Перевод интерфейса пользователя RDS

(на 24.01.18)

К интерфейсу пользователя RDS относятся все тексты, встроенные в основную программу (“rds.exe”) и в модели блоков: названия пунктов меню, заголовки окон и полей ввода, тексты сообщений, выводимых пользователю в отдельных окнах и т. п. К интерфейсу пользователя ***не относятся*** любые тексты в схемах: надписи на блоках, комментарии переменных и т. п. (такие тексты переводятся при помощи уже сделанных функций перевода схем). Важно уметь отличать одно от другого. Например, надпись “Шрифт...” на кнопке в окне настроек какого-либо блока относится к интерфейсу пользователя, а любая надпись на блоке-кнопке, расположенном в схеме – к этой схеме. Текущий язык в схеме и в интерфейсе пользователя устанавливаются независимо: можно, например, работать в RDS с русским интерфейсом со схемой, переведенной на английский, и наоборот.

Тексты элементов интерфейса пользователя главной программы RDS (“rds.exe”) и DLL с моделями блоков хранятся в служебных файлах, входящих в состав RDS (см. п. 1). Каждый набор таких файлов соответствует одному языку интерфейса. Перевод окон настроек в автокомпилируемых моделях сделан по-другому: для каждого элемента интерфейса окна настройки тексты на разных языках хранятся в самом файле модели (“.mdl”), они автоматически подставляются в нужные места программы при ее формировании в момент компиляции. Разработчику модели достаточно для каждого поля ввода указать названия на всех необходимых языках. Если в модели выводятся какие-либо окна с сообщениями пользователю, тексты этих сообщений нужно сделать текстовыми константами, для которых тоже можно задавать разные значения для разных языков. Таким образом, изменения в автокомпилируемых моделях сводятся к минимуму (подробнее см. в п. 5).

В целом процедура перевода интерфейса пользователя будет похожа на процедуру перевода схемы: создается новый язык, экспортируются файлы с текстами, тексты в файлах переводятся на этот новый язык, после чего файлы загружаются обратно. Но есть и принципиальные отличия.

1. **При переводе схем** в каждой схеме может быть свой набор равноправных языков, которые пользователь сам добавляет и удаляет по желанию. **При переводе интерфейса** языки создаются и выбираются централизованно для всей программы (RDS), причем один из них (русский) всегда существует как “язык по умолчанию”, и его тексты не могут быть изменены.
2. **При переводе схем** файл перевода должен постоянно храниться вместе со схемой, тексты на разных языках загружаются из него в схему по команде пользователя. Если в схему добавлены новые блоки с новыми надписями, содержимое схемы синхронизируется (снова по команде пользователя) с файлом перевода, где появляются новые тексты, которые нужно перевести. **При переводе интерфейса** экспортированный файл с текстами нужен только до момента его импорта (обратной загрузки) после перевода. Переведенные тексты встраиваются в служебные файлы RDS или в файлы автокомпилируемых моделей, после чего экспортированный файл можно стирать.
3. **При переводе схем** файл перевода содержит данные всех языков схемы в отдельных столбцах формата CSV. **При переводе интерфейса** файл содержит тексты только для одного языка – того, для которого он был экспортирован.
4. **При переводе схем** файл перевода всегда выгружается в формате CSV (возможно, с дополнительным служебным файлом). **При переводе интерфейса** можно выбрать между форматом CSV и форматом, аналогичным INI-файлам Windows (этот формат удобнее и понятнее для программиста).
5. **При переводе схем** служебная информация в файле перевода закодирована и никак не связана с назначением переводимого текста – например, “0101qj4q2w”. **При переводе**

интерфейса по формату служебной информации можно, в принципе, понять, к чему относится данный текст – например, “General._HelpButton.Caption” (надпись на кнопке вызова справки).

Ниже рассмотрен общий принцип работы, устройство служебных файлов и порядок работы с функциями перевода интерфейса пользователя. В данном тексте п.п. 1–6 предназначены для пользователей и переводчиков, п. 7 – для программистов, создающих модели блоков.

Следует учитывать, что некоторые элементы интерфейса стандартных диалогов и окон сообщений создаются операционной системой, и на них невозможно повлиять. Например, заголовки различных полей в окне выбора шрифта или окне открытия файла всегда будут написаны на текущем языке Windows.

1. Принцип действия и хранение данных для разных языков

Технически перевод интерфейса пользователя выполнен как часть потенциально более универсального механизма загрузки модификаций интерфейса. Сейчас этот механизм будет использоваться только для перевода, в дальнейшем на его основе можно будет также сделать переключение на упрощенный интерфейс со скрытием части возможностей для пугливых пользователей.

Устроен механизм следующим образом. В RDS есть “интерфейс пользователя по умолчанию” – это текущий интерфейс главной программы и всех DLL с моделями блоков. Без загрузки модификаций интерфейса все меню, окна и т. п. будут выглядеть так, как они выглядят сейчас. Все “управляемые” объекты интерфейса пользователя и параметры этих объектов записываются разработчиками (как разработчиком “rds.exe”, так и разработчиками DLL стандартных, не автокомпилируемых, моделей) в специальных текстовых ***файлах описания интерфейса пользователя***, формат которых приведен в п. 2. Эти файлы должны иметь расширение “.dat” и размещаться в папке “Description\” или вложенных в нее папках внутри папки установки RDS. Каких-либо строгих требований к именам этих файлов нет, но на данный момент принята следующая, достаточно простая, схема: файл описания интерфейса называется так же, как исполняемый файл, к которому он относится, и располагается в папке, совпадающей по имени с папкой исполняемого файла. Например, исполняемый файл “rds.exe” находится непосредственно в папке установки RDS, поэтому его файлом описания будет “Description\rds.dat”. Библиотека с моделями стандартных блоков “Common.dll” находится в подпапке “Dll\” внутри папки RDS, поэтому ее файлом описания будет “Description\Dll\Common.dat”. Размещение файлов описаний внутри папки “Description\” будет определять размещение созданных на их основе двоичных файлов модификации интерфейса (о них – ниже), поэтому лучше делать так, чтобы исполняемому файлу было просто “найти” свои файлы модификаций. В описанной схеме это соблюдается – файлы описания, например, располагаются в папке “Description\” точно так же, как сам исполняемый файл внутри папки RDS.

В главную программу жестко закладываются ***типы модификаций интерфейса***, причем каждому типу будет соответствовать конкретное, тоже жестко заданное, имя подпапки внутри папки “Interface\” в папке установки и в папке настроек RDS (на данный момент эти две папки всегда совпадают), в которой хранятся данные разных модификаций этого типа. Сейчас есть единственный тип модификации – “язык”, и ему соответствует папка “Language”. Таким образом, все данные интерфейса пользователя для разных языков будут находиться в папке “Interface\Language\”.

Внутри папки типа модификации (для языка – внутри “Interface\Language\”) находятся папки ***конкретных модификаций*** для данного типа. Каждой модификации (языку) соответствует одна папка, именем модификации считается имя этой папки. Например, если в RDS будет добавлен язык интерфейса с названием “English”, все данные и тексты для этого

языка будут находиться в папке “Interface\Language\English\” внутри папки установки RDS. Имя “Default” зарезервировано для папки хранения данных по умолчанию, никакая модификация интерфейса пользователя (язык) не может так называться.

Поскольку сейчас единственным существующим типом модификации интерфейса является язык, везде далее по тексту вместо словосочетания “модификация интерфейса пользователя” будет употребляться слово “язык”, но нужно иметь в виду, что все написанное будет относиться и к другим типам модификаций, если они со временем появятся.

В папке языка должны располагаться двоичные файлы с текстами, соответствующими файлам описания интерфейса из папки “Description\”, упомянутой выше. Эти файлы можно создавать средствами RDS при помощи экспорта/импорта текстовых файлов специального формата (см. п. 4). Каждому файлу описания интерфейса с расширением “.dat” из папки “Description\” будет соответствовать один двоичный файл перевода с расширением “.bin” в подпапке каждого языка. Двоичными эти файлы сделаны для повышения скорости поиска данных в них: хотя загрузка переводов объектов одного окна и занимает очень небольшое время как в текстовом, так и в двоичном формате, поиск данных в гигантском текстовом файле (только в “rds.exe” на данный момент около ста окон, в каждом из которых не меньше десятка текстов, требующих перевода) последовательным чтением может занимать существенное время. В формате двоичных файлов предусмотрено содержание, располагающееся в начале файла, поэтому поиск данных нужного окна будет выполняться быстрее (загрузка содержания – поиск в нем имени окна – быстрый переход в нужную позицию файла для загрузки данных).

В папке языка должен также находиться настроечный файл с названием “params.ini”, в котором описываются действия, выполняемые RDS при переключении языка пользователем. Обычно при таком переключении мало начать загружать файлы с текстами на другом языке при работе программы, нужно еще поменять и некоторые настройки и служебные файлы. Например, при переключении на английский язык нужно автоматически перенастроить RDS на другую панель блоков с английскими названиями вкладок-папок и, возможно, другими иконками блоков (сейчас на некоторых иконках нарисованы русские буквы). Кроме того, при наличии английской документации (сейчас ее нет) нужно переключить RDS на другие файлы справки, и т. д. Предполагается, что документация, отдельная панель блоков и т. п. для языка также будут находиться в папке этого языка. Если новый язык создается средствами RDS (см. п. 3), “params.ini” записывается автоматически по установленным пользователем флажкам, и, при согласии пользователя, в папку языка копируются дополнительные файлы (список модулей автокомпиляции, расширений и т. д.), которые нужно будет переводить.

При переключении пользователем с текущего языка A на язык B выполняются следующие действия:

- во внутренних настройках RDS запоминается необходимость загружать файлы с текстами из папки “Interface\Language\B\” начиная со следующего запуска;
- все настройки RDS, как-то связанные с языком (например, в какой папке находится панель блоков текущего языка) запоминаются в “Interface\Language\A\params.ini”, чтобы все изменения, сделанные настройках для текущего языка A, не были потеряны;
- эти же настройки RDS копируются из “Interface\Language\B\params.ini” (т. е. из файла параметров нового языка) и становятся текущими настройками RDS.

На этом переключение языков завершается: настройки RDS приведены в соответствие с новым языком, и при следующем запуске тексты объектов интерфейса начнут загружаться из файлов в папке нового языка.

Для примера рассмотрим структуру папок RDS, образующуюся при добавлении языка с названием “English”. В этом примере будем считать, что папка установки RDS совпадает с папкой настроек (на данный момент RDS устанавливается именно в такой конфигурации), и

из всех библиотек с моделями блоков будем рассматривать только “Common.dll”, чтобы не перегружать пример. Все пути будут указаны относительно папки установки RDS.

Структура папок с файлами будет следующей (голубым выделены данные языка “English”, зеленым – общие для всех языков описания интерфейса):

Папка или файл относительно папки установки RDS	***Назначение***
rds.exe	Главная программа
Dll\Common.dll	Стандартные модели блоков
Description\	Папка с описаниями интерфейса пользователя главной программы и стандартных DLL
Description\rds.dat	Описание интерфейса пользователя главной программы
Description\Dll\Common.dat	Описание интерфейса пользователя стандартных моделей блоков
Interface\Language\Default\	Папка резервного копирования параметров “интерфейса по умолчанию” (русского)
Interface\Language\English\	Папка с данными английского языка
Interface\Language\English\params.ini	Настройки параметров переключения на английский язык
Interface\Language\English\rds.bin	Перевод главной программы на английский язык
Interface\Language\English\Dll\Common.bin	Перевод стандартных моделей блоков на английский язык
Interface\Language\English\Panel\	Папка английской панели блоков
Interface\Language\English\Template\	Папка с английскими шаблонами схем и автокомпилируемых моделей
Interface\Language\English*.dat, Interface\Language\English*.lst	Версии служебных файлов RDS для английского языка

На данный момент в RDS не предусмотрено (и, видимо, не будет) никаких средств перевода панели блоков и шаблонов – это нужно выполнять вручную. Перевод названий модулей автокомпиляции, расширений и т. п. тоже должен выполняться вручную, но для этого достаточно после переключения на нужный язык просто отредактировать эти названия стандартными средствами RDS.

Автокомпилируемые модели не используют двоичные файлы перевода из папки языка, поскольку модели пользователя обычно размещаются в одной папке со схемой, и хранить для них данные централизованно не имеет смысла. Вместо этого тексты на разных языках хранятся в самом файле модели с расширением “.mdl”. Эти тексты могут быть экспортированы, переведены и импортированы обратно для загруженной схемы централизованно, через окно настроек RDS (см. п. 5.3). Разработчик модели может и сам, не дожидаясь перевода, указать для объектов тексты на разных языках. Следует помнить, что набор языков и их названия все равно задаются в RDS централизованно – нельзя добавить текст для языка, который не был “создан” заранее.

2. Формат файлов описания интерфейса пользователя

Файлы описания интерфейса пользователя главной программы (“rds.exe”) и стандартных моделей блоков (библиотек в папке “Dll\”) хранятся в папке “Description\” внутри папки установки RDS. Эти файлы создаются разработчиками соответствующих программ и должны поддерживаться ими в актуальном состоянии при изменении этих программ – например, при добавлении новых окон, сообщений или пунктов меню. Работают с этими файлами только разработчики программ, обычным пользователям внутренняя структура этих файлов не важна, даже если эти пользователи сами переводят интерфейс на новый язык. Пользователь-переводчик может, при желании, посмотреть комментарии в файле описания, если их предусмотрел разработчик.

Каждый текст интерфейса пользователя адресуется в файле описания тремя именами: ***именем группы***, ***именем объекта*** и ***именем параметра***. Эти имена не должны содержать знака доллара, знака равенства, точки, запятой, двоеточия, точки с запятой и квадратных скобок, в остальном они выбираются разработчиком программы произвольно. В качестве имени группы обычно используется имя окна или крупного раздела интерфейса пользователя – двоичные файлы перевода, формируемые на основе файла описания, устроены так, что данные одной группы можно достаточно быстро найти по содержанию файла. Имя объекта обычно описывает конкретный объект интерфейса (кнопку, пункт меню и т. п.), а имя параметра – конкретный параметр этого объекта (текст, всплывающую подсказку, список вариантов для выбора и т. п.). Все имена групп и параметров вместе с идентификаторами этих параметров придумываются разработчиком. При этом желательно давать им более-менее осмысленные имена, чтобы переводчик мог догадаться, к чему относится тот или иной текст.

Для всех имен параметров в начале файла должны быть указаны типы (на данный момент поддерживаются только “строка” и “целое число”) и целые идентификаторы, по которым к ним будет обращаться программа для загрузки текстов. Кроме того, в файле указывается, какие имена параметров для какого типа модификации интерфейса используются. На данный момент это не актуально, поскольку сейчас есть только один тип модификации – “язык”. Но позже, когда появятся другие типы, можно будет указывать, что, например, для модификации “видимость пунктов меню и полей ввода” текстовые названия этих пунктов и полей не требуются, и их можно не включать в формируемый двоичный файл.

В целом, файл описания интерфейса имеет формат обычного INI-файла Windows. Начинается он всегда со служебной секции “$Types” (в файле имена секций всегда заключаются в квадратные скобки), в которой указываются типы модификаций интерфейса (на данный момент – только “Language”) и список имен параметров, которые в этой модификации используются. В строках секции имя модификации стоит слева от знака равенства, список имен параметров через запятую – справа. Например, файл “rds.dat” начинается следующим образом:

```
[$Types]
Language=Caption,Text,Hint,Filter,Title,String,List
```

Эта запись означает, что для языков (тип модификации интерфейса “Language”) используются параметры “Caption”, “Text”, “Hint”, “Filter”, “Title”, “String” и “List”.

За секцией “$Types” должна следовать секция “$Params”, в которой указываются типы и целые идентификаторы всех использованных в файле имен параметров. В строках секции слева от знака равенства ставится имя параметра, справа – символ “S” (строка) или “I” (целое число) и, после запятой, целый идентификатор. В “rds.dat” эта секция выглядит так:

```
[$Params]
Caption=S,1
Text=S,2
```

```
Hint=S,3
Filter=S,4
Title=S,5
String=S,6
List=S,7
```

Эта запись означает, что все параметры имеют тип “строка” (символ “S”), параметр “Caption” имеет идентификатор 1, параметр “Text” – идентификатор 2 и т. д.

Далее следует произвольное количество секций следующего формата:

```
[имя_группы]
имя_объекта1.параметр1=значение
имя_объекта1.параметр2=значение
…
имя_объекта2.параметр1=значение
…
```

Каждая секция соответствует одной группе объектов (например, окну), имя группы является именем секции. Далее в строках секции перечисляются все объекты с их параметрами (имя параметра отделено от имени объекта точкой), и, после знака равенства, значение данного параметра данного объекта по умолчанию. Это значение будет попадать в экспортируемый файл перевода (см. п. 4.1), если для него еще нет перевода в двоичном файле.

В “rds.dat” начало секции описания интерфейса пользователя главного окна RDS выглядит так:

```
[TMainForm]
M_File1.Caption=Файл
M_NewSystem.Caption=Новый
M_Templates.Caption=По шаблону
M_LoadSystem.Caption=Загрузить...
M_LoadInCopy.Caption=Загрузить, не закрывая...
M_SaveSystem.Caption=Сохранить
…
```

Здесь “TMainForm” – имя группы, использующееся для описания главного окна. Строка “M_File1.Caption=Файл” означает, что параметр “Caption” (заголовок) объекта “M_File1” (в данном случае это пункт главного меню “Файл”) имеет значение по умолчанию “Файл”.

В любом месте файла может находиться строка комментария, начинающаяся с символа “;” (точка с запятой). При разборе файла описания интерфейса такие строки игнорируются, разработчик может использовать их для того, чтобы оставить в файле какие-то пояснения для переводчика.

Следует учитывать, что для многострочных текстов коды перевода строк заменяются на принятые в языке программирования C сочетания “\n” или “\r\n” (какое именно из этих двух сочетаний использовать решает разработчик, составляющий файл описания). Многострочные тексты чаще всего встречаются в сообщениях пользователю и в описании выпадающих списков. Если с текстами сообщений пользователю все очевидно, то описания выпадающих списков имеют специальный формат и их нужно рассмотреть отдельно.

Рассмотрим выпадающий список выравнивания в параметрах строки текста для картинки блока, описанных в §2.10.5 описания пользователя (рис. 1).

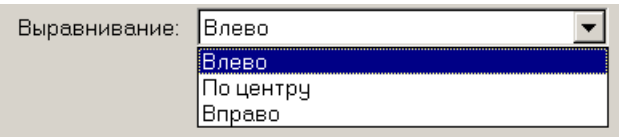

Рис. 1. Выпадающий список с тремя вариантами

Список содержит три варианта: “Влево”, “По центру” и “Вправо”. Соответствующая этому списку запись в файле описания интерфейса пользователя будет выглядеть следующим образом:

```
CB_Alignment.List=0=Влево\n1=По центру\n2=Вправо
```

Здесь “CB_Alignment” – имя объекта, “List” – имя параметра, а многострочный текст после самого левого знака равенства описывает три варианта списка.

Текст “0=Влево\n1=По центру\n2=Вправо” состоит из трех строк: “0=Влево”, “1=По центру” и “2=Вправо” (в качестве разделителя строк в общем тексте используется “\n”). Каждая строка имеет формат “*индекс=значение*”, где “*индекс*” – начинающийся с нуля номер варианта в списке, а “*значение*” – текст этого варианта, видимый пользователю. При переводе этого текста формат строк должен быть сохранен: в качестве переведенного содержимого выпадающего списка в импортируемом файле перевода должен быть записан текст из трех строк формата “*индекс=значение*”, описывающий варианты с индексами 0, 1 и 2 в любом порядке. RDS поддерживает два формата файлов экспорта/импорта перевода (см. п. 4.2 и 4.3), но, независимо от формата файла, сам текст выпадающего списка должен содержать эти три строки с указанием индексов элементов списка. Любые индексы, отличные от заложенных разработчиком в программу, будут игнорироваться: например, если в переведенном тексте будет содержаться строка “3=Other”, в списке не появится дополнительный четвертый пункт.

Если в файле описания необходимо привести очень длинный текст, его для удобства можно разбить на несколько записей. Начало текста, как обычно, записывается после знака равенства в строку “*имя_объекта.параметр*=…”, после которой добавляется новая строка “*имя_объекта.параметр*.1=…” с продолжением текста, затем “*имя_объекта.параметр*.2=…” и т.д пока текст не закончится. В описаниях-продолжениях номера после имени параметра должны идти строго последовательно, их перестановка или вставка между ними других записей не допускается.

Пусть, например, необходимо описать в файле сообщение пользователю со следующим текстом:

> “Для данного блока была создана картинка. Задание прямоугольника с текстом в качестве внешнего вида блока приведет к удалению этой картинки. Задать прямоугольник с текстом?”

Это текст настоящего сообщения RDS, выводимого пользователю при переключении внешнего вида блока в окне его параметров. В файле описания “rds.dat” оно записано как параметр “Text” объекта “PictureToTextBox” в секции “Messages”. Если записать весь этот текст в одну строку, она будет очень длинной, и с ней неудобно будет работать. Поэтому в “rds.dat” описание этого сообщения можно записать следующим образом:

```
PictureToTextBox.Text=Для данного блока была создана картинка.\x20
PictureToTextBox.Text.1=Задание прямоугольника с текстом в качестве\x20
PictureToTextBox.Text.2=внешнего вида блока приведет к удалению этой\x20
PictureToTextBox.Text.3=картинки. Задать прямоугольник с текстом?
```

Здесь длинный текст разбит на четыре отдельных описания: начальное и три продолжения с дополнительными номерами “.1”, “.2” и “.3”. Следует учитывать, что начальные и конечные пробелы в каждой строке выбрасываются, поэтому, чтобы сохранить пробел в месте разбиения текста на отдельные строки, к концу каждой из строк добавлен код пробела “\x20”. Если разбивать текст не по пробелу, то никаких кодов добавлять не нужно. Например, это же описание с тем же самым текстом сообщения может выглядеть следующим образом:

```
PictureToTextBox.Text=Для данного блока была создана картинка
PictureToTextBox.Text.1=. Задание прямоугольника с текстом в качестве в
PictureToTextBox.Text.2=нешнего вида блока приведет к удалению эт
```

```
PictureToTextBox.Text.3=ой картинки. Задать прямоугольник с текстом?
```

Однако, такое описание труднее читать человеку.

3. Создание нового языка

Для создания нового языка и всех связанных с ним папок и служебных файлов (см. п. 1) необходимо запустить RDS, открыть окно настроек пунктом главного меню “Сервис | Настройки RDS” и выбрать в окне вкладку “Инструменты интерфейса” (рис. 2).

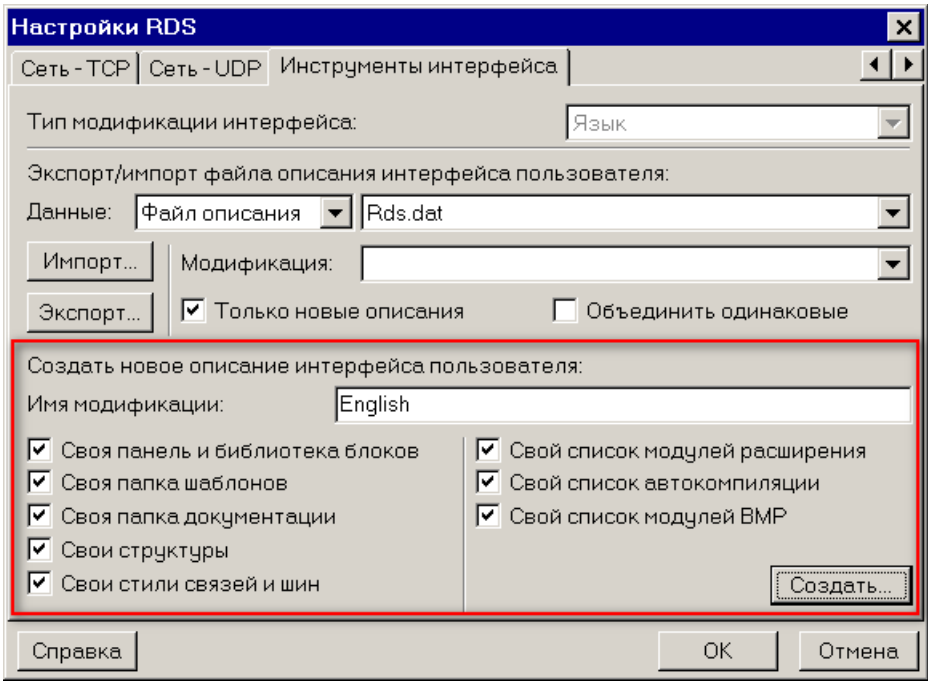

Рис. 2. Панель создания нового языка в окне настроек RDS

Эта вкладка предназначена для работы с любыми модификациями интерфейса пользователя. Пока существует только один тип таких модификаций – “язык”, поэтому выпадающий список выбора типа в верхней части вкладки заблокирован, и все действия на ней будут выполняться только с языками интерфейса пользователя.

В нижней части вкладки находится панель “создать новое описание интерфейса пользователя”, именно там можно добавить в RDS новый язык. В поле ввода “Имя модификации” следует ввести имя создаваемого языка. Это имя будет использоваться и в окне переключения языков (см. п. 6), и в качестве имени папки для хранения данных языка, поэтому имя должно удовлетворять правилам для имен файлов и папок Windows. Имя “Default” использовать нельзя, оно зарезервировано для папки хранения данных интерфейса по умолчанию (в данном случае – русского языка).

Ниже располагается набор флажков, указывающий, какие данные помимо собственно текстов интерфейса пользователя будут храниться независимо для создаваемого языка (эти флажки будут записаны в файл “params.ini” в папке языка, см. п. 1). Флажки управляют следующими данными:

“Своя панель и библиотека блоков” – включение этого флажка указывает на то, что для данного языка будут созданы собственные папки панели и библиотеки блоков (библиотека блоков в настоящее время не используется). Поскольку текущая панель блоков настроена на русский язык, следует включить этот флажок. При создании языка будет предложено скопировать текущую панель блоков внутрь папки данных языка – можно либо согласиться, а затем перевести скопированные блоки вручную, либо отказаться и самостоятельно сформировать другую панель блоков на новом языке.

“Своя папка шаблонов” – включение этого флажка указывает на то, что для данного языка будет использоваться собственная папка с шаблонами схем и автокомпилируемых моделей. В шаблонах могут присутствовать какие-то тексты, поэтому для нового языка лучше иметь свои шаблоны. При создании языка будет предложено скопировать в новую папку шаблонов содержимое текущей папки для дальнейшего перевода.

“Своя папка документации” – включение этого флажка добавит в переключаемые при смене языка данные путь к папке с файлами справки по RDS и по стандартным блокам. Очевидно, что для интерфейса на другом языке требуется и документация на этом языке, однако, пока такая документация отсутствует. Тем не менее, флажок лучше включить, чтобы при переключении языка не выводилась справка на русском.

“Свои структуры” – добавляет в данные языка файл “types.dat”, содержащий описания структур по умолчанию (см. §2.14 описания пользователя). В большинстве случаев этого не требуется, поскольку структуры по умолчанию сейчас нигде не используются, и в описаниях этих структур нет комментариев к полям, т. е. переводить там нечего.

“Свои стили связей и шин” – добавляет в данные языка файл “styles.dat”, содержащий описания стилей связей и шин по умолчанию (см. §2.13 описания пользователя). У этих стилей сейчас русские названия, поэтому флажок лучше включить.

“Свой список модулей расширения” – добавляет в данные языка список модулей расширения (см. §2.19.3 описания пользователя), присутствующих в главном меню RDS как дополнительные пункты подменю “Сервис”. Названия этих дополнительных пунктов, очевидно, нужно переводить, поэтому флажок должен быть включен.

“Свой список автокомпиляции” – добавляет в данные языка список модулей автоматической компиляции моделей (см. §2.19.1 описания пользователя). Самим модулям язык безразличен, поскольку их интерфейс пользователя будет переведен при переводе всех объектов в библиотеке “Common.dll”, но названия модулей, видимые пользователю, могут содержать русские слова, поэтому этот список тоже нужно перевести.

“Свой список модулей BMP” – добавляет в данные языка список модулей преобразования растровых рисунков в другие форматы (см. §2.19.2 описания пользователя). Сейчас существует единственный такой модуль – модуль преобразования в формат PNG, в описании которого есть только название формата “Portable Network Graphics”, которое переводить не требуется. Но, для единообразия, этот флажок тоже лучше включить.

После того, как имя нового языка введено и все необходимые флажки установлены, следует нажать кнопку “Создать”. В процессе создания нового языка может быть задано несколько вопросов о копировании стандартных папок, упомянутых выше, после чего новый язык будет создан и для него можно будет экспортировать и импортировать файлы перевода (см. п. 4), а также вводить для него тексты в редакторе автокомпилируемой модели (см. п. 5).

4. Экспорт и импорт файла перевода программы и стандартных моделей

Экспорт и импорт файла для перевода элементов интерфейса пользователя на другой язык выполняется на той же самой вкладке “Инструменты интерфейса” окна настройки RDS, на которой создаются новые языки. Для этого на ней есть панель “Экспорт/импорт файла описания” (рис. 3).

Экспорт и импорт файлов описания интерфейса (т. е., в данном случае, файлов перевода), делается для одного файла описания из папки “Description\” (см. п. 1) за один раз. Здесь мы рассматриваем перевод “rds.exe” и стандартных моделей блоков (библиотек “Common.dll”, “L57.dll” и др.) Особенности перевода автокомпилируемых моделей рассмотрены в п. 5.

Перевод выполняется в следующей последовательности:

- сначала выбирается DAT-файл описания интерфейса пользователя какой-либо программы или библиотеки, и для него экспортируется текстовый файл перевода на выбранный язык в одном из двух выбранных форматов;
- в текстовом файле перевода русские тексты заменяются на тексты на нужном языке;
- измененный файл перевода импортируется обратно ***для того же самого*** DAT-файла, для которого он был экспортирован ранее.

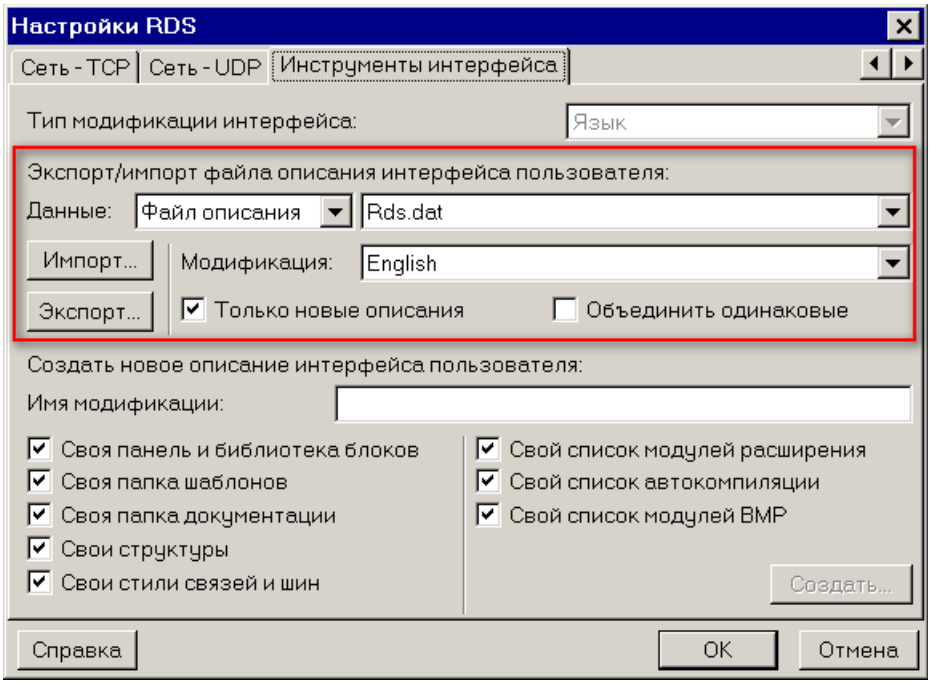

Рис. 3. Панель экспорта/импорта языка в окне настроек RDS

Рассмотрим эти действия подробнее.

4.1. Экспорт файла перевода

Для экспорта файла перевода нужно выбрать в выпадающем списке “Данные” (см. рис. 3) вариант “файл описания”, а справа от него, в другом выпадающем списке, имя файла описания интерфейса, который будет переводиться. В конце концов придется перевести все файлы, но лучше работать с ними по одному.

В выпадающем списке “Модификация” следует выбрать язык, на который будет переводиться интерфейс пользователя. Этот язык должен быть создан заранее, так, как описано в п. 3.

Далее можно выбрать: экспортировать в файл перевода все тексты из выбранного файла описания интерфейса, или только те, которых еще нет в соответствующем двоичном файле внутри папки выбранного языка. Для этого служит флажок “Только новые описания”. Если новый язык только что создан, очевидно, никаких переводов на него еще нет, и установка флажка ни на что не влияет. Однако, если перевод уже был выполнен ранее, но разработчик программы или модели добавил в ее интерфейс пользователя новые объекты, для которых перевода еще нет, разумно будет установить флажок, чтобы уже переведенные тексты не попали в экспортируемый файл. Кроме того, можно установить флажок “Объединять одинаковые”, чтобы одинаковые тексты (например, название кнопки “Шрифт...”, встречающееся в большом числе самых разных окон RDS) не записывались в экспортируемый файл по нескольку раз.

Затем следует нажать кнопку “Экспорт” и выбрать имя для сохраняемого файла, а также один из двух возможных его форматов: “файл INI” или “файл CSV” (эти форматы описаны в п. 4.2 и 4.3 соответственно). Форматы взаимозаменяемы, переводчик может выбрать удобный ему. Единственное ограничение: как и все текстовый файлы в RDS, файлы переводов должны иметь кодировку CP1251, что, на самом деле, ограничивает доступные языки только английским. Но не думаю, что нам когда-нибудь понадобятся другие.

Технически экспорт файла перевода производится следующим образом: в память загружается выбранный файл описания интерфейса (с расширением “dat”), и из него выбрасываются все параметры, имена которых не перечислены через запятую в правой части строки “Language=…” секции “$Types” (см. п. 2). В результате в памяти остаются только параметры, имеющие отношение к переключению языка. Затем для выбранного языка из хранящегося в RDS двоичного файла с переводами (с расширением “bin”) с тем же именем, что и у файла описания, загружаются значения параметров, т. е. уже сделанные переводы. Разумеется, если двоичного файла еще нет, ничего не загружается. Таким образом, значения по умолчанию, загруженные из DAT-файла, заменяются на переводы, если они есть. После этого в текстовый файл перевода записываются либо все параметры с их значениями, либо, если установлен флажок “Только новые описания”, те параметры, которые отсутствовали в двоичном файле перевода.

Весь этот процесс скрыт от пользователя, но ему лучше представлять себе, как формируется файл, с которым ему работать. Пользователь просто выбирает DAT-файл описания и нужный язык, нажимает кнопку “Экспорт”, указывает имя и тип файла для сохранения, и получает текстовый файл, в который ему нужно внести переводы.

4.2. Формат INI-файла перевода

INI-файл перевода по своей структуре очень похож на DAT-файл описания интерфейса пользователя (см. п. 2). У них, фактически, одинаковый формат, только в файле перевода нет секций “$Types” и “$Params” – вместо них он начинается с секции “$Descr”, которой не бывает в DAT-файлах.

Секция “$Descr”, с которой начинается файл перевода, содержит два или три параметра. Она всегда содержит тип модификации интерфейса (“Type=”) и имя той модификации, для которой файл создан (“Mod=”). Кроме того, если файл перевода экспортируется для DAT-файла описания интерфейса (см. п. 2), а не для автокомпилируемых моделей (см. п. 5.3), то в этой секции также будет содержаться имя исходного DAT-файла (“Dat=”), в котором обратная косая черта “\” удвоена. Сохраненное имя DAT-файла используется при импорте файла перевода только для того, чтобы предупредить пользователя, если он позже попытается импортировать этот файл для какого-то другого описания.

При экспорте файла “Dll\Common.dat” для перевода на язык с именем “English”, например, начальная секция файла будет выглядеть так:

```
[$Descr]
Type=Language
Mod=English
Dat=Dll\\Common.dat
```

Здесь “Language” – жестко зашитый в программу и пока единственный тип модификации интерфейса пользователя “язык”, а “Dll\\Common.dat” – имя исходного DAT-файла с замененной обратной косой чертой.

Далее следуют такие же секции для групп объектов интерфейса, как и в DAT-файле:

```
[имя_группы]
имя_объекта1.параметр1=значение
имя_объекта1.параметр2=значение
```

…
имя_объекта2.параметр1=значение
…

Каждая секция соответствует одной группе объектов (например, окну), имя группы является именем секции. Далее в строках секции перечисляются все объекты с их параметрами (имя параметра отделено от имени объекта точкой), и, после знака равенства, значение данного параметра данного объекта. Сразу после экспорта это будет либо значение по умолчанию из DAT-файла, либо, если перевод уже был сделан, предыдущий текст перевода. Задача пользователя – ввести после знаков равенства нужные тексты переводов. Как и в DAT-файлах, для многострочных текстов коды перевода строк необходимо заменять на принятые в языке программирования C сочетания “\n” или “\r\n” (желательно сохранять такую же разбивку на строки, какая была до перевода).

Для приведенного в п. 2 фрагмента файла описания интерфейса “rds.dat” начало секции описания интерфейса пользователя главного окна RDS после перевода может выглядеть так (порядок строк в секции может быть другим):

```
[TMainForm]
M_File1.Caption=File
M_NewSystem.Caption=New
M_Templates.Caption=New by template
M_LoadSystem.Caption=Open...
M_LoadInCopy.Caption=Open separate...
M_SaveSystem.Caption=Save
…
```

Здесь “TMainForm” – имя группы, использующееся для описания главного окна. Строка “M_File1.Caption=File” означает, что параметр “Caption” (заголовок) объекта “M_File1” (в данном случае это пункт главного меню “Файл”) в английском варианте интерфейса имеет значение “File”.

Содержимое выпадающих списков в этом формате указывается точно так же, как и в DAT-файлах, т. е. в виде многострочного текста “0=первый вариант\n1=второй вариант\n…”. Перевод выпадающего списка выравнивания текста из п. 2 будет выглядеть так:

```
CB_Alignment.List=0=Left\n1=Center\n2=Right
```

Поскольку порядок перечисления пунктов списка не важен, можно написать и так:

```
CB_Alignment.List=1=Center\n2=Right\n0=Left
```

Результат будет тем же.

Следует учитывать, что любые индексы, отличные от заложенных в программу разработчиком и указанных в файле описания интерфейса, будут игнорироваться – в выпадающем списке всегда будет столько пунктов, сколько предусмотрено разработчиком, и добавить в список новые пункты через импорт файла перевода невозможно.

Слишком длинные тексты при экспорте могут быть разбиты на несколько описаний с добавлениями к именам параметров дополнительных номеров точно так же, как это описано в п. 2. В процессе перевода можно убрать это разбиение или разбить по-другому, на результат это не влияет.

Если при экспорте был установлен флажок “Объединять одинаковые” (см. рис. 3), то в экспортированном файле могут появиться описания следующего вида:

имя_объекта2.параметр2:=имя_группы1.имя_объекта1.параметр1

Здесь вместо знака равенства используется сочетание двоеточия со знаком равенства. Такая запись означает, что значение параметра “параметр2” объекта “имя_объекта2” из текущей описываемой группы в точности совпадает со значением параметра “параметр1” объекта

“имя_объекта1” группы “имя_групы1” (при этом этот объект “имя_объекта1” со своими параметрами ***обязательно*** должен быть описан в файле до этой записи, т. е. ссылка возможна только на значение уже описанного объекта).

Если, например, экспортировать весь файл описания “rds.dat” в формате INI-файла с включенным флажком “Объединять одинаковые”, пара записей в группе “Messages” (сообщения пользователю) будет выглядеть следующим образом:

```
[Messages]
…
BMPModuleAbout.Text=О модуле
…
ToolModuleAbout.Text:=Messages.BMPModuleAbout.Text
…
```

Здесь указано, что значение параметра “Text” объекта “ToolModuleAbout” (в данном случае это заголовок окна “о модуле” для модуля расширения) в точности совпадает со значением параметра “Text” объекта “BMPModuleAbout” этой же группы “Messages” (заголовок такого же окна для модуля преобразования BMP). В этом примере оба значения находятся в одной группе, но это не обязательно. Главное, чтобы ссылка на значение другого параметра располагалась ниже по тексту файла относительно реального описания этого значения.

Переводчик может либо оставить такую запись без изменения, переведя только текст первого объекта (параметр второго автоматически получит то же значение), либо заменить “**:**=” на “=” и дать второму объекту другой текст перевода.

После редактирования файл перевода должен быть импортирован обратно для слияния с уже имеющимися переводами (см. п. 4.4).

4.3. Формат CSV-файла перевода

CSV-файл перевода похож на файлы переводов схем, но, в отличие от них, он всегда имеет ровно два столбца: служебный столбец и столбец текста. Пользователь может менять тексты только во втором столбце. Первая строка файла – служебная, пользователь не должен менять тексты в обоих ее столбцах.

Данные в файле перевода записываются следующим образом:

RDSINTERFACEDESCRIPTION	CODING\АБВГД\TYPE\Language\MOD*имя_языка…*
группа.объект.параметр	*значение*
группа.объект.параметр	*значение*
…	

В первом столбце первой строки записан текст “RDSINTERFACEDESCRIPTION” – это служебная метка, по которой формат файла опознается при импорте. Во втором столбце закодированы параметры, для которых был экспортирован файл: тип модификации “язык” (после “TYPE\”) и имя языка (после “MOD\”). После “CODING\” в этом столбце записаны русские буквы “АБВГД” для проверки кодировки файла. Если файл перевода экспортируется для DAT-файла описания интерфейса пользователя (см. п. 2), а не для автокомпилируемых моделей (см. п. 5.3), то во втором столбце после “DAT\” будет содержаться имя исходного DAT-файла, в котором обратная косая черта “\” заменена на звездочку “*”. Сохраненное имя DAT-файла используется при импорте файла перевода только для того, чтобы предупредить пользователя, если он позже попытается импортировать этот файл для какого-то другого описания.

Далее во всех строках в первом столбце записаны имена группы объектов (окна или раздела интерфейса), конкретного объекта (поля ввода, пункта меню) и параметра этого

объекта (заголовок, подсказка и т. п.), разделенные точками. Во втором столбце записан текст значения параметра, который пользователь может менять при переводе.

При экспорте файла все тексты берутся в двойные кавычки. По спецификации формата это не обязательно, но так надежнее. При загрузке файла RDS следует спецификациям RFC 4180: в кавычках должны быть тексты, содержащие двойные кавычки, символы разделителя и разрывы строк. В качестве разрыва строки используется комбинация CRLF (0x10, 0x12).

Для приведенного в п. 2 фрагмента файла описания интерфейса пользователя “rds.dat” часть CSV-файла после перевода может выглядеть так (порядок строк может быть другим):

```
"RDSINTERFACEDESCRIPTION","CODING\АБВГД\TYPE\Language\MOD\English\DAT\Rds
.dat"
"TMainForm.M_File1.Caption","File"
"TMainForm.M_NewSystem.Caption","New"
"TMainForm.M_Templates.Caption","New by template"
"TMainForm.M_LoadSystem.Caption","Open..."
"TMainForm.M_LoadInCopy.Caption","Open separate..."
"TMainForm.M_SaveSystem.Caption","Save"
…
```

В данном случае “…MOD\English” во втором столбце первой строки указывает на то, что файл экспортирован для языка “English”, а “…DAT\Rds.dat” – на то, что файл экспортирован для описания интерфейса “rds.dat”.

Текст “TMainForm.M_File1.Caption” в первом столбце второй строки говорит о том, что эта строка описывает параметр “Caption” (заголовок) объекта “M_File1” (в данном случае это пункт главного меню “Файл”) главного окна RDS (“TMainForm”). Во втором столбце этой строки записан заголовок этого пункта меню в английском варианте – “File”.

Списки вариантов выпадающих списков в CSV-файлах описываются многострочными текстами, состоящими из строк вида “*индекс=значение*”, где “*индекс*” – начинающийся с нуля номер варианта в списке, а “*значение*” – текст этого варианта, видимый пользователю. Точно так же эти списки описываются в DAT- (см. п. 2) и INI-файлах (см. п. 4.2), но в CSV-файлах вместо символа перевода строки “\n” используется *настоящий* перевод строки. Например, описание выпадающего списка из DAT-файла вида

```
[TTextProperties]
…
CB_Alignment.List=0=Влево\n1=По центру\n2=Вправо
```

после экспорта в CSV-файл будет выглядеть следующим образом:

```
"TTextProperties_Frame.CB_Alignment.List","0=Влево
1=По центру
2=Вправо"
```

Следует учитывать, что любые строки с индексами, отличными от заложенных в программу разработчиком и указанных в файле описания интерфейса, будут игнорироваться – в выпадающем списке всегда будет столько пунктов, сколько предусмотрено разработчиком, и добавить в список новые пункты через импорт файла перевода невозможно.

Если при экспорте был установлен флажок “Объединять одинаковые” (см. рис. 3), то в первом столбце экспортированного файла может быть указано несколько описаний объектов с одинаковым значением параметра – по одному описанию на строке. Например, запись:

```
"Messages.BMPModuleAbout.Text
Messages.ToolModuleAbout.Text","О модуле"
```

указывает на то, что одно и то же значение “О модуле” имеют параметры “Text” объектов “BMPModuleAbout” и “ToolModuleAbout” секции “Messages”. Если в этом файле перевести

текст “О модуле”, при импорте файла он будет присвоен этим двум объектам одновременно. Переводчик может, при желании, вручную разделить эту запись на две, соблюдая правила формата CSV, и дать параметрам разные значения:

```
"Messages.BMPModuleAbout.Text","About BMP module"
"Messages.ToolModuleAbout.Text","About tool module"
```

После редактирования CSV-файл должен быть импортирован обратно для слияния с уже имеющимися переводами (см. п. 4.4).

4.4. Импорт файла перевода

После того, как в экспортированный файл внесены переводы всех необходимых текстов, этот файл необходимо импортировать обратно. Для этого на вкладке “Инструменты интерфейса” окна настройки RDS следует выбрать в выпадающем списке “Данные” (см. рис. 3) вариант “файл описания”, а справа от него, в другом выпадающем списке, имя файла описания интерфейса, для которого будет импортироваться перевод. ***Здесь важно не ошибиться***: импортировать нужно для того же файла описания, для которого перевод экспортировался (если выбранный в списке файл описания не совпадает с указанным в файле перевода, будет выведено предупреждение). В выпадающем списке модификации язык выбирать не нужно – имя языка записано в самом импортируемом файле.

Далее следует нажать кнопку “Импорт” и выбрать файл, который будет загружаться. После импорта переводы из этого файла заменят уже имеющиеся в RDS или добавятся, если раньше их не было. Сам импортированный файл можно стирать – он больше не нужен.

Технически импорт файла перевода производится следующим образом: сначала из выбранного пользователем файла считывается заголовок и определяется, для какого именно языка этот файл был создан. Затем в память загружается выбранный файл описания интерфейса (с расширением “dat”), из которого выбрасываются все параметры, имена которых не перечислены через запятую в правой части строки “Language=…” секции “$Types” (см. п. 2). Затем загружается имеющийся на данный момент двоичный файл перевода для считанного языка, если он есть, и в него из пользовательского файла добавляются значения тех параметров, которые есть в файле описания. Таким образом, невозможно импортировать параметры или объекты, которых нет в DAT-файле описания интерфейса. После слияния с пользовательской информацией обновленный двоичный файл перевода автоматически сохраняется. Теперь при следующем запуске RDS для данного языка будут использоваться новые переводы.

5. Особенности перевода автокомпилируемых моделей

В автокомпилируемых моделях тексты на разных языках хранятся не в отдельных двоичных файлах, а в самих файлах моделей. Эти тексты можно переводить через процедуру экспорта/импорта, похожую на описанную в п. 4 (формат файлов тот же), либо сразу вводить через интерфейс редактора модели при ее создании. Ниже рассматриваются оба способа.

5.1. Перевод текстов окна настройки блока в редакторе модели

В автокомпилируемой модели к интерфейсу пользователя можно отнести только окно настройки блока, создаваемое средствами редактора модели (см. §3.6.6 описания пользователя) и, возможно, какие-то сообщения, выводимые пользователю в отдельных окнах. Перевод сообщений рассмотрен в п. 5.2, а здесь рассмотрим перевод элементов окна настройки.

В окнах настройки блока может быть четыре текстовых элемента, требующих перевода:

- заголовок самого окна;
- названия вкладок;
- заголовки полей ввода и надписи;
- варианты выбора для выпадающих списков.

В редакторе модели рядом с полями ввода для этих элементов располагаются кнопки с многоточием (рис. 4 и 5), нажатие которых позволяет ввести тексты элементов на разных языках (разумеется, языки уже должны быть добавлены в RDS согласно п. 3).

При нажатии на любую из этих кнопок открывается окно (рис. 6), в левой части которого располагается список существующих языков (включая вариант “по умолчанию”), а справа – поле ввода текста для выбранного в списке слева языка. В файле модели будут сохранены все введенные тексты, при этом в “основном” интерфейсе редактора модели, изображенном на рис. 4 и 5, будет отображаться и редактироваться значение для текущего языка.

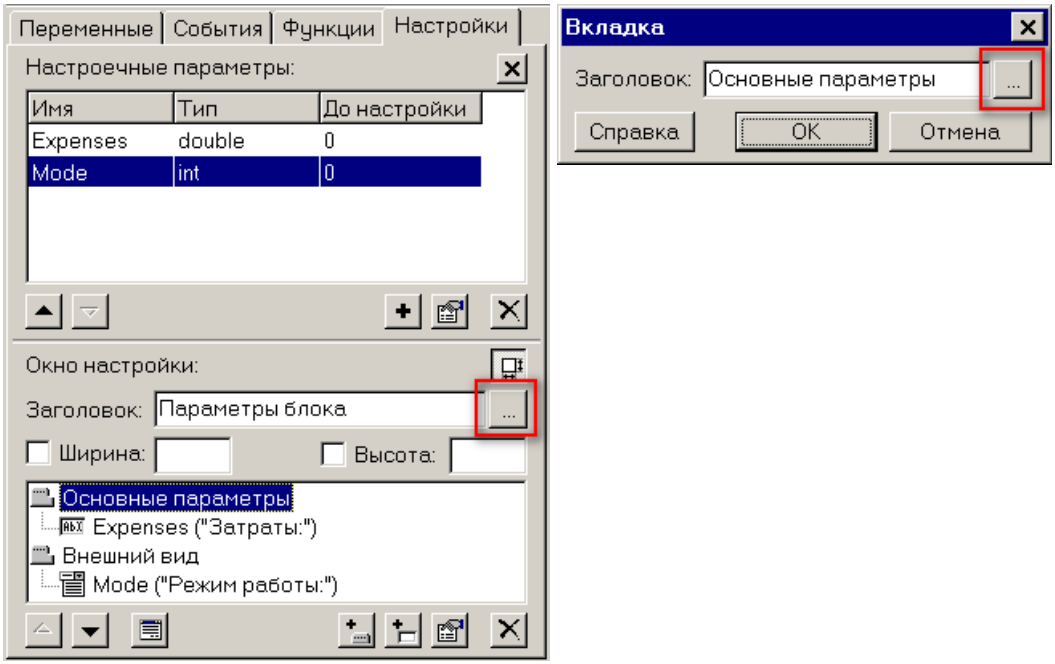

Рис. 4. Кнопки для задания текстов на разных языках для заголовка окна настроек (слева) и названия вкладки (справа)

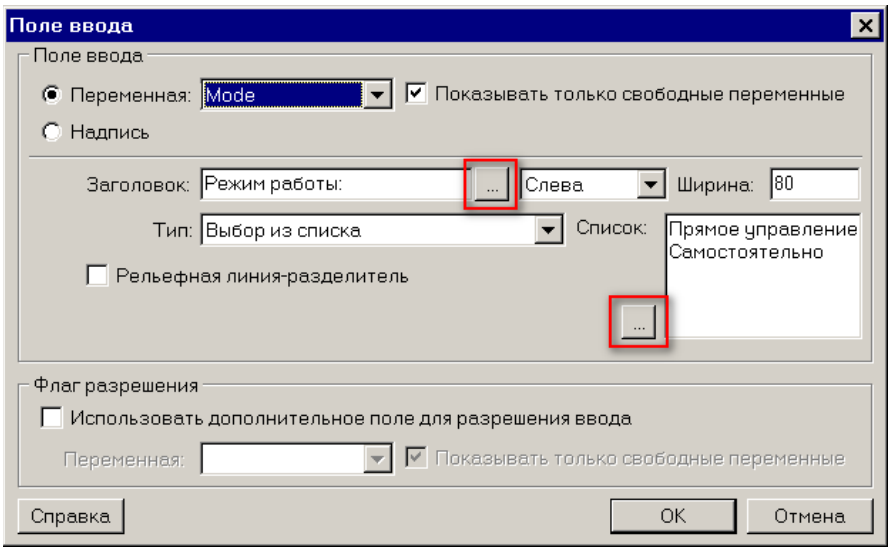

Рис. 5. Кнопки для задания текстов на разных языках для заголовка поля ввода (левее) и списка вариантов выпадающего списка (правее и ниже)

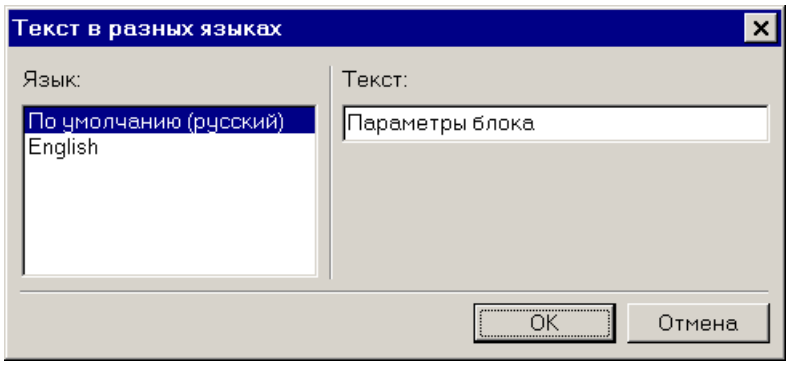

Рис. 6. Окно ввода текстов на разных языках

При смене языка интерфейса пользователя RDS (см. п. 6) в файле модели ничего автоматически изменено не будет. Но при следующей загрузке схемы с этой моделью модуль автокомпиляции обнаружит, что DLL для модели была скомпилирована раньше изменения языка. Это приведет к тому, что DLL модели будет перекомпилирована, а при этом в момент формирования текста программы для компиляции в этот текст будут записаны строки для текущего, т. е. нового, языка. Именно таким образом меняется язык интерфейса пользователя в автокомпилируемых моделях – через перекомпиляцию. Поскольку смена языка выполняется не так уж часто, считаю это приемлемым вариантом.

5.2. Перевод прочих текстовых сообщений

Допустим, где-то в программе автокомпилируемой модели при выполнении (или невыполнении) какого-то условия пользователю выдается окно с сообщением при помощи вызова функции Windows API “MessageBox” или соответствующей ей функции RDS “rdsMessageBox” (см. приложение А.5.5.6 к руководству программиста):

```
rdsMessageBox("Выберите режим работы","Ошибка",
              MB_OK|MB_ICONWARNING);
```

Здесь строки “Выберите режим работы” и “Ошибка” встроены непосредственно в программу модели и не будут зависеть от текущего языка. Для того, чтобы такие строки тоже можно было переводить, в модель можно вводить именованные текстовые константы с разными текстами для разных языков, которые в программе используются по именам. Для ввода или редактирования этих текстовых констант в редакторе модели служит пункт меню “Модель | Текстовые константы”. При его выборе открывается окно, изображенное на рис. 7.

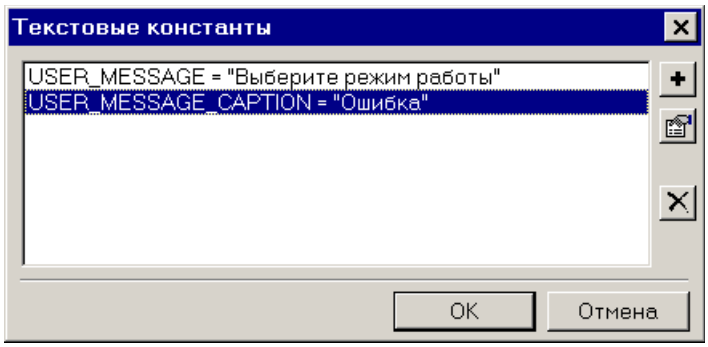

Рис. 7. Окно работы с текстовыми константами

Большую часть окна занимает список констант формата “*имя* = *значение*”, справа находятся кнопки для добавления, изменения и удаления констант. При добавлении константы или редактировании уже существующей открывается окно, изображенное на рис. 8.

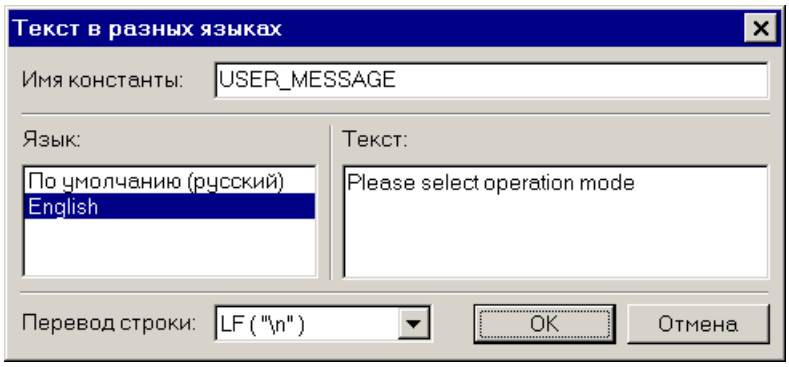

Рис. 8. Редактирование текстов константы

В верхней части окна располагается поле для ввода имени константы, которое будет использоваться в программе модели. Это имя должно удовлетворять правилам языка программирования C, не должно совпадать с именами других используемых в модели объектов (переменных блока, настроечных параметров и т. п.), а также не должно начинаться с букв “rds”, чтобы не пересекаться со служебными функциями и константами.

Ниже слева находится список существующих языков вместе с вариантом “по умолчанию”, как в похожем окне на рис. 6. Справа – многострочное поле для ввода текста значения константы для выбранного в списке слева языка. В нижней левой части окна находится выпадающий список “перевод строки”, в котором можно выбрать разделитель строк для многострочных констант: либо LF (“\n”), либо CRLF (“\r\n”). Какую именно комбинацию выбрать, зависит от того, для чего константа будет применяться, и это должен решать программист-создатель модели.

На рис. 8 в список введены две константы с именами “USER_MESSAGE” и “USER_MESSAGE_CAPTION”. Приведенный выше вызов функции вывода сообщения с использованием этих констант будет выглядеть так:

```
rdsMessageBox(USER_MESSAGE,USER_MESSAGE_CAPTION,
              MB_OK|MB_ICONWARNING);
```

При формировании текста программы для компиляции модели для каждой из этих констант будет автоматически вставлен оператор “`#define`” со значением для текущего языка, поэтому при смене языка и перекомпиляции модели текст сообщения будет автоматически изменяться.

5.3. Экспорт/импорт файла перевода для всех автокомпилируемых моделей схемы

Все тексты интерфейса пользователя всех автокомпилируемых моделей какой-либо схемы, описанные в п. 5.1 и 5.2, могут быть выгружены в файл перевода, переведены все вместе, а потом загружены обратно – примерно так же, как переводятся тексты стандартных, не автокомпилируемых моделей (см. п. 4). Для этого нужно загрузить схему, а затем открыть окно настроек RDS и выбрать в нем вкладку “Инструменты интерфейса” (рис. 9).

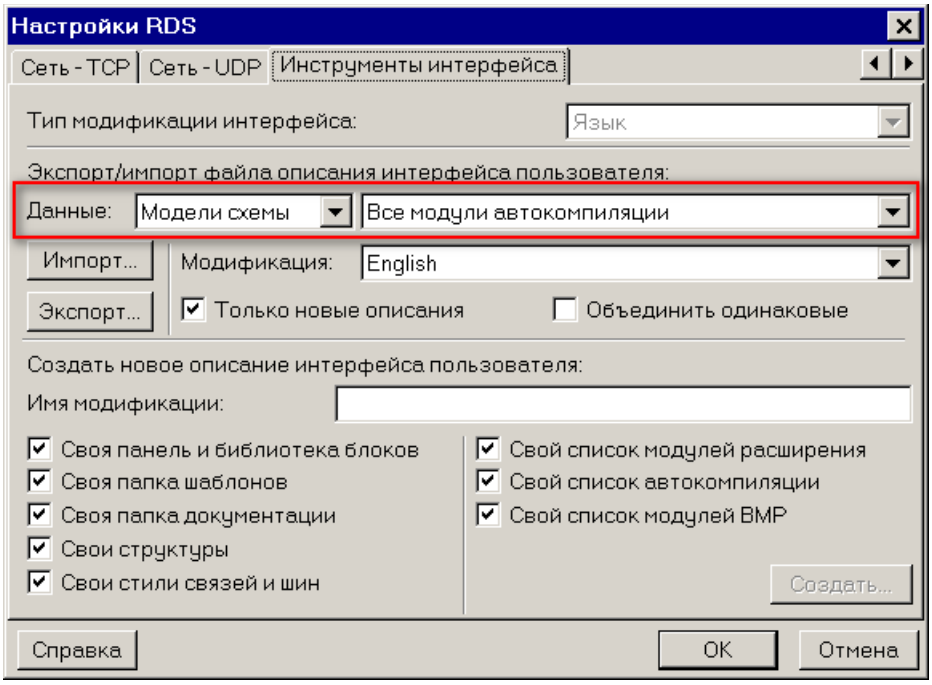

Рис. 9. Экспорт/импорт текстов автокомпилируемых моделей

На этой вкладке на панели “Экспорт/импорт файла описания” в выпадающем списке “Данные” нужно выбрать не “файл описания”, как при переводе программ и стандартных моделей, а “модели схемы”. После этого в выпадающем списке справа можно выбрать модуль автокомпиляции, модели которого будут переводиться. Лучше выбрать вариант “все модули автокомпиляции”, поскольку все существующие на данный момент модули, даже если в схеме их будет использовано несколько (что маловероятно и не рекомендуется), совместимы по формату данных.

Затем в выпадающем списке “Модификация” нужно выбрать язык, для которого экспортируется файл, по желанию включить флажок “только новые описания” (в этом случае в файл будут записаны только тексты, для которых еще нет перевода), нажать кнопку “Экспорт” и выбрать имя и формат записываемого файла. Форматы файлов перевода описаны в п. 4.2 и 4.3, для автокомпилируемых моделей они ничем не отличаются. Имя группы объектов в этом файле формируется из имени модели, имя объекта – это имя текстовой константы, настроечного параметра или автоматически сгенерированный идентификатор для заголовков вкладок и надписей.

После перевода файла его загрузка и слияние с моделями схемы делается точно так же: загружается схема, открывается окно настроек, в списке “Данные” выбирается вариант “модели схемы”, в списке справа от него – “все модули автокомпиляции”, нажимается кнопка “Импорт” и выбирается импортируемый файл.

6. Выбор языка интерфейса пользователя

Для установки текущего языка интерфейса пользователя RDS следует выбрать пункт главного меню “Сервис | Язык интерфейса”. При этом откроется окно со списком доступных языков (рис. 10).

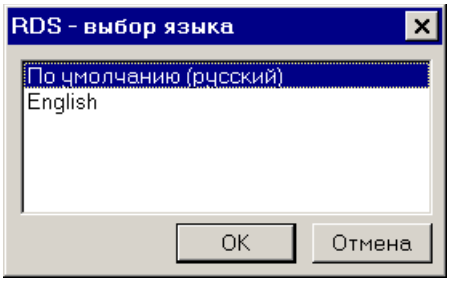

Рис. 10. Окно выбора языка интерфейса пользователя

В этом списке следует выбрать желаемый язык и нажать кнопку “OK”. При этом окно закроется и будет выведено сообщение о том, что язык будет установлен при следующем запуске RDS. Если никакая схема в данный момент не загружена, RDS завершится автоматически.

Если в RDS не создано ни одного языка, пункт меню будет отсутствовать.

7. Включение функций загрузки перевода в модели блоков

Для того, чтобы надписи в окнах настроек блоков или сообщения пользователю, выводимые блоками, всегда показывались на текущем выбранном в RDS языке, необходимо вместо жестких строковых констант (строк в двойных кавычках) использовать в моделях этих блоков вызовы функций, которые будут загружать тексты из двоичных файлов перевода для текущего языка. Ниже рассматриваются эти новые, появившиеся в версии 1.0.520, функции (п. 7.2), а также вспомогательный класс, сделанный на их основе (п. 7.3), который можно использовать в моделях, написанных на C++. В п. 7.4 приведены примеры моделей с переводом интерфейса пользователя при помощи этих функций и вспомогательного класса.

7.1. Константы для стандартных папок

Для стандартных папок, в которых хранятся файлы описания интерфейса (“Description”, “Interface” и т.д, см. п. 1) в файл “RdsDef.h” добавлены следующие строковые константы:

`#define RDS_UIMODD_DEFAULT "Default"`

Папка с параметрами для языка по умолчанию.

`#define RDS_UIMODD_DESCR "Description"`

Папка описаний интерфейса пользователя внутри папки установки RDS.

`#define RDS_UIMODD_MODS "Interface"`

Папка с данными разных типов модификаций интерфейса пользователя (включая язык) внутри папки установки RDS.

`#define RDS_UIMODC_LANGUAGE "Language"`

Папка с данными разных языков интерфейса пользователя внутри папки модификаций (`RDS_UIMODD_MODS`).

Например, путь к папке с данными языка с названием “English” в вызовах различных сервисных функций RDS можно записать так:

```
"$RDS$\\" RDS_UIMODD_MODS "\\" RDS_UIMODC_LANGUAGE "\\English"
```

(здесь “RDS” – символическое имя для папки установки RDS, которое понимают все сервисные функции).

Путь к папке с данными текущего языка интерфейса для передачи в сервисные функции можно указывать с помощью нового символического имени “$CURLANG$”.

7.2. Новые сервисные функции и константы RDS

Для поддержки перевода интерфейса пользователя в RDS был добавлен новый тип вспомогательного объекта (`RDS_HOBJECT`, см. приложение A.5.22.1) и функции для работы с ним. Кроме того, добавлены новые функции общего назначения и новые возможности к уже имеющимся.

7.2.1. Получение имени текущего языка

Имя текущего выбранного языка интерфейса пользователя RDS (это имя совпадает с именем папки, в которой хранятся его данные) можно получить при помощи вызова функции `rdsGetSystemPath` (приложение A.5.2.25) с новой константой `RDS_GSPCURLANGUAGE`:

```
char *lang=rdsGetSystemPath(RDS_GSPCURLANGUAGE);
```

Если выбран не язык по умолчанию, функция возвращает указатель на строку с именем языка во внутреннем буфере RDS (эту строку изменять нельзя). Если выбран язык по умолчанию, функция возвращает указатель на пустую строку.

Название языка по умолчанию, пригодное для показа пользователю, можно получить при помощи вызова той же функции с константой `RDS_GSPDEFLANGTEXT`:

```
char *deftext=rdsGetSystemPath(RDS_GSPDEFLANGTEXT);
```

Для всех прочих языков название для пользователя совпадает с именем языка.

7.2.2. rdsGetUICurrentMod – параметры текущей модификации интерфейса пользователя

Функция rdsGetUICurrentMod считывает параметры текущей модификации интерфейса пользователя заданного типа. На данный момент существует единственный тип модификации интерфейса – язык.

```
LPSTR RDSCALL rdsGetUICurrentMod(
     LPCSTR category,      // Тип модификации
     FILETIME *changetime  // Возвращаемое время изменения
);
```

Тип указателя на эту функцию:

```
RDS_SSFt
```

Параметры:

`category`

Строка с именем папки для запрашиваемого типа модификации интерфейса. Сейчас поддерживается только “Language” (константа RDS_UIMODC_LANGUAGE, см. п. 7.1), т. е. язык интерфейса.

`changetime`

Указатель на структуру FILETIME, в которую функция записывает время последнего изменения интерфейса по данному типу модификации. Для языка это будет временем последнего переключения языка в главном меню RDS, эта информация может быть нужна модулям автокомпиляции для проверки необходимости изменения языка в обслуживаемых моделях (см. п. 5). Если время изменения языка не нужно, в этом параметре можно передать NULL.

Возвращаемое значение:

Указатель на строку с именем текущей модификации (языка) во внутреннем буфере RDS. Если выбрана модификация по умолчанию, возвращается указатель на пустую строку.

7.2.3. rdsGetUIModsList – список модификаций интерфейса пользователя

Функция rdsGetUIModsList возвращает список существующих в RDS модификаций интерфейса пользователя заданного типа. На данный момент существует единственный тип модификации интерфейса – язык.

```
LPSTR RDSCALL rdsGetUIModsList(
     LPCSTR category       // Тип модификации
);
```

Тип указателя на эту функцию:

```
RDS_SS
```

Параметр:

`category`

Строка с именем папки для запрашиваемого типа модификации интерфейса. Сейчас поддерживается только “Language” (константа RDS_UIMODC_LANGUAGE, см. п. 7.1), т. е. язык интерфейса.

Возвращаемое значение:

Динамически сформированный текст со списком существующих модификаций, разделенных кодом перевода строки “\n”.

Примечание:

Эта функция может использоваться для предъявления пользователю списка доступных модификаций для выбора одной из них. Например, в стандартном модуле автокомпиляции она используется для формирования списка языков в левой части окна работы с переводами текстов модели (рис. 6).

Модификация по умолчанию в этот список не входит, в нем будут только дополнительные модификации. Таким образом, если, например, в RDS есть русский (по умолчанию) и английский (папка “English”) языки, то функция вернет текст “English”.

Динамическая строка, созданная функцией `rdsGetUIModsList`, должна быть ***обязательно*** освобождена функцией `rdsFree` (приложение А.5.4.8).

7.2.4. `rdsUIMODCreate` – создание объекта для работы с модификациями интерфейса

Функция `rdsUIMODCreate` создает вспомогательный объект RDS, который может загружать данные из двоичных файлов модификаций интерфейса (файлов перевода) и возвращать значения параметров из этих файлов.

```
RDS_HOBJECT RDSCALL rdsUIMODCreate(void);
```

Тип указателя на эту функцию:

```
RDS_HoV
```

Возвращаемое значение:

Уникальный идентификатор созданного объекта (`RDS_HOBJECT`).

Примечания:

Эта функция создает вспомогательный объект, при помощи которого можно получать тексты на текущем языке для разных элементов интерфейса пользователя. Тексты загружаются из двоичного файла, созданного так, как описано в п. 4. Этот же объект может использоваться и для экспорта/импорта файла перевода. Здесь будет описано только его использование для перевода текстов в моделях блоков, поскольку программный экспорт/импорт файлов нужен только при создании пользовательских модулей автокомпиляции.

Для уничтожения объекта используется стандартная функция `rdsDeleteObject` (см. приложение А.5.22.4).

7.2.5. `rdsUIMODLoadGroup` – загрузка группы

Функция `rdsUIMODLoadGroup` загружает во вспомогательный объект группу объектов интерфейса из указанного двоичного файла. После загрузки можно считывать различные параметры объектов из этой группы.

```
BOOL RDSCALL rdsUIMODLoadGroup(
     RDS_HOBJECT mod, // Объект
     LPCSTR file,     // Имя двоичного файла
     LPCSTR group     // Имя группы
);
```

Тип указателя на эту функцию:

```
RDS_BHoSS
```

Параметры:

`mod`

Идентификатор объекта, ранее созданного вызовом `rdsUIMODCreate` (п. 7.2.4).

`file`

Строка с именем двоичного файла, из которого нужно загрузить группу. В этой строке можно использовать символически обозначения папок RDS, включая “$CURLANG$”. Например, для загрузки данных из двоичного файла “MyDll.bin” в подпапке “Dll\” папки текущего языка интерфейса можно использовать строку “$CURLANG$\\Dll\\MyDll.bin”.

`group`

Строка с именем загружаемой группы.

Возвращаемое значение:

Успешность загрузки (`TRUE`, если существует указанный файл и в нем есть данные указанной группы).

Примечание:

Эта функция обычно используется в тех случаях, когда предполагается работа с несколькими объектами. Например, при открытии окна настройки блока имеет смысл загрузить группу, в которой находятся тексты заголовков объектов, соответствующих полям ввода этого окна, а затем считывать из загруженных данных тексты для каждого поля.

Все тексты, загруженные в объект `mod` до вызова этой функции, стираются из памяти и заменяются загруженными.

7.2.6. `rdsUIMODLoadSingleObject` – загрузка одного объекта

Функция `rdsUIMODLoadSingleObject` загружает во вспомогательный объект данные единственного указанного в параметрах объекта интерфейса из указанного двоичного файла. После загрузки можно считывать различные параметры этого объекта.

```
BOOL RDSCALL rdsUIMODLoadSingleObject(
     RDS_HOBJECT mod, // Объект
     LPCSTR file,     // Имя двоичного файла
     LPCSTR group,    // Имя группы
     LPCSTR object    // Имя объекта
);
```

Тип указателя на эту функцию:

```
RDS_BHoSSS
```

Параметры:

`mod`

Идентификатор объекта, ранее созданного вызовом `rdsUIMODCreate` (п. 7.2.4).

`file`

Строка с именем двоичного файла, из которого нужно загрузить группу. В этой строке можно использовать символически обозначения папок RDS, включая “$CURLANG$”. Например, для загрузки данных из двоичного файла “MyDll.bin” в подпапке “Dll\”

папки текущего языка интерфейса можно использовать строку “$CURLANG$\\Dll\\MyDll.bin”.

`group`

Строка с именем загружаемой группы.

`object`

Строка с именем объекта.

Возвращаемое значение:

Успешность загрузки (`TRUE`, если существует указанный файл и в нем есть данные указанного объекта в указанной группе).

Примечание:

Эта функция обычно используется в тех случаях, когда предполагается работа с единственным объектом. Например, при выводе сообщения пользователю обычно достаточно загрузить только текст этого сообщения, не загружая остальные объекты в группе, в которой этот текст находится. Если объект был загружен успешно, в последующих вызовах функций чтения значений его параметров имя объекта можно не указывать (передается `NULL`).

Все тексты, загруженные в объект `mod` до вызова этой функции, стираются из памяти и заменяются загруженными.

7.2.7. `rdsUIMODGetParamString` – получение текста значения параметра из считанных данных

Функция `rdsUIMODGetParamString` возвращает строку значения указанного параметра указанного объекта из уже загруженной группы (см. `rdsUIMODLoadGroup`, п. 7.2.5) или одиночного объекта (см. `rdsUIMODLoadSingleObject`, п. 7.2.6).

```
BOOL RDSCALL rdsUIMODGetParamString(
     RDS_HOBJECT mod, // Объект
     LPCSTR object    // Имя объекта
     int paramid,     // Идентификатор параметра
     LPCSTR defval    // Значение по умолчанию
);
```

Тип указателя на эту функцию:

```
RDS_SHoSIS
```

Параметры:

`mod`

Идентификатор объекта, ранее созданного вызовом `rdsUIMODCreate` (п. 7.2.4).

`object`

Строка с именем объекта или `NULL` для последнего объекта, из которого читались данные.

`paramid`

Целый идентификатор параметра или константа `RDS_UIMOD_ANYPARAMID` для чтения значения первого попавшегося параметра (используется в тех случаях, когда у объекта есть только один параметр). Идентификаторы параметров указываются в секции “$Params” файла описания интерфейса (см. п. 2).

`defval`

Строка, возвращаемая при отсутствии параметра, объекта или любых загруженных данных.

Возвращаемое значение:

Если данные из двоичного файла перевода были ранее загружены успешно, в этих данных есть указанный объект, и у этого объекта есть указанный параметр, возвращается указатель на строку во внутренних данных объекта `mod` с текстовым значением этого параметра. Эта строка будет существовать до уничтожения объекта или до загрузки в него новых данных. Если же указанного параметра нет, возвращается переданный в параметрах функции указатель `defval`.

Примечание:

Это основная функция получения переведенных строк из загруженных данных. Строка на языке по умолчанию (русском) передается в параметре `defval`. Если перевод успешно загружен, возвращается переведенная строка, если нет – строка по умолчанию.

7.2.8. `rdsUIMODGetParamListString` – получение текста содержимого выпадающего списка из считанных данных

Функция `rdsUIMODGetParamListString` преобразует строку описания выпадающего списка формата “0=вариант1\n1=вариант2\n…” из указанного параметра указанного объекта уже загруженной группы (см. `rdsUIMODLoadGroup`, п. 7.2.5) или одиночного объекта (см. `rdsUIMODLoadSingleObject`, п. 7.2.6) в строку вида “вариант1\nвариант2\n…”, пригодную для передачи в оконные объекты RDS (см. приложение А.5.28.18).

```
BOOL RDSCALL rdsUIMODGetParamListString(
     RDS_HOBJECT mod, // Объект
     LPCSTR object    // Имя объекта
     int paramid,     // Идентификатор параметра
     LPCSTR defval    // Значение по умолчанию
);
```

Тип указателя на эту функцию:

```
RDS_SHoSIS
```

Параметры:

`mod`

Идентификатор объекта, ранее созданного вызовом `rdsUIMODCreate` (п. 7.2.4).

`object`

Строка с именем объекта или `NULL` для последнего объекта, из которого читались данные.

`paramid`

Целый идентификатор параметра или константа `RDS_UIMOD_ANYPARAMID` для чтения значения первого попавшегося параметра (используется в тех случаях, когда у объекта есть только один параметр). Идентификаторы параметров указываются в секции “$Params” файла описания интерфейса (см. п. 2).

`defval`

Строка, возвращаемая при отсутствии параметра, объекта или любых загруженных данных.

Возвращаемое значение:

Если данные из двоичного файла перевода были ранее загружены успешно, в этих данных есть указанный объект, и у этого объекта есть указанный параметр, из этого параметра считывается текст описания выпадающего списка формата “0=вариант1\n1=вариант2\n…” (см. п. 2). Затем в переданном тексте `defval` формата “вариант1\nвариант2\n…” отдельные строки заменяются на считанные согласно индексам перед знаком равенства, получившаяся строка сохраняется во внутренних данных объекта `mod`, и возвращается указатель на нее. Эта строка будет существовать до уничтожения объекта или до загрузки в него новых данных. Если же указанного параметра в загруженных данных нет, возвращается переданный в параметрах функции указатель `defval`.

Примечание:

Эта функция используется для перевода вариантов выпадающих списков для полей ввода RDS в окнах настройки блоков. Список вариантов для поля задается в виде текста, где варианты разделены кодом перевода строки “\n”. В файлах перевода списки вариантов задаются похожим образом, но перед текстом самого варианта указывается его номер и знак равенства. Функция `rdsUIMODGetParamListString` переводит один формат в другой. В параметре `defval` передается список вариантов на языке по умолчанию. Если перевод успешно загружен, возвращается переведенный список вариантов в том же формате, если нет – текст по умолчанию из `defval`.

Пусть, например, список вариантов на русском языке выглядит как “Ручной\nАвтоматический”. Этот текст задает список из двух вариантов: “Ручной” с индексом 0 (самый первый в списке) и “Автоматический” с индексом 1 (второй в списке). Пусть в файле перевода для этого списка хранится строка “1=Automatic\n0=Manual\n2=Other”. Вызов функции `rdsUIMODGetParamListString` в этом случае вернет указатель на текст “Manual\nAutomatic”: в исходном списке строка с индексом 0 заменена на “Manual”, строка с индексом 1 – на “Automatic”. Вариант с индексом 2 (“Other”) из файла перевода будет проигнорирован, поскольку в исходном списке вариантов строки с таким индексом нет.

7.2.9. Команда `RDS_UIMOD_CLEAR` – очистка загруженных данных

Команда `RDS_UIMOD_CLEAR` очищает все загруженные из файла перевода в объект данные.

Вызов команды:

```
rdsCommandObject(mod,RDS_UIMOD_CLEAR);
```

Параметр:

`mod`

Идентификатор объекта, ранее созданного вызовом `rdsUIMODCreate` (п. 7.2.4).

Примечание:

После очистки данных вызовы функций `rdsUIMODGetParamString` (п. 7.2.7) и `rdsUIMODGetParamListString` (п. 7.2.8) будут возвращать переданные в них значения по умолчанию до тех пор, пока в объект не будут загружены новые данные.

7.2.10. Команда `RDS_UIMOD_SETCRLFCORR` – установка способа обработки переводов строк

Команда `RDS_UIMOD_SETCRLFCORR` устанавливает способ обработки переводов строк в загружаемых текстах. Команду нужно отдавать до загрузки данных, т. е. перед вызовом функции `rdsUIMODLoadGroup` (п. 7.2.5) или `rdsUIMODLoadSingleObject` (п. 7.2.6).

Вызов команды:

```
int iMode=... // Константа RDS_UIMOD_CRLF_*
rdsSetObjectInt(mod,RDS_UIMOD_SETCRLFCORR,0,iMode);
```

Параметры:

`mod`

Идентификатор объекта, ранее созданного вызовом `rdsUIMODCreate` (п. 7.2.4).

`iMode`

Одна из констант `RDS_UIMOD_CRLF_*`, указывающая способ обработки переводов строк:

`RDS_UIMOD_CRLF_ANY`	Переводы строк в текстах никак не обрабатываются и остаются в точности такими, какими их записал создатель двоичного файла (переводчик).
`RDS_UIMOD_CRLF_LF`	Переводы строк вида “\r\n” заменяются на одиночный код “\n”.
`RDS_UIMOD_CRLF_BOTH`	Переводы строк вида “\n” заменяются на сочетание “\r\n”.
`RDS_UIMOD_CRLF_NONE`	Любые переводы строк заменяются на пробелы.

Примечание:

Эта команда используется для того, чтобы загрузить тексты с переводами строк из двоичного файла в формате, требуемом программой модели блока. Требуемый формат определяется разработчиком модели.

7.3. Использование вспомогательного класса для загрузки перевода

Если программа модели блока написана на C++, для загрузки текстов из двоичных файлов переводов можно использовать вспомогательный класс `TRdsHelperLanguage`, описанный в файле “RdsClasses.hpp”. Класс берет на себя создание вспомогательного объекта RDS, поэтому обычно программы с ним получаются короче. Все функции класса описаны непосредственно внутри “RdsClasses.hpp”, поэтому для работы достаточно включить этот файл в исходный текст модели командой `#include`.

7.3.1. Конструкторы класса

Класс имеет три конструктора:

```
TRdsHelperLanguage(void);
TRdsHelperLanguage(LPCSTR filepath);
TRdsHelperLanguage(LPCSTR filepath,LPCSTR groupname);
```

Конструктор без параметров просто создает объект, для дальнейшей работы нужно будет передать ему имя двоичного файла функцией `SetBinaryFile`. Конструктор с одним строковым параметром сразу запоминает переданную строку в качестве имени двоичного файла. Конструктор с двумя строковыми параметрами запоминает первый параметр как имя двоичного файла и загружает из этого файла в объект группу с именем, переданным во втором параметре.

Проще всего использовать конструкторы с параметрами: например, чтобы создать объект `Lang`, который будет считывать тексты из файла “MyDll.bin” в подпапке “Dll” папки текущего языка, можно описать его так:

```
TRdsHelperLanguage Lang("$CURLANG$\\Dll\\MyDll.bin");
```

Если в этот объект нужно сразу загрузить все данные группы “MyGroup”, можно описать объект так:

```
TRdsHelperLanguage Lang("$CURLANG$\\Dll\\MyDll.bin","MyGroup");
```

В передаваемом в параметре имени файла можно использовать любые символические имена папок RDS, включая “$CURLANG$” для пути к папке с данными текущего языка интерфейса.

7.3.2. Функция-член `SetBinaryFile` – установка имени двоичного файла

Функция `SetBinaryFile` передает в объект имя двоичного файла, из которого будут загружаться переведенные тексты.

```
void SetBinaryFile(
     LPCSTR filepath   // Имя двоичного файла
);
```

Параметр:

`filepath`

Имя двоичного файла для загрузки текстов перевода. В имени можно использовать любые символические имена папок RDS, включая “$CURLANG$” для пути к папке с данными текущего языка интерфейса.

Примечание:

Если в объект уже были загружены какие-либо тексты, после вызова этой функции они очищаются. Если объект создан конструктором с параметром, в который уже передано имя файла, вызывать `SetBinaryFile` нет необходимости.

7.3.3. Функция-член `Clear` – очистка объекта

Функция `Clear` стирает в объекте все загруженные на данный момент переведенные тексты.

```
void Clear(void);
```

Примечание:

Имя двоичного файла, переданное в объект, этой функцией не очищается.

7.3.4. Функции-члены IsLoaded и IsNotLoaded – проверка наличия в объекте загруженных данных

Функции IsLoaded и IsNotLoaded возвращают результат последней операции загрузки текстов из двоичного файла, т. е. признак наличия в файле запрошенных объектов.

```
BOOL IsLoaded(void);
BOOL IsNotLoaded(void);
```

Возвращаемое значение:

Функция IsLoaded возвращает TRUE, если последний вызов LoadGroup (п. 7.3.6) или LoadObject (п. 7.3.7) успешно загрузил данные из файла, и FALSE в противном случае. Функция IsNotLoaded возвращает противоположное значение (TRUE – данные не загружены, FALSE – загружены).

7.3.5. Функция-член SetCRLFCorrection – установка способа обработки переводов строк

Функция SetCRLFCorrection устанавливает способ обработки переводов строк в загружаемых текстах. Ее нужно вызывать до загрузки данных, т. е. перед вызовом функции LoadGroup (п. 7.3.6) или LoadObject (п. 7.3.7).

```
void SetCRLFCorrection(
      int mode   // Способ обработки
);
```

Параметр:

mode

Одна из констант RDS_UIMOD_CRLF_*, указывающая способ обработки переводов строк:

RDS_UIMOD_CRLF_ANY	Переводы строк в текстах никак не обрабатываются и остаются в точности такими, какими их записал создатель двоичного файла (переводчик).
RDS_UIMOD_CRLF_LF	Переводы строк вида “\r\n” заменяются на одиночный код “\n”.
RDS_UIMOD_CRLF_BOTH	Переводы строк вида “\n” заменяются на сочетание “\r\n”.
RDS_UIMOD_CRLF_NONE	Любые переводы строк заменяются на пробелы.

Примечание:

Функция используется для того, чтобы загрузить тексты с переводами строк из двоичного файла в формате, требуемом программой модели блока. Требуемый формат определяется разработчиком модели.

7.3.6. Функция-член LoadGroup – загрузка всех данных одной группы

Функция LoadGroup загружает в объект группу объектов интерфейса из заранее установленного двоичного файла. После загрузки можно считывать различные параметры объектов из этой группы.

```
void LoadGroup(
      LPCSTR group      // Имя группы
);
```

Параметр:

group

Строка с именем загружаемой группы.

Примечание:

Эта функция обычно используется в тех случаях, когда предполагается работа с несколькими объектами. Например, при открытии окна настройки блока имеет смысл загрузить группу, в которой находятся тексты заголовков объектов, соответствующих полям ввода этого окна, а затем считывать из загруженных данных тексты для каждого поля.

Имя файла передается объекту заранее либо в конструкторе с параметром (п. 7.3.1), либо вызовом SetBinaryFile (п. 7.3.2). Успешность загрузки можно узнать при помощи функций IsLoaded и IsNotLoaded (п. 7.3.4).

Все тексты, загруженные в объект до вызова этой функции, стираются из памяти и заменяются загруженными.

7.3.7. Функция-член LoadObject – загрузка одного объекта

Функция LoadObject загружает в объект данные единственного указанного в параметрах объекта интерфейса из заранее установленного двоичного файла. После загрузки можно считывать различные параметры этого объекта.

```
void LoadObject(
      LPCSTR group,     // Имя группы
      LPCSTR object     // Имя объекта
);
```

Параметры:

group

Строка с именем загружаемой группы.

object

Строка с именем объекта.

Примечание:

Эта функция обычно используется в тех случаях, когда предполагается работа с единственным объектом. Например, при выводе сообщения пользователю обычно достаточно загрузить только текст этого сообщения, не загружая остальные объекты в группе, в которой этот текст находится.

Имя файла передается объекту заранее либо в конструкторе с параметром (п. 7.3.1), либо вызовом SetBinaryFile (п. 7.3.2). Успешность загрузки можно узнать при помощи функций IsLoaded и IsNotLoaded (п. 7.3.4). Если объект был загружен успешно, в дальнейшем можно использовать функции чтения значений его параметров без указания

имени.

Все тексты, загруженные в объект до вызова этой функции, стираются из памяти и заменяются загруженными.

7.3.8. Функция-член SetObject – установка имени объекта для дальнейшего обращения к нему без имени

Функция SetObject запоминает имя текущего объекта для дальнейшего обращения к нему без имени. Ее можно вызывать только после LoadGroup (п. 7.3.6), поскольку любые функции загрузки сбрасывают имя текущего объекта.

```
void SetObject(
     LPCSTR object     // Имя объекта
);
```

Параметр:

object

Строка с именем объекта.

Примечание:

Эта функция обычно используется в тех случаях, когда требуется считать несколько параметров одного и того же объекта, чтобы не передавать имя объекта при каждом обращении.

7.3.9. Функция-член GetString – получение текста значения параметра из считанных данных

Функция GetString возвращает строку значения указанного параметра указанного объекта из уже загруженной группы (см. LoadGroup, п. 7.3.6) или одиночного объекта (см. LoadObject, п. 7.3.7). Она существует в двух вариантах – с указанием имени объекта и без указания (в этом случае возвращается значение параметра текущего объекта).

```
char *GetString(
     LPCSTR object,    // Имя объекта
     int paramid,      // Идентификатор параметра
     LPCSTR defval     // Значение по умолчанию
);

char *GetString(
     int paramid,      // Идентификатор параметра
     LPCSTR defval     // Значение по умолчанию
);
```

Параметры:

object

Строка с именем объекта (в первом варианте функции).

paramid

Целый идентификатор параметра или константа RDS_UIMOD_ANYPARAMID для чтения значения первого попавшегося параметра (используется в тех случаях, когда у объекта есть только один параметр). Идентификаторы параметров указываются в секции “$Params” файла описания интерфейса (см. п. 2). В файле “RdsClasses.hpp” описаны константы RDSHELPERLANG_* для идентификаторов, используемых в стандартных файлах описаний RDS. Для единообразия можно использовать их:

Константа	Значение	Обычное использование
RDSHELPERLANG_CAPTION	1	Заголовки окон и полей ввода
RDSHELPERLANG_TEXT	2	Тексты сообщений пользователю и прочие
RDSHELPERLANG_HINT	3	Всплывающие подсказки
RDSHELPERLANG_FILTER	4	Фильтры (списки файлов) для диалогов
RDSHELPERLANG_TITLE	5	Прочие заголовки
RDSHELPERLANG_STRING	6	Строки общего назначения
RDSHELPERLANG_LIST	7	Выпадающие списки

`defval`

Строка, возвращаемая при отсутствии параметра, объекта или любых загруженных данных.

Возвращаемое значение:

Если данные из двоичного файла перевода были ранее загружены успешно, в этих данных есть указанный объект (при использовании первого варианта функции) или был установлен текущий (при использовании второго варианта функции), и у этого объекта есть указанный параметр, возвращается указатель на строку во внутренних данных объекта с текстовым значением этого параметра. Эта строка будет существовать до уничтожения объекта или до загрузки в него новых данных. Если же указанного параметра нет, возвращается переданный в параметрах функции указатель `defval`.

Примечание:

Это основная функция получения переведенных строк из загруженных данных. Строка на языке по умолчанию (русском) передается в параметре `defval`. Если перевод успешно загружен, возвращается переведенная строка, если нет – строка по умолчанию.

7.3.10. Функция-член `GetListString` – получение текста содержимого выпадающего списка из считанных данных

Функция `GetListString` преобразует строку описания выпадающего списка формата “0=вариант1\n1=вариант2\n…” из указанного параметра указанного объекта уже загруженной группы (см. `LoadGroup`, п. 7.3.6) или одиночного объекта (см. `LoadObject`, п. 7.3.7) в строку вида “вариант1\nвариант2\n…”, пригодную для передачи в оконные объекты RDS (см. приложение А.5.28.18). Функция существует в двух вариантах – с указанием имени объекта и без указания (в этом случае возвращается значение параметра текущего объекта).

```
char *GetListString(
     LPCSTR object     // Имя объекта
     int paramid,      // Идентификатор параметра
     LPCSTR defval     // Значение по умолчанию
);
char *GetListString(
     int paramid,      // Идентификатор параметра
     LPCSTR defval     // Значение по умолчанию
);
```

Параметры:

`object`

Строка с именем объекта (в первом варианте функции).

`paramid`

Целый идентификатор параметра или константа `RDS_UIMOD_ANYPARAMID` для чтения значения первого попавшегося параметра (используется в тех случаях, когда у объекта есть только один параметр). Идентификаторы параметров указываются в секции "[$Params]" файла описания интерфейса (см. п. 2). В файле "RdsClasses.hpp" описаны константы `RDSHELPERLANG_*` для идентификаторов, используемых в стандартных файлах описаний RDS (см. п. 7.3.9).

`defval`

Строка, возвращаемая при отсутствии параметра, объекта или любых загруженных данных.

Возвращаемое значение:

Если данные из двоичного файла перевода были ранее загружены успешно, в этих данных есть указанный объект, и у этого объекта есть указанный параметр, из этого параметра считывается текст описания выпадающего списка формата "0=вариант1\n1=вариант2\n…" (см. п. 2). Затем в переданном тексте `defval` формата "вариант1\nвариант2\n…" отдельные строки заменяются на считанные согласно индексам перед знаком равенства, получившаяся строка сохраняется во внутренних данных объекта, и возвращается указатель на нее. Эта строка будет существовать до уничтожения объекта или до загрузки в него новых данных. Если же указанного параметра в загруженных данных нет, возвращается переданный в параметрах функции указатель `defval`.

Примечание:

Эта функция используется для перевода вариантов выпадающих списков для полей ввода RDS в окнах настройки блоков. Ее действие аналогично действию сервисной функции RDS `rdsUIMODGetParamListString`, описанной в п. 7.2.8.

7.3.11. Статическая функция `MessageBox` – вывод сообщения с переведенным текстом

Статическая функция `MessageBox` выводит пользователю сообщение, текст которого, при наличии перевода, берется из первого попавшегося параметра заданного объекта заданного файла перевода.

```
static int MessageBox(
      LPCSTR filepath   // Имя двоичного файла
      LPCSTR message,   // Текст сообщения по умолчанию
      LPCSTR caption,   // Заголовок окна сообщения
      int flags,        // Флаги Windows API (MB_*)
      LPCSTR object     // Имя объекта
      LPCSTR group=NULL // Имя группы
);
```

Параметры:

`filepath`
: Имя двоичного файла для загрузки текстов перевода. В имени можно использовать любые символические имена папок RDS, включая “$CURLANG$” в качестве пути к папке с данными текущего языка интерфейса.

`message`
: Строка с текстом сообщения на языке по умолчанию.

`caption`
: Строка с заголовком окна сообщения или `NULL` для стандартного (“RDS”).

`flags`
: Битовые флаги сообщения, определяющие набор кнопок в окне, иконку сообщения и т. п. (приложение А.5.5.6).

`object`
: Строка с именем объекта.

`group`
: Необязательный параметр – строка с именем группы. Если параметр опущен, будет загружаться объект из группы “Messages”.

Возвращаемое значение:

Стандартная константа Windows API `ID*`, указывающая на нажатую пользователем кнопку окна сообщения.

Примечание:

Эта статическая функция позволяет упростить вывод переведенных сообщений пользователю, если у объекта с текстом сообщения в двоичном файле есть только один параметр (собственно текст) и если заголовок сообщения переводить не нужно. Ее параметры, за исключением `filepath`, `object` и `group`, совпадают с параметрами сервисной функции RDS `rdsMessageBox` (приложение А.5.5.6).

Поскольку функция статическая, для ее вызова не требуется создавать объект типа `TRdsHelperLanguage`. Ее можно вызвать с указанием префикса класса, т. е. “`TRdsHelperLanguage::MessageBox(…);`”.

Пусть в файле описания интерфейса (см. п. 2) текст сообщения пользователю задан следующим образом:

```
[UserMessages]
…
TestMessage.Text=Сообщение пользователю
…
```

Пусть по этому файлу создан двоичный файл перевода (см. п. 4) “MyDll.bin”, находящийся в подпапке “Dll” папки текущего языка. Тогда функцию вывода этого сообщения со стандартным заголовком окна можно записать так:

```
TRdsHelperLanguage::MessageBox(
     "$CURLANG$\\Dll\\MyDll.bin",  // файл
     "Сообщение пользователю",     // текст по умолчанию
     NULL,                         // заголовок (стандартный)
     MB_OK|MB_ICONINFORMATION,     // флаги сообщения
     "TestMessage",                // объект
     "UserMessages");              // группа
```

Если бы объект “TestMessage” с текстом сообщения находился в группе “Messages”, а не в “UserMessages”, последний параметр функции можно было бы не указывать.

Если у объекта с текстом сообщения несколько параметров, или если у окна сообщения должен быть заголовок, который тоже нужно переводить, эту функцию использовать не следует. Вместо этого нужно создать объект класса `TRdsHelperLanguage`, загрузить в него объект вызовом `LoadObject` (п. 7.3.7), а затем считать переведенные тексты сообщения и заголовка вызовами `GetString` (п. 7.3.9).

7.4. Пример модели с переводом интерфейса пользователя

Рассмотрим пример модели блока с окном настроек, интерфейс пользователя которой нужно перевести. Сначала будет приведен пример модели без перевода, а затем в нее будут добавлены пеерводы при помощи сервисных функций RDS и при помощи вспомогательного класса.

7.4.1. Исходная модель без перевода

Рассмотрим простую модель блока следующего вида (главная функция DLL в примере не приводится).

```
//====== Класс личной области данных ======
class TTestLData
{ public:
    int IOption;      // Номер варианта
    double DParam;    // Вещественный параметр
    int Setup(void);  // Функция настройки параметров
    void MouseDown(void);// Реакция на мышь
    TTestLData(void)  // Конструктор класса
      { IOption=0; DParam=0.0; };
};
//=========================================

//============= Модель блока ==============
extern "C" __declspec(dllexport)
  int RDSCALL TestL(int CallMode, // Режим вызова (событие в RDS)
                    RDS_PBLOCKDATA BlockData, // Структура данных
                    LPVOID /*ExtParam*/) // Доп.параметр
{ TTestLData *data=(TTestLData*)(BlockData->BlockData);
  switch(CallMode)
    { case RDS_BFM_INIT:         // Инициализация
        BlockData->BlockData=new TTestLData();
        break;
      case RDS_BFM_CLEANUP:      // Очистка
        delete data;
        break;
      case RDS_BFM_SETUP:        // Функция настройки
        return data->Setup();
      case RDS_BFM_MOUSEDOWN:    // Кнопка мыши
        data->MouseDown();
        break;
    }
  return RDS_BFR_DONE;
}
//=========================================

// Функция настройки параметров
```

```
int TTestLData::Setup(void)
{ RDS_HOBJECT window; // Идентификатор объекта-окна
  BOOL ok;            // Пользователь нажал "OK"
  // Создание окна
  window=rdsFORMCreate(FALSE,-1,-1,
             "Ввод параметров");
  // Добавление полей ввода
  rdsFORMAddEdit(window,0,1,RDS_FORMCTRL_COMBOLIST,
      "Номер варианта:",200);
  rdsSetObjectStr(window,1,RDS_FORMVAL_LIST,
      "Первый вариант\nВторой вариант\nТретий вариант");
  rdsFORMAddEdit(window,0,2,RDS_FORMCTRL_EDIT,
      "Вещественный параметр:",80);
  // Занесение исходных значений в поля ввода
  rdsSetObjectInt(window,1,RDS_FORMVAL_VALUE,IOption);
  rdsSetObjectDouble(window,2,RDS_FORMVAL_VALUE,DParam);
  // Открытие окна
  ok=rdsFORMShowModalEx(window,NULL);
  if(ok)
    { // Нажата кнопка OK - запись параметров обратно в блок
      IOption=rdsGetObjectInt(window,1,RDS_FORMVAL_VALUE);
      DParam=rdsGetObjectDouble(window,2,RDS_FORMVAL_VALUE);
    }
  // Уничтожение окна
  rdsDeleteObject(window);
  // Возвращаемое значение
  return ok?RDS_BFR_MODIFIED:RDS_BFR_DONE;
}
//=========================================

// Нажатие кнопки мыши
void TTestLData::MouseDown(void)
{
  rdsMessageBox("Нажата кнопка мыши",NULL,
      MB_OK|MB_ICONINFORMATION);
}
//=========================================
```

В этой модели личная область данных оформлена в виде класса `TTestLData` с двумя полями: целым `IOption` и вещественным `DParam`, которые задаются в окне настроек блока. Не будем подробно останавливаться на структуре этой модели – подобные модели во множестве рассматриваются в руководстве программиста.

Тексты интерфейса пользователя, которые необходимо перевести, находятся в функциях `TTestLData::Setup` и `TTestLData::MouseDown`. В функции `TTestLData::Setup` открывается окно настроек блока, все заголовки полей котрого (строки в двойных кавычках) необходимо перевести. В `TTestLData::MouseDown` при нажатии кнопки мыши выводится сообщение пользователю, текст которого также необходимо перевести.

7.4.2. Создание файла описания интерфейса и файла перевода

Создадим для этой модели файл описания интерфейса пользователя (см. п. 2) следующего вида:

```
[$Types]
Language=Caption,Text,Hint,Filter,Title,String,List
```

```
[$Params]
Caption=S,1
Text=S,2
Hint=S,3
Filter=S,4
Title=S,5
String=S,6
List=S,7

[TTestLData]
; Тестовый блок
Setup.Caption=Ввод параметров
IOption.Caption=Номер варианта:
IOption.List=0=Первый вариант\n1=Второй вариант\n2=Третий вариант
DParam.Caption=Вещественный параметр:
MouseDownMessage.Text=Нажата кнопка мыши
```

Секции “$Types” и “$Params” этого файла скопированы из “rds.dat”, поэтому в нем будут использоваться стандартные для файлов описания интерфейса имена и идентификаторы параметров. В секции “TTestLData” каждой из строк в кавычках, используемых в программе модели, соответствует какой-либо параметр какого-либо объекта. Например, заголовок окна настроек присвоен параметру “Caption” объекта “Setup”, сообщение пользователю о нажатии кнопки мыши – параметру “Text” объекта “MouseDownMessage” и т. д.

Следует обратить внимание на содержимое выпадающего списка для параметра `IOption`, которое присвоено параметру “List” объекта “IOption”. Оно отличается от строки в модели тем, что туда, как описано в п. 2, добавлены индексы для каждого варианта, т. е. “0=…”, “1=…” и т. д.

Поместим этот файл в папку “Description\Dll\” внутри папки установки RDS и дадим ему имя “MyDll.dat”.

На данный момент в RDS добавлен английский язык интерфейса пользователя с именем “English”. Экспортируем файл описания “Dll\MyDll.dat” в INI-файл для перевода согласно п. 4.1, в результате чего получим файл со следующим содержимым:

```
[$Descr]
Type=Language
Mod=English
Dat=Dll*MyDll.dat

[TTestLData]
DParam.Caption=Вещественный параметр:
IOption.Caption=Номер варианта:
IOption.List=0=Первый вариант\n1=Второй вариант\n2=Третий вариант
MouseDownMessage.Text=Нажата кнопка мыши
Setup.Caption=Ввод параметров
```

Откроем его в любом текстовом редакторе и переведем на английский язык все значения параметров в секции “TTestLData”:

```
[$Descr]
Type=Language
Mod=English
Dat=Dll*MyDll.dat

[TTestLData]
DParam.Caption=Real parameter:
```

```
IOption.Caption=Option:
IOption.List=0=First option\n1=Second option\n2=Third option
MouseDownMessage.Text=Mouse button pressed
Setup.Caption=Parameters
```

Теперь импортируем этот файл для описания “Dll\MyDll.dat” так, как описано в п. 4.4. В папке “Interface\Language\English\Dll\” появится двоичный файл “MyDll.bin”, из которого можно загружать тексты окна настроек и сообщения пользователю на английском языке. В общем виде путь к файлу перевода нашей модели для текущего языка будет записываться как “$CURLANG$\\Dll\\MyDll.bin”: если текущим языком будет английский, этот путь будет указывать на только что импортированный нами файл.

7.4.3. Загрузка перевода сервисными функциями RDS

Для того, чтобы добавить в модель блока функции загрузки перевода, достаточно изменить только те части модели, в которых есть строки текста для интерфейса пользователя. В нашем случае это `TTestLData::Setup` и `TTestLData::MouseDown`. Перепишем их (изменения выделены цветом):

```
// Функция настройки параметров
int TTestLData::Setup(void)
{ RDS_HOBJECT window; // Идентификатор объекта-окна
  BOOL ok;            // Пользователь нажал "OK"
  RDS_HOBJECT lang;   // Перевод интерфейса

  // Загрузка группы "TTestLData" из языкового файла
  lang=rdsUIMODCreate();
  rdsUIMODLoadGroup(lang,"$CURLANG$\\Dll\\MyDll.bin","TTestLData");

  // Создание окна
  window=rdsFORMCreate(FALSE,-1,-1,
           rdsUIMODGetParamString(lang,"Setup",
             RDS_UIMOD_ANYPARAMID,
             "Ввод параметров"));
  // Добавление полей ввода
  rdsFORMAddEdit(window,0,1,RDS_FORMCTRL_COMBOLIST,
      rdsUIMODGetParamString(lang,"IOption",1,"Номер варианта:"),
      200);
  rdsSetObjectStr(window,1,RDS_FORMVAL_LIST,
      rdsUIMODGetParamListString(lang,"IOption",7,
        "Первый вариант\nВторой вариант\nТретий вариант"));
  rdsFORMAddEdit(window,0,2,RDS_FORMCTRL_EDIT,
      rdsUIMODGetParamString(lang,"DParam",
        RDS_UIMOD_ANYPARAMID,"Вещественный параметр:"),
      80);
  // Занесение исходных значений в поля ввода
  rdsSetObjectInt(window,1,RDS_FORMVAL_VALUE,IOption);
  rdsSetObjectDouble(window,2,RDS_FORMVAL_VALUE,DParam);
  // Открытие окна
  ok=rdsFORMShowModalEx(window,NULL);
  if(ok)
    { // Нажата кнопка OK - запись параметров обратно в блок
      IOption=rdsGetObjectInt(window,1,RDS_FORMVAL_VALUE);
      DParam=rdsGetObjectDouble(window,2,RDS_FORMVAL_VALUE);
    }
  // Уничтожение окна
  rdsDeleteObject(window);
```

```
  // Уничтожение интерфейсного объекта
  rdsDeleteObject(lang);
  // Возвращаемое значение
  return ok?RDS_BFR_MODIFIED:RDS_BFR_DONE;
}
//=======================================

// Реакция на мышь
void TTestLData::MouseDown(void)
{ RDS_HOBJECT lang; // Перевод интерфейса

  // Загрузка одного объекта интерфейса
  lang=rdsUIMODCreate();
  rdsUIMODLoadSingleObject(lang,"$CURLANG$\\Dll\\MyDll.bin",
    "TTestLData","MouseDownMessage");

  rdsMessageBox(
    rdsUIMODGetParamString(lang,NULL,
        RDS_UIMOD_ANYPARAMID,"Нажата кнопка мыши"),
    NULL,MB_OK|MB_ICONINFORMATION);

  // Уничтожение интерфейсного объекта
  rdsDeleteObject(lang);
}
//=======================================
```

В функцию `TTestLData::Setup` добавлен новый вспомогательный объект `lang`, который создается вызовом `rdsUIMODCreate`. Затем вызовом `rdsUIMODLoadGroup` в него загружается группа объектов “TTestLData” из файла “$CURLANG$\\Dll\\MyDll.bin”, т. е. из файла “MyDll.bin” подпапки “Dll” в папке текущего языка интерфейса пользователя. Если текущий язык – английский, будут загружены данные из созданного нами файла “MyDll.bin”, и все последующие вызовы будут возвращать тексты из этого файла. Если текущий язык – любой другой, включая русский, соответствующий файл найден не будет, и данные не загрузятся. В результате все последующие вызовы будут возвращать переданные в параметрах строки по умолчанию, т. е. тексты на русском языке.

Далее везде в функциях, где раньше стояли строки в двойных кавычках, теперь вместо самих строк записаны вызовы сервисных функций `rdsUIMODGetParamString` или `rdsUIMODGetParamListString`, в которые прежние строки передаются как параметры по умолчанию. Для тех объектов, у которых в нашем файле описания интерфейса есть только один параметр (“Setup” и “DParam”) в качестве идентификатора параметра передается константа `RDS_UIMOD_ANYPARAMID`. У объекта “IOption” два параметра: заголовок поля ввода “Caption” с идентификатором 1 и содержимое списка “List” с идентификатором 7. Из-за этого в вызовы функций получения текстов для “IOption” передаются конкретные числа 1 и 7 – считывать “первый попавшийся” параметр нельзя.

В самом конце функции `TTestLData::Setup` созданный объект `lang` уничтожается вызовом `rdsDeleteObject`. Уничтожать его раньше нельзя, поскольку необходимые окну настройки тексты, загруженные из двоичного файла перевода, хранятся в его внутренней памяти и будут уничтожены месте с ним.

В функции `TTestLData::MouseDown` переведенный текст сообщения пользователю точно таким же образом загружается из объекта “MouseDownMessage” той же секции “TTestLData”.

Внесенных в модель изменений достаточно для того, чтобы при выборе пользователем русского языка интерфейса RDS окно настройки блока и сообщение были на русском языке, при выборе английского – на английском.

7.4.4. Загрузка перевода объектом вспомогательного класса

Теперь изменим исходную модель так, чтобы переведенные тексты загружались при помощи объекта класса `TRdsHelperLanguage`. Для этого нужно будет включить в текст модели файл заголовка “RdsClasses.hpp” и снова переписать функции `TTestLData::Setup` и `TTestLData::MouseDown`. Изменения выделены цветом:

```
// Включение описания вспомогательного класса
#include <RdsClasses.hpp>
// Функция настройки параметров
int TTestLData::Setup(void)
{ RDS_HOBJECT window; // Идентификатор бъекта-окна
  BOOL ok;            // Пользователь нажал "OK"
  // Объект перевода интерфейса
  TRdsHelperLanguage Lang("$CURLANG$\\Dll\\MyDll.bin");

  // Загрузка группы "TTestLData" из языкового файла
  Lang.LoadGroup("TTestLData");

  // Создание окна
  window=rdsFORMCreate(FALSE,-1,-1,
            Lang.GetString("Setup",
                   RDS_UIMOD_ANYPARAMID,
                   "Ввод параметров"));
  // Добавление полей ввода
  rdsFORMAddEdit(window,0,1,RDS_FORMCTRL_COMBOLIST,
      Lang.GetString("IOption",1,"Номер варианта:"),
      200);
  rdsSetObjectStr(window,1,RDS_FORMVAL_LIST,
      Lang.GetListString("IOption",7,
             "Первый вариант\nВторой вариант\nТретий вариант"));
  rdsFORMAddEdit(window,0,2,RDS_FORMCTRL_EDIT,
      Lang.GetString("DParam",RDS_UIMOD_ANYPARAMID,
             "Вещественный параметр:"),
      80);
  // Занесение исходных значений в поля ввода
  rdsSetObjectInt(window,1,RDS_FORMVAL_VALUE,IOption);
  rdsSetObjectDouble(window,2,RDS_FORMVAL_VALUE,DParam);
  // Открытие окна
  ok=rdsFORMShowModalEx(window,NULL);
  if(ok)
    { // Нажата кнопка OK - запись параметров обратно в блок
      IOption=rdsGetObjectInt(window,1,RDS_FORMVAL_VALUE);
      DParam=rdsGetObjectDouble(window,2,RDS_FORMVAL_VALUE);
    }
  // Уничтожение окна
  rdsDeleteObject(window);
  // Возвращаемое значение
  return ok?RDS_BFR_MODIFIED:RDS_BFR_DONE;
}
//=======================================
```

```
// Реакция на мышь
void TTestLData::MouseDown(void)
{
  TRdsHelperLanguage::MessageBox(
    "$CURLANG$\\Dll\\MyDll.bin",
    "Нажата кнопка мыши",
    NULL,
    MB_OK|MB_ICONINFORMATION,
    "MouseDownMessage",
    "TTestLData");
}
//=====================================
```

В начале функции `TTestLData::Setup` создается объект `Lang` типа `TRdsHelperLanguage`, причем в его конструкторе сразу передан путь к двоичному файлу описания интерфейса пользователя нашей модели для текущего языка "$CURLANG$\\Dll\\MyDll.bin". Как и в предыдущем примере, если текущий язык – английский, дале в программе в `Lang` будут загружены данные из созданного нами файла "MyDll.bin", и все последующие вызовы функций-членов этого объекта будут возвращать тексты из этого файла. Если текущий язык – любой другой, включая русский, соответствующий файл найден не будет, данные не загрузятся, и все последующие вызовы будут возвращать переданные в параметрах строки по умолчанию, т. е. тексты на русском языке.

Для загрузки в объект данных секции "TTestLData" используется функция объекта `Lang.LoadGroup` (п. 7.3.6).

Далее везде вместо строк в кавычках подставлены вызовы функций-членов объекта `Lang.GetString` (п. 7.3.9) и `Lang.GetListString` (п. 7.3.10), в которые передаются имена объектов и идентификаторы их параметров для чтения, а также строки по умолчанию, которые будут возвращаться при отсутствии данных текущего языка или если текущим выбран язык по умолчанию. Для объектов, у которых только один параметр, вместо идентификатора этого параметра передается константа `RDS_UIMOD_ANYPARAMID` (безразлично, данные какого параметра загружать – все равно он единственный).

В отличие от примера в п. 7.4.3, в конце функции `TTestLData::Setup` нет команды уничтожения объекта – он уничтожится сам, когда выйдет из области видимости.

В функции `TTestLData::MouseDown` вызов сервисной функции `rdsMessageBox` заменен на вызов статической функции вспомогательного класса `TRdsHelperLanguage` `TRdsHelperLanguage::MessageBox`, которая использует собственный объект для загрузки данных. По этой причине описания дополнительного объекта в функции `TTestLData::MouseDown` нет, `TRdsHelperLanguage::MessageBox` делает все сама. В нее передается путь к двоичному файлу перевода "$CURLANG$\\Dll\\MyDll.bin", имя объекта "MouseDownMessage" и имя группы с этим объектом "TTestLData", а также флаги сообщения, заголовок его окна и текст сообщения для языка по умолчанию, с которым вызывалась `rdsMessageBox` в исходной версии модели. Идентификатор параметра не передается: функция `TRdsHelperLanguage::MessageBox` требует, чтобы у указанного в ее вызове объекта был единственный параметр. В нашем случае это выполняется.

Как и в предыдущем примере, внесенных в модель изменений достаточно для того, чтобы при выборе пользователем русского языка интерфейса RDS окно настройки блока и сообщение были на русском языке, при выборе английского – на английском.