поле флажков Multi user Development (разработка многих пользователей ниже Setup > Application Options) обеспечивают поддержку для открытия словаря данных и REGISTRY.TRF в режиме “только для чтения”, поэтому многие разработчики с отдельными файлами .APP могут работать с одним и тем же самым словарем данных. У всех членов команды должен быть один и тот же общий REGISTRY.TRF и один набор файлов шаблонов исходной программы. Руководитель команды должен отвечать за словарь и набор шаблонов.



Когда уже существует словарь данных, имеются три базовых подхода, которые ваша команда может принять, чтобы использовать Clarion for Windows как инструмент групповой разработки:

◆ Процедурно-ориентированный:

Команда делит приложение на процедуры, как это перечислено в дереве приложений. Они должны быть организованы вокруг различных окон, диалоговых полей, позиций меню и командных кнопок, формирующих интерфейс пользователя.

Руководитель команды готовит “ядро .APP” (или головной модуль), над которым надстраиваются все остальные. Каждый член команды получает копию файла .APP, затем работает над процедурой (или процедурами). Руководитель программы импортирует завершённые процедуры в мастер-файл .APP для компиляции. Этот подход годится для проектов с размерами от малого до среднего.

◆ Модульно-ориентированный:

Команда делит приложение на его компоненты цель-файл-уровень (.DLL, .LIB и .EXE). Каждый член команды создает одиночный целевой файл. Отдельные файлы проекта (.PRJ) компилируют отдельные компоненты. Файл мастер-проекта включает все другие проектные файлы, строя все целевые файлы сразу. Такой подход подходит для размеров проектов от среднего до большого.

◆ Суб-Приложение:

Команда разделяет приложение на его компоненты на уровне целевых файлов (.DLL). Каждый член команды создает одиночное приложение или динамически загружаемую библиотеку (.DLL). Главное приложение вызывает каждую .DLL. Этот подход годится для проектов от среднего до большого размера. Данный метод обеспечивает наибольшую гибкость и минимизирует хлопоты по управлению версиями.

## **Приведение в действие и организация командных проектов**

---

В данном разделе рассказывается, как установить Clarion Development Environment на каждом рабочем месте и о том, где хранить файлы, необходимые для обоих подходов групповой разработки:

1. Создайте словарь данных в общем каталоге.

Все члены команды, работающие над проектом, должны иметь право чтения этого каталога. Те, кому разрешено редактировать словарь, должны также иметь привилегию записи, хотя может быть лучше будет, если только руководителю команды будет разрешено редактировать его.

2. Создайте общий каталог для ресурсных файлов.

Обеспечьте права чтения всем членам команды в отношении файлов пиктограмм, курсоров, bitmap и других ресурсных файлов.

3. Внутри ID, на каждом рабочем месте, выберите Setup > Application Options.

Появится диалоговое окно Application Options (параметры приложения).

4. Установите флажок в поле Multi User Development (много-пользовательская разработка).

Эта позиция устанавливает, что при работе в генераторе приложений копия Clarion for Windows, находящаяся на каждом рабочей станции, открывает файл словаря системы в режиме “только-для-чтения”. Целью этого является гарантировать, чтобы никто не удалил случайно поле, файл или ключ, необходимые для другого члена команды. По этой причине мы рекомендуем, чтобы только руководитель команды имел привилегии записывать в каталог, содержащий словарь. Чтобы изменить словарь, все члены команды должны закрыть все приложения, которые используют словарь. Руководитель команды должен очистить поле флажков Multi User Development (разработка многих пользователей) для того, чтобы изменить словарь и, по завершении, отметить этот поле снова.

5. Чтобы закрыть диалоговое окно, нажмите кнопку ОК.

6. Создайте каталог на локальном диске каждого рабочего места для хранения индивидуальных файлов .APP и файлов исходного текста каждого члена команды.

Реальной работой является планирование того, как разделить разработку проекта, что и является предметом обсуждения оставшейся части данной главы.

**Внимание:** Если DLL вашего приложения используют файлы вашего приложения, определения файлов и все глобальные переменные и структуры должны быть объявлены в .DLL (не в .EXE) и экспортированы. Смотрите дополнительную информацию в Суб-приложенческом подходе.

## **Процедурно-ориентированный подход**

---

Генератор приложений дает вам возможность импортировать и экспортировать процедуры из других файлов .APP. При внимательном руководстве руководитель команды может организовать разработку таким образом, что каждый член команды может компилировать и испытывать копию приложения, которое включает ту часть, над которой работает он или она. Каждый просматривает все меню и наиболее важные диалоговые поля приложения, однако исполняет только те процедуры, за которые ответственен данный член команды.

Чтобы выполнить это, каждому члену команды нужна копия “мастер-файла” .APP,

содержащая главную процедуру MAIN (которой скорее всего будет процедура “Рамка приложения”), а также другие процедуры, вставленные ниже ее как процедуры “ToDo”. Каждый член команды затем “включает” процедуры, за которые он ответственен.

Чтобы собрать полное приложение, с помощью команды File > Import from Application, руководитель команды импортирует каждую законченную процедуру в мастер-файл .APP.

Ниже дается вариант осуществления процедурно-ориентированного подхода:

1. Создайте словарь данных и установите рабочие места в соответствии со сказанным выше.

2. Создайте “мастер-файл” .APP в каталоге, в который только руководитель команды имеет право записи.

3. Внутри файла .APP отредактируйте наиболее важные элементы интерфейса пользователя в главной процедуре MAIN и объявите ее глобальные переменные.

Элементы интерфейса пользователя могут включать любые поля диалога или окна особой важности для приложения. Когда вы назначаете вызовы процедур к позициям меню и/или к элементам управления панели инструментов, генератор приложений автоматически добавляет командные действия к дереву приложения в качестве процедур “ToDo”.

4. Сохраните и скопируйте файл .APP на локальный накопитель каждого члена команды. Если команда предпочитает это, вы можете переименовать копии; например, MASTER01.APP, MASTER02.APP и т.д. или JIM.APP, JANE.APP и т.д..

5. Каждый член команды работает над процедурой, за которую он ответственен, используя при этом свою копию .APP файла.

Имея .APP файл, содержащий полный интерфейс пользователя, каждый член команды может компилировать промежуточное состояние разработки локально и испытывать свои собственные процедуры в процессе их разработки.

6. Каждый член команды синхронизирует свой локальный каталог с эквивалентным каталогом сети в конце каждого дня или копирует переименованные .APP файлы в “главный” каталог.

7. Чтобы обновить мастер-файл .APP в соответствии с последней работой разработчика, руководитель команды заменяет процедуру “ToDo” в дереве приложения завершенной процедурой в .APP члена команды путем ее импортирования. Руководитель команды выбирает File > Import from Application, указывая ту же процедуру в .APP файле в сетевом каталоге разработчика.

Любые под-процедуры, добавленные членами команды, будут принесены как процедуры “ToDo”. По мере того, как руководитель команды “выращивает” мастер-файл .APP, он или она может скопировать файл .APP обратно в индивидуальные директории членов команды (но только если вся работа, выполненная отдельным членом команды, была импортирована). При таком образе действий каждый член команды может получить доступ ко всей работе, выполненной к этому времени другими членами команды. Помните, что каждый модуль другого члена команды потребует компоновки на локальном драйвере этого члена команды.

Если руководитель команды является также членом команды - то есть он или она также отвечает за кодированные процедуры - лучше вести совершенно отдельный директорий и копию мастер-файла .APP для этой работы.

8. После импортирования обновленной процедуры руководитель команды выясняет, не прибавила ли она новые процедуры “ToDo” к дереву приложения, а если да, то импортирует их.

Коммуникация на этой стадии имеет первостепенную важность. Фактически, с помощью сообщений электронной почты внутри команды руководитель команды может по необходимости импортировать “работы в стадии развития”.

9. Руководитель команды компилирует проект, так чтобы он теперь включал работу каждого члена команды, добавленную через процедуры импортирования.

10. Руководитель команды повторяет последние три описанных шага периодически, пока не завершена работа каждого члена команды и может быть испытано все приложение.

## **Модульно-ориентированный подход**

---

При таком подходе каждый член команды создает отдельный целевой файл. Это требует разделения приложения на “главную” исполняемую и “вторичные” исполняемые программы или динамически загружаемые библиотеки. Отдельные члены команды ведут отдельные проектные файлы (.PRJ) для каждой компоненты. Руководитель команды создает файл мастер-проекта для построения всех целевых файлов сразу.

Ключом к успешному выполнению этой стратегии является интенсивное предпланирование “разделения труда” между отдельными целевыми файлами, созданными приложением. В разделе замечаний ниже приведены несколько полезных предложений.

Следующие пункты дают схему осуществления этой стратегии:

1. Создайте словарь данных и установите рабочие места в соответствии с вышесказанным.
2. Каждый член команды создает свои собственные файлы .APP и .PRJ, устанавливая

файл словаря в сети в качестве словаря данных, а каталог на локальном накопителе в качестве каталога по умолчанию для файла .APP. Каждый член команды устанавливает различный целевой файл.

Один конкретный файл .APP или .PRJ создает исполняемую программу, которая запускает или вызывает библиотечные функции или процедуры в других. Для конечного пользователя это .EXE программа, которую нужно запустить при работе со всем приложением.

3. Каждый член команды в конце каждого дня синхронизирует свой локальный каталог с эквивалентным сетевым.

4. Руководитель команды создает в сетевом подкаталоге мастер-файл .PRJ , который включает все другие .PRJ файлы.

Руководитель команды вставляет имя каждого файла .PRJ (ранее скопированного в сеть) в позиции Projects to Include (проекты для включения) в дереве проекта.

5. Руководитель команды компилирует мастер-проект, который в свою очередь компилирует один за другим все целевые файлы.

6. Руководитель программы повторяет периодически последний шаг до тех пор, пока не завершена вся работа всеми разработчиками и пока все приложение не может быть испытано.

### **Замечания относительно деления проекта**

Существуют, по-видимому, много способов разделения проекта; в данном разделе даются несколько предложений по этому вопросу.

◆ Если задача, связанная с командой меню, требует значительного кодирования, сохраните ее в ее собственной внешней .DLL так, чтобы только один разработчик мог работать над ней.

Типичным примером может быть программа бухгалтерского учета, которая могла бы хранить все процедуры и функции, связанные с получаемыми счетами в одном .DLL файле, с оплачиваемыми счетами в другом и так далее.

◆ Организуйте .DLLs функционально: например, поместите служебные процедуры и функции, такие как Backup процедуры и процедуры экспорта файлов, в UTILITIES.DLL.

◆ Сохраните определенные пользователем функции в файле .LIB; распределите скомпилированные файлы .LIB каждому члену команды по мере их поступления, так чтобы каждый мог проверить работу этих функций в их собственной среде.

### **Замечания относительно управления файлами**

Каждый проект с участием многих разработчиков имеет свои собственные свойства, так что вы несомненно примете следующие предположения для осуществления требований:

◆ Создайте каталог для каждого члена команды на сетевом диске, на том же уровне или ниже чем уровень, где находится словарь. Дайте каждому разработчику права только на его собственный каталог, используйте утилиты сети для синхронизации каталогов в конце дня.

Это не только служит как резервная копия, но и дает руководителю команды доступ к последним разработкам членов команды.

Если разрабатываемая задача создает .INI файл, его копия должна располагаться в сетевом каталоге, в который каждый член команды имеет право записи, так что если у кого-то есть необходимость добавить переменную в файл, другие члены команды могли ее видеть

### **Суб-приложение**

В данном разделе описываются шаги, которые необходимо выполнить для создания программы, использующей одно главное приложение и несколько суб-приложений, скомпилированных и скомпонованных как внешние .DLL. Разделение большого проекта на многие .DLL дает много преимуществ:

- Каждое суб-приложение может быть модифицировано и проверено независимо.
- Разработчики могут работать каждый над своей частью проекта, не мешая другим членам команды разработчиков.
- Каждое суб-приложение может быть скомпилировано как .DLL и испытано в главной программе без перекомпилирования всего проекта. Это уменьшает время, требуемое на компилирование и компоновку.
- Пределы динамического пула в больших проектах избегаются.
- Будущие обновления могут быть осуществлены отправкой новой .DLL, уменьшая тем самым расходы по отправке.

При таком подходе каждый член команды создает отдельную .DLL, которая вызывается приложением “хозяином”. Это требует расщепления приложения на “Главное” исполняемое и “вторичные”.DLL. Отдельные члены команды ведут отдельные файлы приложения для каждой компоненты. Руководитель команды создает приложение-хозяин, которое вызывает суб-приложения, а также приложение “данных”, которое содержит (и экспортирует) все определения файла и глобальные переменные. По желанию члены команды могут вызвать процедуры из .DLL других членов.

Этот метод требует также интенсивного пред-планирования “разделения труда” между различными целевыми файлами, созданными приложением. Предыдущий раздел дает

несколько полезных предложений.

Возможная реализация этой стратегии:

1. Создайте словарь данных и установите рабочие станции в соответствии с вышесказанным.

Создайте приложение “данные” для хранения и экспорта всех объявлений данных. Все глобальные переменные и структуры, а также все определения файлов определяются (и экспортируются) в этом приложении. Используйте следующие установки:

**В свойствах приложения:**

|                                         |                   |
|-----------------------------------------|-------------------|
| Dictionary File (файл словаря)          | Основной словарь. |
| Destination Type (тип файла назначения) | .DLL              |

**В Глобальных свойствах приложения:**

|                                                                               |             |
|-------------------------------------------------------------------------------|-------------|
| Generate Global Data as External<br>(генерация глобальных данных как внешних) | OFF (ВЫКЛ.) |
|-------------------------------------------------------------------------------|-------------|

File Control Flags (флажки управления файла)

|                                                                        |                                      |
|------------------------------------------------------------------------|--------------------------------------|
| Generate All File declarations:<br>(генерировать все объявления файла) | ON (ВКЛ)                             |
| External: (внешний)                                                    | NONE EXTERNAL (НИКАКОГО<br>ВНЕШНЕГО) |

|                                                                         |          |
|-------------------------------------------------------------------------|----------|
| Export All File declarations:<br>(экспортировать все объявления файла:) | ON (ВКЛ) |
|-------------------------------------------------------------------------|----------|

**Внимание:** Вам нет нужды ( и не появится ) в **OWNER**, **NAME** или **PASSWORD** для тех файлов, которые имеют атрибут **EXTERNAL** (внешний). Эти атрибуты должны иметься только для **ФАЙЛОВ**, который не объявлены внешними.

3. Член команды создает .APP файлы своего собственного суб-приложения, устанавливая файл словаря в сети как словарь данных, а каталог на локальном диске как каталог по умолчанию для .APP файла. Каждый член команды устанавливает разный целевой файл с помощью следующих установок:

**В свойствах приложения:**

|                                 |                                      |
|---------------------------------|--------------------------------------|
| Dictionary File (файл словаря): | Главный словарь, находящийся в сети. |
|---------------------------------|--------------------------------------|

Destination Type (тип файла назначения): .EXE во время фазы конструирования и

испытания.DLL при передаче к главному каталогу.

**Внимание:** Изменение типа цели позволяет процедурам быть экспортированными. Убедитесь, что каждая процедура, которая вызвана главным приложением или другой .DLL, имеет поле флажков экспортной процедуры в Свойствах процедуры отмеченным (поле флажков доступно только после изменения типа цели).

### В глобальных свойствах приложения:

Generate Global Data as External

(генерировать глобальные данные как внешние): ON

File Control Flags (флажки управления файла)

Generate All File declarations

(генерировать все объявления файла) OFF (ВЫКЛ)

External (внешний):

ALL EXTERNAL (ВСЕ ВНЕШНИЕ)

All Files declared in another .App

(все файлы объявлены в других приложениях): ON (ВКЛ)

Declaring Module (объявление модуля):

Оставьте это пустым

### В дереве модулей приложения:

Выберите Application > Insert Module и отберите соответствующую .LIB для .DLL, содержащую определения данных.

Одно конкретное .APP создает исполняемую программу, которая запускает или вызывает функции библиотеки или процедуры в других. Для конечного пользователя это запуск программы .EXE при работе с полным приложением.

4. Члены команды синхронизируют свои локальные каталоги с эквивалентом сети в конце каждого дня.

5. Члены команды выдают свои компилированные и скомпонованные .DLL руководителю команды.

Каждое суб-приложение имеет “пустую” рамку (не экспортированную), которая вызывает процедуры суб-приложения, так что член команды может испытать суб-приложение, компилируя его как .EXE. Когда оно проходит испытания, член команды компилирует его в .DLL путем изменения типа целевого файла свойств приложения на



**Совет:** Если вы редактируете файл переадресации для включения “.” в начале поисковых путей \*.DLL и \*.LIB, Clarion будет генерировать файлы \*.DLL и \*.LIB в локальном подкаталоге суб-приложения вместо \CW20\BIN и \CW20\OBJ. Это небольшая мера предосторожности, которая предупреждает попадание \*.DLL и \*.LIB в руки других членов команды до того, как они завершены. Кроме того, добавление главного каталога в конец этих путей поиска позволяет суб-приложению или главному приложению найти завершенные .LIB и .DLL, принадлежащие другим суб-приложениям в главном субкаталоге.

6. Руководитель команды копирует выданные .DLL в главный каталог и создает главный .APP файл, который вызывает процедуры точки ввода в .DLL.

Главный .APP обычно простой скелет приложения с красочным экраном и главной рамкой с меню и линейкой инструментов. .DLL вызываются во время исполнения, поэтому вам нет необходимости компилировать большой главный .EXE. Главное .APP должно иметь те же установки, что и суб-приложение, за исключением того, что оно всегда компилируется как .EXE.

## Главное .APP должно иметь следующие установки:

### В свойствах приложения:

Dictionary File (файл словаря):                      Главный словарь.

Destination Type (тип файла назначения): .EXE

### В глобальных свойствах приложения:

.Generate Global Data as External  
(генерировать глобальные данные как внешние): ON (ВКЛ)

### File Control Flags (флажки управления файла):

|                                                                        |                            |
|------------------------------------------------------------------------|----------------------------|
| Generate All File declarations<br>(генерировать все объявления файлов) | OFF (ВЫКЛ)                 |
| External (внешние):                                                    | ALL EXTERNAL (ВСЕ ВНЕШНИЕ) |

All Files declared in another .APP  
(все файлы объявлены в другом приложении): ON (ВКЛ)

Declaring Module (Объявление модуля): Оставьте это пустым

**В дереве модулей приложения:**

Выберите Application > Insert Module , отберите Внешнюю LIB, затем выберите соответствующую .LIB для .DLL, содержащей определения данных.

Выберите Application > Insert Module , отберите Внешнюю LIB, затем выберите соответствующую .LIB для суб-приложения .DLL. Повторите этот шаг для каждого суб-приложения.

Для каждой процедуры главное приложение вызывает, редактирует процедуру ToDo следующим образом:

Шаблон: внешний шаблон.

Имя модуля: Выберите соответствующую .LIB для DLL из выпадающего списка.

При необходимости удалите любые пустые генерированные модули.

7. Руководитель команды компилирует главное .APP и испытывает вызовы .DLLs.

8. Руководитель команды периодически повторяет последний шаг до тех пор, пока вся работа всех разработчиков не завершена, а все приложение может быть испытано в целом.

## ***.EXE, состоящий из одного куска***

.EXE, состоящий из одного куска - это файл .EXE, который содержит все, что ему нужно для того, чтобы работать, за исключением файлов, которые не могут быть скомпонованы (.VBX, файлы данных и т.п.). Это означает, что для .EXE не нужны ассоциированные .DLL, .ICO и т.п., так как они уже скомпонованы в .EXE.

### **Когда осуществлять**

---

#### **Аргументы за и против использования во время разработки**

Потенциально никаких, так как окончательное состояние .EXE не обязательно влияет на организацию проекта во время разработки.

Например, приложение может быть разработано с помощью нескольких файлов .APP для генерации отдельных исполняемых (.EXE или .LIB). В конце концов отдельные исполняемые оказываются скомпонованными вместе в .EXE, состоящий из одного куска.

Время изготовления для “моно” .EXE больше, чем время изготовления его отдельных компонент, поэтому в точке цикла разработки, где вы компонуete и испытываете все компоненты, время изготовления возрастает.

#### **Аргументы за при эксплуатации**

Вы развертываете одиночный файл. Нет никакого риска забыть требуемый исполняемый файл или развернуть несовместимый набор .EXE и .DLL, нет риска конфликта с различными версиями .DLL, имеющими одно и то же имя.

#### **Аргументы против при эксплуатации**

Ваши изменения требуют перекомпоновки всего проекта плюс развертывания большего файла. Более крупные файлы обычно труднее развертывать.

#### **Выводы**

Используйте .EXE, состоящие из одного куска, для малых и средних приложений, которые не часто разворачиваются. Это значит, что приложение достаточно стабильно или что оно распределяется только небольшому числу конечных пользователей.

Используйте .LIBs тогда, когда вы хотите разделить свой процесс разработки на много приложений, но при этом вы хотите разворачивать свое приложение как “моно” .EXE. Вы можете вначале сделать каждое приложение .EXE, так чтобы оно могло быть независимо разработано и испытано. Когда вы готовы интегрировать все приложения вместе, сделайте вызываемые приложения в .LIBs, затем скомпонуйте их в родительское приложение.

## **Как осуществлять**

Создайте .EXE, состоящее из одного куска, из одиночного .APP файла или из многих .APP файлов. Самый простой способ - это создать .EXE из одиночного .APP файла. Или же вы можете использовать много .APP файлов для того, чтобы создать один или более .LIBs, которые скомпонованы в “моно” .EXE.

**Совет: Вы можете предпочесть создать .EXE вместо .LIB во время цикла разработки для того, чтобы уменьшить время компоновки и разрешить изолированное испытание и отладку несвязанных исполняемых. Ближе к концу проекта преобразуйте .LIBs, изменив этот выбор.**

Чтобы создать “моно” .EXE из одного или более файлов .APP:

### **Установите тип файла назначения на .EXE в родительском приложении**

Самый высокий уровень (родительский) типа файла назначения (Destination Type) для приложения должен быть установлен на .EXE в диалоговом окне Application Properties (свойства приложения). Родительское приложение - это приложение, чьи процедуры не вызываются процедурами любых других приложений.

1. Откройте родительское приложение.
2. Выберите из меню Application > Properties.
3. В диалоговом окне Application Properties (свойства приложения) выберите исполняемый (.EXE) из выпадающего списка Destination Type (тип файла назначения).

**Внимание: Важно, чтобы вы установили Тип файла назначения приложения, а не тип цели проекта, так как Генератор приложения и шаблоны принимают решения о генерации кода на основе значения типа файла назначения приложения, а не типа цели проекта.**

### **Установите библиотеку поддержки родительского приложения на Local**

Установите библиотеку поддержки родительского приложения (Run-Time Library) на Local в диалоговом окне редактора проекта Global Options (глобальные опции). Это компонует функции поддержки Clarion for Windows, включая все драйверы файлов, на которые есть ссылка, в .EXE приложения.

1. Откройте родительское приложение.
2. Выберите из меню Project > Properties.
3. В диалоговом окне Project Editor (редактор проекта) нажмите кнопку Properties.

4. В диалоговом окне Global Options (глобальные опции) выберите Local из ниспадающего списка Run-Time Library (библиотека поддержки).

**Внимание: Библиотека поддержки в данном контексте включает все драйверы файлов, используемые вашим приложением!**

Если нет .LIBs, тогда вы готовы изготовить “моно” .EXE. Перейдите к последнему шагу в разделе ниже Сделайте и испытайте свое “моно” .EXE.

### **Изготовьте .LIB**

Изготовьте и испытайте свое .LIB приложение также как вы это сделаете с любым другим приложением. Отметьте имя вашего .LIB и его процедур, так как вы будете ссылаться на эти имена в вашем родительском приложении.

**Совет: Чтобы избежать возможного предупреждения компоновщика, вы можете назвать первую процедура как-нибудь иначе, чем Главная. Для того, чтобы сделать это, воспользуйтесь диалоговым окном Application Properties (свойства приложения).**

1. Выберите из меню Application > Properties.
2. Выберите Library(.LIB) из ниспадающего списка Destination Type (тип файла назначения) и нажмите ОК.

### **Установите переключатель главного кода (Maincode) в положение выключено (Off)**

Смотрите более полную информацию о Maincode в Руководстве программиста.

### **Установите библиотеку поддержки приложения .LIB на Local**

Установите Run-Time Library (библиотека поддержки) .LIB приложения на Local в диалоговом окне Global Options (глобальные опции) редактора проекта. Это командует компоновщику, что функции поддержки Clarion for Windows , включая все драйверы файлов, на которые есть ссылка, находятся в .EXE родительского приложения.

1. Выберите из меню Project > Properties.
2. В диалоговом окне Project Editor (редактор проекта) нажмите кнопку Properties.
3. В диалоговом окне Global Options (глобальные опции) выберите Local из ниспадающего списка Run-Time Library (библиотека поддержки), затем нажмите ОК.
4. Выберите Project > Make из меню для того, чтобы изготовить .LIB.

### **Добавьте .LIB модули к вашему родительскому приложению**

Следующие шаги говорят вашему родительскому приложению, что некоторые внешние процедуры вызваны из конкретного .LIB.

1. Откройте родительское приложение.
2. Выберите из меню Application > Insert Module.
3. В диалоговом окне Select Module Type (выбор типа модуля) выберите ExternalLIB.
4. В диалоговом окне Module Properties (свойства модуля) наберите имя .LIB, затем нажмите ОК.

### **Добавьте внешние процедуры к своему родительскому приложению**

1. Откройте родительское приложение.
2. Выберите из меню Procedure > New.
3. В диалоговом окне New Procedure (новая процедура) наберите имя внешней процедуры и нажмите ОК.
4. В диалоговом окне Select Procedure Type (выбор типа процедуры) выберите External.

Появится диалоговое окно Procedure Properties (свойства процедуры). Кнопки Files (файлы), Procedures (процедуры), Formulas (формулы) и Extensions (расширения) на этом диалоговом окне не имеют значения для внешних процедур и вы должны их игнорировать.

5. В ниспадающем списке Module Name (имя модуля) выберите .LIB, содержащий внешнюю процедуру, и нажмите ОК.

Это скажет родительскому приложению, откуда вызывать внешнюю процедуру.

### **Добавьте файлы .RSC к проекту вашего родительского приложения**

Только для 16-битовых приложений добавьте .RSC, .ICO, .BMP и т.д. файлы, используемые .LIB, к проекту родительского приложения.

1. Откройте родительское приложение.
2. Выберите из меню Project > Edit.
3. Отметьте Library and Object files (библиотечные и объектные файлы) и нажмите кнопку Add File... (добавить файл).

4. Перейдите к .RSC файлу, а затем нажмите ОК.

.RSC файлы находятся в том же каталоге, что и .OBJ файлы для .LIB. По умолчанию каталог ....\CW20\OBJ.

**Совет:** Вы можете скопировать Генератору приложений генерировать .RSC файлы в каталоге вашего приложения путем редактирования файла переадресовки (..\CW20\BIN\CW20.RED). Вставьте точку и точку с запятой в строку \*.rsc следующим образом

\*.rsc = .:\cw20\obj

### **Изготовьте и испытайте свое “моно” .EXE.**

1. Откройте родительское приложение.

2. Выберите из меню Project > Run.

Генератор приложений и система проекта генерируют исходную программу, затем компилируют и компонуют одно-кусковый .EXE, основанный на:

- типе файла назначения Executable(.EXE) -свойство приложения
- типе мишени .EXE - редактор проекта , глобальные опции
- local библиотека поддержки - редактор проекта, глобальные опции
- модулях .LIB и внешних процедурах, которые вы вставили в свое дерево приложения.

**Внимание:** Не смешивайте 16-битовое и 32-битовое приложения пытаюсь компоновать 16-битовые .LIB или .DLL в 32-битовые .EXE, или наоборот.

## ***.EXE плюс .DLLs (.EXE Plus .DLLs)***

.EXE плюс .DLLs (.EXE Plus .DLLs) - это .EXE, требующее некоторых ассоциированных файлов, которые могут быть скомпонованы в .EXE во время компиляции. То есть, .EXE нужны ассоциированные .DLL, .ICO и т.д., так как они еще не скомпонованы в .EXE,

Использование этой конфигурации в первую очередь влияет на цикл эксплуатации приложения; однако, создание одного или более .DLL подразумевает наличие многих .APP файлов, что также влияет на цикл разработки.

.DLL часто включают библиотеку поддержки Clarion for Windows .DLL и один или более .DLL драйверов файлов. Пожалуйста, помните, что эти .DLL могут быть спрятаны созданием “моно” .EXE или компоновкой их в другой .DLL, который вы создаете! Смотрите ниже Размещение библиотеки поддержки Clarion for Windows.

### **Когда осуществлять**

---

Используйте .EXE плюс .DLLs для двух или более приложений, которые делят общий .DLL. Это сэкономит место на диске. Хорошим примером этому являются два приложения Clarion for Windows, которые используют один и тот же драйвер файла TopSpeed: CW2TPS32.DLL. Приложение и делимые .DLL должны быть довольно стабильны, у вас нет двух различных .DLL с одним и тем же именем на одной машине. Различные .DLL с одинаковым именем могут приводить к путанице и ухудшению ситуации, если файлами .DLL не управлять должным образом.

Используйте .EXE плюс .DLLs для очень больших приложений или для приложений, которые часто разворачиваются. Это даст вам возможность разворачивать только части приложения, которые действительно изменяются.

### **Как осуществлять**

---

Чтобы создать .EXE плюс .DLLs:

#### **Установите тип файла назначения .EXE в родительском приложении**

Установите тип файла назначения (Destination Type) для приложения на .EXE в диалоговом окне Application Properties (свойства приложения).

**Внимание: Обычно .EXE вызывает .DLLs, однако одно .DLL может вызвать другие .DLL.**

1. Откройте родительское приложение.
2. Выберите из меню Application > Properties.
3. В диалоговом окне Application Properties (свойства приложения) выберите execut-



able (исполняемую) (.EXE) из ниспадающего списка Destination Type (тип файла назначения).

**Внимание: Важно, чтобы вы установили Тип файла назначения приложения, а не тип цели проекта, так как Генератор приложения и шаблоны принимают решения о генерации кода на основе значения типа файла назначения приложения, а не типа цели проекта.**

### **Установите библиотеку поддержки родительского приложения на Standalone**

Библиотека поддержки родительского приложения (Run-Time Library) должна быть установлена на Standalone в диалоговом окне редактора проекта Global Options (глобальные опции). Это позволяет .EXE компоновать внешние функции Clarion for Windows во время работы из CW2RUNxx.DLL.

1. Откройте родительское приложение.
2. Выберите из меню Project > Properties.
3. В диалоговом окне Project Editor (редактор проекта) нажмите кнопку Properties.
4. В диалоговом окне Global Options (глобальные опции) выберите Standalone из ниспадающего списка Run-Time Library (библиотека поддержки).

Если вы не изготавливаете свои собственные .DLLs, т.е., если имеется только один .APP файл, тогда вы готовы изготовить .EXE плюс .DLLs. Перейдите к последнему шагу в этом разделе Сделайте и испытайте свое .EXE плюс .DLLs.

### **Сгенерируйте глобальные данные родительского приложения как внешние (EXTERNAL).**

**Внимание: Если .DLLs вашего приложения используют файлы вашего приложения, определения файлов и все глобальные переменные должны быть объявлены в .DLL (а не в .EXE) и экспортированы. Смотрите более полную информацию выше в Суб-приложении.**

Генерируйте глобальные данные родительского приложения как EXTERNAL (внешние). Это позволит разделять переменные GlobalRequest и GlobalResponse, а также глобальные переменные, которые вы назначите, между локальными и внешними функциями.

1. Откройте родительское приложение.
2. Выберите из меню Application > Global Properties.
3. В диалоговом окне Global Properties (глобальные свойства) отметьте поле Generate Global Data as EXTERNAL (генерировать глобальные данные как внешние).

4. Если имеется зеленый флажок около кнопки Data (данные), нажмите ее.
5. В диалоговом окне Global Data(глобальные данные) нажмите кнопку Properties (свойства).
6. В диалоговом окне Field Properties (свойства поля) отберите закладку Attributes (атрибуты).
7. В выпадающем списке Storage Class (класс хранения) выберите EXTERNAL-DLL. Повторите шаги с 5 по 7 для каждой позиции глобальных данных.

### **Изготовьте .DLLs**

Изготовьте и испытайте свое .DLL приложение так же, как вы испытали бы другое приложение. Заметьте имя вашего . DLL и его процедур, так как вы будете ссылаться на эти имена в вашем родительском приложении.

**Совет: Чтобы избежать возможного предупреждения компоновщика, не экспортируйте процедуру Main (главная).**

1. Выберите из меню Application > Properties.
2. Выберите Dynamic Link Library(.DLL) из выпадающего списка Destination Type (тип файла назначения) и нажмите OK.

Выбор Destination Type .DLL автоматически экспортирует (делает доступной) каждую процедуру приложения.

**Совет: Чтобы оптимизировать исполнение, экспортируйте только процедуры, вызываемые другой исполняемой программой.**

3. В диалоговом окне Procedure Properties (свойства процедуры) очистите поле Export Procedures (экспортные процедуры) для каждой процедуры, не вызываемой внешне.

Процедуры, которые вызываются только внутри .DLL приложения, не должны экспортироваться.

### **Установите библиотеку поддержки приложения .DLL на Standalone**

Установите библиотеку поддержки (Run-Time Library) приложения .DLL на Standalone в диалоговом окне редактора проекта Global Options (глобальные опции). Это позволяет .DLL компоновать функции Clarion for Windows во время работы из CW2RUNxx.DLL.

1. Выберите из меню Project > Properties.
2. В диалоговом окне Project Editor (редактор проекта) нажмите кнопку Properties.

3. В диалоговом окне Global Options (глобальные опции) выберите Standalone из ниспадающего списка Run-Time Library (библиотека поддержки), затем нажмите ОК.

4. Выберите из меню Project > Make.

#### **Добавьте модули .DLL к вашему родительскому приложению**

Следующие шаги скажут вашему родительскому приложению, что некоторые внешние процедуры вызваны из соответствующей .DLL.

1. Откройте родительское приложение.

2. Выберите из меню Application > Insert Module.

3. В диалоговом окне Select Module Type (выбор типа модуля) выберите ExternalLIB.

Не выбирайте ExternalDLL. Выбирайте ExternalDLL только тогда, когда вы используете .DLL, для которого не имеется соответствующей .LIB.

**Внимание: Когда вы изготавливаете .DLL с помощью Clarion for Windows, вы также генерируете соответствующую .LIB. Компоновщик компонуется .LIB во время компиляции, а .LIB вызывает .DLL во время работы.**

4. В диалоговом окне Module Properties (свойства модуля) наберите имя .DLL, затем нажмите ОК.

#### **Добавьте внешние процедуры к вашему родительскому приложению**

1. Откройте родительское приложение.

2. Выберите из меню Procedure > New.

3. В диалоговом окне New Procedure (новая процедура) наберите имя внешней процедуры и нажмите ОК.

4. В диалоговом окне Select Procedure Type (выбор типа процедуры) выберите External.

Появится диалоговое окно Procedure Properties (свойства процедуры).

Кнопки Files (файлы), Procedures (процедуры), Formulas (формулы) и Extensions (расширения) на этом диалоговом окне не имеют значения для внешних процедур и вы должны их игнорировать.

5. В ниспадающем списке Module Name (имя модуля) выберите .DLL, содержащий внешнюю процедуру, и нажмите ОК.

Это скажет родительскому приложению, откуда вызывать внешнюю процедуру.

### **Изготовьте и испытайте свой .EXE плюс .DLLs.**

1. Откройте родительское приложение.

2. Выберите из меню Project > Run.

Генератор приложений и система проекта генерируют исходную программу, затем компилируют и компонуют .EXE плюс .DLLs, основанный на:

- Типе файла назначения Executable(.EXE)- свойства приложения
- Типе мишени .EXE - редактор проекта, глобальные опции
- Standalone библиотека поддержки- редактор проекта, глобальные опции
- Модулях .DLLs и внешних процедурах, которые вы вставили в свое дерево приложения.

**Внимание: Не смешивайте 16-битовое и 32-битовое приложения пытайтесь скомпоновать 16-битовые .LIB или .DLL в 32-битовые .EXE,**

## **Размещение библиотеки поддержки Clarion for Windows**

Изготовление “моно” .EXE прячет библиотеку поддержки Clarion for Windows внутри .EXE. Вы можете также спрятать библиотеку поддержки путем компоновки ее в другие .DLL следующим образом:

### **Установите тип файла назначения на .EXE в родительском приложении**

Тип файла назначения (Destination Type) для родительского приложения должен быть установлен на .EXE в диалоговом окне Application Properties (свойства приложения). Родительское приложение - это приложение, чьи процедуры не вызываются процедурами любых других приложений.

1. Откройте родительское приложение.

2. Выберите из меню Application > Properties.

3. В диалоговом окне Application Properties (свойства приложения) выберите исполняемый (.EXE) из выпадающего списка Destination Type (тип файла назначения).

**Внимание: Важно, чтобы вы установили Тип файла назначения приложения, а не тип цели проекта, так как Генератор приложения и шаблоны принимают решения о генерации кода на основе значения типа файла назначения приложения, а не типа цели проекта.**

### **Установите библиотеку поддержки родительского приложения на External**

Установите библиотеку поддержки родительского приложения (Run-Time Library) на External в диалоговом окне редактора проекта Global Options (глобальные опции). Это командует .EXE компоновать внешние функции поддержки Clarion for Windows во время работы из .DLL ,отличного от CW2RUNxx.DLL.

1. Откройте родительское приложение.
2. Выберите из меню Project > Properties.
3. В диалоговом окне Project Editor (редактор проекта) нажмите кнопку Properties.
4. В диалоговом окне Global Options (глобальные опции) выберите Local из ниспадающего списка Run-Time Library (библиотека поддержки).

**Внимание: Библиотека поддержки в данном контексте включает все драйверы файлов, используемые вашим приложением!**

### **Установите библиотеку поддержки приложения .DLL на External**

Библиотека поддержки (Run-Time Library) родительского приложения должна быть установлена на External в диалоговом окне редактора проекта Global Options (глобальные опции). Это говорит .EXE компоновать внешние функции Clarion for Windows во время работы из .DLL иных, чем CW2RUNxx.DLL.

1. Откройте родительское приложение.
2. Выберите из меню Project > Properties.
3. В диалоговом окне Project Properties (свойства проекта) нажмите кнопку Properties.
4. В диалоговом окне Global Options (глобальные опции) выберите External из ниспадающего списка Run-Time Library (библиотека поддержки).

**Внимание: Библиотека поддержки в этом контексте автоматически включает все драйверы файлов, используемые вашим приложением!**

### **Сгенерируйте глобальные данные родительского приложения как внешние (EXTERNAL).**

**Внимание: Если .DLL вашего приложения используют файлы вашего приложения, определения файла и все глобальные переменные должны быть объявлены в .DLL (а не в .EXE) и экспортированы. Смотрите более полную информацию выше в Суб-приложении.**

Генерируйте глобальные данные родительского приложения как EXTERNAL (внешние). Это позволит делить переменные GlobalRequest и GlobalResponse, а также глобальные

переменные, которые вы назначите, между локальными и внешними функциями.

1. Откройте родительское приложение.
2. Выберите из меню Application > Global Properties.
3. В диалоговом окне Global Properties (глобальные свойства) отметьте поле Generate Global Data as EXTERNAL (генерировать глобальные данные как внешние).
4. Если имеется зеленый флажок около кнопки Data (данные), нажмите ее.
5. В диалоговом окне Global Data(глобальные данные) нажмите кнопку Properties (свойства).
6. В диалоговом окне Field Properties (свойства поля) отберите закладку Attributes (атрибуты).
7. В ниспадающем списке Storage Class (класс хранения) выберите EXTERNAL-DLL. Повторите шаги с 5 по 7 для каждой позиции глобальных данных.

### **Изготовьте .DLL, содержащую функции поддержки Clarion**

1. Выберите из меню File > New.
2. В диалоговом окне New (новый) отберите закладку Application (приложение).
3. В поле File Name (имя файла) наберите имя приложения.
4. Нажмите кнопку Create (создать).  
Появится диалоговое окно Application Properties.
5. Если применимо, выберите словарь данных.
6. Выберите Dynamic Link Library (.DLL) из ниспадающего списка Destination Type (тип файла назначения), затем нажмите ОК.

### **Установите библиотеку поддержки приложения .DLL на Local**

1. Выберите из меню Project > Properties.
2. В диалоговом окне Project Editor (редактор проекта) нажмите кнопку Properties.
3. В диалоговом окне Global Options (глобальные опции) выберите Local из ниспадающего списка Run-Time Library (библиотека поддержки), затем нажмите ОК.

4. Выберите Project > Make из меню для того, чтобы изготовить .LIB.

Это компонует функции Clarion for Windows в ваше .DLL приложение во время компиляции, так что они могут в конце концов быть скомпонованы в вашем .EXE во время работы. Это также генерирует файл определения модуля (.EXP) для вашей .DLL. Вы редактируете файл .EXP следующим шагом для того, чтобы экспортировать функции поддержки из вашего .DLL.

### **Экспортируйте функции поддержки Clarion for Windows**

1. Откройте родительское приложение.

2. Выберите Project > Make из меню.

Так как функции поддержки не были экспортированы, компоновщик генерирует ошибку “не определена” для каждой функции. Ожидайте сообщений об ошибке от нескольких сотен до тысячи, в зависимости от вашего приложения.

В следующих шагах мы используем текстовый редактор Clarion for Windows для преобразования сообщений ошибки в операторы EXPORT.

3. Нажмите кнопку Edit Errors (редакция ошибок), затем нажмите ОК чтобы отредактировать сообщения об ошибках.

4. Удалите все сообщения, кроме сообщений “...function is unresolved...” (функция не определена).

5. Выберите Search > Replace из меню.

6. В поле Find what (что найти) наберите is unresolved in file <filename.ext> (неразрешена в файле <имя файла.ext>).

Замените <filename.ext> на имя файла и расширение в первом сообщении.

7. В поле Replace with (заменить на) наберите знак @ и знак вопроса без пробелов, типа: @?.

8. Нажмите кнопку Replace All (заменить все).

Это преобразует сообщения ошибки для файла <filename.ext> в операторы EXPORT. Повторите шаги с 5 по 8 до тех пор, пока все ...function is unresolved... не будут преобразованы.

9. Удалите дублирующие операторы.

Отберите все операторы и скопируйте их в буфер, а затем скопируйте их в систему подготовки текстов или электронную таблицу, которая обладает способностями сортировки. Отсортируйте операторы так, чтобы дубликаты операторов появились вместе, затем удалите дубликаты. Отберите все оставшиеся операторы, скопируйте их в буфер обмена и сохраните свою работу.

Вставьте операторы EXPORT в экспортный файл (.EXP) для .DLL приложения в следующих шагах.

10. Если вы еще не сделали этого, выберите Exit! (выйти) из меню для того, чтобы выйти из текстового редактора.

11. Выберите из меню File > Open.

12. В поле File Name (имя файла) наберите \*.exp, затем нажмите ввод (ENTER).

13. Выберите файл .EXP, ассоциированный с вашей .DLL (не ваше родительское приложение) и нажмите ОК.

Это открывает файл .EXP с текстовым редактором.

14. Прокрутите до конца этот файл и щелкните клавишей мыши для того, чтобы установить точку вставки.

15. Выберите из меню File > Paste.

**Совет: Вы можете модифицировать файл .EXP путем вставки текста в глобальные точки встраивания, связанные с .EXP.**

Изготовьте .DLL приложение с экспортированными функциями

1. Откройте .DLL приложение.

2. Выберите из меню Project > Make.

Это компонует функции поддержки Clarion for Windows в вашу .DLL и экспортирует их, так что они могут быть вызваны во время работы вашим .EXE.

### **Изготовьте родительское приложение**

1. Откройте родительское приложение.

2. Выберите из меню Project > Make.

Теперь, когда функции поддержки Clarion for Windows экспортированы, компоновщик разрешает вызовы к этим функциям и больше не выдает сообщений о неопределенных функциях .



## **Приложение Д DDE - Динамический обмен данными**



Данная глава представляет Динамический обмен данными (DDE). Полное обсуждение Clarion и DDE вы можете найти в Справочнике языка. Данное приложение имеет своей целью только предоставить описание того, что может делать DDE, продемонстрировать, что его легко реализовать и предложить один-два совета в помощь вашему исследованию DDE.

DDE (Dynamic Data Exchange) является межпроцессным коммуникационным (IPC) протоколом Windows. “Диалог” DDE состоит из двух приложений, обменивающихся сообщениями. В DDE “Диалоге” одно приложение действует как клиент, другое как сервер.

Приложение, которое начинает “Диалог”, запрашивая данные или услуги от другого приложения, является клиентом. Контактируемое приложение является сервером. Сервер должен “зарегистрировать” в Windows, что он имеет возможности сервера.

Clarion for Windows дает вам возможность создать и DDE клиента и DDE сервера. Приложение может быть и тем и другим. На деле ваше приложение может действовать и как клиент и как сервер, хотя это требует двух отдельных “Диалогов” DDE.

Чтобы получить информацию об использовании Clarion for Windows в качестве DDE сервера, смотрите Руководство программиста.

#### Возможности

Запуск DDE “Диалога” - клиента с сервером

Инициализация “Диалога

Посылка команд DDE

Отсылка данных от клиента к серверу

## **Возможности**

### **В качестве клиента DDE ваше приложение может:**

- ◆ Инициировать DDE “Диалог” с DDE- сервером через функцию DDECLIENT.
- ◆ Получать данные от сервера через оператор DDEREAD. EVENT:DDEdata сообщает вашему приложению, что в данный момент для него имеются данные для чтения.
- ◆ Посылать командную строку символов серверу через оператор DDEEXECUTE.

Многие существующие приложения Windows позволяют получить доступ к их функциональным возможностям через командные сообщения. Например, вы можете исполнить любой макро оператор Microsoft Excel, заключая его в квадратные скобки и отсылая его как параметр в виде строки символов в оператор DDEEXECUTE.

- ◆ Отослать незапланированные данные серверу с оператором DDEPOKE.

Обычно вы обеспечиваете сервер описанием “позиции” и ее величины (строка символов). Например, чтобы поместить величину в особую ячейку в таблице Excel, позиция является адресом ячейки в формате R1C1. Эта величина является действительной величиной, которую вы хотите поместить в ячейку.

### **В качестве DDE сервера ваше приложение может:**

- ◆ Проверить “тему”, которую определяет клиент, контактирующий с вашим приложением-сервером.

Когда клиент контактирует с вашим приложением-сервером, он определяет в строке символов, о чем должен быть “диалог”. Вы кодируете в вашем приложении необходимость проверки строки символов относительно устанавливаемого вами списка, а затем предпринимаете соответствующее действие, когда тема совпадает с позицией в вашем списке.

“Стандартные темы” де-факто Windows DDE - это имя текущего документа и тема “Система”. Текущий документ - это имя любого открытого файла, связанного с приложением-сервером. Тема “система” обычно переключает возвращаемое сообщение, перечисляя через запятую имеющиеся “темы”, которые поддерживает ваш сервер.

- ◆ Обеспечить автоматическое обновления данных через оператор DDEWRITE, когда “режим” установлен на DDE:auto.

Это дает вам возможность установить переменную. Сервер автоматически пошлет сообщение клиенту, когда величина переменной изменяется.

- ◆ Предоставляет доступ к “командам”.

Приложение-сервер отыскивает командную строку символов с помощью функции DDEITEM. Вы кодируете в вашем приложении необходимость проверки строки символов относительно устанавливаемого вами списка, а затем предпринимаете соответствующее действие, когда тема совпадает с позицией в вашем списке.

В Справочнике языка вы найдете описания всех операторов и функций DDE. Оставшиеся разделы данной главы описывают эти возможности с помощью обобщенных примеров, в которых клиент Clarion DDE посылает образцы запроса Клиента к Microsoft Excel, затем посылает данные для размещения в одной ячейке большой таблицы.

## **Запуск DDE “Диалога” - клиента с сервером**

Запуск DDE “Диалога” дело такое же простое, как использование функции DDECLIENT. Единственно, что оба приложения должны уже работать, чтобы канал мог быть открыт.

Самый простой способ убедиться, что “Диалог” происходит во время прогона, это использовать структуру IF. Функция DDECLIENT возвращает нуль, если приложение-сервер не работает. Проверьте возвращаемое ею значение и используйте оператор RUN, чтобы запустить приложение-сервер, если оно возвращает нуль.

Многие из DDE процедур и функций требуют, чтобы вы установили номер канала DDE, который является целым числом, возвращаемым Windows, когда вы открываете DDE “Диалог”. Создайте локальную переменную для сохранения возвращаемой величины. Начните в диалоговом окне Procedure Properties (свойства процедуры) процедуры, которую вы хотите иметь для DDE-”Диалога”.

□ Объявите переменную для хранения номера канала DDE.

1. Нажмите кнопку Data (данные) в диалоговом окне Procedure Properties (свойства процедуры).

2. Нажмите кнопку Insert (вставка) в диалоговом окне Local Data (локальные данные).

3. Наберите Channel (канал) в поле Name (имя).

4. Выберите LONG из выпадающего списка Type (тип).

5. Нажмите кнопку ОК, чтобы закрыть диалоговое окно Field Properties (свойства поля).

6. Чтобы закрыть диалоговое окно Local Data (локальные данные), нажмите кнопку Close (закрыть).

### **Инициализация “Диалога”**

---

Вы должны встроить код, который для того, чтобы инициализировать DDE беседу, запустит приложение - сервер, если оно еще не запущено. Предполагая, что приложение начинает “Диалог” после выбора пользователем соответствующей позиции меню, встройте код на событии поля, которое связано с позицией в меню Принятого события.

1. Выберите соответствующее событие поля в списке Embedded Source (вставка исходного текста).

2. Нажмите кнопку Edit (редактировать).

3. Выберите позицию Source (исходный текст) в диалоговом окне Embedded Source (вставка исходного текста).

4. Нажмите кнопку Add (добавить).

5. Наберите следующий код, заменяя “Excel” на имя файла (без расширения) сервера.

|                                       |                           |
|---------------------------------------|---------------------------|
| Channel = DDECLIENT('Excel','System') | ! Обратитесь к            |
|                                       | ! Excel через тему System |
| IF Channel < 1                        | ! Если не было            |
|                                       | ! контакта                |
| RUN('Excel')                          | ! Попробуйте              |
|                                       | ! запустить Excel         |
| Channel = DDECLIENT('Excel','System') | ! И попробуйте снова      |
| ELSE                                  |                           |
| RETURN                                | ! Оставьте, если нет      |
|                                       | !контакта -               |

END

Пример кода намеренно упрощен; было бы более эффективно выполнить попытку установить контакт дважды в цикле, а затем, если необходимо, предупредить конечного пользователя о неудаче.

Данный фрагмент программы попытается открыть Excel в качестве сервера. Функция DDECLIENT возвращает величину, соответствующую каналу; неважно, каков номер канала. Если он меньше единицы, это неудача. Вы должны, следовательно, запустить сервер и попытаться вновь открыть “Диалог”.

Вторым параметром функции DDECLIENT является DDE “тема”. Этот параметр сообщает серверу, что “должен начаться” DDE “Диалог”. В большинстве случаев тема - это имя файла. В данном случае в программе используется тема “System”, что говорит Excels, что беседа не касается конкретного документного файла.

## Посылка команд DDE

Когда канал DDE открыт, вы можете использовать функции DDE для посылки серверу команд, данных или запросов.

Пример кода, приведенный ниже, посылает команду Excel открыть новый файл и сохранить его под установленным именем файла. Это обычная задача DDE при работе с коммерческими приложениями. Зачастую приложение сервер позволяет доступ к “документным” функциям только тогда, когда вы установите имя документа в функции DDECLIENT. Имя документа должно быть уже существующим файлом.

В этом конкретном случае, чтобы выполнить какие-либо “документные” действия, такие, как ввод величины в ячейку, Excel (и многие другие приложения) требует, чтобы “тема”

канала DDE была именем документа. Следовательно, если ваше приложение обеспечивает новые данные, относительно которых оно хочет, чтобы сервер сохранил их в новом документном файле, ваше приложение должно:

- Открыть “Диалог” с темой “System”.
- Отправить команду, которая потребует от сервера сохранения документного файла под установленным именем.
- Закрыть “Диалог”.
- Открыть второй “Диалог” с сервером, на этот раз устанавливая имя вновь созданного файла как тему.
- Послать “незапланированные” (потому что сервер не запрашивал их) данные и затем сообщить DDE-серверу (Excel) о необходимости выполнить команды или другие запросы о данных, которые относятся к данному файлу.
- Закрыть “Диалог”.

Нижеследующее, следовательно, должно выполняться только в том случае, если выполнение приведенного выше примера программы было успешным.

DDEEXECUTE(Channel,'[NEW(1)]')!Команда Excel File/New (новый файл)

Оператор DDEEXECUTE берет номер канала DDE в качестве своего первого параметра, строку символов команды в качестве второго параметра. Excel требует, чтобы все DDE команды были заключены в квадратные скобки (стандартное условие DDE). Эта команда создает пустую большую таблицу.

Строка символов команды Excel, заключенная в квадратные скобки, является макрооператором Excel. Excel и многие другие приложения позволяют посылать макрооператор через оператор DDEEXECUTE. В данном конкретном случае вам не нужно знать имя открытого Excel файла, чтобы выполнить оператор.

**Совет: Многие коммерческие приложения со своими собственными макроязыками позволяют вам записывать и редактировать макросы. Используйте приложение для “холостого выполнения” действий, которые вам нужно иметь исполненными при включенной макрозаписи. Отредактируйте получившееся макро и используйте буфер обмена для копирования каждого макрооператора в ваше окно вставок исходного текста. Поместите каждый макрооператор во второй параметр оператора DDEEXECUTE, и вы можете быть уверены в правильном его синтаксисе для DDE команды!**

1. В следующей вставке исходного текста укажите Excel о необходимости сохранить новую (пустую) таблицу под именем, устанавливаемым вами.





## **Приложение Е Изготовление вызовов API**



**(Интерфейс программирования приложения)**

Данное приложение является введением к вызовам API (Интерфейс программирования приложения). Данное приложение предназначено чтобы дать только общую схему того, что называется вызовами API, того, что они делают, а также дать для начала некоторые базовые примеры.

Вызовы API дают вам метод вызова для вашей программы функций, которые являются внешними для вашего приложения. Иными словами, вызов API - это просто вызов функции из чьей-либо динамически загружаемой библиотеки (DLL). DLL - это файл, содержащий исполняемый код, который подключается к вашему .EXE во время исполнения. Вы можете использовать вызовы API для выполнения Объектной компоновки и Встраивания (OLE), а также для выполнения мультимедиа в ваших приложениях.

В общем случае, вызовов API из Clarion включает два этапа: макетирование функций API и компоновка API функций в вашу программу. Однако многие вызовы Windows API уже скомпонованы для вас. Смотрите ниже Компоновка API функций.

Создание прототипов API функций

Компоновка API функций

API функции Windows

Другие API функции

Функция вызова CALL

## Создание прототипов API функций

Для каждой API функции, которую вы хотите вызвать, должен быть указан прототип в структуре MAP Clarion. На функции, написанные не на языке Clarion, ссылки в программе Clarion могут быть реализованы путем создания эквивалентного прототипа Clarion. Прототипы помещаются в структуру MODULE, которая идентифицирует имя библиотеки DLL как параметр MODULE. Например, если имя DLL - WIN32.DLL, тогда структура модуля и прототипа для функции GetWindowsDirectory (получить каталог Windows):

```
MAP
  MODULE('WIN32.LIB')
    GetWindowDirectory(*CSTRING,USHORT),USHORT,RAW,PASCAL
  END
END
```

Для того, чтобы продолжить ваше макетирование, вам понадобится техническая ссылка, описывающая функции DLL, цели и параметры. Для вызовов Windows API мы обеспечили некоторые примеры прототипов в C:\CW20\LIBSRC\WINDOWS.CLW. Вам может быть захочется почитать Как использовать DLL, не созданные в Clarion for Windows в разделе Часто задаваемые вопросы оперативной помощи Clarion.

Смотрите также Прототипы процедур и Функций в Справочнике языка.

Имеется несколько вопросов, требующих рассмотрения при создании прототипов Clarion, зависящих от языка исходной программы DLL. Первым моментом является нахождение эквивалентных типов данных в двух языках. Вы можете определить эквивалентные типы данных рассматривая глубинное машинное представление данных. Например, тип данных SREAL Clarion хранится в четырех байтах со знаком и плавающей точкой в формате Intel 8087, в то время как BFLOAT4 хранится в четырех байтах со знаком и плавающей точкой в формате Microsoft Basic.

Имеется несколько эквивалентов типов данных между Clarion и C или C++:

### C.C++

unsigned char  
short  
unsigned short  
long  
unsigned long  
float  
double

### Clarion

BYTE  
SHORT  
USHORT  
LONG  
ULONG  
SREAL  
REAL

```

Struct {
    unsigned long ul1
    unsigned long ul2
}   структ1;

Struct1 GROUP
ul1      ULONG
ul2      ULONG
END

```

Вторым важным вопросом построения прототипов является условие вызова функции, используемое другим языком. Clarion обеспечивает поддержку для трех различных вызывающих условий: PASCAL, C, TopSpeed's Register Based.

## **Компоновка API функций**

Для того, чтобы вызвать API функцию, вы должны сначала скомпоновать эту функцию в программу. Это может быть выполнено несколькими способами. Некоторые функции (WIN16.LIB и WIN32.LIB) скомпонованы автоматически. Другие .DLL функции должны быть явно скомпонованы из соответствующего .LIB файла. Наконец, функции могут быть также динамически загружены с помощью функции Clarion CALL.

## **API функции Windows**

---

С точки зрения языка Clarion, вызовы API могут быть разделены на две категории: вызовы Windows API и другие вызовы API. Вызовы Windows API - это вызовы функций, которые существуют в трех главных библиотеках Windows (USER.EXE, GDI.EXE и KERNEL.EXE).

Так как язык Clarion широко использует вызовы Windows API, многие функции API уже скомпонованы в рабочие библиотеки Clarion. Это означает, что вы можете выполнить вызовы Windows API из ваших программ Clarion просто указанием прототипов и вызовом. Компоновка уже сделана за вас.

Clarion предоставляет C:\CW20\LIB\WIN16.LIB и C:\CW20\LIB\WIN32.LIB с данной версией, поэтому ссылки на эти функции могут быть решены во время процесса компилирования и компоновки. Многие Windows-ориентированные технические справочники или Windows API Bible могут обеспечить информацию по функциям, имеющимся в этих библиотеках Windows.

Вот пример того, как вызывать функцию API Windows:

```

PROGRAM
MAP
    MODULE('WIN16.LIB')
        MessageBox(USHORT,*CSTRING,*CSTRING,USHORT),PASCAL,RAW
    END
END

```

Caption            CSTRING(18)  
MessageText       CSTRING(32)

#### CODE

```
Caption = 'Title'  
MessageText = 'This is the text'  
MessageBox(0,MessageText,Caption,30)
```



## Другие API функции

Для API функций не из WIN16.LIB или WIN32.LIB вы должны иметь библиотечный файл (.LIB), который соответствует .DLL. Как только вы получите .LIB, который соответствует .DLL, добавьте к вашему файлу проекта .LIB, так чтобы вы могли разрешить внешнюю ссылку во время процесса компилирования и компоновки. Укажите прототипы функций, которые вызывает ваше приложение, затем скомпилируйте и скомпонуйте как обычно. Функция после этого может быть динамически загружена в вашу программу во время выполнения.

### Создание .LIB из .DLL

Изготовление и использование .LIB, которое соответствует .DLL, может быть выполнено с помощью следующих шагов:

1. Создайте файл экспорта (.EXP) для DLL.
2. Создайте Библиотечный файл (.LIB) для DLL.
3. Сошлитесь на Библиотечный файл (.LIB) в Системе проекта.

### Создайте файл экспорта для DLL

Технический комплект TopSpeed включает программу (TSIMPLIB.EXE), которая может быть использована для создания файлов .LIB из .DLL. Технический комплект TopSpeed распространяется вместе с TopSpeed C, C++, Modula-2 и Pascal.

Вы можете также выделить набор доступных имен функций DLL из DLL с помощью утилитой программы командной строки EXEHDR.EXE DOS. Эта программа появляется на большинстве дискет DOS более ранних, чем версия 6.0. Если эта программа-утилита недоступна, тогда ее можно заменить другой утилитой или методом, которые обеспечивают тот же список имен.

Приложите выделенные имена функций (отделенные от любого окружающего текста) к информации заголовка файла экспорта, даваемого ниже. Подставьте соответствующее DLL имя вместо слова “dllname” на строке 1 информации заголовка. Сохраните экспортный файл под тем же файловым именем, что и DLL, с расширением .EXP.

```
-----Start of EXPORT File -----
LIBRARY dllname
CODE MOVEABLE DISCARDABLE PRELOAD
DATA MOVEABLE SINGLE PRELOAD
HEAPSIZE 1024
STACKSIZE 32678
SEGMENTS
    ENTERCODE MOVEABLE DISCARDABLE PRELOAD
EXETYPE WINDOWS
EXPORTS
    сюда вставить функцию и имена функции (одно имя на строку)
-----End of Export File -----
```

### **Создайте библиотечный .LIB файл Clarion for Windows для DLL**

Как только вы создали экспортный (.EXP) файл, вы можете создать файл .LIB для DLL, используя для этого команду системы проекта #implib. Команда #implib создает или обновляет библиотечный файл (.LIB), основанный на информации, содержащейся в экспортном (.EXP) файле. Синтаксис команды:

```
#implib      <имя библиотечного файла>      <имя экспортного файла>
```

Где

<имя библиотечного файла> - имя DLL с расширением .LIB

<имя экспортного файла> - имя DLL с расширением .EXP

С помощью текстового редактора создайте файл проекта Clarion (.PRJ) и введите одну строку в файл, содержащую команду проекта системы #implib с соответствующими параметрами. Обратите внимание, что #implib должно быть в нижнем регистре. Сохраните проектный файл под соответствующим именем (т.е. именем DLL с расширением .PRJ). Более полную информацию о #implib вы можете найти в Руководстве программиста.

В Clarion for Windows поставьте проектный файл, который вы только что создали, в качестве текущего проекта:

1. Выберите Project III Set.
2. Выберите проектный файл и нажмите кнопку ОК.
3. Изготовьте проектный файл, нажав для этого кнопку Make на линейке инструментов.

Если команда Make была успешной и библиотечный (.LIB) файл был создан, появится окно подтверждения с зеленой отметкой в правом нижнем углу и соответствующим сообщением о завершении. Библиотечный файл (.LIB) готов к использованию.

### **Соплтитеся на файл .LIB в системе проекта**

Поместите библиотечный файл (.LIB) в дерево проекта ( в раздел “библиотечные и объектные файлы”) любого проекта при использовании вами ассоциированных DLL функций. Во время фазы компоновки Make компоновщик распознает любые цитируемые функции в библиотечном (.LIB) файле.

## **Функция вызова CALL**

Если у вас нет .LIB файла, соответствующего .DLL, к которой вы хотите получить доступ, и вы не можете сделать такой файл, Clarion функция CALL предоставляет вам ограниченный доступ к функциям .DLL без явной компоновки функции. С функцией CALL связано многое лишнее по сравнению с использованием непосредственно функции API. Функция CALL использует промежуточную API функцию для получения доступа к целевой API функции и требует, чтобы вы знали, где в .DLL находится нужная функция. Вы не можете передать параметры вызванной с помощью CALL функции, не можете вы и вернуть какие-либо величины. Дополнительную информацию смотрите в Справочнике языка.





# Содержание

|                                                                             |           |
|-----------------------------------------------------------------------------|-----------|
| <b>Глава 1 Введение .....</b>                                               | <b>1</b>  |
| <b><i>Добро пожаловать на обзор по разработке Windows приложений. .</i></b> | <b>3</b>  |
| <b><i>Что вы найдете в этой книге .....</i></b>                             | <b>5</b>  |
| Где получить дополнительную информацию .....                                | 7         |
| <b><i>Условные обозначения в документации .....</i></b>                     | <b>8</b>  |
| Шрифтовые условные обозначения : .....                                      | 8         |
| Условные обозначения для клавиатуры: .....                                  | 9         |
| Техническое обслуживание .....                                              | 9         |
| Факс-система TopSpeed .....                                                 | 10        |
| <b>Глава 2 Установка .....</b>                                              | <b>11</b> |
| <b><i>Требования к системе .....</i></b>                                    | <b>13</b> |
| <b><i>Программа установки .....</i></b>                                     | <b>13</b> |
| Начало установки .....                                                      | 14        |
| Варианты установки .....                                                    | 14        |
| <b><i>Запуск Clarion For Windows .....</i></b>                              | <b>15</b> |
| <b><i>Быстрый доступ к файлам и функциям .....</i></b>                      | <b>16</b> |
| Основная панель инструментов .....                                          | 16        |
| Список выбора .....                                                         | 17        |
| <b>Глава 3 Последовательность разработки .....</b>                          | <b>19</b> |
| <b><i>Схема этапов разработки приложения .....</i></b>                      | <b>21</b> |
| <b><i>Программирование в Clarion .....</i></b>                              | <b>22</b> |
| Шаблоны и генерация исходных текстов .....                                  | 22        |
| Процесс разработки .....                                                    | 24        |
| <b><i>Среда разработки Clarion .....</i></b>                                | <b>25</b> |
| Редактор словаря данных .....                                               | 25        |
| Генератор приложений .....                                                  | 27        |
| Форматер окна .....                                                         | 29        |
| Форматер отчета .....                                                       | 30        |
| Редактор текста .....                                                       | 30        |
| Редактор формул .....                                                       | 31        |
| Проектная система .....                                                     | 31        |
| Отладчик .....                                                              | 31        |
| Контекстный справочник .....                                                | 32        |

|                                                                       |            |
|-----------------------------------------------------------------------|------------|
| <b>Глава 4 Редактор словаря данных .....</b>                          | <b>33</b>  |
| <b>Что такое словарь данных .....</b>                                 | <b>36</b>  |
| Преимущества использования словаря данных .....                       | 36         |
| Функции редактора словаря .....                                       | 37         |
| <b>Разработка словаря и базы данных .....</b>                         | <b>37</b>  |
| Нормализация .....                                                    | 37         |
| Ключи .....                                                           | 38         |
| Реляционные операции .....                                            | 40         |
| Редактор словаря данных .....                                         | 41         |
| <b>Создание словаря данных .....</b>                                  | <b>42</b>  |
| <b>Открывая редактор словаря .....</b>                                | <b>44</b>  |
| <b>Добавление в словарь файлов .....</b>                              | <b>46</b>  |
| Несколько слов о быстрой загрузке Quick Load .....                    | 47         |
| Импорт определений файлов .....                                       | 47         |
| Свойства нового файла .....                                           | 48         |
| <b>Добавление в словарь .....</b>                                     | <b>52</b>  |
| <b>Добавление и изменение полей .....</b>                             | <b>54</b>  |
| Определение свойств поля .....                                        | 55         |
| <b>Добавление и изменение ключей .....</b>                            | <b>66</b>  |
| Задание свойств ключа .....                                           | 68         |
| Компоненты ключа .....                                                | 71         |
| <b>Добавление и изменение связей .....</b>                            | <b>72</b>  |
| Установка требований по обеспечению ссылочной целостности .....       | 74         |
| <b>Управление вашим словарем .....</b>                                | <b>75</b>  |
| Копирование и вставка .....                                           | 75         |
| Версии словаря .....                                                  | 76         |
| Средства настройки Редактора словаря .....                            | 76         |
| <b>Глава 5 Использование Генератора Приложений .....</b>              | <b>79</b>  |
| <b>Установка режимов Генератора приложений. ....</b>                  | <b>81</b>  |
| <b>Создание APP файла. ....</b>                                       | <b>89</b>  |
| <b>Обзор: Создание приложения .....</b>                               | <b>92</b>  |
| <b>Определение глобальных характеристик .....</b>                     | <b>97</b>  |
| Глобальные переменные .....                                           | 97         |
| Общие характеристики приложения .....                                 | 98         |
| Поддержка файла .INI .....                                            | 99         |
| Определение других глобальных характеристик - Управление файлом ..... | 100        |
| Точки вставок в окне глобальных характеристик .....                   | 103        |
| <b>Добавление процедур к вашему приложению .....</b>                  | <b>104</b> |
| Определение типа процедуры .....                                      | 104        |

|                                                                |                   |
|----------------------------------------------------------------|-------------------|
| Определение характеристик процедуры .....                      | 105               |
| Файлы процедуры .....                                          | 108               |
| Окна процедуры .....                                           | 108               |
| Документы процедуры .....                                      | 109               |
| Данные процедуры .....                                         | 109               |
| Вызовы других процедур .....                                   | 111               |
| Встраивание исходного текста (Embedded Source) .....           | 112               |
| Формулы в процедуре .....                                      | 120               |
| Использование в процедуре распределенных шаблонов .....        | 120               |
| <b>Прототипы и передача параметров .....</b>                   | <b>121</b>        |
| Добавление параметров к прототипу процедуры .....              | 121               |
| Добавление параметров к оператору PROCEDURE или FUNCTION ..... | 122               |
| Вызов процедур и функций с параметрами .....                   | 123               |
| <b>Работа с деревом приложения .....</b>                       | <b>125</b>        |
| Использование меню Edit .....                                  | 125               |
| Использование меню Application .....                           | 126               |
| Использование меню Procedure .....                             | 128               |
| Использование страниц просмотра .....                          | 129               |
| Всплывающее меню .....                                         | 130               |
| <b>Обслуживание ваших шаблонов .....</b>                       | <b>132</b>        |
| Настройка регистра шаблонов .....                              | 132               |
| Открытие Регистра шаблонов .....                               | 133               |
| Обслуживание регистра шаблонов .....                           | 134               |
| <b>Команды импорта / экспорта приложений .....</b>             | <b>136</b>        |
| <b><u>Глава 6 Мастера и шаблоны процедур .....</u></b>         | <b><u>137</u></b> |
| <b>WIZARDS (МАСТЕРА) .....</b>                                 | <b>139</b>        |
| Мастер быстрого старта .....                                   | 139               |
| Мастер приложения .....                                        | 142               |
| Мастер процедуры просмотра .....                               | 143               |
| Мастер формы .....                                             | 144               |
| Мастер отчета .....                                            | 145               |
| Мастер печати словаря .....                                    | 146               |
| Оптимизация для Мастеров .....                                 | 146               |
| <b>Шаблоны процедур .....</b>                                  | <b>150</b>        |
| Шаблон Window .....                                            | 151               |
| Шаблон Frame .....                                             | 152               |
| Шаблон заставки Splash .....                                   | 154               |
| Шаблон Menu .....                                              | 154               |
| Шаблон исходной программы Source .....                         | 155               |
| Шаблон процесса (Process) .....                                | 155               |
| Шаблон External .....                                          | 157               |

|                     |     |
|---------------------|-----|
| Шаблон Browse ..... | 158 |
| Шаблон Form .....   | 164 |
| Шаблон Report ..... | 167 |
| Шаблон Viewer ..... | 170 |

## **Глава 7 Шаблоны элементов управления, программы и расширений .... 173**

### **Шаблоны элементов управления ..... 176**

|                                                                             |     |
|-----------------------------------------------------------------------------|-----|
| Добавление шаблонов элементов управления .....                              | 176 |
| Поле просмотра (BrowseBox) .....                                            | 176 |
| Шаблоны управления обновлением поля просмотра (BrowseUpdateButtons) .....   | 185 |
| Шаблон управления выбором в поле просмотра (BrowseSelectButton) .....       | 185 |
| Шаблон кнопки публикации процедуры просмотра .....                          | 186 |
| Шаблон управления сохранением данных. Кнопка OK. (SaveButton) .....         | 187 |
| Шаблон управления Кнопки Cancel (прервать) .....                            | 190 |
| Кнопка Close (закрыть) .....                                                | 190 |
| Просмотр текстовых файлов ( ASCIIBox) .....                                 | 191 |
| Кнопка управления печатью ASCII файлов (ASCII Print Button) .....           | 191 |
| Кнопка поиска в ASCII - файлах (ASCII Search ) .....                        | 191 |
| Доступ к стандартному диалогу выбора файлов (DOSFileLookup) .....           | 192 |
| OOPASCIIViewer .....                                                        | 194 |
| Шаблон элемента управления FileDrop .....                                   | 194 |
| Шаблон элемента управления FileDropCombo .....                              | 198 |
| Шаблон Дерево отношений (Relation Tree) .....                               | 202 |
| Шаблон Кнопок обновления дерева отношений (RelationTreeUpdateButtons) ..... | 205 |

### **Кодовые шаблоны ..... 206**

|                                                                      |     |
|----------------------------------------------------------------------|-----|
| Инициировать процесс (thread) .....                                  | 206 |
| Вызов процедуры доступа к справочнику (CollProcedureAsLOOKUP ) ..... | 206 |
| Шаблон проверки достоверности данных (ControlValueValidation) .....  | 207 |
| Поиск записи в несвязанном файле (LookupNonRelatedRecord) .....      | 207 |
| Закрыть текущее окно (CloseCurrentWindow) .....                      | 208 |

### **Шаблоны расширений ..... 209**

|                                                              |     |
|--------------------------------------------------------------|-----|
| Шаблон показа даты, времени (DateTimeDisplay) .....          | 209 |
| Шаблон проверки правильности записи (RecordValidation) ..... | 210 |

## **Глава 8 Форматер окна ..... 211**

### **Обзор: создание окон вашего приложения ..... 214**

### **Выбор типа окна ..... 216**

|                                         |     |
|-----------------------------------------|-----|
| Окна приложения - SDI или MDI .....     | 216 |
| Окна документов и диалоговые поля ..... | 216 |
| Структуры окна по умолчанию .....       | 217 |

### **Определение окон вашего приложения ..... 221**

|                                                |     |
|------------------------------------------------|-----|
| Процедуры Форматера окна .....                 | 221 |
| Инструменты форматера окна .....               | 222 |
| Меню Форматера окна .....                      | 229 |
| Диалог Свойства окна (Window Properties) ..... | 240 |
| Размещение элементов управления в окне .....   | 251 |

## **Глава 9 Меню и панели инструментов ..... 253**

|                                                            |            |
|------------------------------------------------------------|------------|
| <b>Меню интерфейса многих документов (MDI) .....</b>       | <b>256</b> |
| Сливающиеся меню .....                                     | 256        |
| Планирование и выполнение меню .....                       | 257        |
| <b>Вызов редактора меню .....</b>                          | <b>259</b> |
| <b>Создание меню вашего приложения .....</b>               | <b>261</b> |
| <b>Другие функции редактора меню .....</b>                 | <b>264</b> |
| Реализация стандартного поведения Windows .....            | 264        |
| Положения меню и поведение слияния .....                   | 265        |
| Добавление горячих клавиш .....                            | 266        |
| Другие поведения меню - блокирование и переключение .....  | 267        |
| Управление вашими меню .....                               | 268        |
| <b>Панели инструментов .....</b>                           | <b>269</b> |
| Слияние панелей инструментов .....                         | 269        |
| Кнопки .....                                               | 270        |
| Кнопки с фиксацией .....                                   | 271        |
| Взаимоисключающие кнопки .....                             | 272        |
| Предварительный просмотр меню и панелей инструментов ..... | 274        |

## **Глава 10 Элементы управления и их свойства ..... 275**

|                                                  |            |
|--------------------------------------------------|------------|
| <b>Обзор .....</b>                               | <b>277</b> |
| Типы элементов управления .....                  | 277        |
| <b>Общие атрибуты элементов управления .....</b> | <b>280</b> |
| Установка атрибута USE .....                     | 280        |
| Установка атрибута AT .....                      | 283        |
| Установка атрибута TEXT .....                    | 284        |
| Редактор шаблона изображения .....               | 285        |
| Установка атрибута цвета .....                   | 291        |
| Установка атрибута KEY (клавиша) .....           | 293        |
| Установка атрибута ALRT .....                    | 294        |
| Установка атрибута FONT .....                    | 295        |
| Установка режимов элементов управления .....     | 296        |
| Установка атрибутов помощи .....                 | 297        |
| <b>Интерактивные элементы управления .....</b>   | <b>300</b> |
| Свойства кнопки (Button) .....                   | 300        |

|                                                                           |                   |
|---------------------------------------------------------------------------|-------------------|
| Свойства радиокнопки (Radio Button) .....                                 | 305               |
| Свойства поля флажков (Check Box) .....                                   | 311               |
| Создание полей списков (Creating List Boxes) .....                        | 317               |
| Свойства списка (List Properties) .....                                   | 318               |
| Свойства комбинированного поля списка (ComboBox) .....                    | 323               |
| Свойства spin-поля .....                                                  | 325               |
| Свойства поля ввода (Entry Box) .....                                     | 329               |
| Свойства текстового поля (Text) .....                                     | 336               |
| Свойства Листа (SHEET) .....                                              | 338               |
| Свойства элемента управления “Закладка” .....                             | 343               |
| <b>Неинтерактивные элементы управления .....</b>                          | <b>346</b>        |
| Свойств строки символов .....                                             | 346               |
| Свойства элемента управления Приглашение (PROMPT) .....                   | 349               |
| Свойства элемента управления Групповое поле (GROUP BOX) .....             | 349               |
| Свойства строки индикатора выполнения (Progress Bar) .....                | 351               |
| Свойства изображения (Image) .....                                        | 353               |
| Свойства элемента управления Область (Region) .....                       | 354               |
| Свойства элемента управления Линия (Line) .....                           | 356               |
| Свойства элемента управления Прямоугольная область (Box) .....            | 357               |
| Свойства элемента управления Эллипс (Ellipse) .....                       | 358               |
| Свойства панели (Panel Properties) .....                                  | 359               |
| <b>Элементы управления OLE .....</b>                                      | <b>361</b>        |
| Обзор .....                                                               | 361               |
| Свойства элемента управления OLE (OLE Control Properties) .....           | 362               |
| Размещение Электронной таблицы/Графика/Документа в вашем приложении ..... | 368               |
| Размещение в каждой записи электронной таблицы/Графика/Документа .....    | 371               |
| <b>Элементы управления OLE с OCX (OLE Controls with OCXs) .....</b>       | <b>377</b>        |
| Использование OCX с глобальными функциями обратного вызова .....          | 377               |
| Использование OCX с локальными функциями обратного вызова .....           | 383               |
| <b>Элементы управления VBX (VBX Controls) .....</b>                       | <b>387</b>        |
| Свойства VBX .....                                                        | 388               |
| Работа элемента управления VBX .....                                      | 390               |
| <b><u>Глава 11 Форматер поля списка .....</u></b>                         | <b><u>393</u></b> |
| <b>Обзор .....</b>                                                        | <b>395</b>        |
| <b>Работа с форматером поля списка .....</b>                              | <b>398</b>        |
| Диалог Свойства поля списка (List Field Properties) .....                 | 400               |
| Группы столбцов .....                                                     | 405               |
| Создание заголовка над соседними столбцами .....                          | 407               |
| Создание линейки прокрутки под соседними столбцами .....                  | 408               |
| Многострочные записи .....                                                | 410               |
| Улавливание двойного щелчка на поле списка .....                          | 411               |

|                                                                                 |            |
|---------------------------------------------------------------------------------|------------|
| Способность drag and drop (тащить и бросать) .....                              | 411        |
| Поддержка редактирования на месте .....                                         | 414        |
| Идентификация выбранных строк .....                                             | 416        |
| <b>Глава 12 Форматер отчета .....</b>                                           | <b>417</b> |
| <b>Как пользоваться данной главой .....</b>                                     | <b>419</b> |
| <b>Процессор печати Clarion .....</b>                                           | <b>420</b> |
| <b>Интерфейс форматера отчета .....</b>                                         | <b>422</b> |
| Открытие форматера отчета .....                                                 | 422        |
| В виде полос .....                                                              | 422        |
| Панели инструментов .....                                                       | 423        |
| Меню .....                                                                      | 429        |
| <b>Структуры и свойства отчета .....</b>                                        | <b>438</b> |
| Свойства отчета .....                                                           | 438        |
| Форма (Form) .....                                                              | 441        |
| Заголовок страницы .....                                                        | 443        |
| Group Breaks (Групповые прерывания) .....                                       | 445        |
| Групповой Заголовок .....                                                       | 446        |
| Detail (тело отчета) .....                                                      | 449        |
| Групповой Колонтитул (Group Footer) .....                                       | 453        |
| Колонтитул страницы (Page Footer) .....                                         | 455        |
| <b>Возможности форматера отчета .....</b>                                       | <b>457</b> |
| Создание Процедуры отчета .....                                                 | 457        |
| Назначение файлов и ключей (Порядок сортировки) .....                           | 457        |
| Назначение размера и ориентации бумаги .....                                    | 460        |
| Назначение полей отчета .....                                                   | 460        |
| Установка положения и выравнивание .....                                        | 462        |
| Ограниченное перемещение .....                                                  | 463        |
| Назначение формы "предпечатать" .....                                           | 464        |
| Определение страничных заголовков и колонтитулов .....                          | 464        |
| Определение заголовков столбцов и заголовков отчета .....                       | 464        |
| Назначение полей для печати (переменный текст) .....                            | 466        |
| Назначение Групповых прерываний (Group Breaks) .....                            | 467        |
| Назначение поведения страничных прерываний (Page Breaking Behavior) .....       | 470        |
| Создание Итогов и Вычисляемых полей (Totals and Calculated Fields) .....        | 470        |
| Показ номеров страниц (Displaying Page Numbers) .....                           | 473        |
| Показ Дат печати (Displaying Print Dates) .....                                 | 474        |
| Выполнение предварительного просмотра печати (Implementing Print Preview) ..... | 475        |
| Печатание бирок (динамически масштабируемых) (Printing Labels) .....            | 476        |
| Печать одной записи на страницу (Printing One Record per Page) .....            | 481        |
| Печать слитых документов (Printing Mail-Merge Documents) .....                  | 481        |
| Печатание графики .....                                                         | 481        |

|                                                                                        |     |
|----------------------------------------------------------------------------------------|-----|
| Печатание многострочного текста с помощью автоматического перехода на новую строку ... | 486 |
| Отчеты, которые выглядят как окна (Reports That Look Like Windows) .....               | 487 |

## **Глава 13 Текстовый редактор ..... 495**

|                                                              |            |
|--------------------------------------------------------------|------------|
| <b>Открываем текстовый редактор .....</b>                    | <b>497</b> |
| <b>Управление окнами текстового редактора .....</b>          | <b>498</b> |
| <b>Использование инструментов текстового редактора .....</b> | <b>499</b> |
| Меню редактирования .....                                    | 499        |
| Линейка инструментов .....                                   | 502        |
| Поля .....                                                   | 502        |
| Меню поиска .....                                            | 502        |
| Файловое меню .....                                          | 505        |
| Отступ блока .....                                           | 506        |
| <b>Настройка текстового редактора .....</b>                  | <b>506</b> |
| Опции вставки .....                                          | 507        |
| Опции блока .....                                            | 508        |
| Опции цвета .....                                            | 508        |
| Опции сохранения .....                                       | 509        |
| Редактирование ошибок .....                                  | 510        |

## **Глава 14 Редактор формул ..... 511**

|                                              |            |
|----------------------------------------------|------------|
| <b>Компоненты выражения .....</b>            | <b>514</b> |
| <b>Инструменты редактора формул .....</b>    | <b>516</b> |
| Диалог Formulas (Формулы) .....              | 516        |
| Редактор формул .....                        | 516        |
| Кнопки оператора .....                       | 517        |
| <b>Создание выражения присваивания .....</b> | <b>517</b> |
| <b>Условные выражения .....</b>              | <b>519</b> |
| Создание структуры IF .....                  | 520        |
| Создание структуры CASE .....                | 523        |
| Вложенные структуры .....                    | 526        |

## **Глава 15 Система управления проектом ..... 527**

|                                                                             |            |
|-----------------------------------------------------------------------------|------------|
| <b>Меню проекта .....</b>                                                   | <b>530</b> |
| <b>Редактирование файла переадресации .....</b>                             | <b>532</b> |
| <b>Создание файла проекта для приложения, закодированного вручную .....</b> | <b>533</b> |
| <b>Сопровождение проекта .....</b>                                          | <b>535</b> |
| Пиктограмма приложения .....                                                | 535        |
| Добавление файлов исходной программы .....                                  | 536        |
| Добавление ресурсов, объектных файлов и библиотек .....                     | 537        |
| Добавление внешних ресурсов .....                                           | 538        |



|                                                         |            |
|---------------------------------------------------------|------------|
| Добавление других проектов .....                        | 539        |
| Добавление исполняемых программ .....                   | 539        |
| <b>Распределение файлов .....</b>                       | <b>540</b> |
| Выбор конфигурации .....                                | 540        |
| Инсталлирование и доступ к DLLs вашего приложения ..... | 542        |
| <b>Целевой файл .....</b>                               | <b>542</b> |
| Файлы .LIB .....                                        | 543        |
| Файлы .DLL .....                                        | 544        |
| Файл списка загрузки .....                              | 545        |
| Глобальные опции компиляции и компоновки .....          | 545        |
| Опции компилирования отдельных исходных модулей .....   | 550        |
| <b>Глава 16 Отладчики .....</b>                         | <b>553</b> |
| <b>Процесс отладки .....</b>                            | <b>556</b> |
| <b>Подготовка ваших проектов для отладки .....</b>      | <b>557</b> |
| <b>16-битовый отладчик .....</b>                        | <b>559</b> |
| Запуск отладчика .....                                  | 559        |
| Загрузка исходных файлов .....                          | 560        |
| Установка параметров отладки .....                      | 561        |
| Окна отладчика .....                                    | 565        |
| Установка точек прерывания .....                        | 570        |
| Выполнение программы .....                              | 574        |
| Работа с исходной программой .....                      | 575        |
| Редактирование выражений наблюдения .....               | 576        |
| Редактирование переменных во время прогона .....        | 578        |
| <b>32-битовый отладчик .....</b>                        | <b>579</b> |
| Запуск отладчика .....                                  | 579        |
| Загрузка исходных файлов .....                          | 580        |
| Установка опций отладчика .....                         | 581        |
| Назначение точек прерывания .....                       | 586        |
| Исполнение программы .....                              | 587        |
| Редактирование переменных во время работы .....         | 589        |
| <b>Глава 17 Менеджер базы данных .....</b>              | <b>591</b> |
| <b>Обзор .....</b>                                      | <b>593</b> |
| <b>Просмотр файлов данных .....</b>                     | <b>594</b> |
| <b>Заккрытие файла данных .....</b>                     | <b>596</b> |
| <b>Порядок сортировки .....</b>                         | <b>596</b> |
| <b>Статистика файла .....</b>                           | <b>597</b> |
| <b>Работа со столбцами .....</b>                        | <b>598</b> |
| Соккрытие столбцов .....                                | 598        |

|                                                                                          |                   |
|------------------------------------------------------------------------------------------|-------------------|
| Показ столбцов .....                                                                     | 598               |
| Использование команды <i>Reformat</i> .....                                              | 598               |
| Выравнивание столбца .....                                                               | 599               |
| Ширина столбца .....                                                                     | 599               |
| Изменение шаблона показа столбца .....                                                   | 600               |
| Заголовки столбцов .....                                                                 | 600               |
| <b>Работа с файлами данных .....</b>                                                     | <b>601</b>        |
| Движение по файлу .....                                                                  | 601               |
| Команда <i>Locate (Key)</i> .....                                                        | 602               |
| Поиск и нахождение следующего .....                                                      | 602               |
| Посылка строки символов драйверу .....                                                   | 603               |
| Сохранение определения файла в виде текста .....                                         | 604               |
| <b>Использование запроса-по-образцу .....</b>                                            | <b>604</b>        |
| <b>Редактирование данных .....</b>                                                       | <b>605</b>        |
| Редактирование записей .....                                                             | 605               |
| Менеджер базы данных дает вам возможность вводить новые записи в файл. ....              | 606               |
| Редактирование мемо-полей .....                                                          | 606               |
| Показ удаленных записей .....                                                            | 607               |
| Возвращение удаленных записей .....                                                      | 607               |
| Удержание и освобождение записей .....                                                   | 608               |
| <b>Преобразование файла данных .....</b>                                                 | <b>608</b>        |
| Немедленное преобразование .....                                                         | 608               |
| Генерирование исходной программы для преобразования файла .....                          | 609               |
| <b>Редактирование исходной программы для выполнения присваиваний<br/>    полей .....</b> | <b>611</b>        |
| <b>Печать данных .....</b>                                                               | <b>616</b>        |
| <b><u>Приложение А Вопросы конструирования Окон .....</u></b>                            | <b><u>617</u></b> |
| Обзор .....                                                                              | 619               |
| Принципы конструирования .....                                                           | 619               |
| Управление, осуществляемое пользователем .....                                           | 620               |
| Программирование, управляемое событиями .....                                            | 621               |
| Фоновая обработка .....                                                                  | 622               |
| Windows и элементы окна .....                                                            | 623               |
| Окно приложения .....                                                                    | 623               |
| MDI .....                                                                                | 624               |
| Диалоговые поля .....                                                                    | 624               |
| Кнопки .....                                                                             | 625               |
| Поля флажков .....                                                                       | 626               |
| Радиокнопки .....                                                                        | 626               |
| Поля списка .....                                                                        | 626               |

|                                                        |                   |
|--------------------------------------------------------|-------------------|
| Комбинированные поля .....                             | 626               |
| Выпадающие списковые поля .....                        | 626               |
| Текстовые поля .....                                   | 627               |
| Spin-Поля .....                                        | 627               |
| Статическое поле .....                                 | 628               |
| Групповые поля .....                                   | 628               |
| Листы и закладки .....                                 | 628               |
| Мастера .....                                          | 628               |
| Метки элементов управления .....                       | 629               |
| Курсоры .....                                          | 630               |
| <b>Меню .....</b>                                      | <b>630</b>        |
| Меню файлов .....                                      | 631               |
| Меню редактирования Edit .....                         | 632               |
| Меню помощи .....                                      | 633               |
| Меню просмотра .....                                   | 634               |
| Меню окна .....                                        | 634               |
| Ускоряющие клавиши .....                               | 634               |
| <b>Цвет .....</b>                                      | <b>635</b>        |
| <b><u>Приложение Б Драйверы базы данных .....</u></b>  | <b><u>637</u></b> |
| <b>Файлы ASCII .....</b>                               | <b>639</b>        |
| Поддерживаемые типы данных .....                       | 639               |
| Характеристики файла/максимумы .....                   | 639               |
| Разное .....                                           | 640               |
| Строки символов драйвера и функции SEND .....          | 640               |
| Поддерживаемые атрибуты файла, команды и функции ..... | 642               |
| <b>Basic файлы .....</b>                               | <b>645</b>        |
| Поддерживаемые типы данных .....                       | 645               |
| Характеристики файла/максимумы .....                   | 646               |
| Строки символов драйвера и функции SEND .....          | 646               |
| Разное .....                                           | 649               |
| Поддерживаемые атрибуты файла, команды и функции ..... | 649               |
| <b>Файлы Btrieve .....</b>                             | <b>652</b>        |
| Типы данных .....                                      | 653               |
| Характеристики файла/максимумы: .....                  | 655               |
| Строки символов драйвера и функции SEND .....          | 656               |
| Поддерживаемые атрибуты файла, команды и функции ..... | 659               |
| Замечания .....                                        | 662               |
| Разное .....                                           | 663               |
| <b>Файлы Clarion .....</b>                             | <b>666</b>        |
| Типы данных .....                                      | 666               |
| Максимальные характеристики файла .....                | 666               |

|                                                              |            |
|--------------------------------------------------------------|------------|
| Строки символов драйвера и функции SEND .....                | 667        |
| Разное .....                                                 | 668        |
| Поддерживаемые атрибуты файла, команды и функции .....       | 668        |
| Замечания .....                                              | 671        |
| <b>Файлы Clipper.....</b>                                    | <b>671</b> |
| Типы данных .....                                            | 672        |
| Характеристики файла/максимумы .....                         | 673        |
| Строки символов драйвера и функции SEND .....                | 674        |
| Поддерживаемые атрибуты файла, команды и функции .....       | 675        |
| Замечания .....                                              | 678        |
| Разное .....                                                 | 680        |
| <b>Файлы dBase III.....</b>                                  | <b>683</b> |
| Типы данных .....                                            | 683        |
| Характеристики файла/максимумы .....                         | 685        |
| Строки символов драйвера и функции SEND .....                | 685        |
| Поддерживаемые атрибуты файла, команды и функции .....       | 686        |
| Замечания .....                                              | 689        |
| Разное .....                                                 | 691        |
| <b>Файлы dBase IV .....</b>                                  | <b>694</b> |
| Типы данных .....                                            | 695        |
| Характеристики файла/максимумы .....                         | 696        |
| Строки символов драйвера и функции SEND .....                | 696        |
| Поддерживаемые атрибуты файла, команды и функции .....       | 698        |
| Замечания .....                                              | 700        |
| Разное .....                                                 | 703        |
| <b>Файлы DOS .....</b>                                       | <b>707</b> |
| Типы данных .....                                            | 707        |
| Характеристики файла/максимумы .....                         | 708        |
| Строки символов драйвера и функции SEND .....                | 708        |
| Разное .....                                                 | 709        |
| Поддерживаемые атрибуты файла, команды и функции .....       | 709        |
| Файлы FoxPro и FoxBased .....                                | 712        |
| Типы данных .....                                            | 712        |
| Характеристики файла/максимумы .....                         | 713        |
| Строки символов драйвера и функции SEND .....                | 714        |
| Поддерживаемые атрибуты файла, команды и функции .....       | 715        |
| Замечания .....                                              | 718        |
| Разное .....                                                 | 720        |
| <b>ODBC- Открытый интерфейс базы данных .....</b>            | <b>724</b> |
| ODBC, за и против .....                                      | 725        |
| Как работает ODBC .....                                      | 726        |
| Добавление поддержки ODBC к вашему приложению - Основы ..... | 727        |

|                                                                                    |                   |
|------------------------------------------------------------------------------------|-------------------|
| <i>Использование SQL-вставок .....</i>                                             | <i>729</i>        |
| <i>Типы данных .....</i>                                                           | <i>730</i>        |
| <i>Посылка оператора SQL .....</i>                                                 | <i>732</i>        |
| <i>Посылка строки драйвера /WHERE .....</i>                                        | <i>732</i>        |
| <i>Проверка вашего ODBC приложения .....</i>                                       | <i>733</i>        |
| <i>Поддерживаемые атрибуты файла, команды и функции .....</i>                      | <i>734</i>        |
| <i>Замечания .....</i>                                                             | <i>737</i>        |
| <i>Разное .....</i>                                                                | <i>737</i>        |
| <i>MS доступ и ODBC .....</i>                                                      | <i>739</i>        |
| <b><i>Файлы TopSpeed .....</i></b>                                                 | <b><i>739</i></b> |
| <i>Типы данных .....</i>                                                           | <i>740</i>        |
| <i>Характеристики файла/максимумы .....</i>                                        | <i>740</i>        |
| <i>Строки символов драйвера и функции SEND .....</i>                               | <i>740</i>        |
| <i>Поддерживаемые атрибуты файла, команды и функции .....</i>                      | <i>742</i>        |
| <i>Замечания .....</i>                                                             | <i>745</i>        |
| <i>Разное .....</i>                                                                | <i>745</i>        |
| <i>Сохранение многих таблиц (файлов данных) в одном файле TPS .....</i>            | <i>748</i>        |
| <b><i>Утилита восстановления базы данных TopSpeed .....</i></b>                    | <b><i>749</i></b> |
| <i>Использование утилиты восстановления в интерактивном режиме .....</i>           | <i>749</i>        |
| <i>Параметры командной строки .....</i>                                            | <i>751</i>        |
| <i>Использование утилиты в вашем приложении .....</i>                              | <i>752</i>        |
| <i>Работа с утилитой восстановления базы данных TopSpeed .....</i>                 | <i>752</i>        |
| <b><i>Утилита копирования базы данных TopSpeed .....</i></b>                       | <b><i>753</i></b> |
| <i>Пример использования .....</i>                                                  | <i>753</i>        |
| <i>Рассмотрение совместного использования файлов .....</i>                         | <i>754</i>        |
| <i>Интерфейс утилиты копирования .....</i>                                         | <i>754</i>        |
| <i>Параметры командной строки утилиты копирования .....</i>                        | <i>755</i>        |
| <b><u>Приложение В Стратегии разработки и развертывания - EXE, .LIB и .DLL</u></b> | <b><u>757</u></b> |
| <b><i>Обзор .....</i></b>                                                          | <b><i>759</i></b> |
| <i>Разработка с участием многих программистов .....</i>                            | <i>760</i>        |
| <i>Приведение в действие и организация командных проектов .....</i>                | <i>761</i>        |
| <i>Процедурно-ориентированный подход .....</i>                                     | <i>762</i>        |
| <i>Модульно-ориентированный подход .....</i>                                       | <i>764</i>        |
| <i>Суб-приложение .....</i>                                                        | <i>766</i>        |
| <b><i>.EXE, состоящий из одного куска .....</i></b>                                | <b><i>771</i></b> |
| <i>Когда осуществлять .....</i>                                                    | <i>771</i>        |
| <b><i>.EXE плюс .DLLs (.EXE Plus .DLLs) .....</i></b>                              | <b><i>776</i></b> |
| <i>Когда осуществлять .....</i>                                                    | <i>776</i>        |
| <i>Как осуществлять .....</i>                                                      | <i>776</i>        |
| <i>Размещение библиотеки поддержки Clarion for Windows .....</i>                   | <i>780</i>        |

**Приложение Д DDE - Динамический обмен данными ..... 785*****Возможности ..... 787******Запуск DDE “Диалога” - клиента с сервером ..... 789****Инициализация “Диалога” ..... 789**Посылка команд DDE ..... 790**Отсылка данных от клиента к серверу ..... 792***Приложение Е Изготовление вызовов API ..... 793*****Создание прототипов API функций ..... 795******Компоновка API функций ..... 796****API функции Windows ..... 796**Другие API функции ..... 797**Функция вызова CALL ..... 799*