

SMILE VISUAL CTI

Руководство пользователя

Оглавление

1. Описание среды разработки.....	4
1.1. Элементы графического интерфейса.....	4
1.2. Создание кубиков и построение связей.....	6
1.3. Настройка параметров кубиков.....	7
1.4. Пример простейшего алгоритма.....	8
1.5. Разработка и отладка алгоритмов.....	9
1.6. Переменные и константы.....	10
1.7. Список системных переменных и констант.....	10
Список переменных:.....	10
Список констант:.....	12
2. Работа с базами данных.....	13
2.1. Соединение с СУБД.....	13
2.2. Построение источника ODBC.....	14
3. Выбор драйвера устройства.....	16
4. Описание кубиков.....	17
4.1. Начало алгоритма.	17
Свойства.....	17
4.2. Завершение алгоритма.....	18
Свойства.....	18
4.3. Ветвление алгоритма.....	19
Свойства.....	19
4.4. Ожидание звонка.....	20
Свойства.....	20
Результат исполнения.....	21
4.5. Набор номера.....	22
Свойства.....	22
Результат исполнения.....	23
4.6. Прием цифр.....	24
Свойства.....	24
Результат исполнения.....	25
4.7. Воспроизведение.....	26
Свойства.....	26
Результат исполнения.....	27
Синтез речи.....	28
4.8. Запись.....	29
Свойства.....	29
Результат исполнения.....	30
4.9. Передача факса.....	31
Свойства.....	31
Результат исполнения.....	33
4.10. Прием факса.....	34
Свойства.....	34
Результат исполнения.....	35

4.11. Коммутатор.....	36
Свойства.....	36
Механизм работы кубика «Коммутатор».....	38
Результат исполнения.....	38
4.12. Операции с переменными.....	39
Свойства.....	39
Функции.....	40
Математические функции.....	40
Функции работы со строками.....	40
Функции работы с текстовыми переменными.....	41
4.13. Выборка из базы данных.....	42
Свойства.....	42
Результат исполнения.....	43
4.14. Перемещение по выборке.....	44
Свойства.....	44
Результат исполнения.....	44
4.15. Запись в базу данных.....	45
Свойства.....	45
Результат исполнения.....	46
4.16. Удаление строк из таблицы.....	47
Свойства.....	47
Результат исполнения.....	47
4.17. Добавление строк в таблицу.....	48
Свойства.....	48
Результат исполнения.....	48
4.18. Запись в текстовый файл.....	49
Свойства.....	49
4.19. Отправка E-mail.....	50
Свойства.....	50
Результат исполнения.....	52
4.20. Вызов подпрограммы.....	53
Свойства.....	53
4.21. Операции с файлами.....	55
Свойства.....	55
4.22. Старт приложения.....	56
Свойства.....	56
4.23. Запрос Интернет-ресурса.....	57
Свойства.....	57
Результат исполнения.....	60
4.24. Запрос к серверу RADIUS.....	61
Свойства.....	61
Результат исполнения.....	63
<i>Приложение A. Phone Emulator.....</i>	<i>64</i>
Эмулятор факс-аппарата.....	65
Эмулятор удаленного телефонного аппарата.....	65

1. Описание среды разработки.

1.1. Элементы графического интерфейса.

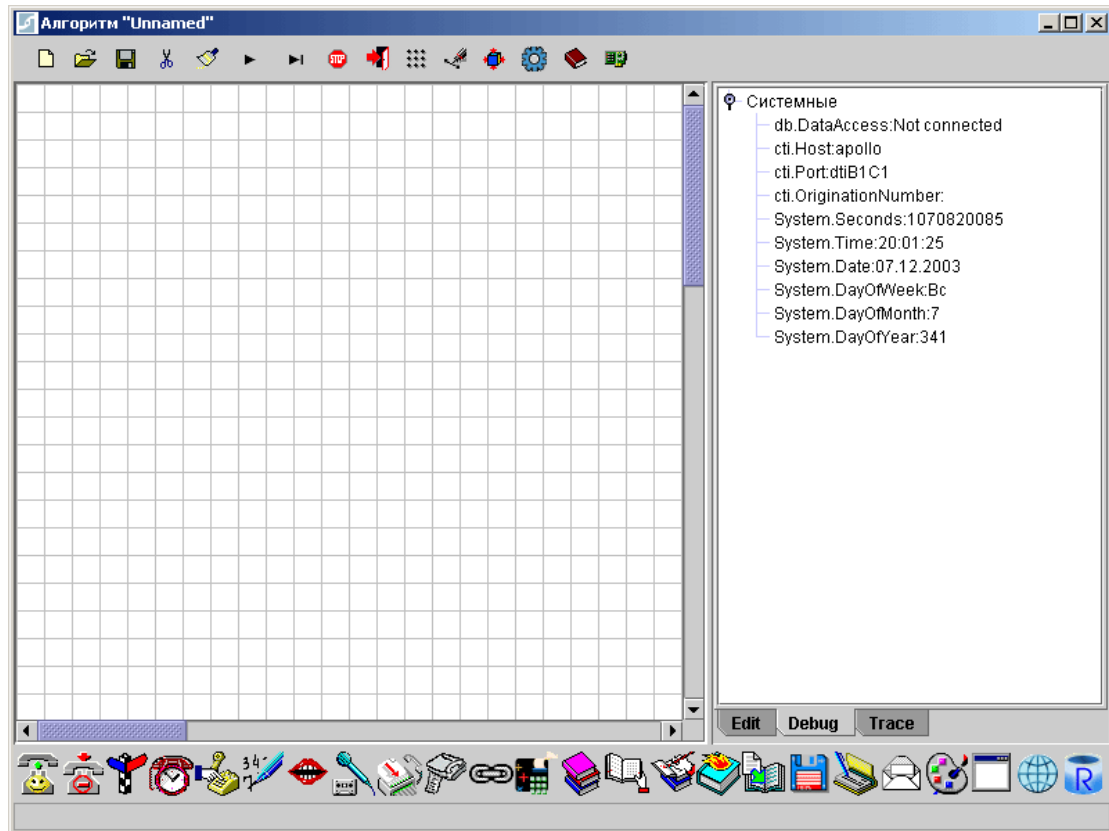

















Рис. 1.1. Окно графического интерфейса *Smile Visual CTI*.

Большая панель в центре окна – это *рабочий стол*, на котором производится сборка алгоритма. Над рабочим столом расположена панель пиктограмм элементов управления:

	Создать новый алгоритм
	Загрузить алгоритм с диска
	Сохранить алгоритм в файл
	Удалить/ вырезать группу кубиков
	Удалить выбранный кубик
	Начать выполнение алгоритма
	Пошаговое выполнение алгоритма
	Остановить выполнение алгоритма
	Выход из <i>Smile Visual CTI</i>
	Установить/убрать разметочную сетку на рабочем столе
	Добавить связь между кубиками
	Установить активный кубик в центр рабочего стола

	Настройка общих параметров
	Настройка параметров соединения с СУБД
	Настройка параметров драйвера CTI

Прямоугольная панель с тремя закладками, расположенная в правой части окна программы, называется **информационной панелью**. В зависимости от выбранной закладки, эта панель предназначена для редактирования параметров кубиков (закладка *Edit*), отображения значений системных и пользовательских (закладка *Debug*) или вывода отладочной информации (закладка *Trace*). В нижней части окна *Smile Visual CTI* расположена **панель кубиков**:

	Начало алгоритма
	Завершение алгоритма
	Ветвление алгоритма
	Ожидание звонка
	Набор номера
	Прием цифр
	Запись звука
	Воспроизведение
	Коммутатор
	Передача факса
	Прием факса
	Операции с переменными
	Выборка из базы данных
	Чтение записи из базы данных
	Запись в базу данных
	Удаление из базы данных
	Добавление записи в базу данных
	Запись в текстовый файл
	Отправка E-mail
	Вызов подпрограммы
	Операции с файлами
	Выполнить приложение
	Операции с Интернет-ресурсами
	Запрос к серверу RADIUS

1.2. Создание кубиков и построение связей.

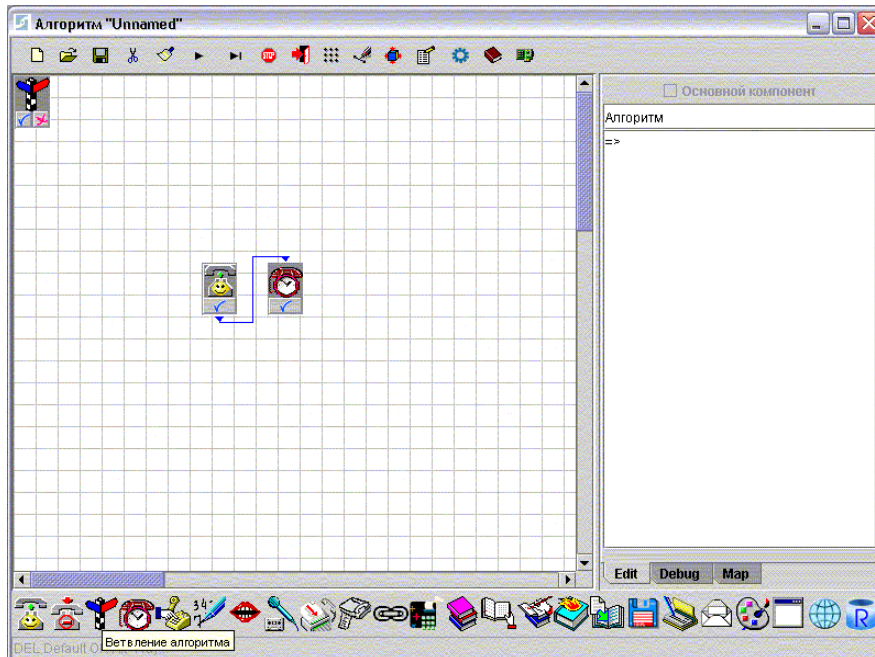




Рис. 1.2. Создание кубиков и размещение их на рабочем столе.

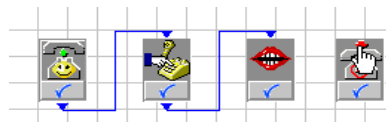
Чтобы создать кубик, нужно щелкнуть левой кнопкой мыши на соответствующей пиктограмме на панели выбора кубиков. Кубик появится в левом верхнем углу рабочего стола. Чтобы поместить появившийся кубик в нужное место рабочего стола, щелкните в этом месте кнопкой мыши. Кубик можно перетаскивать по рабочему столу посредством метода “*drag & drop*”.


Визуальное представление кубика на рабочем столе состоит из иконки, определяющей тип кубика, и области построения связей:

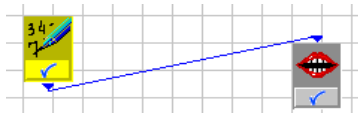
-  область построения связей кубиков;
-  область построения связей кубика «Логическое ветвление».

Кубик, выделенный желтым цветом, называется **активным**. Чтобы сделать кубик активным, нужно сделать двойной щелчок кнопкой мыши на его иконке.


Связи между кубиками определяют последовательность выполнения операций. Связи можно строить двумя способами. Первый способ - щелкнуть левой кнопкой мыши в области построения связей одного кубика, а затем щелкнуть на иконке другого кубика.



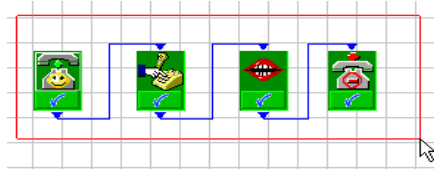
Второй способ - сделать активным кубик, от которого строится связь, щелкнуть левой кнопкой мыши на пиктограмме  панели управления и, затем, щелкнуть на иконке кубика, к которому строится связь.




Чтобы отменить связь, нужно сделать двойной щелчок на области построения связи кубика.

Чтобы удалить кубик с рабочего стола, нужно отметить его как активный и воспользоваться пиктограммой удаления . Связь от этого кубика и все связи к нему будут также удалены.

При построении алгоритма может возникнуть необходимость перемещения или удаления группы кубиков. Для этого нужно «обвести» левой кнопкой мыши нужную группу. Цвет выделенных кубиков станет зеленым.



Выделенная группа кубиков перетаскивается по рабочему столу вместе с их связями. Для удаления выделенной группы используется пиктограмма . Чтобы отменить выделенную группу, обведите любой участок пустого пространства на рабочем столе.

1.3. Настройка параметров кубиков.

Каждый кубик выполняет определенную операцию в процессе работы алгоритма. Эта операция выполняется с учетом заданных параметров кубика. Далее, параметры кубиков мы будем называть *свойствами*. Каждый тип кубика имеет строго определенный набор свойств. Свойства кубиков могут задаваться как постоянные значения (*константы*), или *переменные*, значения которых вычисляются в процессе выполнения алгоритма. Для просмотра и редактирования свойств кубика нужно выбрать закладку *Edit*.

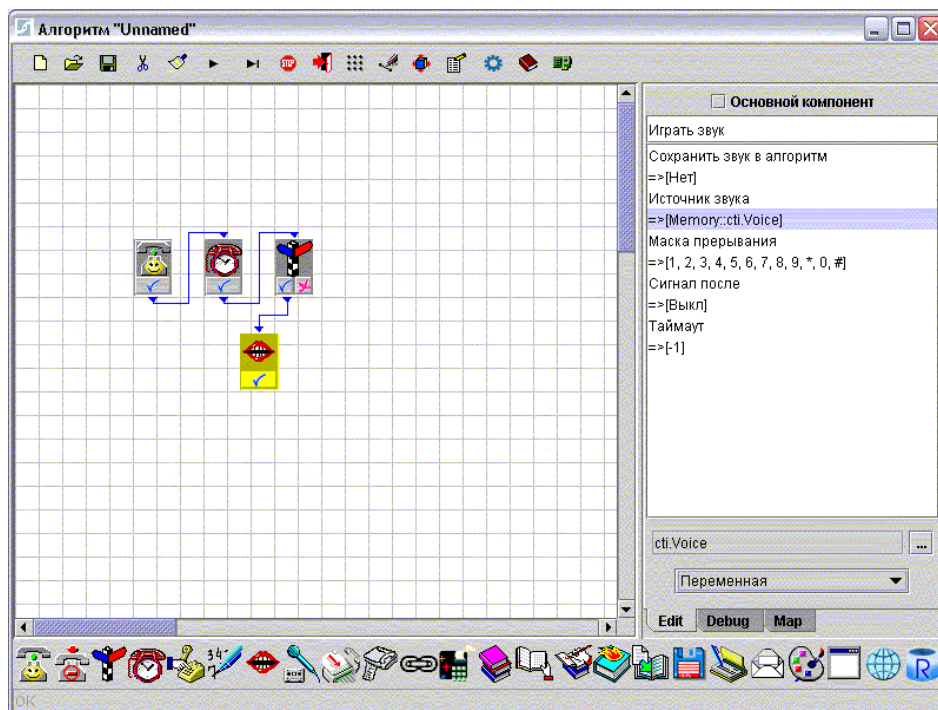
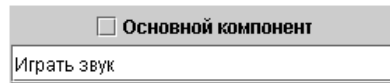


Рис. 1.3. Редактирование свойств кубика.

В верхней части окна редактирования находится выбор «Основной компонент» и текстовое поле комментария. Эти поля определяют два свойства, которые имеются в кубиках всех типов.



В процессе отладки алгоритма, любой из кубиков (но только один!) может быть помечен, как «Основной компонент». С этого кубика будет стартовать алгоритм после нажатия кнопки ▶ панели управления. Как правило, таким кубиком должен быть кубик «Начало алгоритма», но при тестировании можно выбрать любой из кубиков. Свойство «Комментарий» предназначено для описания кубика и может содержать любой текст.

Свойства кубиков подробно описаны в главе 4 «Описание кубиков».

1.4. Пример простейшего алгоритма.

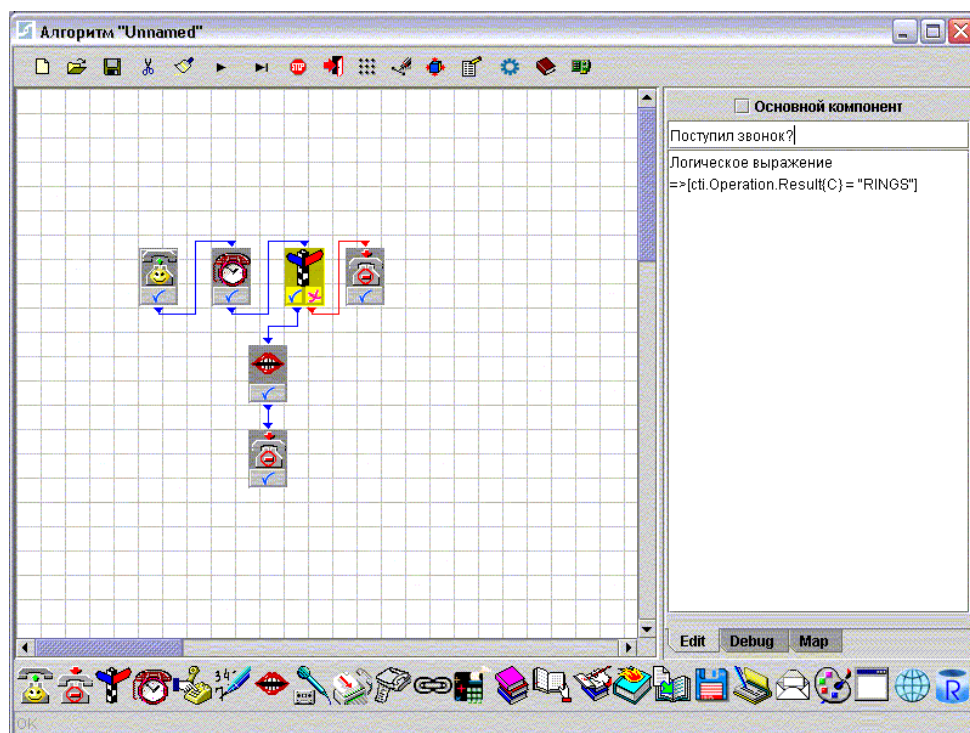



Рис. 1.4. Пример простейшего алгоритма.

На рисунке приведен пример простейшего алгоритма. Алгоритм ожидает звонок в течение заданного промежутка времени и, если звонок поступил, снимает трубку и проигрывает голосовое приветствие.

Первым кубиком, с которого начинается выполнение алгоритма, всегда должен быть кубик «Начало алгоритма». В алгоритме может быть только один такой кубик.

Оканчиваться алгоритм всегда должен кубиком «Завершение алгоритма». Кубиков «Завершение алгоритма» может быть в алгоритме любое количество.

1.5. Разработка и отладка алгоритмов.

Чтобы начать выполнение алгоритма, нужно нажать кнопку  на панели управления. Последовательность выполнения алгоритма можно наблюдать визуально на экране монитора. Выполняющийся в каждый момент времени кубик будет выделен желтым цветом.

Отладчик *Smile Visual CTI* фиксирует состояние всех переменных после выполнения каждого кубика. Данная информация отображается в окне **Debug**.

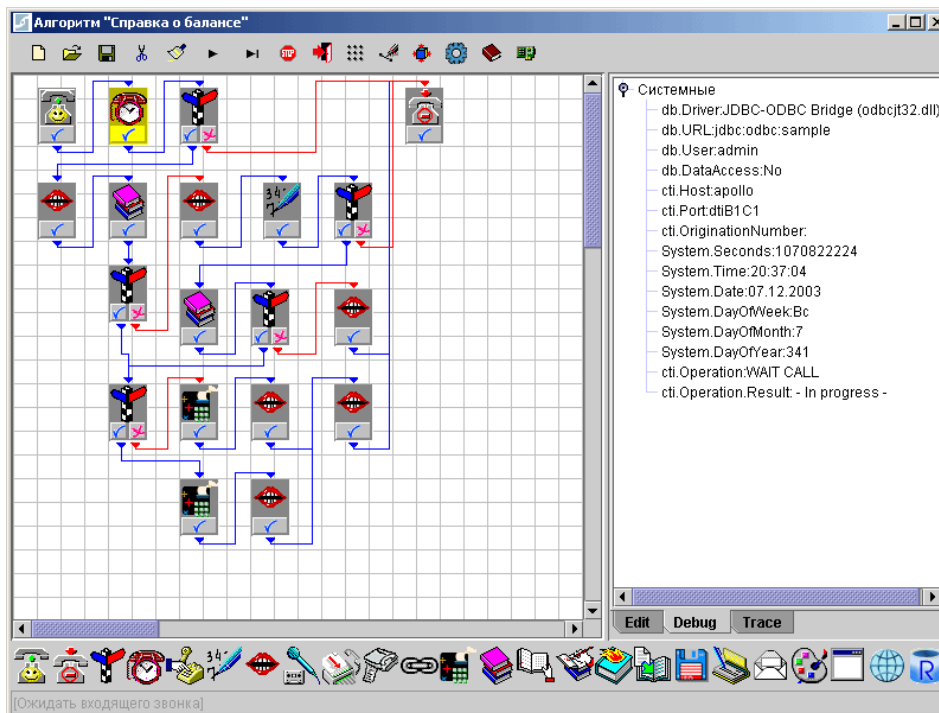





Рис. 1.5. Визуальная отладка алгоритма.

Выполнение алгоритма можно остановить в любой момент, чтобы посмотреть состояние переменных или переключиться в пошаговый режим отладки. Для остановки алгоритма нужно нажать кнопку  панели управления.

В пошаговом режиме за один шаг выполняется один кубик. Чтобы выполнить кубик, нужно выбрать его щелчком кнопки мыши и, затем нажать кнопку  панели управления.

Для анализа хода выполнения алгоритма можно воспользоваться журналом трассировки. Журнал трассировки алгоритма последовательно фиксирует результат выполнения каждого кубика и фиксирует сообщения об ошибках. Для просмотра трассировки, нужно выбрать закладку **Trace**.

Чтобы сохранить алгоритм на диск, нужно нажать кнопку  на панели управления. При первом сохранении алгоритма, на экране появится диалог выбора файла. В результате операции сохранения, на диске создаются файлы с расширениями ".alg" и ".application". Файл с расширением ".alg" - это файл для редактирования в среде *Smile Visual CTI*, а файл с расширением ".application" – для установки алгоритма в *Smile CTI Server*. Имена этих файлов всегда соответствуют имени алгоритма. Имя алгоритма можно изменить в соответствующем свойстве кубика «Начало алгоритма». После изменения имени алгоритма, его необходимо сохранить под новым именем.

1.6. Переменные и константы.

Переменные и константы используются при вычислении логических, арифметических и строковых выражений в процессе выполнения кубиков. Константой может быть любое число или текстовая строка. Существует три вида переменных: *системные переменные, пользовательские переменные и поля объектов баз данных.*

Значения системных переменных устанавливаются в процессе выполнения алгоритма. Список системных переменных приведен в главе 1.6. Пользовательские переменные объявляются в кубиках «**Операции с переменными**». Значения пользовательских переменных устанавливаются в процессе выполнения этих кубиков.

Системные и пользовательские переменные бывают следующих типов:

- Число {N} – число с плавающей или фиксированной точкой
- Строка {C} – строка символов
- Текст {T} – массив строк
- Бинарный {B} – массив байт
- Голос {V} – массив байт с образом звукового файла “WAV”
- Факс {F} – массив байт с образом файла формата “TIFF”

Поля объектов (таблиц и представлений) базы данных добавляются в список переменных при настройке кубика «**Выборка из базы данных**». Поле базы данных может быть числового, строкового или бинарного типа.

Значения системных и пользовательских переменных типа Голос {V} или Факс {F} можно присваивать бинарным {B} полям базы данных и наоборот. С переменными числового типа можно выполнять арифметические операции сложения, вычитания, умножения и деления. Переменные строкового типа можно складывать, в том числе с числовыми переменными и константами.

1.7. Список системных переменных и констант.

Список переменных:

db.Driver{C} – имя класса-драйвера соединения с СУБД (*Строка*);

db.URL{C} – строка соединения с СУБД (*Строка*);

db.User{C} – имя пользователя (login) соединения с СУБД (*Строка*);

db.DataAccess{C} – признак доступности данных после выполнения кубика выборки из БД или кубика перемещения по выборке. Если соединение с БД отсутствует, переменная имеет значение “*Not connected*”. (*Строка*);

db.RowCount{N} – количество записей БД к которым было применено действие. Устанавливается кубиками добавления, удаления и изменения записей БД (*Число*).

cti.Host{C} – имя или ip адрес компьютера(*Строка*);

cti.Port{C} – имя порта, на котором выполняется алгоритм (*Строка*);

cti.Operation{C} – название текущей телефонной операции (*Строка*);

cti.Operation.Result{C} – результат выполнения кубика из группы телефонных операции (*Строка*);

cti.Timestamp {N} – время поступления вызова (или инициализации исходящего вызова). Количество миллисекунд от 01.01.1970. (*Число*);

cti.CallID {C} – глобальный идентификатор вызова. (*Строка*);

cti.CalledNumber {C} – вызываемый номер. Устанавливается кубиком «Ожидание звонка» после поступления звонка (*Строка*);

cti.CallingNumber {C} – вызывающий номер. Устанавливается кубиком «Ожидание звонка» после поступления звонка (*Строка*);

cti.RemoteHost {C} - ip адрес удаленного шлюза. Устанавливается кубиком «Ожидание звонка» после поступления звонка по IP (*Строка*);

cti.Input {C} – принятые цифры тонального набора. Устанавливается после выполнения кубика «Прием цифр» (*Строка*);

cti.BreakDigit {C} – цифра, вызвавшая завершение кубика «Прием цифр» (*Строка*);

cti.Port2 {C} – имя порта, через который скомутировано соединение после выполнения кубика «Коммутатор» (*Строка*);

cti.CallID2 {C} – глобальный идентификатор вызова, выполненного кубиком «Коммутатор» (*Строка*);

cti.CalledNumber2 {C} – вызываемый номер в исходящем вызове кубика «Коммутатор» (*Строка*);

cti.CallingNumber2 {C} – вызывающий номер в исходящем вызове кубика «Коммутатор» (*Строка*);

cti.RemoteHost2 {C} - ip адрес удаленного шлюза, на который был выполнен вызов из кубика «Коммутатор» (*Строка*);

cti.Cause {N} – код завершения звонка. Значение этой переменной устанавливается протоколом связи при получении команды «отбой». Если протокол не поддерживает передачу кода завершения (например, аналоговая телефонная линия), данное значение всегда будет равно 16. Коды завершения соответствуют рекомендации Q.931. (*Число*);

cti.Voice {V} – переменная, в которую сохраняется звук в виде двоичных данных. После исполнения кубика «Запись» звуковые данные сохраняются в эту переменную, вне зависимости от свойства «Сохранить звук в...» (*Голос*);

cti.Fax {F} – переменная, в которую сохраняется факс в виде двоичных данных после исполнения кубика «Прием факса» (*Факс*);

cti.Fax.Pages {N} – количество страниц в факсимильном документе. Значение этой переменной устанавливается при выполнении кубика передачи факса (*Число*);

cti.Fax.TransferredPages {N} – количество переданных/принятых страниц факсимильного документа. (*Число*);

cti.Fax.RemoteID {C} – идентификатор (ID) удаленного факс-аппарата (*Строка*);

cti.File {B} – переменная, в которую можно сохранить массив двоичных данных. (*Массив байт*);

cti.Text{T} – переменная, в которую можно сохранить текст в виде строк, разделенных символом перевода строки. (*Массив строк*);

file.Access {C} – результат выполнения файловой операции. Переменная устанавливается при выполнении кубика «Операции с файлами». (*Строка*);

file.Type {C} – тип файла-источника. Переменная устанавливается при выполнении кубика «Операции с файлами». (*Строка*);

file.Length {N} – размер файла в байтах. Переменная устанавливается при выполнении кубика «Операции с файлами». Если источником является каталог файлов, переменная содержит количество файлов в каталоге. (*Строка*);

file.List {T} – текстовая переменная, предназначенная для сохранения списка файлов из каталога. (*Массив строк*);

file.Date {C} – дата последней модификации файла. Переменная устанавливается при выполнении кубика «Операции с файлами». (*Строка*);

file.Time {C} – время последней модификации файла. Переменная устанавливается при выполнении кубика «Операции с файлами». (*Строка*);

file.Seconds {N} – время последней модификации файла в секундах от 1 января 1970 года. (*Число*);

System.Date {C} – текущая дата. Формат даты по умолчанию соответствует формату, установленному в операционной системе. (*Строка*);

System.Time {C} – текущее время. Формат времени по умолчанию соответствует формату, установленному в операционной системе. (*Строка*);

System.Seconds {N} – текущее время в секундах от 1.01.1970 (*Число*);

System.DayOfWeek {C} – название текущего дня недели. (*Строка*);

System.DayOfMonth {N} – текущий день месяца. (*Число*);

System.DayOfYear {N} – текущий день года. (*Число*);

System.Exit {N} – код завершения внешней программы. (*Строка*);

System.Out {C} – переменная для вывода сообщений. (*Строка*);


Список констант:

TIMEOUT	RINGS
FORMAT	EOF
DTMF	DISCONNECT
CONNECT	SILENCE
COMPATIBILITY	COMMUNICATION ERROR
USER STOP	POLLING
NO POLL	VOICE
FAX	BUSY
NO ANSWER	NO RINGBACK
TERMINATION	NOT AVAILABLE
ERROR	OK
Not found	Not permitted
Target exists	Infinity
Yes	No

2. Работа с базами данных.

2.1. Соединение с СУБД.

Для того, чтобы алгоритм получил доступ к объектам базы данных, необходимо выполнить процедуру соединения с СУБД, описанную ниже.

 Для вызова диалога соединения с СУБД нажмите на эту пиктограмму

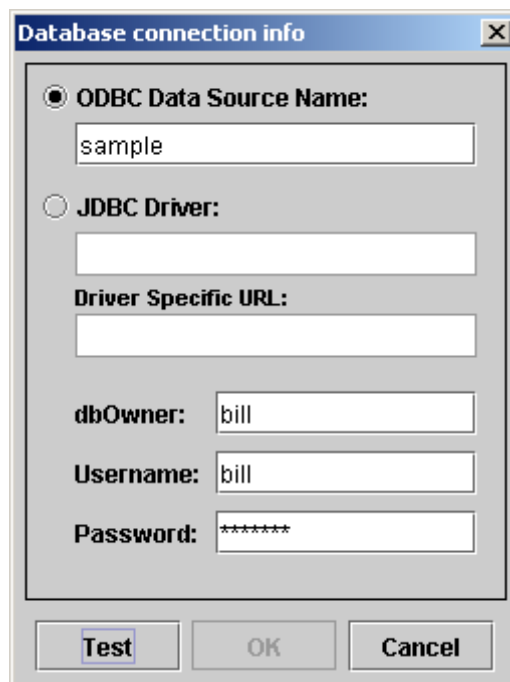


Рис. 2.1. Диалог соединения с СУБД.

Для подключения к СУБД можно использовать механизм **ODBC** (стандартный механизм доступа к базам данных, использующийся в Windows) или специальный драйвер **JDBC**, который обычно поставляется фирмой-производителем СУБД. При работе с источником **ODBC**, нужно указать его имя. При выборе драйвера **JDBC**, нужно указать имя драйвера и строку связи в поле **Driver Specific URL** (имя драйвера и формат строки связи описываются в документации к соответствующему драйверу).

В поле **dbOwner** вводится имя схемы (имя владельца схемы), к объектам которой нужно получить доступ. В поля **Login** и **Password** вводится логин и пароль пользователя БД. Пользователь должен быть или владельцем схемы, указанной в **dbOwner**, или иметь права доступа к объектам этой схемы. Обычно, имя схемы совпадает с логином пользователя.

После заполнения всех полей диалога, необходимо нажать на кнопку **[Test]**. Далее, система попытается соединиться с СУБД и считать информацию об объектах базы данных (таблицы, представления, колонки). Если соединение прошло успешно, кнопка **[Test]** станет недоступной, а кнопка **[Ok]** доступной. Введенные параметры соединения с СУБД запоминаются в алгоритме при его сохранении на диск. При последующей загрузке алгоритма с диска, соединение с СУБД будет производиться автоматически.

2.2. Построение источника ODBC.

Для построения ODBC-источника нужно выполнить следующие действия:

1. В панели управления найдите иконку менеджера ODBC (в Windows2000/XP она находится в средствах Администратора) и запустите его на выполнение.

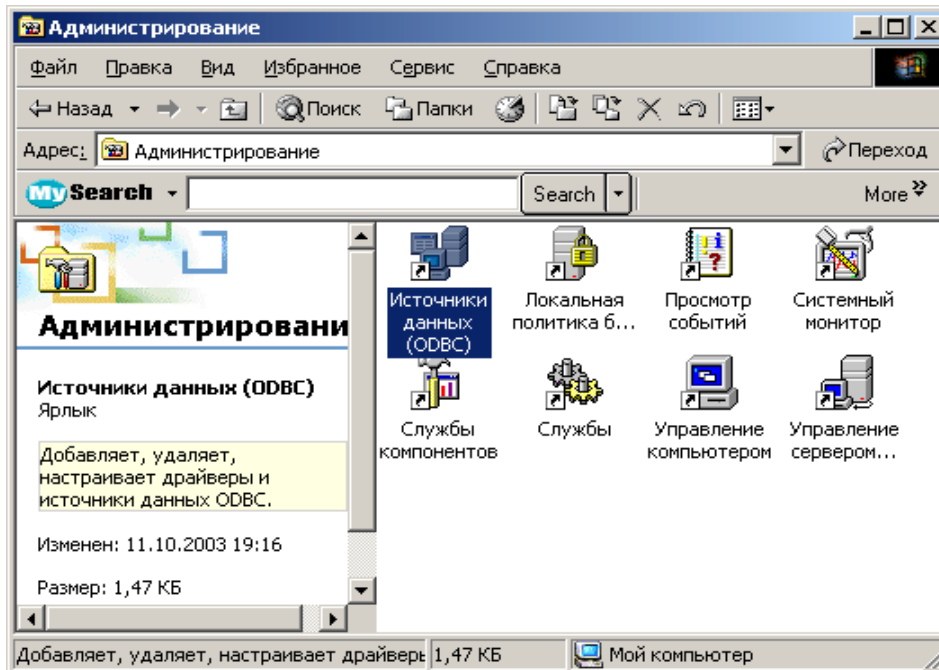


Рис. 2.2 Иконка ODBC-менеджера в панели управления

2. Выберите закладку System DSN и нажмите кнопку [Добавить]:

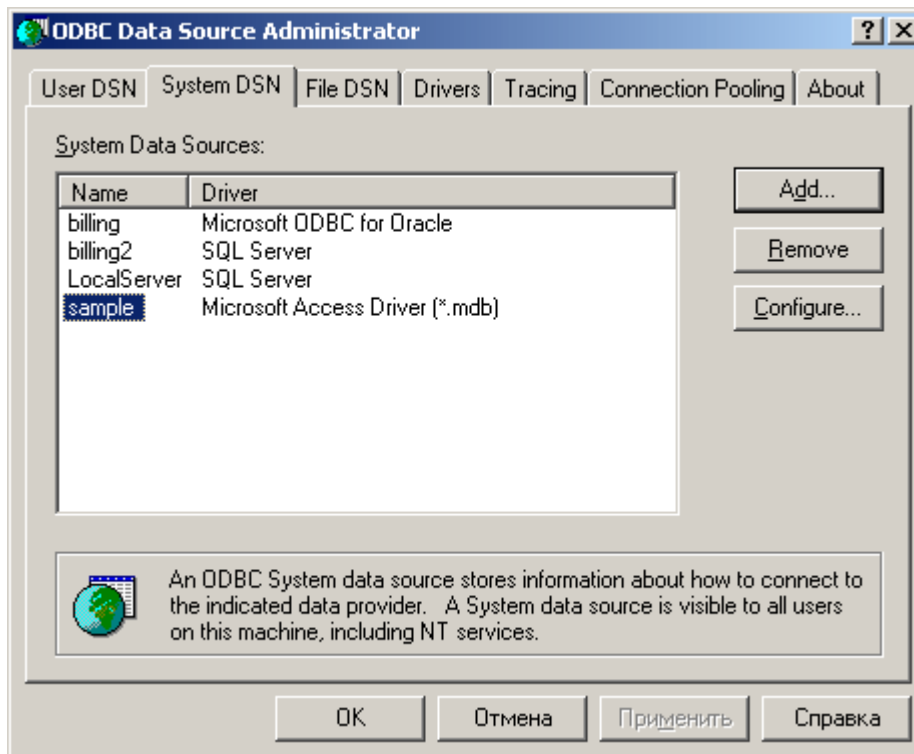


Рис. 2.3 Окно ODBC-менеджера

3. Выберите драйвер и нажмите кнопку [Finish]:

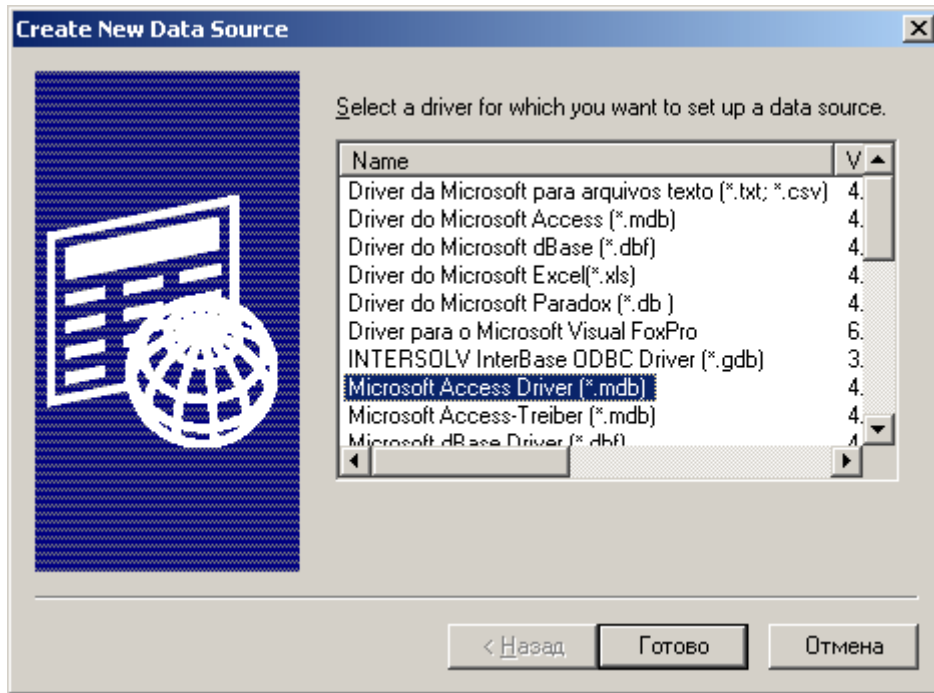


Рис. 2.4 Окно выбора драйвера ODBC

4. Настройте параметры драйвера. Ниже приведена форма настройки драйвера Microsoft Access. Здесь нужно задать имя источника данных, выбрать путь на базу данных (кнопка [Select]). Для установки логина/пароля нажмите кнопку [Advanced].

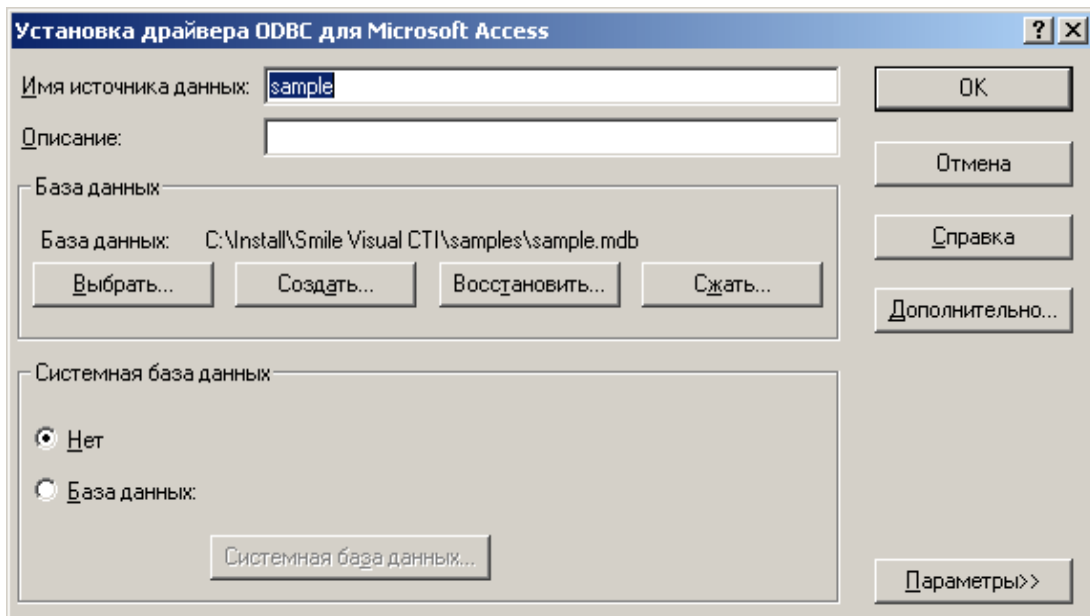


Рис. 2.5 Окно настройки источника ODBC для драйвера MS Access

3. Выбор драйвера устройства.

Программа работает с CTI-оборудованием через *драйвера устройств*. Драйвер устройства обеспечивает адекватное выполнение кубиков группы телефонных операций на данном оборудовании.

 Для вызова диалога выбора и настройки драйвера, нажмите эту кнопку



Рис. 3.1. Диалог выбора драйвера устройства.

Список драйверов включает в себя все доступные драйвера устройств. Выберите драйвер устройства. Если оборудование, обслуживаемое драйвером, присутствует и готово к работе, списки «**Master**» и «**Resource**» будут содержать имена портов. В списке «**Master**» выберите порт, на котором будет выполняться отладка алгоритма. В списке «**Resource**» - порт, который будет использоваться, как ресурс, при выполнении кубика «Коммутатор».

В **Smile Visual CTI** предусмотрена возможность разработки и отладки алгоритмов без использования реального оборудования. Функции порта платы моделируются специальным драйвером-эмулятором. Для записи и воспроизведения звука используется стандартная звуковая карта с микрофоном и колонками. Для установки эмулятора, в списке драйверов выберите строку «**Phone Emulator**». Описание работы с эмулятором приведено в Приложении А.

При выборе драйвера оборудования, в верхнем правом углу появится окно статуса выбранного порта. В окне отображается текущее состояние порта и выполняемая телефонная операция. Кнопка с иконкой отображает статус «трубки» порта. С помощью этой кнопки, можно также «повесить трубку», если она снята (только в состоянии “IDLE”).

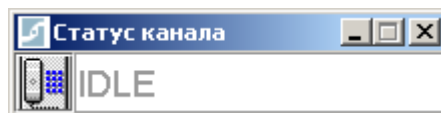


Рис. 3.2 Окно статуса порта

4. Описание кубиков



4.1. Начало алгоритма.

Алгоритм должен начинаться с этого кубика. По умолчанию, в кубике установлено свойство «*Основной компонент*».

Свойства.

Имя алгоритма

<p>Имя Алгоритма =>[Unnamed] Версия =>[1.0] Параметр 1 =>[]</p> <p>Unnamed</p>	<p>Имя алгоритма определяет имя файла, с которым будет сохраняться алгоритм. Файл алгоритма имеет расширение “.alg” . По умолчанию, алгоритм сохраняется под именем “<i>Unnamed.alg</i>” в каталог “.algorithm” . Одновременно с файлом алгоритма, создается файл с расширением “.application” для выполнения в <i>Smile CTI Server</i>.</p>
---	--

Версия

<p>Имя Алгоритма =>[Unnamed] Версия =>[1.0] Параметр 1 =>[]</p> <p>1.0</p>	<p>Информация о номере версии.</p>
---	------------------------------------

Параметры

<p>Имя Алгоритма =>[Unnamed] Версия =>[1.0] Параметр 1 =>[]</p> <p>Добавить Изменить Удалить</p>	<p>Входные параметры алгоритма. Входные параметры могут быть двух типов – строковые {C} и числовые {N}.</p>
---	---



4.2. Завершение алгоритма.

Кубик завершает работу алгоритма.

Свойства.

Повесить трубку

<p>Повесить трубку =>[Да] Значение "Cause" (Q.931) =>[16]</p> <p><input checked="" type="radio"/> Да <input type="radio"/> Нет</p>	<p>Если данное свойство установлено в значение [Да], перед завершением алгоритма выполняется телефонная операция «повесить трубку».</p>
--	--



Код завершения

<p>Повесить трубку =>[Да] Значение "Cause" (Q.931) =>[16]</p> <p>16</p> <p><input checked="" type="radio"/> Const <input type="radio"/> Sys <input type="radio"/> User <input type="radio"/> Db</p>	<p>В данном свойстве можно задать код завершения, который будет передан в линию протоколом связи.</p>
---	---

При отладке алгоритма в среде *Smile Visual CTI*, можно строить связь от этого кубика к любому другому, в том числе и к кубику «Старт алгоритма». Однако, при работе алгоритма в *Smile CTI Server*, эти связи не будут учитываться, и алгоритм завершит работу.

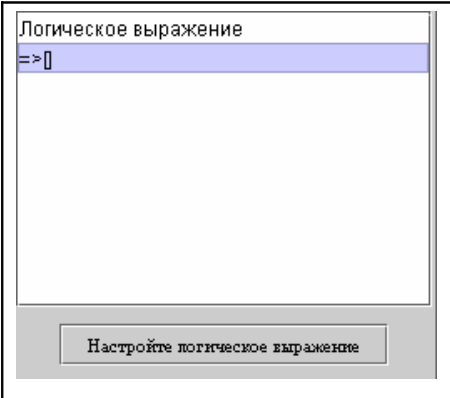
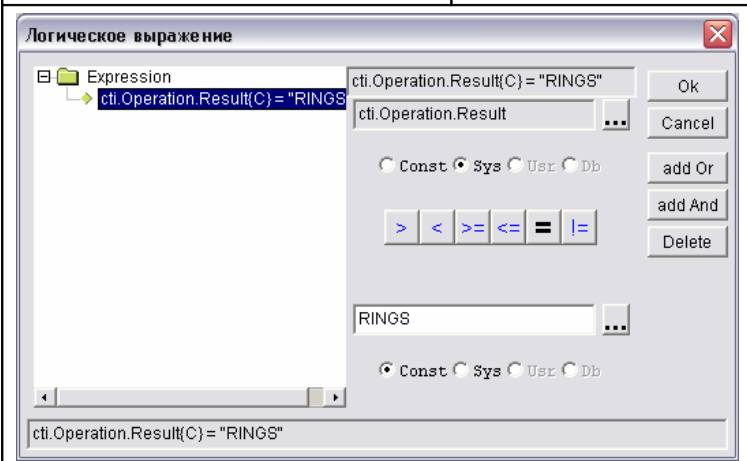
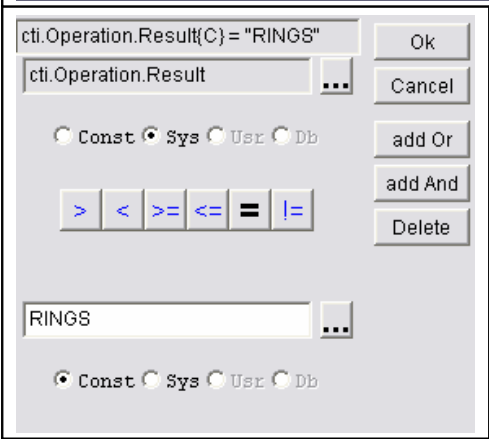


4.3. Ветвление алгоритма.

Кубик выполняет функцию ветвления алгоритма по результату логического выражения. Если логическое выражение истинно, выполняется переход на следующий кубик по связи , в противном случае - по связи .

Свойства.

Логическое выражение

	<p>В окне редактирования выберите левой кнопкой мыши строку логического выражения. В нижней части окна появится кнопка «Настройте логическое выражение». Нажатие на кнопку вызовет на экран диалог построения логического выражения. Логическое выражение может состоять из одного или нескольких <i>логических условий</i>, связанных между собой отношениями «И»/«ИЛИ».</p>
	<p>Кнопка «add And» добавляет дополнительное логическое условие в выражение по связи «И».</p> <p>Кнопка «add Or» добавляет дополнительное условие по связи «ИЛИ».</p> <p>Кнопка «Delete» удаляет выбранное условие из логического выражения.</p>
	<p>Логическое условие строится путем выбора операндов и определения отношения между ними. Операндами могут быть переменные или константы. Для выбора операнда, нужно определить его группу, установив флажок:</p> <p>Const – Константа</p> <p>Sys – Системная переменная</p> <p>User – Пользовательская переменная</p> <p>Db – Поле объекта базы данных</p>
<p>Чтобы построить логическое условие, нужно выбрать операнды и нажать кнопку со знаком нужного отношения. Операнды выбираются из списка. Операнд-константу можно вводить вручную в текстовом поле. Строки сравниваются в лексикографическом порядке. Если операнды разного типа, второй операнд приводится к типу первого.</p>	



4.4. Ожидание звонка.

В зависимости от настройки, кубик «Ожидание звонка» может использоваться для:

- ◆ ожидания входящего вызова («звонка»);
- ◆ установки соединения с вызывающим абонентом («снятия трубки»);
- ◆ отсчета интервала времени.

Свойства.

Время ожидания

<div style="border: 1px solid gray; padding: 5px;"> <p>Время ожидания =>[60]</p> <p>Количество звонков =>[1]</p> <p>Отвечать на вызов =>[Да]</p> <hr/> <p>60</p> <p><input checked="" type="radio"/> Const <input type="radio"/> Sys <input type="radio"/> User <input type="radio"/> Db</p> </div>	<p>Максимальное время, по истечении которого кубик завершает работу. Время ожидания устанавливается в секундах. Значение вводится в текстовом поле редактора свойства.</p>
--	--

Количество звонков

<div style="border: 1px solid gray; padding: 5px;"> <p>Время ожидания =>[60]</p> <p>Количество звонков =>[1]</p> <p>Отвечать на вызов =>[Да]</p> <hr/> <p>1</p> <p><input checked="" type="radio"/> Const <input type="radio"/> Sys <input type="radio"/> User <input type="radio"/> Db</p> </div>	<p>Если значение данного свойства больше нуля, кубик выполняет операцию ожидания входящего вызова («звонка»). При поступлении входящего вызова, отсчитывается заданное число звонков, после чего кубик может установить соединение («снять трубку»). Кубик можно также использовать для отсчета интервалов времени. Для этого, нужно установить «Количество звонков»=>[0].</p>
---	---

Отвечать на вызов

<div style="border: 1px solid gray; padding: 5px;"> <p>Время ожидания =>[60]</p> <p>Количество звонков =>[1]</p> <p>Отвечать на вызов =>[Да]</p> <hr/> <p><input checked="" type="radio"/> Да <input type="radio"/> Нет</p> </div>	<p>При поступлении входящего вызова, если свойство «Отвечать на вызов»=>[Да], кубик устанавливает соединение («снимает трубку»). Снять трубку можно, также, с помощью отдельного кубика «Ожидание звонка». Для этого, установите «Время ожидания»=>[0], «Количество звонков»=>[0], «Отвечать на вызов»=>[Да]</p>
---	--

Результат исполнения.

В результате выполнения кубика «Ожидание», системная переменная **cti.Operation.Result** устанавливается в одно из следующих значений:

- **RINGS** – поступил входящий вызов (звонок);
- **TIMEOUT** – истекло время ожидания;
- **DISCONNECT** – поступил отбой соединения;
- **TERMINATE** – поступил отбой на скоммутированном порту (см. **36**);

При поступлении входящего вызова, автоматически устанавливаются значения системных переменных:

cti.CalledNumber - номер вызываемого абонента

cti.CallingNumber - номер вызывающего абонента.

Если протокол телефонной сигнализации не обеспечивает получение номера вызывающего номера, значение переменной **cti.CallingNumber** будет равно пустой строке.

cti.RemoteHost - ip адрес шлюза (для портов IP телефонии)

Если «Количество звонков» > 0, а соединение уже было установлено, кубик завершается с результатом **RINGS**.



4.5. Набор номера.

Данный кубик может выполнять одну из двух операций – исходящий звонок или генерацию сигналов DTMF.

Свойства.

Номер

	<p>Номер может быть задан, как строковая константа или переменная. Для набора номера могут использоваться следующие символы:</p> <ul style="list-style-type: none"> • цифры 0 – 9 • символы тонов * # A B C D • флэш - & • ожидание тонального сигнала станции - W • пауза - ,
--	---

Для выполнения исходящего звонка в системах **IP-телефонии**, строка номера может иметь следующий вид: [номер абонента]@[ip-адрес шлюза].

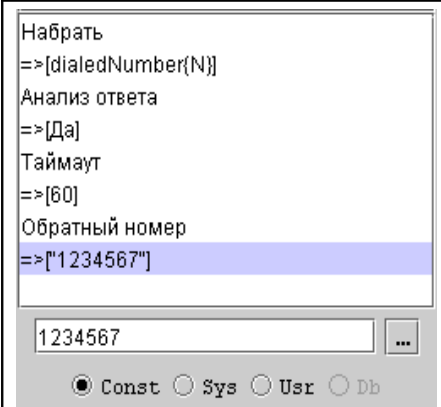
Анализ ответа

	<p>Данное свойство определяет тип операции:</p> <p>«Да» - выполнение исходящего звонка</p> <p>«Нет» - генерация сигналов DTMF.</p> <p>Выполнение операции зависит от текущего состояния порта. Сигнал DTMF генерируется только в случае, если порт находится в состоянии соединения («трубка снята»), а исходящий звонок – только при отсутствии соединения.</p>
--	--

Таймаут

	<p>Данное свойство имеет значение, если только установлено «Анализ ответа»=>[Да]. Таймаут определяет максимальное время ожидания ответа на исходящий вызов.</p>
--	--

Обратный номер

	<p>Номер вызывающего абонента, который передается телефонной сигнализацией при установлении исходящего соединения.</p>
---	--

Результат исполнения.

После выполнения исходящего звонка, в системной переменной **cti.Operation.Result** устанавливается одно из следующих значений:

- **VOICE** – голосовой ответ;
- **FAX** – сигнал факса на линии;
- **BUSY** – номер занят;
- **NO ANSWER** – номер не отвечает;
- **NO RINGBACK** – номер недоступен или сбой сети;
- **NO DIAL TONE** – нет соединения с сетью или шлюзом.

Если значение равно **BUSY**, переменная **cti.Cause** может содержать код завершения (в соответствии с рекомендацией Q.931, Appendix I).



4.6. Прием цифр.

Кубик выполняет функцию ожидания и приема стандартных сигналов тонального набора (0 – 9, *, #) с телефонной линии.

Свойства.

Тоновый сигнал вначале

<p>Тоновый сигнал вначале =>[Вкл] Количество цифр =>[1] Таймаут =>[5] Маска прерывания =>[]</p> <p><input checked="" type="radio"/> Вкл <input type="radio"/> Выкл</p>	<p>Если данное свойство включено (выбрано значение [Вкл]), в начале выполнения кубика в телефонную линию генерируется тональный сигнал.</p>
--	---

Количество цифр

<p>Тоновый сигнал вначале =>[Вкл] Количество цифр =>[1] Таймаут =>[5] Маска прерывания =>[]</p> <p>1</p> <p><input checked="" type="radio"/> Const <input type="radio"/> Sys <input type="radio"/> Usr <input type="radio"/> Db</p>	<p>В данном свойстве задается количество цифр, после приема которых, выполнение кубика завершается. Последовательность принятых цифр помещается в виде строки в системную переменную cti.Input.</p>
---	--

Ждать цифру

<p>Тоновый сигнал вначале =>[Вкл] Количество цифр =>[1] Таймаут =>[5] Маска прерывания =>[]</p> <p>5</p> <p><input checked="" type="radio"/> Const <input type="radio"/> Sys <input type="radio"/> Usr <input type="radio"/> Db</p>	<p>Значение данного свойства определяет максимальное время ожидания ввода каждой цифры (в секундах). Системная переменная cti.Input будет содержать строку цифр, принятых до наступления таймаута или пустую строку (если не было принято ни одной цифры).</p>
---	--

Прерывающие цифры

<p>Тоновый сигнал вначале =>[Вкл]</p> <p>Количество цифр =>[1]</p> <p>Таймаут =>[5]</p> <p>Маска прерывания =>[*,#]</p>	<p>Свойство определяет подмножество цифр, досрочно завершающих выполнение кубика. При приеме одной из этих цифр, исполнение кубика завершается. Переменная cti.Input будет содержать все принятые данным кубиком цифры, исключая прервавшую цифру. Маска прерывания строится установкой-снятием флажка на соответствующих цифрах в редакторе свойства.</p>
<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9 <input checked="" type="checkbox"/> * <input type="checkbox"/> 0 <input checked="" type="checkbox"/> #	

Результат исполнения.

В результате выполнения кубика «Прием цифр», системная переменная **cti.Operation.Result** устанавливается в одно из следующих значений:

- **OK** - принято ожидаемое количество цифр;
- **TIMEOUT** - время ожидания истекло;
- **DTMF** - принята прерывающая цифра;
- **FAX** - принят сигнал факса;
- **DISCONNECT** - поступил отбой соединения;
- **TERMINATE** - поступил отбой на скоммутированном порту (см. 36)

Последовательность принятых цифр (кроме цифр из свойства «Прерывающие цифры») сохраняется в виде строки в системной переменной **cti.Input**. Если ввод закончился прерывающей цифрой, эта цифра сохраняется в системной переменной **cti.BreakDigit**.



4.7. Воспроизведение.

Кубик выполняет воспроизведение звуковой информации в телефонную линию.

Свойства.

Сохранить звук в алгоритм

<p>Сохранить звук в алгоритм =>[Да] Источник звука =>[File:] Маска прерывания =>[1, 2, 3, 4, 5, 6, 7, 8, 9, *, 0, #] Сигнал после =>[Выкл] Таймаут =>[-1]</p> <p><input checked="" type="radio"/> Да <input type="radio"/> Нет</p>	<p>Используя это свойство, можно сохранить выбранный звуковой файл внутри алгоритма. Свойство полезно в случаях, если звуковой файл не должен изменяться. Сохранив файл внутри алгоритма, можно не беспокоиться о том, что этот файл может быть кем-то удален с диска или изменен.</p>
---	--

Источник звука

<p>Сохранить звук в алгоритм =>[Нет] Источник звука =>[Memory:cti.Voice] Маска прерывания =>[1, 2, 3, 4, 5, 6, 7, 8, 9, *, 0, #] Сигнал после =>[Выкл] Таймаут =>[-1]</p> <p>cti.Voice <input type="button" value="..."/></p> <p>Переменная ▼</p>	<p>Источником звука может служить звуковой файл, переменная типа «Голос»{V}, или синтезатор речи. Тип звукового ресурса выбирается из выпадающего списка в нижней части окна редактирования. Чтобы выбрать источник звука, нужно выбрать его тип и нажать кнопку <input type="button" value="..."/>.</p>
--	--

Типы звукового ресурса:

«**Переменная**». Источник звука – двоичные данные из переменной типа «Голос» {V};

«**Файл на диске**». Источник - звуковой файл формата “WAV”;

«**Имя файла в переменной**». Источник - звуковой файл, имя которого находится в строковой переменной;

«**Синтез речи**». Источник - значение числовой или строковой переменной, преобразуемое “на лету” в голосовое сообщение.

Прерывающие цифры

<p>Сохранить звук в алгоритм =>[Нет]</p> <p>Источник звука =>[Memory::cti.Voice]</p> <p>Маска прерывания =>[1, 2, 3, 4, 5, 6, 7, 8, 9, *, 0, #]</p> <p>Сигнал после =>[Выкл]</p> <p>Таймаут =>[-1]</p> <p><input checked="" type="checkbox"/> 1 <input checked="" type="checkbox"/> 2 <input checked="" type="checkbox"/> 3 <input checked="" type="checkbox"/> 4 <input checked="" type="checkbox"/> 5 <input checked="" type="checkbox"/> 6 <input checked="" type="checkbox"/> 7 <input checked="" type="checkbox"/> 8 <input checked="" type="checkbox"/> 9 <input checked="" type="checkbox"/> * <input checked="" type="checkbox"/> 0 <input checked="" type="checkbox"/> #</p>	<p>Свойство определяет набор цифр тонального набора, прием которых прерывает запись. Набор цифр задается установкой-снятием флажков. При приеме одной из установленных цифр, исполнение кубика завершается. Принятая цифра может быть получена с помощью кубика «Принять цифры».</p>
---	--

Тоновый сигнал по окончании

<p>Сохранить звук в алгоритм =>[Нет]</p> <p>Источник звука =>[Memory::cti.Voice]</p> <p>Маска прерывания =>[1, 2, 3, 4, 5, 6, 7, 8, 9, *, 0, #]</p> <p>Сигнал после =>[Выкл]</p> <p>Таймаут =>[-1]</p> <p><input type="radio"/> Вкл <input checked="" type="radio"/> Выкл</p>	<p>Если данное свойство установлено в значение [Вкл], после исполнения кубика генерируется тональный сигнал.</p>
--	--

Таймаут

<p>Сохранить звук в алгоритм =>[Нет]</p> <p>Источник звука =>[Memory::cti.Voice]</p> <p>Маска прерывания =>[1, 2, 3, 4, 5, 6, 7, 8, 9, *, 0, #]</p> <p>Сигнал после =>[Выкл]</p> <p>Таймаут =>[-1]</p> <p>-1</p> <p><input checked="" type="radio"/> Const <input type="radio"/> Sys <input type="radio"/> User <input type="radio"/> Db</p>	<p>Свойство определяет максимальное время воспроизведения в секундах. Если данное свойство имеет значение -1, таймаут на воспроизведение отсутствует.</p>
---	---

Результат исполнения.

В результате исполнения кубика, системная переменная **cti.Operation.Result** устанавливается в одно из следующих значений:

- **EOF** - воспроизведение завершено полностью;
- **TIMEOUT** - время воспроизведения истекло;
- **FORMAT** - неверный формат звукового файла;
- **DTMF** - принята цифра из «Прерывающие цифры»;
- **FAX** - принят сигнал факса;
- **DISCONNECT** - поступил отбой соединения;
- **TERMINATE** - поступил отбой на скоммутированном порту.

Синтез речи.

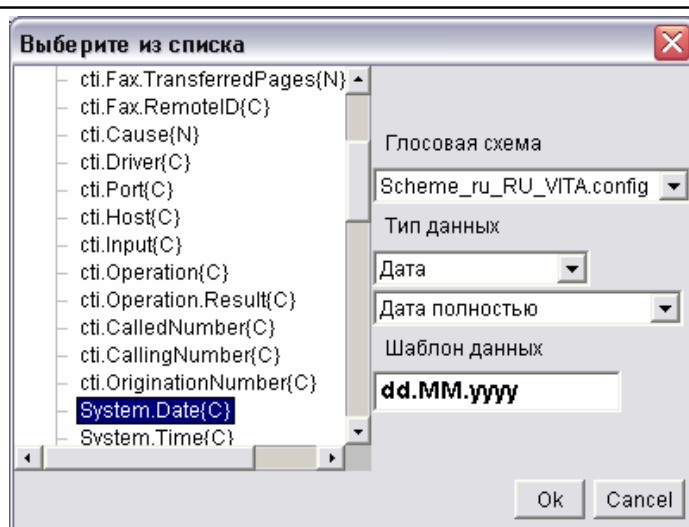
Диалог выбора источника синтеза речи:

Голосовая схема определяет комплект звуковых файлов, из которых синтезируются голосовые сообщения. Стандартный пакет *Smile Visual CTI* поставляется с тремя разновидностями одной русскоязычной голосовой схемы. Эти разновидности отличаются между собой способом проговаривания денежной суммы и имеют следующие названия:

ru_US_VITA – денежная сумма проговаривается в долларах;

ru_RU_VITA – денежная сумма проговаривается в рублях;

ru_UA_VITA – денежная сумма проговаривается в гривнах.



Данные для синтеза речи выбираются из переменных числового или строкового типа. Эти данные могут интерпретироваться как:

Дата;

Время;

Номер телефона;

Число;

Деньги.

Выбор форматов даты:

Дата полностью
Число, месяц
День недели, число месяца
Дата полностью

Строковая переменная должна иметь формат, установленный в поле «Шаблон данных», где:

ММ – месяц (01-12), **dd** – день (01-31), **yyyy** – год.

В качестве разделителя может быть любой символ.

Выбор форматов времени:

Время с секундами
Время кратко
Длительность
Время без секунд

Строковая переменная должна иметь формат, установленный в поле «Шаблон данных», где:

НН – часы (00-23), **mm** – минуты, **ss** – секунды.

В качестве разделителя может быть любой символ.

Номер телефона должен состоять из цифр 1 – 9 и может быть взят из числовой или строковой переменной. Шаблон данных для номера телефона обозначает, какими порциями цифр проговаривать номер. Строка '###-##-##' означает, что последовательно проговариваются числа из трех, двух и двух цифр.

Число должно быть целым и может быть взято из числовой или строковой переменной

Денежная сумма должна иметь формат целого числа или числа с десятичной точкой. Валюта, в которой проговаривается денежная сумма, зависит от выбранной «голосовой схемы».



4.8. Запись.

Кубик выполняет запись звука с телефонной линии.

Свойства.

Сохранить звук в...

Сохранить звук в...

=>[Memory::cti.Voice]

Таймаут

=>[60]

Тишина

=>[10]

Маска прерывания

=>[]

Тоновый сигнал вначале

=>[Вкл]

cti.Voice

Переменная

Свойство определяет, куда сохранять записанный звук. Ресурс-приемник звука можно задать тремя способами:

«**Переменная**». Звук сохраняется в переменную типа «Голос» {V}.

«**Файл на диске**»;

«**Имя файла в переменной**». Имя файла находится в переменной строкового типа.

Для выбора ресурса нужно нажать кнопку .

Таймаут

Сохранить звук в...

=>[Memory::cti.Voice]

Таймаут

=>[60]

Тишина

=>[10]

Маска прерывания

=>[]

Тоновый сигнал вначале

=>[Вкл]

60

Const Sys Usr Db

Свойство определяет максимальное время записи в секундах.

Тишина

Сохранить звук в...

=>[Memory::cti.Voice]

Таймаут

=>[60]

Тишина

=>[10]

Маска прерывания

=>[]

Тоновый сигнал вначале

=>[Вкл]

10

Const Sys Usr Db

Свойство определяет максимальное время отсутствия звука («тишины») в телефонной линии, после которого запись прекращается.

Прерывающие цифры

<p>Сохранить звук в... =>[Memory::cti.Voice] Таймаут =>[60] Тишина =>[10] Маска прерывания =>[]</p> <p><input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9 <input type="checkbox"/> * <input type="checkbox"/> 0 <input type="checkbox"/> #</p>	<p>Свойство определяет набор цифр тонального набора, прием которых прерывает запись. Набор цифр задается установкой-снятием флажков. При приеме одной из установленных цифр, исполнение кубика завершается. Принятая цифра может быть получена с помощью кубика «Принять цифры».</p>
--	--

Тоновый сигнал вначале

<p>Сохранить звук в... =>[Memory::cti.Voice] Таймаут =>[60] Тишина =>[10] Маска прерывания =>[] Тоновый сигнал вначале =>[Вкл]</p> <p><input checked="" type="radio"/> Вкл <input type="radio"/> Выкл</p>	<p>Если свойство установлено в значение [Да], перед началом записи в линию генерируется тоновый сигнал.</p>
--	---

Результат исполнения.

В результате исполнения кубика «Воспроизведение», системная переменная **cti.Operation.Result** устанавливается в одно из следующих значений:

- **TIMEOUT** - время записи истекло;
- **DTMF** - принята цифра из «Прерывающие цифры»;
- **FAX** - принят сигнал факса;
- **SILENCE** - отсутствие звука в линии («тишина»);
- **DISCONNECT** - принят сигнал «отбой» ;
- **TERMINATE** - «отбой» на скоммутированном порту.

Системная переменная **cti.Voice** содержит записанный звук.



4.9. Передача факса.

Свойства.

Сохранить факс в алгоритм

Используя это свойство, можно сохранить выбранный факсимильный файл внутри алгоритма. Свойство полезно в случаях, если файл не должен изменяться. Сохранив файл внутри алгоритма, можно не беспокоиться о том, что этот файл может быть кем-то удален с диска или изменен.

Источник факса

Источник факсимильных данных можно задать следующими способами:

«**Переменная**». Источник – переменная типа «**Факс**» {F}.

«**Файл на диске**».

«**Имя файла в переменной**». Имя файла находится в строковой переменной.

Чтобы выбрать источник факсимильных данных, нужно нажать кнопку .

Вызывать оператора

Если свойство установлено в значение [Да], после завершения передачи факса на удаленный факс-аппарат поступает звуковой сигнал («вызов оператора»).

(для плат “Ольха” не реализовано).

Качество

Сохранить факс в алгоритм
=>[Нет]
Источник факса
=>[Memory::cti.Fax]
Вызывать оператора
=>[Нет]
Качество
=>[Стандартное]
Заголовок факса
=>[""]
Идентификатор факс-аппарата
=>[""]

Повышенное Стандартное

Свойство определяет выбор между передачей факса в стандартном или повышенном качестве.
(только для плат Dialogic)

Заголовок факса

Сохранить факс в алгоритм
=>[Нет]
Источник факса
=>[Memory::cti.Fax]
Вызывать оператора
=>[Нет]
Качество
=>[Стандартное]
Заголовок факса
=>[""]
Идентификатор факс-аппарата

Const Sys Usr Db

Заголовок факса можно задать строкой в текстовом поле ввода или указать переменную строкового типа. Для выбора переменной, нужно выбрать ее группу и нажать кнопку .
(только для плат Dialogic).

Идентификатор факс-аппарата

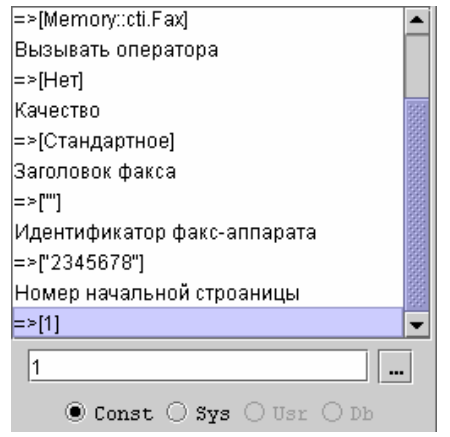

=>[Memory::cti.Fax]
Вызывать оператора
=>[Нет]
Качество
=>[Стандартное]
Заголовок факса
=>[""]
Идентификатор факс-аппарата
=>["2345678"]
Номер начальной строаницы
=>[1]

2345678

Const Sys Usr Db

Идентификатор факса можно задать строкой в текстовом поле ввода или указать переменную строкового типа. Для выбора переменной, нужно выбрать ее группу и нажать кнопку .
(для плат "Ольха" не реализовано).

Номер начальной страницы

	<p>Свойство указывает, с какой страницы начнется передача факсимильного документа. Значение начальной страницы можно задать в виде числовой переменной. Для выбора переменной, нужно выбрать ее группу и нажать кнопку  .</p> <p>Нумерация страниц факса начинается с 1.</p>
---	---

Результат исполнения.

В результате исполнения кубика «Передача факса», системная переменная **cti.Operation.Result** устанавливается в одно из следующих значений:

- **EOF** - факс передан нормально;
- **TIMEOUT** - удаленная сторона не стала принимать факс;
- **USER STOP** - на удаленном факс-аппарате была нажата клавиша «STOP»;
- **DISCONNECT** - обрыв связи.
- **COMMUNICATION ERROR** - ошибка связи в процессе передачи факса;
- **POLLING** - с удаленного факс-аппарата поступил запрос поллинга.
- **COMPATIBILITY** - удаленный факс-аппарат, не может принять данные, с заданным разрешением.
- **FORMAT** - неверный формат файла – факсимильного документа или факсимильные функции не поддерживаются оборудованием;

Другие системные переменные, содержимое которых устанавливается кубиком «Передача факса»:

cti.Fax.Pages - количество страниц в передаваемом факсе;

cti.Fax.TransferredPages - количество реально переданных страниц;

cti.Fax.RemoteID – идентификатор удаленного факс-аппарата.



4.10. Прием факса.

Свойства.

Сохранить факс в

Сохранить факс в
=>[Memory::cti.Fax]
Вызывать оператора
=>[Нет]
Режим опроса
=>[Нет]

cti.Fax ...

Переменная

Свойство определяет, куда сохранять принятый факс. Ресурс-приемник факса можно задать тремя способами:

«**Переменная**». Факс сохраняется в переменную типа «**Факс**» {F}.

«**Файл на диске**»;

«**Имя файла в переменной**». Имя файла находится в строковой переменной.

Для выбора ресурса нужно нажать кнопку .

Вызывать оператора

Сохранить факс в
=>[Memory::cti.Fax]
Вызывать оператора
=>[Нет]
Режим опроса
=>[Нет]

Да Нет

Если свойство установлено в значение «**Да**», после завершения приема факса на удаленный факс-аппарат поступает звуковой сигнал («вызов оператора»).

(только для плат Dialogic).

Режим опроса

Сохранить факс в
=>[Memory::cti.Fax]
Вызывать оператора
=>[Нет]
Режим опроса
=>[Нет]

Да Нет

Если свойство установлено в значение «**Да**», для приема факса используется режим опроса (поллинг). Режим опроса обычно применяется для получения факса от удаленного факс-аппарата в автоматическом режиме.

(только для плат Dialogic).

Результат исполнения.

В результате исполнения кубика «Прием факса», системная переменная **cti.Operation.Result** устанавливается в одно из следующих значений:

- **EOF** - прием факса завершен нормально;
- **DISCONNECT** - обрыв связи.
- **COMMUNICATION ERROR** - ошибка связи в процессе приема факса;
- **USER STOP** - на удаленном факс-аппарате была нажата клавиша «STOP»;
- **NO POLL** - отказ от удаленного факс-аппарата при попытке приема по запросу (поллинга);
- **FORMAT** – факсимильные функции не поддерживаются оборудованием;

Другие системные переменные, содержимое которых устанавливается кубиком «Прием факса»:

cti.Fax - содержит образ принятого факса;

cti.Fax.Pages - количество принятых страниц факса;

cti.Fax.TransferredPages - количество принятых страниц факса;

cti.Fax.RemoteID – идентификатор (ID) удаленного факс-аппарата.



4.11. Коммутатор.

Кубик выполняет операцию «набрать номер и соединить абонента».

Свойства.

Вызываемый номер

	<p>Номер может быть задан, как строковая константа или переменная. Для набора номера могут использоваться следующие символы:</p> <ul style="list-style-type: none"> • 0 – 9, *, # • & - флэш • W - ожидание тонального сигнала станции • , - пауза • ! – удержание активного соединения (HOLD) • ^ – установление режима конференции <p>Кубик с пустым значением вызываемого номера выполняет разрыв соединения.</p>
--	--

Таймаут

	<p>Таймаут определяет максимальное время ожидания ответа на исходящий вызов. Если вызываемый номер не ответил в течение заданного промежутка времени, кубик завершается с результатом NO ANSWER.</p>
--	---

Играть звук

	<p>Указывается звуковой файл, который будет воспроизводиться в линию во время ожидания ответа. Имя файла может быть указано явно или быть значением строковой переменной.</p>
--	---

Маска прерывания

<p>Вызываемый номер =>[CalledNumber{C}]</p> <p>Таймаут =>[60]</p> <p>Играть звук =>[File::soundstringback.wav]</p> <p>Маска прерывания =>[#]</p> <p><input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3</p> <p><input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6</p> <p><input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9</p> <p><input type="checkbox"/> * <input type="checkbox"/> 0 <input checked="" type="checkbox"/> #</p>	<p>Свойство определяет набор цифр тонального набора, прием которых прерывает операцию установки исходящего соединения. Набор цифр задается установкой-снятием флажков. При приеме одной из установленных цифр, исполнение кубика завершается с результатом “NO ANSWER”.</p>
---	---

Обратный номер

<p>=>[CalledNumber{C}]</p> <p>Таймаут =>[60]</p> <p>Играть звук =>[File::soundstringback.wav]</p> <p>Маска прерывания =>[#]</p> <p>Обратный номер =>[cti.CallingNumber{C}]</p> <p>Приложение cti.CallingNumber</p> <p><input type="radio"/> Const <input checked="" type="radio"/> Sys <input type="radio"/> User <input type="radio"/> Db</p>	<p>Устанавливает номер вызывающего абонента, который передается телефонной сигнализацией при установлении исходящего соединения.</p>
---	--

Приложение

<p>=>[60]</p> <p>Играть звук =>[File::soundstringback.wav]</p> <p>Маска прерывания =>[#]</p> <p>Обратный номер =>[cti.CallingNumber{C}]</p> <p>Приложение =>[File::algorithm\SimpleMenu.application]</p> <p>}\R}\algorithm\SimpleMenu.application</p> <p>Файл</p>	<p>Алгоритм, который начнет исполняться на ресурсном порту, после того, как установится исходящее соединение. Параметром данного свойства является имя файла с расширением .application или строковая переменная, содержащая имя файла.</p>
--	--

Механизм работы кубика «Коммутатор».

При выполнении кубика, используется дополнительный (**ресурсный**) порт. В Visual CTI ресурсный порт выбирается в диалоге выбора устройства. При работе алгоритма в *Smile CTI Server*, ресурсный порт выбирается автоматически, в соответствии с таблицей маршрутизации. Выполнение кубика происходит следующим образом:

1. Выбирается ресурсный порт для выполнения исходящего звонка.
2. Ресурсный порт выполняет исходящий звонок по номеру, заданному в свойстве **Вызываемый номер**.
3. После набора номера, ресурсный порт находится в состоянии ожидания ответа в течение времени, заданного в свойстве **Таймаут**.
4. Если номер занят, не отвечает или недоступен по каким-либо причинам, исполнение кубика завершается с соответствующим результатом. Если соединение установлено, производится коммутация портов и кубик завершается с результатом VOICE.

После завершения кубика «Коммутатор», алгоритм продолжается, и могут выполняться любые кубики. Если в свойстве **Приложение** задан алгоритм для ресурсного порта, он начнет выполняться сразу после установки соединения. Порты остаются скоммутированными между собой до тех пор, пока на одном из них не произойдет отбой. Раскоммутировать порты можно, также, выполнив кубик «Коммутатор» с пустым **Вызываемым номером**.

При помощи кубиков «Коммутатор» можно создавать одновременно несколько активных соединений, переключаться между ними или объединять в конференцию. Специальный символ ‘!’ в поле вызываемого номера, выполняет постановку активного соединения на удержание (HOLD) или переключение между соединениями. Символ ‘^’ предназначен для объединения активных соединений в конференцию.

Результат исполнения.

Результатом выполнения кубика «Коммутатор» является результат выполнения исходящего звонка с ресурсного порта. Результат записывается в системной переменной **cti.Operation.Result** и может принимать следующие значения:

- **VOICE** – установлено соединение;
- **BUSY** – вызываемый номер занят;
- **NO ANSWER** – вызываемый номер не отвечает;
- **NO RINGBACK** – номер недоступен или сбой сети;
- **NO DIAL TONE** – нет соединения с сетью или шлюзом.

Если кубик завершился с результатом **BUSY**, в переменной **cti.Cause** устанавливается код завершения (в соответствии с рекомендацией Q.931). Если протокол сигнализации не передает код завершения, значение **cti.Cause** = 16.



4.12. Операции с переменными.

Свойства.

	<p>Кубик предназначен выполнения следующих операций:</p> <ul style="list-style-type: none"> объявление пользовательских переменных с присваиванием начального значения - для всех типов переменных. арифметические операции с числовыми переменными и константами; операции сложения строк для строковых переменных и констант.
	<p>В кубике можно выполнять несколько последовательных операций. Операции отображаются в виде списка свойств с именами – «Шаг 1», «Шаг 2» и т.д. Нажатие кнопок «Добавить» или «Изменить» вызывает окно настройки выражения. Кнопка «Удалить» служит для удаления выбранной операции.</p> <p>При добавлении свойства, оно размещается в порядке свойств, следующим после отмеченного.</p>
	<p>Окно настройки выражения содержит три текстовых поля:</p> <ul style="list-style-type: none"> C – имя переменной; A – поле первого операнда; B – поле второго операнда.
<p>Имя пользовательской переменной и операнды можно выбирать из списка переменных, нажав кнопку . Операндами могут быть любые переменные и константы соответствующего типа. Для интерпретации операнда, как строковой или числовой константы, необходимо установить флажок «Конст». Для выбора типа арифметической операции нужно нажать соответствующую кнопку.</p> <p>Однажды объявленная, переменная далее присутствует в списке пользовательских переменных и может использоваться другими кубиками алгоритма. При объявлении переменной, ее тип определяется автоматически, в зависимости от типа выражения. Если операндом является константа, тип операнда устанавливается при вводе первого символа в текстовое поле. Если первый набранный символ – цифра, операнд будет числового типа {N}, иначе – строкового {C}. Если тип переменной уже определен, операнды автоматически приводятся к заданному типу.</p>	

Функции.

Список функций вызывается нажатием кнопки [F]. В списке каждый раз *отображаются только те функции, которые могут быть применены к заданной комбинации типа переменной и типа и количества операндов*. Если в выражении два операнда и установлен знак операции (+, -, x, /), функция вычисляется от результата. Правильное выражение отображается в строке синим шрифтом. Если выражение не может быть выполнено, оно отображается красным шрифтом. Такое выражение не будет сохранено в свойство кубика. Если при редактировании выражения изменяется тип одного из операндов, функция сбрасывается.

Математические функции.

- **ABS (A)** Модуль числа A.
- **CEIL (A)** Округление числа A до целого в большую сторону.
- **FLOOR (A)** Округление числа A до целого в меньшую сторону.
- **ROUND (A)** Округление числа A до ближайшего целого.
- **POW (A, B)** Возведение A в степень B
- **RANDOM (A, B)** Случайное целое число в промежутке между A и B.

Функции работы со строками.

- **STRLEN (A)** Длина строки A.
- **STRNUM (A)** Преобразование строки A в число. Если строка не может быть преобразована в число, результат равен константе **Infinity**.
- **STRNUMEX (A, B)** Преобразование строки A в число. Если строка не может быть преобразована в число, результат равен операнду B.
- **STRINDEX_B (A, B)** Номер позиции первого символа первой найденной подстроки B в строке A. Нумерация символов в строке начинается с 1. Если строка A не содержит подстроку B, результат равен 0.
- **STRLASTINDEX_B (A, B)** Номер позиции первого символа последней найденной подстроки B в строке A. Нумерация символов в строке начинается с 1. Если строка A не содержит подстроку B, результат равен 0.
- **STRINDEX_E (A, B)** Номер позиции после последнего символа первой найденной подстроки B в строке A.
- **STRLASTINDEX_E (A, B)** Номер позиции после последнего символа последней найденной подстроки B в строке A.
- **SUBSTR_B (A, B)** Результатом является подстрока от начала строки A до первой найденной подстроки B в строке A. Если строка A не содержит подстроку B, возвращается строка A.
- **SUBSTR_L_B (A, B)** Результатом является подстрока от начала строки A до последней найденной подстроки B в строке A. Если строка A не содержит подстроку B, возвращается строка A.
- **SUBSTR_E (A, B)** Результатом является подстрока от первого символа первой найденной подстроки B в строке A и до конца строки A. Если строка A не содержит подстроку B, возвращается пустая строка.

- **SUBSTRL_E (A, B)** Результатом является подстрока от первого символа последней найденной подстроки **B** в строке **A** и до конца строки **A**. Если строка **A** не содержит подстроку **B**, возвращается пустая строка.
- **SUBSTR_X (A, B)** Результатом является подстрока, начинающаяся после последнего символа первой найденной подстроки **B** в строке **A** и до конца строки **A**. Если строка **A** не содержит подстроку **B**, возвращается пустая строка.
- **SUBSTRL_X (A, B)** Результатом является подстрока, начинающаяся после последнего символа последней найденной подстроки **B** в строке **A** и до конца строки **A**. Если строка **A** не содержит подстроку **B**, возвращается пустая строка.
- **SUBSTRN_B (A, B)** Результатом является подстрока от начала строки **A** до символа с номером **B-1**. Нумерация символов в строке начинается с 1.
- **SUBSTRN_E (A, B)** Результатом является подстрока от символа с номером **B** до конца строки **A**. Нумерация символов в строке начинается с 1.

Функции работы с текстовыми переменными.

- **LINECOUNT (A)** Количество строк в текстовой переменной **A**.
- **STRLINDEX_B (A, B)** Номер первой строки из текста **A** в которой встретилась подстрока **B**.
- **STRLINDEX_E (A, B)** Номер последней строки из текста **A** в которой встретилась подстрока **B**.
- **STRLINE (A, B)** Выбор из текста **A** строки с номером **B**
- **SUBSTRLINE_B (A, B)** Возвращает первую строку из текста **A**, в которой найдена подстрока **B**. При отсутствии таковой, возвращает пустую строку.
- **SUBSTRLINE_BX (A, B)** Возвращает подстроку первой строки из текста **A**, в которой найдена подстрока **B**. Возвращаемая подстрока начинается с символа, следующего после подстроки **B**.
- **SUBSTRLINE_E (A, B)** Возвращает последнюю строку из текста **A**, в которой найдена подстрока **B**.
- **SUBSTRLINE_EX (A, B)** Возвращает подстроку последней строки из текста **A**, в которой найдена подстрока **B**. Возвращаемая подстрока начинается с символа, следующего после подстроки **B**.
- **APPENDTEXT (A, B)** Добавить текст **B** к тексту **A**.
- **APPENDLINE (A, B)** Добавить строку **B** к тексту **A**
- **SUBTXTN_B (A, B)** Результат - текст, полученный из текста **A** до строки с номером **B** (не включая строку с номером **B**).
- **SUBTXTN_E (A, B)** Результат - текст, полученный из текста **A** начиная со строки с номером **B**.
- **SUBSTRTXT_B (A, B)** Результат - текст, полученный из текста **A** до строки, в которой найдена подстрока **B** (не включая эту строку).
- **SUBSTRTXT_E (A, B)** Результат - текст, полученный из текста **A** начиная со строки, в которой найдена подстрока **B**.

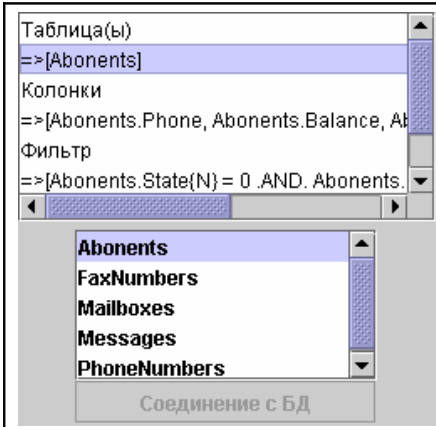


4.13. Выборка из базы данных.

Данный кубик выполняет операцию выборки из базы данных (операция SELECT). Для настройки свойств кубика, нужно установить соединение с базой данных (как это сделать, описано в п.2.1. Соединение с СУБД.).

Свойства.

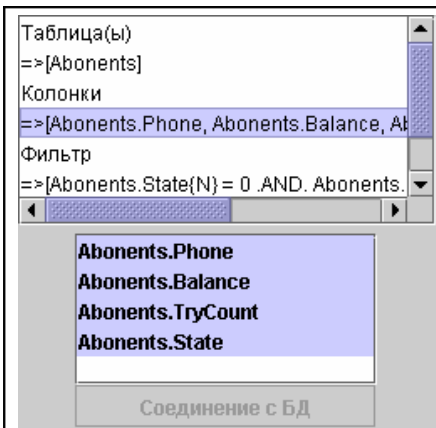
Таблицы



Свойство определяет список объектов (таблиц и представлений) в базе данных, из которых производится выборка строк. Редактор свойства содержит список всех доступных пользователю объектов базы данных. Для выбора нужного объекта, его необходимо отметить с помощью кнопки мыши. Одновременно можно выбрать любое количество объектов из списка.

Если соединение с СУБД отсутствует, список объектов будет пустым. Для вызова диалога соединения с СУБД нажмите кнопку **[Соединение с БД]**. После того, как соединение установлено, в нижнем окне появится список доступных объектов БД (таблиц и представлений). Если соединение установлено успешно, а список объектов остался пустым, проверьте, имеет ли пользователь привилегии на объекты запрашиваемой схемы. Нужно также иметь в виду, что в некоторых СУБД имя пользователя может зависеть от регистра. Например, в Оракл 8/8i нужно вводить имя пользователя большими буквами.

Колонки



Свойство содержит список полей (колонок) выбранных объектов, которые должны присутствовать в выборке. Поля обозначаются в виде:

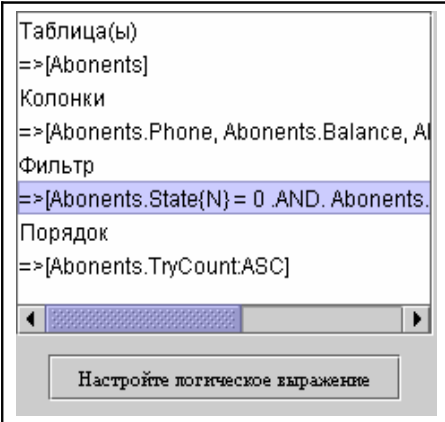
[имя таблицы].[имя колонки]

Редактор свойства содержит список всех доступных полей всех объектов, выбранных в свойстве «Таблицы». Чтобы добавить нужное поле, нужно отметить его в списке.

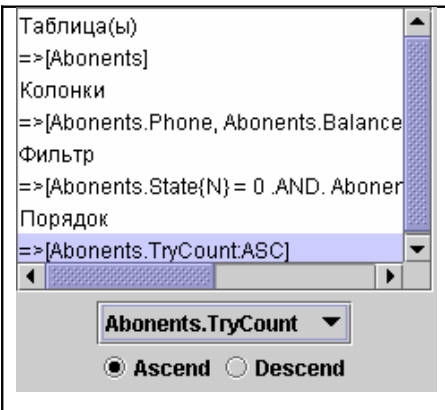
Поля объектов, заданные в кубиках «**Выборка из базы данных**», добавляются в список переменных алгоритма и могут затем использоваться в вычислениях.

Значения полей действительны после выполнения кубика, в котором эти поля определены, и до выполнения следующего кубика «Выборка из базы данных».

Фильтр

	<p>Фильтр определяет набор условий, в виде логического выражения, по которым строится выборка. Настройка этого свойства полностью аналогична настройке свойства кубика «Логическое ветвление». Если фильтр не задан, то выборка будет включать все строки объектов, выбранных в свойстве «Таблицы».</p>
---	--

Порядок

	<p>Свойство определяет порядок строк в выборке. Выпадающий список выбора содержит имена колонок, заданных в свойстве «Колонки». Выборка может быть отсортирована в порядке возрастания (Ascend) или убывания (Descend) значений выбранной колонки.</p>
--	--

Результат исполнения.

Результатом исполнения является выборка из базы данных всех строк, которые удовлетворяют условиям свойства «Фильтр».

Если выборка содержит хотя бы одну строку, системная переменная **db.DataAccess** устанавливается в значение «Yes». Если ни одна строка базы данных не удовлетворяет условиям свойства «Фильтр», системная переменная **db.DataAccess** устанавливается в значение «No».

Если **db.DataAccess**=«Yes», значения полей первой строки выборки могут использоваться алгоритмом. Доступ к остальным строкам выборки осуществляется кубиком «Перемещение по выборке». Значение переменной **db.RowCount** равно количеству строк выборки.



4.14. Перемещение по выборке.

Кубик выполняет перемещение курсора по выборке, построенной кубиком «Выборка из базы данных» и зачитывает из базы соответствующую запись.

Свойства.

Тип перемещения

	<p>Свойство определяет тип перемещения <i>курсора</i> (указателя текущей строки) по выборке:</p> <p>NEXT – прямое перемещение; PREVIOUS – обратное перемещение; FIRST – на первую строку выборки; LAST – на последнюю строку выборки; ABSOLUTE – перемещение на строку с номером, указанным в «Шаг перемещения»</p>
--	---

Шаг перемещения

	<p>Если «Тип перемещения» установлен в значение NEXT или PREVIOUS, курсор перемещается на указанное количество записей в соответствующем направлении.</p> <p>Если «Тип перемещения» установлен в значение ABSOLUTE, курсор перемещается на строку с заданным номером.</p>
--	---

Результат исполнения.

Системная переменная **db.DataAccess** устанавливается в одно из следующих значений:

«Yes» - указанная запись была выбрана и значения соответствующих полей установлены;

«No» - указанная запись не найдена;

«ERROR» - ошибка при чтении записи из базы данных.

Системная переменная **db.RowNumber** содержит номер текущей строки в выборке.



4.15. Запись в базу данных.

Изменение значений полей в таблице. Операция применяется одновременно ко всем строкам таблицы, удовлетворяющим заданному условию.

Свойства.

Таблицы

	<p>Свойство определяет таблицу базы данных, в которой будут производиться изменения. Редактор свойства содержит список всех доступных пользователю таблиц базы данных. Для выбора таблицы, ее необходимо отметить с помощью кнопки мыши.</p>
--	--

Если соединение с СУБД отсутствует, список объектов будет пустым. Для вызова диалога соединения с СУБД нажмите кнопку **[Соединение с БД]**. После того, как соединение установлено, в нижнем окне появится список доступных объектов БД (таблиц и представлений). Если соединение установлено успешно, а список объектов остался пустым, проверьте, имеет ли пользователь привилегии на объекты запрашиваемой схемы. Нужно также иметь в виду, что в некоторых СУБД имя пользователя может зависеть от регистра. Например, в Оракл 8/8i нужно вводить имя пользователя большими буквами.

Фильтр


	<p>Фильтр определяет набор условий, в виде логического выражения. Изменение будет применено для всех строк таблицы, которые удовлетворяют условию фильтра. Настройка этого свойства полностью аналогична настройке свойства кубика «Логическое ветвление». Если фильтр не задан, изменение значений полей будет применено для всех строк таблицы.</p>
--	--

Шаг N.

<p>Таблица(ы) =>[Abonents]</p> <p>Фильтр =>[Abonents.Phone(C) = Phone(C)]</p> <p>Шаг 1 =>[Abonents.TryCount(N) = Abonents.TryCount</p> <p>Добавить Изменить Удалить</p>	<p>Цепочка выражений, которые задают новые значения полей таблицы. Каждое выражение располагается в виде списка свойств с соответствующими именами «Шаг 1», «Шаг 2» и т.д.</p> <p>Нажатие кнопок «Добавить» или «Изменить» вызывает окно настройки выражения.</p>
--	---

Окно настройки выражения.

<p>Настройте выражение</p> <p>C Messages.Messages ... Принять</p> <p>= f C = A + B Отмена</p> <p>A Name ... <input type="checkbox"/> Конст</p> <p>+ + - x /</p> <p>B counter ... <input type="checkbox"/> Конст</p>	<p>Окно настройки выражения содержит три текстовых поля:</p> <p>C – имя переменной;</p> <p>A – поле первого операнда;</p> <p>B – поле второго операнда.</p>
---	--

Имя пользовательской переменной и операнды можно выбирать из списка переменных, нажав кнопку . Операндами могут быть любые переменные и константы соответствующего типа. Для интерпретации операнда, как строковой или числовой константы, необходимо установить флажок «Конст». Для выбора типа арифметической операции нужно нажать соответствующую кнопку.

При построении выражения можно использовать функции (см. 39).

Результат исполнения.

Новые значения полей записываются во все строки таблицы, удовлетворяющие условиям фильтра.

В переменную **db.RowCount** заносится количество измененных строк .



4.16. Удаление строк из таблицы.

Свойства.

Таблицы

<p>Таблица(ы) =>[Messages]</p> <p>Фильтр =>[Messages.Mailbox{C} = Phone{C}]</p> <p>Abonents FaxNumbers Mailboxes Messages PhoneNumbers</p> <p>Соединение с БД</p>	<p>Свойство определяет таблицу базы данных, из которой будут удаляться строки. Редактор свойства содержит список всех доступных пользователю таблиц базы данных. Для выбора нужной таблицы, ее необходимо отметить с помощью кнопки мыши.</p>
<p>Если соединение с СУБД отсутствует, список доступных таблиц будет пустым. Для вызова диалога соединения с СУБД нажмите кнопку «Соединение».</p>	

Фильтр

<p>Таблица(ы) =>[Messages]</p> <p>Фильтр =>[Messages.Mailbox{C} = Phone{C}]</p> <p>Настройте логическое выражение</p>	<p>Фильтр определяет набор условий, в виде логического выражения. При исполнении кубика, из выбранной таблицы будут удалены все строки, удовлетворяющие условию фильтра. Настройка этого свойства полностью аналогична настройке свойства кубика «Логическое ветвление».</p> <p>Если фильтр не задан, из таблицы будут удалены все данные.</p>
---	---

Результат исполнения.

После исполнения кубика из таблицы будут удалены строки, удовлетворяющие условию фильтра.

Переменная **db.RowCount** примет значение равное количеству удаленных строк.



4.17. Добавление строк в таблицу.

Свойства.

Таблицы

	<p>Свойство определяет таблицу базы данных, в которую будет добавлена строка. Редактор свойства содержит список всех доступных пользователю таблиц базы данных. Для выбора нужной таблицы, ее необходимо отметить с помощью кнопки мыши.</p> <p>Если соединение с СУБД отсутствует, список таблиц будет пустым. Для вызова диалога соединения с СУБД нажмите кнопку «Соединение».</p>
--	---

Шаг N.

	<p>Цепочка выражений, в которых вычисляются значения полей для новой строки. Каждое выражение располагается в виде списка свойств с соответствующими именами «Шаг 1», «Шаг 2» и т.д. Нажатие кнопок «Добавить» или «Изменить» вызывает окно настройки выражения.</p>
	<p>Окно настройки выражения содержит три текстовых поля: С – имя переменной; А – поле первого операнда; В – поле второго операнда.</p>

Имя пользовательской переменной и операнды можно выбирать из списка переменных, нажав кнопку . Операндами могут быть любые переменные и константы соответствующего типа. Для интерпретации операнда, как строковой или числовой константы, необходимо установить флажок «Конст». Для выбора типа арифметической операции нужно нажать соответствующую кнопку.

При построении выражения можно использовать функции (см. 39).

Результат исполнения.

Если операция выполнена успешно, системная переменная **db.RowCount** будет равна **1**, если нет - **0**.



4.18. Запись в текстовый файл.

Кубик выполняет добавление строки в указанный текстовый файл.

Свойства.

Имя файла

Имя файла может быть задано явно или взято из строковой переменной. Если файл не существует, он будет создан. Кроме этого, можно производить запись в переменную типа текст {T}.

Специальная переменная **System.Out** предназначена для вывода текста в строку состояния порта *Smile CTI Server*.

Строка

Строка может быть задана константой или любой переменной строкового типа. В строке можно использовать в качестве знака табуляции символ ` `.

Строка добавляется в конец текстового файла.



4.19. Отправка E-mail.

Свойства.

To

	<p>Адрес получателя. Может быть задан, как строковая константа или переменная.</p>
--	--

From

	<p>Адрес отправителя. Может быть задан, как строковая константа или переменная.</p>
--	---

Subject

	<p>Тема. Может быть задана, как строковая константа или переменная.</p>
--	---

Body

Тело письма. Текст письма может быть введен в строке ввода свойства или взят из строковой или текстовой переменной.


Attachment

Присоединенный файл, который также может содержать данные типа «Голос» и «Факс». Источник данных можно задать следующими способами:

«**Переменная**». Источник – переменная типа **{V}** **{F}** **{T}** или **{B}**.

«**Файл на диске**».

«**Имя файла в переменной**». Имя файла находится в строковой переменной.

Для выбора источника, нужно нажать кнопку 

Сервер SMTP

IP адрес или полное доменное имя сервера SMTP, через который будет отправлено письмо.

Режим

Письмо может быть отправлено в одном из двух режимов – синхронном или асинхронном. Синхронный режим означает, что кубик завершается только после выполнения операции отправки письма.

Результат исполнения.

В результате исполнения кубика, системная переменная *cti.Operation.Result* устанавливается в одно из следующих значений:

- **OK** - сообщение отправлено;
- **COMMUNICATION ERROR** – ошибка.

В асинхронном режиме, кубик сразу завершается с результатом “**OK**”, а отправкой письма занимается в фоновом режиме отдельный процесс – *sendmail*. Результаты работы *sendmail* фиксируются в соответствующем журнале. Если в процессе отправки сообщения возникает ошибка, *sendmail* будет пытаться отправить это сообщение каждые 10 минут. Неотправленные сообщения хранятся в виде файлов с расширением “.msg” в каталоге **\unsent** .



4.20. Вызов подпрограммы.

Посредством данного кубика можно вызывать из одного алгоритма другой алгоритм, как подпрограмму.

Свойства.

Вставить в алгоритм

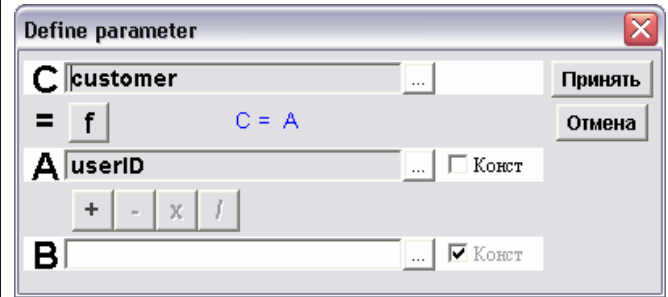

<p>Вставить в алгоритм => [Нет] Источник => [File::algorithm\test2.application] Параметр 1 => [DIAL_TRYCOUNT{N}= 3] Параметр 2 => [ORIGINATION_NUMBER{C}= cti.Input{C}]</p> <p><input type="radio"/> Да <input checked="" type="radio"/> Нет</p>	<p>Установив это свойство в значение [Да], можно вставить выбранный алгоритм внутрь основного алгоритма.</p>
---	--

Источник

<p>Вставить в алгоритм => [Нет] Источник => [File::algorithm\test2.application] Параметр 1 => [DIAL_TRYCOUNT{N}= 3] Параметр 2 => [ORIGINATION_NUMBER{C}= cti.Input{C}]</p> <p><input type="text" value="-HOME DIR\algorithm\test2.application"/> ...</p> <p>Файл</p>	<p>Выбирает файл алгоритма с расширением “<i>application</i>”. Имя файла может быть задано явно или взято из строковой переменной.</p>
--	--

Параметры

<p>Вставить в алгоритм => [Нет] Источник => [File::algorithm\test2.application] Параметр 1 => [DIAL_TRYCOUNT{N}= 3] Параметр 2 => [ORIGINATION_NUMBER{C}= cti.Input{C}]</p> <p>Добавить Изменить Удалить</p>	<p>Цепочка выражений, в которых определяются значения фактических параметров, которые передаются в подпрограмму. (Формальные параметры подпрограммы задаются в кубике «Начало алгоритма»).</p> <p>Каждое выражение располагается в виде списка свойств с именами «Параметр 1», «Параметр 2» и т.д. Нажатие кнопок «Добавить» или «Изменить» вызывает окно определения параметров вызова.</p>
---	--

	<p>Окно настройки выражения содержит три текстовых поля:</p> <p>С – имя переменной; А – поле первого операнда; В – поле второго операнда.</p>
<p>Если свойство «Источник» указывает на файл подпрограммы явным образом, имя переменной вызываемой подпрограммы выбирается из списка (нажать кнопку ). Если свойство «Источник» задает имя файла через строковую переменную, имя переменной вызываемой подпрограммы можно ввести вручную.</p> <p>Операндами могут быть любые переменные основного алгоритма или константы, имеющие соответствующий тип. Для интерпретации операнда, как строковой или числовой константы, необходимо установить флажок «Конст». Для выбора типа арифметической операции нужно нажать соответствующую кнопку.</p> <p>При построении выражения можно использовать функции (см. 4.12 Операции с переменными).</p>	



4.21. Операции с файлами.

Посредством данного кубика можно копировать, удалять и переименовывать файлы, а также получать информацию о файлах и содержимом каталогов. Кроме этого, содержимое файла может, с помощью данного кубика, быть загружено в переменную типа {T}, {B}, {V} или {F}.

Свойства.

Источник

<p>Источник =>[File::cti.cfg]</p> <p>Приемник =>[Memory::cti.Text]</p> <p>Удалить источник =>[Нет]</p> <p>[HOME DIR]\cti.cfg</p> <p>Файл</p>	<p>Источником может быть файл или переменная типа {T}, {B}, {V} или {F}. Имя файла может быть указано явно или взято из строковой переменной. После выполнения кубика, системные переменные группы “file.” будут содержать информацию о данном файле. Если задано имя каталога, значение переменной file.Type{C} будет равно “DIR”. Список имен файлов каталога будет загружен в переменную file.List{T}. Если во время выполнения кубика файл не найден, переменная file.Access примет значение “Not found”.</p>
---	---

Приемник

<p>Источник =>[File::cti.cfg]</p> <p>Приемник =>[Memory::cti.Text]</p> <p>Удалить источник =>[Нет]</p> <p>cti.Text</p> <p>Переменная</p>	<p>Приемником может быть файл или переменная типа {T}, {B}, {V} или {F}. Имя файла может быть указано явно или взято из строковой переменной. Если файл уже существует, операция завершается с результатом “Target exists” в переменной file.Access. Если источник – каталог, допускается только операция переименования (с установленным свойством «Удалить источник»).</p> <p>Если операция выполнена успешно, значение file.Access будет равно “Yes”.</p>
---	--

Удалить источник

<p>Источник =>[File::cti.cfg]</p> <p>Приемник =>[Memory::cti.Text]</p> <p>Удалить источник =>[Нет]</p> <p><input type="radio"/> Да <input checked="" type="radio"/> Нет</p>	<p>Данное свойство позволяет удалять файлы или каталоги. Каталог может быть удален, если он пуст. Если файл или каталог не может быть удален, операция завершается с результатом “Not permitted” в переменной file.Access.</p> <p>Если операция выполнена успешно, значение file.Access будет равно “Yes”.</p>
--	--



4.22. Старт приложения.

При помощи данного кубика можно стартовать внешние программы.

Свойства.

Ждать завершения программы

	<p>Внешние программы можно исполнять в синхронном или асинхронном режиме.</p> <p>В синхронном режиме кубик ожидает завершения программы и записывает код завершения в переменную System.Exit.</p> <p>В асинхронном режиме кубик выполняет только старт программы и на этом его выполнение завершается.</p>
--	---

Исполняемый файл

	<p>В свойстве указывается имя исполняемого файла или строковая переменная, содержащая имя такого файла. Исполняемый файл должен иметь расширение “.exe”.</p>
--	--

Параметр

	<p>Кубик позволяет передать стартующей программе до пяти параметров. Параметр может быть задан строковой константой или переменной.</p>
--	---



4.23. Запрос Интернет-ресурса.

Свойства.

Протокол

Протокол
=>["http"]
Сервер
=>["192.168.1.10"]
Порт
=>[80]
Ресурс
=>["/servlet/billinginfo"]
Данные
=>[Memory::cti.Input]
Логин
=>[user{C}]
Пароль
=>[password{C}]
Таймаут
=>[10]

http

Const Sys Usr Db

Выбор протокола. Строка может иметь одно из следующих значений:

- 1) http
- 2) https
- 3) ftp

Сервер

Протокол
=>["http"]
Сервер
=>["192.168.1.10"]
Порт
=>[80]
Ресурс
=>["/servlet/billinginfo"]
Данные
=>[Memory::cti.Input]
Логин
=>[user{C}]
Пароль
=>[password{C}]
Таймаут
=>[10]

192.168.1.10

Const Sys Usr Db

Имя или ip адрес сервера, к которому выполняется запрос. Для запроса http можно использовать адрес прокси-сервера.

Порт

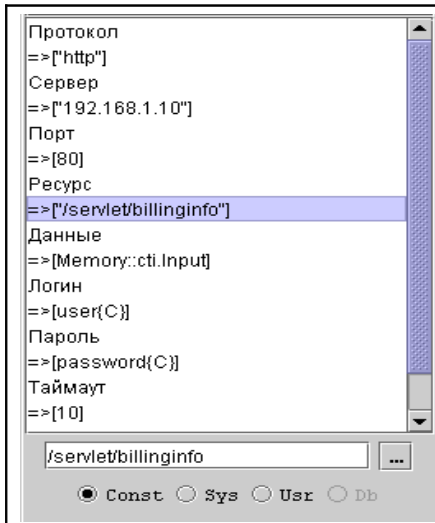
Протокол
=>["http"]
Сервер
=>["192.168.1.10"]
Порт
=>[80]
Ресурс
=>["/servlet/billinginfo"]
Данные
=>[Memory::cti.Input]
Логин
=>[user{C}]
Пароль
=>[password{C}]
Таймаут
=>[10]

80

Const Sys Usr Db

Порт сервера. Если порт не задан, по умолчанию используются стандартные порты для соответствующего протокола (ftp - 21, http - 80, https - 443).

Ресурс



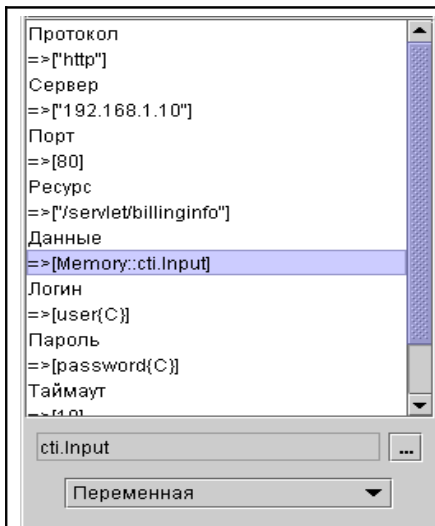
Универсальный идентификатор ресурса (URI). Строка, указывающая на определенный ресурс сервера (файл или другой объект). В данном свойстве можно также указывать параметры метода GET для http-запроса. Например:

/search/java/?qt=forte&Search.x=39&Search.y=9

Если запрос выполняется через прокси-сервер, в строке «Ресурс» нужно указывать абсолютный URI. Например:

http://www.javasoft.com/search/java/?qt=forte

Данные



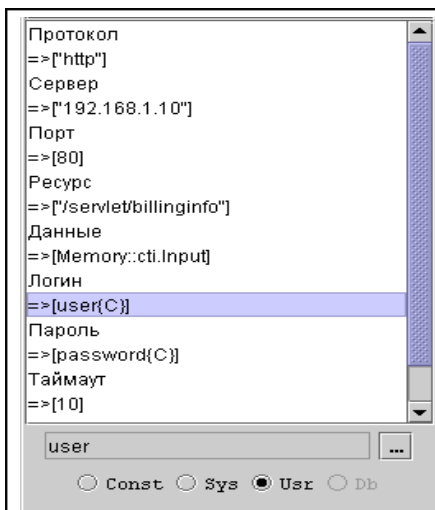
Данные, передаваемые на сервер. Источником данных может быть файл или переменная типа {C}, {T}, {B}, {V} или {F}. Имя файла может быть указано явно или взято из переменной.

В протоколе http, если источником является строковая переменная {C}, формируется запрос методом POST. Содержимое переменной, в этом случае, является параметром запроса, например:

PostMessage: user=Bob&password=123

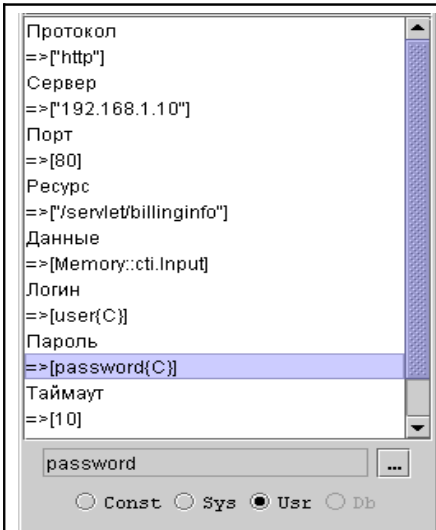
Если источником является переменная другого типа или файл, http формирует запрос методом PUT.

Логин



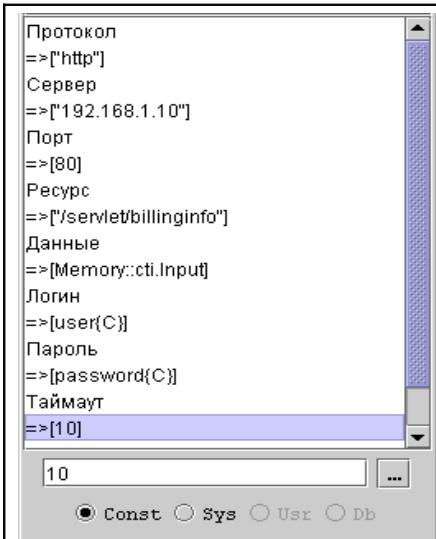
Имя пользователя, используемое при доступе к защищенным серверам. При анонимном доступе к ftp серверу данное свойство можно не устанавливать.

Пароль



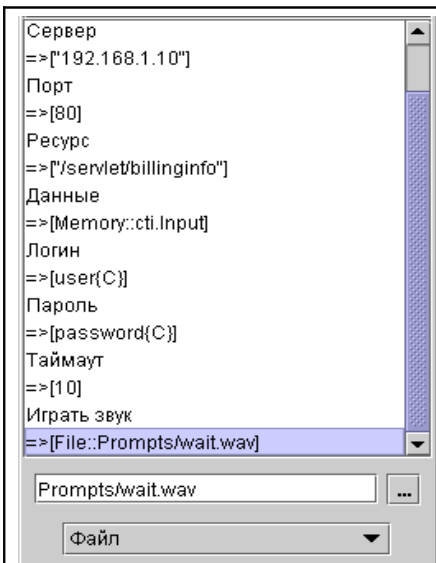
Пароль, используемый вместе с логином для доступа к защищенным серверам.

Таймаут



Таймаут запроса в секундах.

Играть звук



Свойство определяет звуковой файл, который будет воспроизводиться во время выполнения запроса. Путь на файл может быть указан явно или через строковую переменную (“имя файла в переменной”)

Результат исполнения.

Результат выполнения запроса отображается в переменной *url.Access* {C}, которая может принимать следующие значения:

- **OK** - запрос выполнен успешно и получен ответ сервера;
- **NOT AVAILABLE** – сервер с указанным адресом не найден или ресурс не найден или недоступен;
- **COMMUNICATION ERROR** – ошибка в процессе связи с сервером;
- **TIMEOUT** – таймаут.
- **FORMAT** – ошибка в параметрах запроса.

После выполнения кубика могут также устанавливаться следующие переменные:

url.ErrorMessage {C} – строка с описанием ошибки. Содержимое данной переменной носит информативный характер.

http.ResponseCode {N} – код, возвращаемый http-сервером (см. RFC 2616).

http.ResponseMessage {C} – строка с описанием, возвращаемая http-сервером.

Следующие переменные могут устанавливаться в случае успешного выполнения запроса (*url.Access*= **OK**):

url.ContentType {C} – тип полученных данных (содержимое заголовка *Content-Type*);

url.ContentLength {N} – количество байт данных (*Content-Length*);

url.ContentEncoding {C} – тип кодирования данных (*Content-Encoding*);

url.LastModified {N} – дата последнего изменения данных на сервере (количество секунд с 1.1.1970).

Сами данные, полученные от сервера, сохраняются, в зависимости от их типа в переменную *cti.Text* (если тип данных – текст) или в переменную *cti.File* в остальных случаях.



4.24. Запрос к серверу RADIUS.

Данный кубик реализует клиентскую часть протокола RADIUS (Remote Authentication Dial In User Service) в соответствии со стандартами Интернет RFC2865 и RFC2866.

Свойства.

Тип запроса

<p>Тип запроса =>[Access]</p> <p>Сервер =>["192.168.1.1"]</p> <p>Секретный код =>["secret123"]</p> <p>Таймаут =>[3]</p> <p>Число попыток =>[1]</p> <p>Атрибут 1 =>[User-Name(C)= cti.CalledNumber(C)]</p> <p>Атрибут 2 =>[User-Password(C)= "R2w56Ya9"]</p> <p><input type="radio"/> Accounting <input checked="" type="radio"/> Access</p>	<p>Выбор типа формируемого запроса:</p> <p>4) Access-Request</p> <p>5) Accounting-Request</p>
--	---

Сервер

<p>Тип запроса =>[Access]</p> <p>Сервер =>["192.168.1.1"]</p> <p>Секретный код =>["secret123"]</p> <p>Таймаут =>[3]</p> <p>Число попыток =>[1]</p> <p>Атрибут 1 =>[User-Name(C)= cti.CalledNumber(C)]</p> <p>Атрибут 2</p> <p>192.168.1.1</p> <p><input checked="" type="radio"/> Const <input type="radio"/> Sys <input type="radio"/> Usr <input type="radio"/> Db</p>	<p>IP адрес сервера RADIUS. По умолчанию, используются стандартные порты (1812 для Access-Request и 1813 для Accounting-Request). Если используется нестандартный порт, номер порта нужно добавить к адресу через двоеточие. Например 192.168.1.1:1912</p>
--	--

Секретный код

<p>Тип запроса =>[Access]</p> <p>Сервер =>["192.168.1.1"]</p> <p>Секретный код =>["secret123"]</p> <p>Таймаут =>[3]</p> <p>Число попыток =>[1]</p> <p>Атрибут 1 =>[User-Name(C)= cti.CalledNumber(C)]</p> <p>Атрибут 2</p> <p>secret123</p> <p><input checked="" type="radio"/> Const <input type="radio"/> Sys <input type="radio"/> Usr <input type="radio"/> Db</p>	<p>Секретный код (shared secret), установленный на сервере.</p>
--	---

Таймаут

Тип запроса
=>[Access]
Сервер
=>["192.168.1.1"]
Секретный код
=>["secret123"]
Таймаут
=>[3]
Число попыток
=>[1]
Атрибут 1
=>[User-Name(C)= cti.CalledNumber(C)]
Атрибут 2

3

Const Sys Usr Db

Максимальное время ожидания ответа сервера (в секундах).

Число попыток

Тип запроса
=>[Access]
Сервер
=>["192.168.1.1"]
Секретный код
=>["secret123"]
Таймаут
=>[3]
Число попыток
=>[1]
Атрибут 1
=>[User-Name(C)= cti.CalledNumber(C)]
Атрибут 2

1

Const Sys Usr Db

Максимальное число попыток запроса. Повторный запрос выполняется автоматически, если не получен ответ сервера на предыдущий запрос.

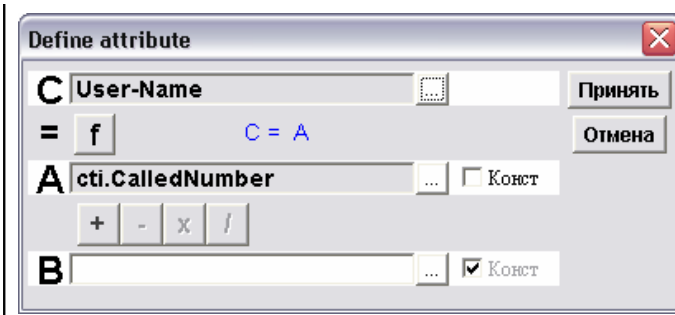
Атрибут

Тип запроса
=>[Access]
Сервер
=>["192.168.1.1"]
Секретный код
=>["secret123"]
Таймаут
=>[3]
Число попыток
=>[1]
Атрибут 1
=>[User-Name(C)= cti.CalledNumber(C)]
Атрибут 2
=>[User-Password(C)= "R2w56Ya9"]

Добавить Изменить Удалить

Список выражений, в которых вычисляются значения атрибутов, включаемых в запрос. Каждое выражение располагается в виде списка свойств с соответствующими именами «Атрибут1», «Атрибут 2» и т.д. Нажатие кнопок «Добавить» или «Изменить» вызывает окно настройки выражения.

Выражения формируются таким же способом, как и в кубике «Операции с переменными» (см. 4.12).




Окно настройки выражения содержит три текстовых поля:

C – имя атрибута;

A – поле первого операнда;

B – поле второго операнда.

Имя атрибута выбирается из списка (нажать кнопку ). Список имен атрибутов загружается из текстового файла “radius”, который находится в каталоге установки Visual CTI.

Результат исполнения.

После исполнения кубика, инициализируется системная переменная *radius.Response*, которая может принимать одно следующих значений:

- **OK** - запрос отправлен и получен ответ сервера (*Access-Accept* для *Access-Request* или *Accounting-Response* для *Accounting-Request*);
- **REJECT** – получен *Access-Reject* в ответ на *Access-Request*;
- **NOT AVAILABLE** – сервер с указанным адресом не найден;
- **ERROR** – ошибка протокола в процессе связи с сервером;
- **TIMEOUT** – таймаут ожидания ответа сервера;
- **FORMAT** – от сервера получен некорректный ответ.

Если от сервера принят *Access-Accept* или *Accounting-Response*, инициализируется системная переменная *radius.Attributes* (тип *Text*), в которую записывается содержимое соответствующего пакета. Содержимое переменной *radius.Attributes* представляет собой набор текстовых строк в формате: *<Атрибут>=<Значение>*. Последовательность строк соответствует последовательности атрибутов в принятом пакете.

Приложение A. Phone Emulator.

Драйвер виртуального устройства - “**Phone Emulator**” - предназначен для разработки и отладки приложений компьютерной телефонии на обычном ПК, оснащённом звуковой картой.

При выборе драйвера устройства “**Phone Emulator**”, на экране появляется телефонный аппарат, предназначенный для тестирования алгоритмов.

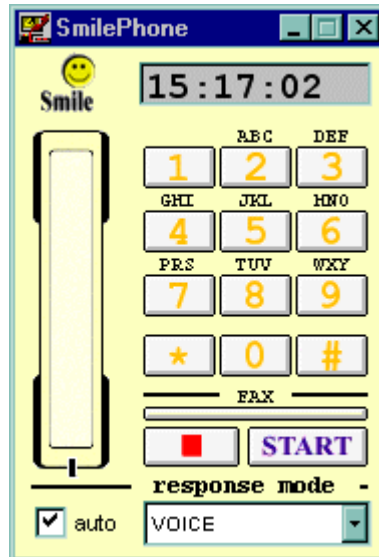



Рис. А.1 Телефонный аппарат SmilePhone.

С помощью **SmilePhone** можно позвонить или принять звонок от виртуального “порта” эмулятора, на котором «выполняется» алгоритм.

Позвонить на алгоритм.

Алгоритм должен находиться в состоянии ожидания звонка. Чтобы позвонить, нужно кнопкой мыши «снять трубку».

Выполняя звонок, можно передать алгоритму «номер вызывающего абонента» и «номер вызываемого абонента». После того, как алгоритм примет звонок и выполнится кубик «Ожидание звонка», эти номера запишутся, соответственно, в переменные **cti.CallingNumber** и **cti.CalledNumber**.

Чтобы установить в **SmilePhone** «номер вызывающего абонента», нужно щелкнуть мышью на иконке . В появившемся окне свойств установите нужное значение в текстовом поле “Caller ID”.

«**Номер вызываемого абонента**» нужно просто набрать на клавиатуре перед тем, как снять трубку.

Принять звонок от алгоритма.

Эмулятор принимает звонки, генерируемые кубиком «Набор номера», в двух режимах – ручном и автоматическом. Установленная птичка “**auto**” соответствует режиму автоматического ответа на звонок. Тип ответа выбирается из списка. Чтобы ответить на звонок в ручном режиме, нужно просто снять трубку.

Эмулятор факс-аппарата.

Эмулировать прием факса **SmilePhone** может как в ручном, так и в автоматическом режиме.


Факс принимается от алгоритма при выполнении кубика «**Передача факса**». Для эмуляции приема факса нужно нажать кнопку **[START]**. Во время приема можно эмулировать нажатие кнопки **[STOP]** или обрыв соединения (положить трубку).

Для эмуляции приема факса в автоматическом режиме, нужно установить на **SmilePhone** режим автоответа и выбрать тип ответа “FAX”. Алгоритм должен выполнить исходящий звонок (кубик «**Набор номера**») и затем, выполнить кубик «**Передача факса**».

Факс передается алгоритму при выполнении кубика «**Прием факса**». Для эмуляции передачи факса нужно «вставить документ» (выбрать файл с факсом) и нажать кнопку **[START]**. Чтобы «вставить документ», нажмите кнопку с надписью “**fax**” и выберите файл формата “**tiff**” с помощью окна диалога выбора файла. Если при выполнении кубика «**Прием факса**» нажать **[START]**, не выбрав предварительно файл, алгоритм получит встречный сигнал приема факса и кубик завершится с результатом “POLLING”.

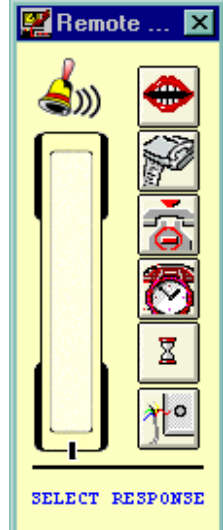
Во время передачи факса можно эмулировать нажатие кнопки **[STOP]** или обрыв соединения (положить трубку).

Если нажать **[START]** во время выполнения алгоритмом кубиков «**Прием цифр**», «**Воспроизведение**» или «**Запись**», эти кубики завершаются с результатом “FAX”.

Чтобы установить идентификатор факс-аппарата **SmilePhone**, нужно щелкнуть мышью на иконке  и установить значение в текстовом поле “Fax ID”. После выполнения кубиков «**Прием факса**» «**Передача факса**» то значение будет записано в переменную **cti.Fax.RemoteID**.







Эмулятор удаленного телефонного аппарата.

Для отладки алгоритмов, включающих в себя кубик «**Коммутатор**», эмулируется звонок с «ресурсного» порта на «удаленный аппарат».



SELECT RESPONSE

При выполнении кубика «**Коммутатор**», на экране появляется «удаленный телефонный аппарат». На этот аппарат поступает звонок. Ответ от удаленного абонента эмулируется нажатием одной из кнопок:

 VOICE	 FAX	 BUSY
 NO ANSWER	 NO RINGBACK	 NO DIAL TONE

Если выбрать “VOICE” или “FAX”, аппарат “снимет трубку” и установится условное соединение между абонентами. Разорвать соединение можно, положив трубку на одном из аппаратов. При этом, если положить трубку на **SmilePhone**, текущий кубик алгоритма завершится с результатом “DISCONNECT”. Если положить трубку на «удаленном аппарате», кубик завершится с результатом “TERMINATE”.

