

**Федеральное государственное бюджетное учреждение науки
ИНСТИТУТ ПРОБЛЕМ УПРАВЛЕНИЯ
им. В.А. ТРАПЕЗНИКОВА
Российской академии наук**

А.А. Рощин

**Расчет Динамических Систем (РДС)
Руководство для программистов
Приложение: описание функций и структур**

Москва 2016

Редактор: заведующий лабораторией Института проблем управления им. Трапезникова РАН, профессор М.Х. Дорри.

Работа выполнена в рамках проектов, выполняемых Учреждением Российской академии наук Институт проблем управления им. В.А. Трапезникова РАН:

- проект 3121: "Разработка теории и алгоритмов синтеза управления объектами морской техники и методологии создания исследовательских стендов", (Перечень комплексных проектов, разрабатываемых в 2010-2012 гг.)
- проект 400-09: "Методы анализа и синтеза информационно-управляющих систем с интеллектуальным интерфейсом" (Программа фундаментальных исследований № 15, 2009-2011 гг.).

В приложении к руководству для программистов инструментального комплекса РДС (Расчет Динамических Систем) подробно описываются функции, структуры и константы, используемые при создании моделей блоков и модулей автоматической компиляции моделей комплекса.

Оглавление

Приложение А. Функции, константы и структуры РДС.....	23
А.1. Идентификаторы объектов и вспомогательные типы.....	23
А.2. События блока и связанные с ними описания.....	26
А.2.1. Функция модели блока.....	26
А.2.2. Главная функция DLL.....	27
А.2.3. RDS_BLOCKDATA – структура данных блока.....	28
А.2.4. События общего назначения.....	31
А.2.4.1. RDS_BFM_CALCMODE – переход из режима редактирования в режим моделирования.....	31
А.2.4.2. RDS_BFM_CHECKFUNCSUPPORT – проверка поддержки блоком функции с заданным идентификатором.....	32
А.2.4.3. RDS_BFM_CLEANUP – очистка данных блока.....	33
А.2.4.4. RDS_BFM_DYNVARCHANGE – изменение динамической переменной.....	34
А.2.4.5. RDS_BFM_EDITMODE – переход в режим редактирования.....	35
А.2.4.6. RDS_BFM_FUNCTIONCALL – вызов функции блока.....	35
А.2.4.7. RDS_BFM_INIT – инициализация блока.....	38
А.2.4.8. RDS_BFM_LOADSTATE – загрузка состояния блока.....	39
А.2.4.9. RDS_BFM_MODEL – выполнение такта расчета.....	40
А.2.4.10. RDS_BFM_PREMODEL – вызов модели перед тактом расчета.....	41
А.2.4.11. RDS_BFM_REMOTEMSG – вызов от внешнего приложения.....	42
А.2.4.12. RDS_BFM_RESETCALC – сброс расчета.....	44
А.2.4.13. RDS_BFM_SAVESTATE – запись состояния блока.....	45
А.2.4.14. RDS_BFM_STARTCALC – запуск расчета.....	46
А.2.4.15. RDS_BFM_STOPCALC – остановка расчета.....	46
А.2.4.16. RDS_BFM_TIMER – срабатывание таймера блока.....	47
А.2.4.17. RDS_BFM_UNLOADSYSTEM – схема будет выгружена из памяти.....	47
А.2.4.18. RDS_BFM_VARCHHECK – проверка допустимости структуры статических переменных блока.....	48
А.2.5. События загрузки и сохранения схемы и отдельных блоков.....	50
А.2.5.1. RDS_BFM_AFTERLOAD – завершена загрузка схемы.....	50
А.2.5.2. RDS_BFM_AFTERSAVE – завершено сохранение схемы.....	51
А.2.5.3. RDS_BFM_BEFORESAVE – начато сохранение схемы.....	51
А.2.5.4. RDS_BFM_LOADBIN – загрузка данных блока в двоичном формате.....	52
А.2.5.5. RDS_BFM_LOADTXT – загрузка данных блока в текстовом формате.....	52
А.2.5.6. RDS_BFM_SAVEBIN – запись данных блока в двоичном формате.....	53
А.2.5.7. RDS_BFM_SAVETXT – запись данных блока в текстовом формате.....	54
А.2.6. События пользовательского интерфейса и рисования внешнего вида блоков.....	55
А.2.6.1. RDS_BFM_BLOCKPANEL – уведомление от панели блока в подсистеме.....	55
А.2.6.2. RDS_BFM_CONTEXTPOPUP – вызов контекстного меню блока.....	56
А.2.6.3. RDS_BFM_DRAW – рисование внешнего вида блока.....	57
А.2.6.4. RDS_BFM_DRAWADDITIONAL – дополнительное рисование.....	59
А.2.6.5. RDS_BFM_KEYDOWN – нажатие клавиши.....	60
А.2.6.6. RDS_BFM_KEYUP – отпускание клавиши.....	62
А.2.6.7. RDS_BFM_MENUFUNCTION – выбор пользователем пункта меню.....	63
А.2.6.8. RDS_BFM_MOUSEDBLCLICK – двойной щелчок мыши.....	64
А.2.6.9. RDS_BFM_MOUSEDOWN – нажатие кнопки мыши.....	67
А.2.6.10. RDS_BFM_MOUSEMOVE – перемещение курсора мыши.....	68
А.2.6.11. RDS_BFM_MOUSEUP – отпускание кнопки мыши.....	69
А.2.6.12. RDS_BFM_POPUPHINT – вывод всплывающей подсказки.....	70

A.2.6.13. RDS_BFM_SETUP – вызов функции настройки блока.....	71
A.2.6.14. RDS_BFM_WINDOWKEYDOWN – реакция подсистемы на нажатие клавиши в своем окне.....	72
A.2.6.15. RDS_BFM_WINDOWKEYUP – реакция подсистемы на отпускание клавиши в своем окне.....	73
A.2.6.16. RDS_BFM_WINDOWMOUSEDCLICK – реакция подсистемы на двойной щелчок мыши в своем окне.....	73
A.2.6.17. RDS_BFM_WINDOWMOUSEDOWN – реакция подсистемы на нажатие кнопки мыши в своем окне.....	74
A.2.6.18. RDS_BFM_WINDOWMOUSEMOVE – реакция подсистемы на перемещение курсора мыши в своем окне.....	75
A.2.6.19. RDS_BFM_WINDOWMOUSEUP – реакция подсистемы на отпускание кнопки мыши в своем окне.....	76
A.2.6.20. RDS_BFM_WINDOWOPERATION – открытие и закрытие окна подсистемы.....	76
A.2.6.21. RDS_BFM_WINREFRESH – обновление окон блока.....	77
A.2.7. События, связанные с изменением схемы пользователем.....	78
A.2.7.1. RDS_BFM_MANUALDELETE – удаление блока пользователем.....	78
A.2.7.2. RDS_BFM_MANUALINSERT – вставка блока пользователем.....	79
A.2.7.3. RDS_BFM_MOUSESELECT – возможность выбора блока мышью.....	80
A.2.7.4. RDS_BFM_MOVED – перемещение блока.....	81
A.2.7.5. RDS_BFM_RENAME – переименование блока.....	82
A.2.7.6. RDS_BFM_RESIZE – размер блока изменен пользователем.....	82
A.2.7.7. RDS_BFM_RESIZING – изменение размеров блока пользователем.....	84
A.2.8. События обмена данными по сети.....	84
A.2.8.1. RDS_BFM_NETCONNECT – установка соединения.....	84
A.2.8.2. RDS_BFM_NETDATAACCEPTED – получение данных сервером.....	85
A.2.8.3. RDS_BFM_NETDATARECEIVED – получение данных блоком.....	87
A.2.8.4. RDS_BFM_NETDISCONNECT – разрыв соединения.....	88
A.2.8.5. RDS_BFM_NETERROR – ошибка при работе с сетью.....	89
A.3. События модуля автокомпиляции и связанные с ними структуры.....	91
A.3.1. Функция модуля автокомпиляции.....	91
A.3.2. RDS_COMPMODULEDATA – структура данных модуля.....	92
A.3.3. RDS_COMPMODELDATA – структура данных модели.....	93
A.3.4. События модуля автокомпиляции.....	95
A.3.4.1. RDS_COMPM_ATTACHBLOCK – подключение модели к блоку.....	95
A.3.4.2. RDS_COMPM_CANATTACHBLK – проверка возможности подключения модели к блоку.....	96
A.3.4.3. RDS_COMPM_CANRENMODEL – проверка возможности переименования модели.....	98
A.3.4.4. RDS_COMPM_CLEANUP – очистка данных модуля.....	99
A.3.4.5. RDS_COMPM_CLOSEALLWIN – закрытие всех окон.....	99
A.3.4.6. RDS_COMPM_COMPILE – компиляция моделей.....	100
A.3.4.7. RDS_COMPM_DETACHBLOCK – отключение модели от блока.....	101
A.3.4.8. RDS_COMPM_EXECFUNCTION – реакция на действия пользователя.....	101
A.3.4.9. RDS_COMPM_GETOPTIONS – описание возможностей модуля.....	103
A.3.4.10. RDS_COMPM_INIT – инициализация модуля.....	103
A.3.4.11. RDS_COMPM_MODECHANGE – изменение режима РДС.....	104
A.3.4.12. RDS_COMPM_MODELCLEANUP – очистка данных модели.....	104
A.3.4.13. RDS_COMPM_MODELINIT – инициализация модели.....	105
A.3.4.14. RDS_COMPM_MODELRENAMED – модель переименована.....	105

A.3.4.15. RDS_COMPM_OPENEDITOR – вызов редактора модели.....	106
A.3.4.16. RDS_COMPM_PREPARE – подготовка модели к компиляции.....	107
A.3.4.17. RDS_COMPM_SAVEBLOCK – сохранение блока.....	108
A.3.4.18. RDS_COMPM_SAVESYSTEM – сохранение схемы.....	109
A.3.4.19. RDS_COMPM_SETUP – настройка модуля автокомпиляции.....	110
A.3.4.20. RDS_COMPM_STRUCTCHANGE – изменение структуры.....	110
A.4. Структуры РДС.....	112
A.4.1. RDS_ARRAYACCESSDATA – описание матрицы/массива.....	112
A.4.2. RDS_BLOCKDESCRIPTION – описание блока.....	113
A.4.3. RDS_BLOCKDIMENSIONS – размеры и положение блока или связи.....	117
A.4.4. RDS_CONNAPPEARANCE – внешний вид связи или шины.....	118
A.4.5. RDS_CONNDESCRIPTION – описание связи или шины.....	119
A.4.6. RDS_DYNVARLINK – подписка на динамическую переменную.....	121
A.4.7. RDS_EDITORPARAMETERS – параметры окна подсистемы.....	122
A.4.8. RDS_EDITORTOOLBARS – описание панелей окна подсистемы.....	126
A.4.9. RDS_FINDBYEXTIDDATA – результаты поиска по идентификатору.....	126
A.4.10. RDS_FORMSERVFUNCDATA – параметр функции обратного вызова модального окна.....	128
A.4.11. RDS_FUNCPROVIDERLINK – подписка на исполнителя функции.....	129
A.4.12. RDS_LINEDESCRIPTION – описание отрезка внутри связи или шины.....	129
A.4.13. RDS_PANDESCRIPTION – описание панели блока в окне подсистемы.....	131
A.4.14. RDS_POINTDESCRIPTION – описание точки связи или шины.....	133
A.4.15. RDS_SERVFONTPARAMS – описание шрифта.....	135
A.4.16. RDS_TIMERDESCRIPTION – описание таймера.....	136
A.4.17. RDS_VARDESCRIPTION – описание переменной блока.....	138
A.5. Сервисные функции РДС.....	141
A.5.1. Доступ к сервисным функциям РДС.....	141
A.5.2. Управление работой РДС и функции общего назначения.....	142
A.5.2.1. Макрос RDS_GETFLAG – получение битового флага.....	142
A.5.2.2. Макрос RDS_INITSERVSIZE – инициализация поля размера в стандартных структурах РДС.....	142
A.5.2.3. Макрос RDS_INTVERSION – преобразование версии РДС в целое число.....	143
A.5.2.4. Макрос RDS_SETFLAG – установка битового флага.....	145
A.5.2.5. rdsApplicationIsActive – активность приложения РДС.....	145
A.5.2.6. rdsBadSystemTime – ошибка в системных часах.....	146
A.5.2.7. rdsBlockModalWinClose – сообщение о закрытии модального окна.....	146
A.5.2.8. rdsBlockModalWinOpen – сообщение об открытии модального окна.....	147
A.5.2.9. rdsBringAppToFront – перемещение РДС на передний план.....	148
A.5.2.10. rdsCalcProcessIsRunning – РДС в режиме расчета.....	148
A.5.2.11. rdsCalcProcessNeverStarted – запускался ли расчет.....	149
A.5.2.12. rdsCancelPaste – отмена вставки блоков из буфера обмена.....	149
A.5.2.13. rdsChangeRegWinTitle – изменение названия немодального окна.....	150
A.5.2.14. rdsEnableCommandQueue – приостановить/продолжить выполнение очереди команд.....	151
A.5.2.15. rdsExecuteCommand – выполнить общесистемную команду.....	151
A.5.2.16. rdsFindCmdParam – найти номер параметра командной строки.....	153
A.5.2.17. rdsGetAppInstance – дескриптор приложения РДС.....	154
A.5.2.18. rdsGetAppWindowHandle – дескриптор главного окна РДС.....	154
A.5.2.19. rdsGetCmdParam – параметр командной строки по номеру.....	155
A.5.2.20. rdsGetCmdParamCount – число слов в командной строке РДС.....	156
A.5.2.21. rdsGetCustomColors – получение массива пользовательских цветов.....	156

A.5.2.22. rdsGetHugeDouble – получение значения-индикатора математической ошибки.....	157
A.5.2.23. rdsGetMainWindow – дескриптор главного окна пользовательского интерфейса РДС.....	158
A.5.2.24. rdsGetSystemInt – получить целый системный параметр.....	158
A.5.2.25. rdsGetSystemPath – получить системную строку.....	160
A.5.2.26. rdsIsValidVarName – проверка синтаксиса имени переменной.....	161
A.5.2.27. rdsListVarTypes – список названий типов переменных.....	162
A.5.2.28. rdsMainWindowVisible – видимость главного окна РДС.....	164
A.5.2.29. rdsModalWindowExists – наличие открытых модальных окон.....	164
A.5.2.30. rdsModalWindowMustClose – проверка принудительного закрытия модальных окон.....	165
A.5.2.31. rdsRegisterWindow – регистрация немодального окна.....	166
A.5.2.32. rdsRegWinActivateNotify – уведомление об активации зарегистрированного окна.....	167
A.5.2.33. rdsRunWithoutEvents – приостановить обработку некоторых некритических событий.....	168
A.5.2.34. rdsServiceVersion – версия РДС.....	169
A.5.2.35. rdsSetExclusiveCalc – выделенный расчет подсистемы.....	169
A.5.2.36. rdsSetModifiedFlag – установка флага наличия изменений в схеме.....	171
A.5.2.37. rdsSetSystemInt – установка целого системного параметра.....	171
A.5.2.38. rdsSetSystemUpdate – разрешить/запретить обновление вспомогательных данных.....	172
A.5.2.39. rdsShowBlockPanelTab – управление вкладками панели блоков.....	173
A.5.2.40. rdsShowMainWindow – управление видимостью главного окна РДС.....	174
A.5.2.41. rdsStartCalc – запуск расчета.....	174
A.5.2.42. rdsStopCalc – остановка расчета.....	175
A.5.2.43. rdsSystemInEditMode – РДС в режиме редактирования.....	175
A.5.2.44. rdsUnregisterWindow – отмена регистрации немодального окна.....	175
A.5.2.45. rdsUpdateExtIdsRange – обновление диапазонов идентификаторов.....	176
A.5.3. Синхронизация потоков РДС.....	176
A.5.3.1. rdsBlockDataSyncCall – вызвать функцию с блокировкой данных.....	176
A.5.3.2. rdsCallerThreadType – тип вызвавшего потока.....	178
A.5.3.3. rdsLockBlockData – включение блокировки данных.....	178
A.5.3.4. rdsUnlockAndCall – вызвать функцию, сняв блокировку данных.....	180
A.5.3.5. rdsUnlockBlockData – выключение блокировки данных.....	181
A.5.4. Отведение памяти и преобразование строк.....	182
A.5.4.1. rdsAddToDynStr – добавление строки к динамически отведенной строке.....	182
A.5.4.2. rdsAllocate – динамическое отведение области памяти.....	183
A.5.4.3. rdsAtoD – преобразование строки в вещественное число.....	184
A.5.4.4. rdsAtoI – преобразование строки в целое число.....	184
A.5.4.5. rdsDtoA – преобразование вещественного числа в строку.....	185
A.5.4.6. rdsDynStrCat – сложение двух строк.....	186
A.5.4.7. rdsDynStrCopy – создание динамической копии строки.....	187
A.5.4.8. rdsFree – освобождение отведенной динамической памяти.....	187
A.5.4.9. rdsGetFullFilePath – сокращенный путь к файлу в полный.....	188
A.5.4.10. rdsGetRelFilePath – полный путь к файлу в сокращенный.....	190
A.5.4.11. rdsItoA – преобразование целого числа в строку.....	191
A.5.4.12. rdsProcessText – обработка строки.....	192
A.5.4.13. rdsStringReplace – замена фрагментов строки.....	194
A.5.4.14. rdsTransformFileName – преобразование имени файла.....	196

A.5.5. Вызов стандартных диалогов.....	197
A.5.5.1. rdsCallColorDialog – вызов диалога выбора цвета.....	197
A.5.5.2. rdsCallDirDialog – вызов диалога выбора папки.....	197
A.5.5.3. rdsCallFileDialog – вызов диалога выбора файла.....	198
A.5.5.4. rdsExecutePrintDialog – открыть окно печати подсистемы.....	200
A.5.5.5. rdsInputString – окно ввода строки.....	200
A.5.5.6. rdsMessageBox – вывод окна сообщения.....	201
A.5.6. Операции с блоками и связями.....	203
A.5.6.1. rdsActivateOutputConnections – активация выходных связей блока.....	203
A.5.6.2. rdsAltConnAppearanceOp – операции с альтернативным внешним видом связи или шины.....	204
A.5.6.3. rdsBlockByFullName – блок по его полному имени.....	206
A.5.6.4. rdsBlockOrConnByExtId – блок или связь по внешнему идентификатору.....	206
A.5.6.5. rdsCountBlocks – подсчет блоков.....	208
A.5.6.6. rdsCreateBlockFromFile – загрузить блок из файла.....	209
A.5.6.7. rdsCreateFullBlockNameString – полное имя блока.....	210
A.5.6.8. rdsDeleteBlock – удаление блока.....	210
A.5.6.9. rdsDeleteConnection – удаление связи или шины.....	211
A.5.6.10. rdsDuplicateBlock – сделать копию блока.....	212
A.5.6.11. rdsEnumBlocks – перебрать все блоки подсистемы.....	212
A.5.6.12. rdsEnumConnectedBlocks – перебрать все соединенные блоки.....	214
A.5.6.13. rdsEnumConnectedBlocksByVar – перебрать все блоки, соединенные с заданной переменной.....	216
A.5.6.14. rdsFindNextConnectedLine – найти отрезок связи, соединенный с точкой.....	218
A.5.6.15. rdsForceBlockRedraw – перерисовать изображение блока.....	219
A.5.6.16. rdsGetBlockDescription – получить описание блока.....	220
A.5.6.17. rdsGetBlockDimensions – получить размеры и положение блока (устаревшая).....	221
A.5.6.18. rdsGetBlockDimensionsEx – получить размеры и положение блока.....	221
A.5.6.19. rdsGetBlockFlags – получить флаги параметров блока.....	222
A.5.6.20. rdsGetBlockLink – найти очередную связь, соединенную с блоком.....	223
A.5.6.21. rdsGetChildBlockByName – блок подсистемы по имени.....	224
A.5.6.22. rdsGetConnAppearance – получить внешний вид связи.....	225
A.5.6.23. rdsGetConnDescription – получить описание связи.....	225
A.5.6.24. rdsGetConnDimensions – получить размеры и положение связи.....	226
A.5.6.25. rdsGetConnStyleAppearance – получить внешний вид стиля связи/шины.....	227
A.5.6.26. rdsGetFirstBlock – первый блок в подсистеме.....	228
A.5.6.27. rdsGetFirstConn – первая связь в подсистеме.....	228
A.5.6.28. rdsGetIOBlockByVarName – внешний вход/выход по имени переменной подсистемы.....	229
A.5.6.29. rdsGetIOBlockLink – очередная связь снаружи подсистемы по идентификатору внутреннего блока-входа или выхода.....	230
A.5.6.30. rdsGetLineDescription – получить описание отрезка связи.....	231
A.5.6.31. rdsGetMouseObjectId – элемент векторной картинки блока под курсором мыши.....	232
A.5.6.32. rdsGetNextBlock – следующий блок в подсистеме.....	233
A.5.6.33. rdsGetNextConn – следующая связь в подсистеме.....	234
A.5.6.34. rdsGetParentBlock – родительская подсистема блока.....	235
A.5.6.35. rdsGetPictureObjectId – элемент векторной картинки блока.....	236
A.5.6.36. rdsGetPointDescription – получить описание точки связи.....	237
A.5.6.37. rdsGetRootSystem – корневая подсистема.....	238

A.5.6.38. rdsIsRoot – является ли подсистема корневой.....	238
A.5.6.39. rdsMakeUniqueBlockName – создать уникальное имя блока.....	239
A.5.6.40. rdsMoveBlock – переместить блок.....	240
A.5.6.41. rdsParentIsRoot – является ли родительская подсистема корневой.....	240
A.5.6.42. rdsRenameBlock – переименовать блок.....	241
A.5.6.43. rdsSelectBlock – выделить блок в редакторе.....	241
A.5.6.44. rdsSetBlockAltNameText – вывод текста вместо имени блока.....	242
A.5.6.45. rdsSetBlockComment – задать комментарий блока.....	243
A.5.6.46. rdsSetBlockFlags – установить флаги параметров блока.....	243
A.5.6.47. rdsSetBlockLayer – задать слой блока.....	244
A.5.6.48. rdsSetBlockModel – подключить к блоку модель.....	245
A.5.6.49. rdsSetBlockRect – задать прямоугольник блока.....	246
A.5.6.50. rdsSetBlockSetupFuncName – задать имя функции настройки блока.....	247
A.5.6.51. rdsSetConnAppearance – задать внешний вид связи.....	247
A.5.6.52. rdsSetConnLayer – задать слой связи.....	248
A.5.6.53. rdsSetHintText – текст всплывающей подсказки.....	249
A.5.6.54. rdsSetPointPosition – задать координаты точки связи.....	249
A.5.7. Сохранение и загрузка состояния схемы.....	250
A.5.7.1. rdsDeleteSystemState – удалить сохраненное состояние.....	250
A.5.7.2. rdsLoadSystemState – загрузить сохраненное состояние.....	251
A.5.7.3. rdsResetSystemState – сбросить состояние блока или подсистемы.....	252
A.5.7.4. rdsSaveSystemState – сохранить состояние блока/подсистемы.....	252
A.5.8. Работа с окнами подсистем.....	254
A.5.8.1. rdsCheckRectVisibility – проверить видимость прямоугольника.....	254
A.5.8.2. rdsCheckSystemWindow – открыто ли окно подсистемы.....	255
A.5.8.3. rdsCloseSystemWindow – закрыть окно подсистемы.....	255
A.5.8.4. rdsEnableWindowRefresh – разрешение/запрет обновления окон.....	255
A.5.8.5. rdsGetEditorFont – получить параметры шрифта окна подсистемы.....	257
A.5.8.6. rdsGetEditorParameters – получить описание окна подсистемы.....	258
A.5.8.7. rdsGetEditorToolBars – состояние панелей окна подсистемы (устаревшая).....	259
A.5.8.8. rdsGetEditorWindowFlags – получить флаги панелей окна подсистемы.....	259
A.5.8.9. rdsGetScreenCoords – вычислить координаты на экране по координатам на рабочем поле.....	260
A.5.8.10. rdsGetTopWindowBlock – блок-владелец активного окна.....	261
A.5.8.11. rdsHideAllEditorToolBars – скрыть все панели окна подсистемы.....	261
A.5.8.12. rdsOpenSystemWindow – открыть окно подсистемы.....	261
A.5.8.13. rdsOpenSystemWindowEx – открыть окно подсистемы с указанием его координат.....	262
A.5.8.14. rdsRefreshBlockWindows – обновить немодальные окна.....	263
A.5.8.15. rdsScrollWindowToBlock – показать блок в окне подсистемы.....	263
A.5.8.16. rdsScrollWindowToRect – показать область в окне подсистемы.....	264
A.5.8.17. rdsSetEditorToolBars – задать состояние панелей окна подсистемы (устаревшая).....	265
A.5.8.18. rdsSetEditorWindowFlags – задать флаги панелей окна подсистемы.....	265
A.5.8.19. rdsSetSystemWindowBounds – задать границы окна подсистемы.....	267
A.5.8.20. rdsSetSystemWindowCaption – задать заголовок окна подсистемы.....	267
A.5.8.21. rdsSetSystemWindowRect – задать границы окна подсистемы.....	268
A.5.8.22. rdsSetZoomPercent – задать масштаб окна подсистемы.....	269
A.5.9. Работа со слоями.....	270
A.5.9.1. rdsAddLayer – добавить слой.....	270
A.5.9.2. rdsGetLayerConfigName – имя конфигурации слоев по номеру.....	270

A.5.9.3. rdsGetLayerId – идентификатор слоя по имени.....	271
A.5.9.4. rdsGetLayerIdInConfig – идентификатор слоя по номеру в конфигурации.....	271
A.5.9.5. rdsGetLayerName – имя слоя по идентификатору.....	272
A.5.9.6. rdsGetLayerParams – параметры слоя в заданной конфигурации.....	273
A.5.9.7. rdsSetCurLayerConfig – установить текущую конфигурацию.....	274
A.5.9.8. rdsSetCurLayerConfigByName – установить текущую конфигурацию по имени.....	274
A.5.9.9. rdsSetLayerParams – задать параметры слоя в конфигурации.....	275
A.5.9.10. rdsSetLayerPosition – задать положение слоя в заданной конфигурации.....	276
A.5.10. Загрузка и сохранение данных блока.....	277
A.5.10.1. rdsReadBlockData – считать данные блока в двоичном формате.....	277
A.5.10.2. rdsReportTextLoadError – сообщение об ошибке текстового формата.....	278
A.5.10.3. rdsStructToFontText – формирование описания шрифта.....	279
A.5.10.4. rdsWriteBlockData – записать данные блока в двоичном формате.....	280
A.5.10.5. rdsWriteBlockDataText – добавление текста к сохраняемым в текстовом формате данным блока.....	280
A.5.10.6. rdsWriteColorText – запись цвета в текстовом формате.....	281
A.5.10.7. rdsWriteFontText – запись параметров шрифта в текстовом формате.....	282
A.5.10.8. rdsWriteHexText – запись блока двоичных данных в текстовом формате.....	285
A.5.10.9. rdsWriteLineStyleText – запись стиля линии в текстовом формате.....	286
A.5.10.10. rdsWriteWordDoubleText – запись вещественного числа в текстовом формате.....	287
A.5.10.11. rdsWriteWordStringText – запись строки в текстовом формате.....	288
A.5.10.12. rdsWriteWordValueText – запись целого числа в текстовом формате.....	289
A.5.11. Разбор текста.....	290
A.5.11.1. rdsFontTextToStruct – разбор описания шрифта.....	290
A.5.11.2. rdsGetTextWord – извлечение слова из текста.....	291
A.5.11.3. rdsGetTextWordDyn – извлечение слова из текста.....	293
A.5.11.4. rdsReadColorText – разбор описания цвета.....	294
A.5.11.5. rdsReadFontText – разбор описания шрифта.....	295
A.5.11.6. rdsReadHexText – разбор шестнадцатеричного блока текста.....	297
A.5.11.7. rdsReadLineStyleText – разбор стиля линии.....	298
A.5.12. Таймеры блоков.....	298
A.5.12.1. rdsDeleteBlockTimer – удалить таймер.....	298
A.5.12.2. rdsGetBlockTimerDescr – получить описание таймера.....	299
A.5.12.3. rdsRestartBlockTimer – перезапустить таймер.....	299
A.5.12.4. rdsSetBlockTimer – создать таймер.....	300
A.5.12.5. rdsStopBlockTimer – остановить таймер.....	302
A.5.13. Вызов функций блоков.....	302
A.5.13.1. Макрос RDS_FUNCPARAMCAST – приведение параметра функции к нужному типу.....	302
A.5.13.2. Макрос RDS_FUNCPARAMPVOID – поле параметра функции.....	304
A.5.13.3. Макрос RDS_FUNCPROVIDERLINK_SUCCESS – проверка успешности подписки на исполнителя функции.....	305
A.5.13.4. rdsBroadcastFuncCallsDelayed – отложенный вызов функции всех блоков подсистемы.....	305
A.5.13.5. rdsBroadcastFunctionCalls – прямой вызов функции всех блоков подсистемы (устаревшая).....	307
A.5.13.6. rdsBroadcastFunctionCallsEx – прямой вызов функции всех блоков подсистемы.....	308
A.5.13.7. rdsCallBlockFunction – прямой вызов функции блока.....	310

A.5.13.8. rdsCallBlockFunctionDelayed – отложенный вызов функции блока (устаревшая).....	311
A.5.13.9. rdsCheckBlockFunctionSupport – проверка поддержки функции блока.....	313
A.5.13.10. rdsQueueCallBlockFunction – отложенный вызов функции блока.....	313
A.5.13.11. rdsRegisterFuncProvider – регистрация блока как исполнителя функции.....	315
A.5.13.12. rdsRegisterFunction – регистрация функции блока.....	316
A.5.13.13. rdsSubscribeToFuncProvider – подписка на блок-исполнитель функции.....	317
A.5.13.14. rdsUnregisterFuncProvider – отмена регистрации блока как исполнителя функции.....	317
A.5.13.15. rdsUnsubscribeFromFuncProvider – отмена подписки на блок-исполнитель функции.....	318
A.5.14. Общие функции работы с переменными блока.....	319
A.5.14.1. rdsBlockVarFromMem – считать значение переменной блока из буфера в памяти.....	319
A.5.14.2. rdsBlockVarToMem – записать значение переменной блока в буфер в памяти.....	320
A.5.14.3. rdsClearRuntimeType – очистка переменной произвольного типа.....	321
A.5.14.4. rdsCopyRuntimeType – копировать переменную произвольного типа.....	322
A.5.14.5. rdsCopyVarGeneral – копировать значение переменной в другую переменную.....	323
A.5.14.6. rdsCreateVarDescriptionString – текстовое описание переменной.....	324
A.5.14.7. rdsCreateVarTypeText – название типа переменной.....	325
A.5.14.8. rdsFindBlockVar – найти переменную блока по имени.....	326
A.5.14.9. rdsFindStructVar – найти структуру по имени типа.....	327
A.5.14.10. rdsGetBlockVar – переменная блока по номеру.....	328
A.5.14.11. rdsGetBlockVarBase – базовый адрес переменной блока по ее номеру.....	329
A.5.14.12. rdsGetBlockVarDefValueStr – получить значение переменной блока по умолчанию.....	330
A.5.14.13. rdsGetRuntimeTypeData – получить фактические данные переменной произвольного типа.....	330
A.5.14.14. rdsGetStructVar – структура по номеру.....	331
A.5.14.15. rdsGetVarField – поле структуры или элемент массива по номеру.....	332
A.5.14.16. rdsSetBlockVarDefValueByCur – сделать текущее значение переменной блока значением по умолчанию.....	333
A.5.14.17. rdsSetBlockVarDefValueStr – установить значение переменной по умолчанию.....	334
A.5.14.18. rdsSetBlockVarFlags – установить флаги переменной.....	335
A.5.14.19. rdsSetRuntimeType – установить фактический тип переменной произвольного типа.....	335
A.5.14.20. rdsVarUsesStructType – проверить использование структуры внутри переменной.....	336
A.5.15. Работа с матрицами и массивами.....	337
A.5.15.1. Макрос RDS_ARRAYCOLS – число столбцов матрицы/массива.....	337
A.5.15.2. Макрос RDS_ARRAYDATA – указатель на первый элемент матрицы/массива.....	338
A.5.15.3. Макрос RDS_ARRAYEXISTS – проверка наличия элементов в матрице/массиве.....	339
A.5.15.4. Макрос RDS_ARRAYITEM – элемент матрицы с заданными индексами.....	339
A.5.15.5. Макрос RDS_ARRAYITEMADDR – указатель на элемент матрицы с заданными индексами.....	340
A.5.15.6. Макрос RDS_ARRAYROWS – число строк матрицы/массива.....	341

A.5.15.7. rdsCopyVarArray – копировать одну матрицу/массив в другую.....	342
A.5.15.8. rdsGetVarArrayAccessData – заполнить структуру описания матрицы/массива.....	343
A.5.15.9. rdsGetVarArrayParams – получить размеры матрицы/массива и указатель на первый элемент.....	344
A.5.15.10. rdsResizeVarArray – изменить размер матрицы/массива.....	344
A.5.15.11. rdsVarArrayIndexCheck – проверить допустимость индексов матрицы/массива.....	345
A.5.16. Работа с динамическими переменными.....	347
A.5.16.1. rdsCreateAndSubscribeDV – создать динамическую переменную и подписаться на нее.....	347
A.5.16.2. rdsCreateDynamicVar – создать динамическую переменную.....	348
A.5.16.3. rdsDeleteDVByLink – удалить динамическую переменную и прекратить подписку на нее.....	349
A.5.16.4. rdsDeleteDynamicVar – удалить динамическую переменную.....	350
A.5.16.5. rdsEnumDynVarSubscribers – перебрать всех подписчиков переменной.....	351
A.5.16.6. rdsNotifyDynVarSubscribers – уведомить подписчиков об изменении переменной.....	352
A.5.16.7. rdsSubscribeToDynamicVar – создать подписку на динамическую переменную.....	353
A.5.16.8. rdsUnsubscribeFromDynamicVar – прекратить подписку на динамическую переменную.....	354
A.5.17. Системное меню и контекстное меню блока.....	354
A.5.17.1. rdsAdditionalContextMenuItem – добавить временный пункт в контекстное меню блока.....	354
A.5.17.2. rdsAdditionalContextMenuItemEx – добавить временный пункт в контекстное меню блока.....	355
A.5.17.3. rdsChangeMenuItem – изменить параметры пункта меню.....	356
A.5.17.4. rdsEnableMenuItem – установить видимость и разрешенность пункта меню.....	358
A.5.17.5. rdsExecMenuItem – имитировать выбор пункта меню.....	359
A.5.17.6. rdsRegisterContextMenuItem – создать постоянный пункт контекстного меню блока.....	360
A.5.17.7. rdsRegisterContextMenuItemEx – создать постоянный пункт контекстного меню блока.....	361
A.5.17.8. rdsRegisterMenuItem – создать пункт системного меню РДС.....	362
A.5.17.9. rdsSetMenuItemOptions – установить флаги пункта меню.....	363
A.5.17.10. rdsUnregisterMenuItem – удалить постоянный пункт меню.....	364
A.5.18. Графические функции.....	365
A.5.18.1. Применимость графических функций.....	365
A.5.18.2. rdsXGArc – дуга эллипса или окружности.....	365
A.5.18.3. rdsXGChord – сегмент эллипса или окружности.....	366
A.5.18.4. rdsXGDrawBlockPicture – рисование картинки блока.....	367
A.5.18.5. rdsXGDrawStdIcon – рисование стандартной иконки.....	368
A.5.18.6. rdsXGEllipse – эллипс или окружность.....	369
A.5.18.7. rdsXGFillRect – заполненный прямоугольник.....	369
A.5.18.8. rdsXGFontSizeToHeight – размер шрифта в высоту в точках.....	370
A.5.18.9. rdsXGGetStdIconSize – получить размеры стандартной иконки.....	371
A.5.18.10. rdsXGGetTextSize – получить размеры строки текста.....	371
A.5.18.11. rdsXGGetVisibleRect – получить координаты видимой области.....	372
A.5.18.12. rdsXGIInvertRect – инвертировать прямоугольник.....	373

A.5.18.13. rdsXGLineTo – отрезок прямой.....	373
A.5.18.14. rdsXGMoveTo – установить текущую точку рисования.....	374
A.5.18.15. rdsXGPie – сектор эллипса или окружности.....	374
A.5.18.16. rdsXGPolygon – многоугольник.....	375
A.5.18.17. rdsXGPolyline – ломаная линия.....	375
A.5.18.18. rdsXGRectangle – прямоугольник.....	376
A.5.18.19. rdsXGRoundRect – скругленный прямоугольник.....	377
A.5.18.20. rdsXGSetBrushStyle – установить стиль заливки.....	377
A.5.18.21. rdsXGSetClipRect – установить область отсечения.....	378
A.5.18.22. rdsXGSetFont – установить шрифт.....	379
A.5.18.23. rdsXGSetFontByParStr – установить шрифт по структуре описания.....	381
A.5.18.24. rdsXGSetLogFont – установить шрифт по структуре LOGFONT.....	381
A.5.18.25. rdsXGSetPenStyle – установить стиль линии.....	382
A.5.18.26. rdsXGSetPixel – точка.....	384
A.5.18.27. rdsXGTextOut – строка текста.....	384
A.5.18.28. rdsXGTextRect – строка текста с отсечением.....	385
A.5.18.29. rdsXGTriangle – треугольник.....	385
A.5.19. Работа с временными файлами.....	386
A.5.19.1. rdsTMPCreateEmptyFile – создать временный файл.....	386
A.5.19.2. rdsTMPCreateFileSet – создать набор временных файлов.....	387
A.5.19.3. rdsTMPDeleteFile – удалить временный файл.....	388
A.5.19.4. rdsTMPDeleteFileSet – удалить набор временных файлов.....	388
A.5.19.5. rdsTMPRememberFileName – запомнить файл как временный.....	388
A.5.20. Сетевые функции.....	389
A.5.20.1. rdsNetBroadcastData – передача данных всем блокам канала.....	389
A.5.20.2. rdsNetCloseConnection – разорвать соединение.....	391
A.5.20.3. rdsNetConnect – установка сетевого соединения.....	391
A.5.20.4. rdsNetSendData – передача данных конкретному блоку канала.....	392
A.5.20.5. rdsNetServer – запуск сервера и установка соединения с ним.....	394
A.5.21. Функции поддержки внешнего управления.....	395
A.5.21.1. rdsExecutesRemoteOpsSet – регистрация блока как исполнителя операции внешнего управления.....	395
A.5.21.2. rdsGetRemoteControllerName – получить имя внешней управляющей программы.....	396
A.5.21.3. rdsGetRemoteControllerString – получить строку, установленную внешней программой.....	396
A.5.21.4. rdsHasRemoteController – проверка наличия внешнего управления.....	397
A.5.21.5. rdsRemoteControllerCall – передача сообщения управляющей программе.....	397
A.5.21.6. rdsRemoteReply – возврат строки управляющему приложению.....	398
A.5.22. Общие функции вспомогательных объектов.....	399
A.5.22.1. Использование вспомогательных объектов РДС.....	399
A.5.22.2. rdsCommandObject – команда объекту.....	400
A.5.22.3. rdsCommandObjectEx – команда объекту.....	400
A.5.22.4. rdsDeleteObject – удалить объект.....	401
A.5.22.5. rdsGetObjectArray – получить массив из объекта.....	402
A.5.22.6. rdsGetObjectDouble – получить вещественное число.....	403
A.5.22.7. rdsGetObjectDoubleP – получить вещественное число.....	404
A.5.22.8. rdsGetObjectInt – получить целое число.....	404
A.5.22.9. rdsGetObjectStr – получить строку.....	405
A.5.22.10. rdsSetObjectDouble – установить вещественное число.....	406
A.5.22.11. rdsSetObjectInt – установить целое число.....	407

A.5.22.12. rdsSetObjectStr – установить строку.....	407
A.5.23. Вспомогательный объект для изменения связей и шин.....	408
A.5.23.1. rdsCECreateEditor – создать объект-редактор связи/шины.....	408
A.5.23.2. rdsCEAddBezier – добавление кривой Безье.....	409
A.5.23.3. rdsCEAddBlockPoint – добавление точки соединения с блоком.....	410
A.5.23.4. rdsCEAddBusPoint – добавление точки соединения с шиной.....	411
A.5.23.5. rdsCEAddChannel – добавление канала шины.....	412
A.5.23.6. rdsCEAddInternalPoint – добавление промежуточной точки.....	413
A.5.23.7. rdsCEAddLine – добавление отрезка прямой.....	414
A.5.23.8. rdsCECreateConnBus – создание связи или шины по данным объекта.....	415
A.5.23.9. rdsCEEditConnBus – изменение связи или шины по данным объекта.....	416
A.5.23.10. Команда RDS_HCE_RESET – очистка вспомогательного объекта.....	417
A.5.23.11. Макрос rdsCEClearEditor – очистка вспомогательного объекта.....	418
A.5.23.12. Макрос rdsCECreateBus – создание шины.....	418
A.5.23.13. Макрос rdsCECreateConnection – создание связи.....	419
A.5.23.14. Макросы rdsCEEditConnection и rdsCEEditBus – изменение связи и шины по данным объекта.....	419
A.5.24. Вспомогательный объект для работы со списком блоков и связей.....	420
A.5.24.1. rdsBCLCreateList – создать объект для хранения списка блоков и связей.....	420
A.5.24.2. rdsBCLAddBlock – добавление блока в список.....	421
A.5.24.3. rdsBCLAddConn – добавление связи или шины в список.....	422
A.5.24.4. rdsBCLExecuteGroupSetDialog – вызов окна групповой установки.....	423
A.5.24.5. Команда RDS_HBCL_AUTODELETE – отслеживание удаления блоков и связей.....	424
A.5.24.6. Команда RDS_HBCL_BLOCKARRAY – получение массива блоков.....	425
A.5.24.7. Команда RDS_HBCL_BLOCKCOUNT – число блоков в списке.....	426
A.5.24.8. Команда RDS_HBCL_CLEAR – очистка списка.....	426
A.5.24.9. Команда RDS_HBCL_CONNARRAY – получение массива связей/шин.....	427
A.5.24.10. Команда RDS_HBCL_CONNCOUNT – число связей и шин в списке.....	428
A.5.24.11. Макрос rdsBCLGetBlockArray – получение массива блоков.....	428
A.5.24.12. Макрос rdsBCLGetConnArray – получение массива связей и шин.....	429
A.5.25. Вспомогательный объект для изменения структуры переменных блока.....	430
A.5.25.1. rdsVSCreateEditor – создать объект-редактор переменных.....	430
A.5.25.2. rdsVSAddAutoConn – добавить связь с управляющей переменной.....	430
A.5.25.3. rdsVSAddTypeRename – добавить переименование типов структур.....	431
A.5.25.4. rdsVSAddVar – добавить переменную.....	432
A.5.25.5. rdsVSAddVarByDescr – добавить переменную по строке описания.....	433
A.5.25.6. rdsVSAddVarByTypeText – добавить переменную по текстовому описанию типа.....	434
A.5.25.7. rdsVSAddVarRename – добавить переименование переменной.....	435
A.5.25.8. rdsVSApplyToBlock – создать структуру переменных блока.....	436
A.5.25.9. rdsVSCreateByDescr – заполнить набор переменных по тексту описания.....	437
A.5.25.10. rdsVSCreateFromBlock – считать структуру переменных блока.....	438
A.5.25.11. rdsVSExecuteEditor – открыть окно редактора переменных.....	438
A.5.25.12. rdsVSFindAutoConn – найти имя связанной переменной.....	440
A.5.25.13. rdsVSGetVarDefValueStr – получить строку значения переменной по умолчанию.....	441
A.5.25.14. rdsVSGetVarDescription – получить описание переменной.....	441
A.5.25.15. rdsVSInstallStruct – добавить структуру в общий список структур.....	442
A.5.25.16. rdsVSSetVarFlags – установить флаги переменной.....	443
A.5.25.17. rdsVSUsesStructType – используется ли структура в объекте.....	444

A.5.25.18. Команда RDS_HVAR_CLEARPYPEREN – очистка списка переименований структур.....	444
A.5.25.19. Команда RDS_HVAR_CLEARVARREN – очистка списка переименований переменных.....	445
A.5.25.20. Команда RDS_HVAR_DELAUTO – удалить связь с управляющей переменной.....	445
A.5.25.21. Команда RDS_HVAR_DELVAR – удалить переменную.....	446
A.5.25.22. Команда RDS_HVAR_GETAUTOCONN – получить имя связанной переменной по номеру связи.....	447
A.5.25.23. Команда RDS_HVAR_GETAUTOCOUNT – получить число связей основных и вспомогательных переменных.....	447
A.5.25.24. Команда RDS_HVAR_GETAUTOMAIN – получить имя основной переменной по номеру связи.....	448
A.5.25.25. Команда RDS_HVAR_GETFIELDCOUNT – получить число переменных в объекте.....	449
A.5.25.26. Команда RDS_HVAR_GETTYPENAME – получить имя типа всей структуры переменных.....	449
A.5.25.27. Команда RDS_HVAR_GETTYPESTRING – получить строку типа всей структуры переменных.....	450
A.5.25.28. Команда RDS_HVAR_GETVARFLAGS – получить флаги переменной.....	450
A.5.25.29. Команда RDS_HVAR_GETVARRANK – получить уровень всей структуры переменных.....	451
A.5.25.30. Команда RDS_HVAR_RESET – очистка объекта.....	451
A.5.25.31. Команда RDS_HVAR_SETTYPENAME – установить имя типа всей структуры переменных.....	452
A.5.25.32. Команда RDS_HVAR_SETVARFLAGS – одновременно установить все флаги переменной.....	452
A.5.25.33. Макрос rdsVSClearEditor – очистка объекта.....	453
A.5.25.34. Макрос rdsVSClearTypeRenames – очистка списка переименований структур.....	454
A.5.25.35. Макрос rdsVSClearVarRenames – очистка списка переименований переменных.....	454
A.5.25.36. Макрос rdsVSDeleteAutoConn – удалить связь между основной и управляющей переменными.....	455
A.5.25.37. Макрос rdsVSDeleteVar – удалить переменную.....	455
A.5.25.38. Макрос rdsVSGetAutoConn – получить имя связанной переменной по номеру связи.....	456
A.5.25.39. Макрос rdsVSGetAutoCount – получить число связей основных и вспомогательных переменных.....	456
A.5.25.40. Макрос rdsVSGetAutoMain – получить имя основной переменной по номеру связи.....	457
A.5.25.41. Макрос rdsVSGetFieldCount – получить число переменных в объекте.....	457
A.5.25.42. Макрос rdsVSGetStructName – получить имя типа всей структуры переменных.....	458
A.5.25.43. Макрос rdsVSGetStructRank – получить уровень всей структуры переменных.....	458
A.5.25.44. Макрос rdsVSSetStructName – установить имя типа всей структуры переменных.....	459
A.5.26. Вспомогательный объект для разбора текста.....	460
A.5.26.1. rdsSTRCreateTextReader – создать объект для разбора текста.....	460
A.5.26.2. rdsSTRAddKeyword – добавление ключевого слова.....	460

A.5.26.3. rdsSTRAddKeywordsArray – добавление набора ключевых слов.....	461
A.5.26.4. rdsSTRGetWord – считать из текста очередное слово.....	462
A.5.26.5. Команда RDS_HSTR_ENDOFLINEID – идентификатор конца строки.....	466
A.5.26.6. Команда RDS_HSTR_ENDOFTEXTID – идентификатор конца текста....	466
A.5.26.7. Команда RDS_HSTR_GETLASTWORD – получить последнее считанное слово.....	467
A.5.26.8. Команда RDS_HSTR_GETRESTOFTEXT – получить остаток текста.....	467
A.5.26.9. Команда RDS_HSTR_IGNORECASE – учет регистра символов ключевых слов.....	468
A.5.26.10. Команда RDS_HSTR_READDOUBLE – получить из текста вещественное число.....	468
A.5.26.11. Команда RDS_HSTR_READINT – получить из текста целое число.....	469
A.5.26.12. Команда RDS_HSTR_SETTEXT – передать в объект текст для разбора.	470
A.5.26.13. Команда RDS_HSTR_UNKNOWNID – идентификатор неопознанного слова.....	471
A.5.26.14. Макрос rdsSTRGetDoubleWord – получить из текста слово как вещественное число.....	471
A.5.26.15. Макрос rdsSTRGetIntWord – получить из текста слово как целое число.	472
A.5.26.16. Макрос rdsSTRSetTextToRead – передать в объект текст для разбора....	472
A.5.27. Вспомогательный объект для работы с текстом в формате INI-файла.....	473
A.5.27.1. rdsINICreateTextHolder – создать объект для работы с текстом.....	473
A.5.27.2. rdsINIOpenSection – установить текущую секцию.....	474
A.5.27.3. rdsINIReadDouble – получить вещественное значение параметра.....	475
A.5.27.4. rdsINIReadDoubleP – получить вещественное значение параметра.....	476
A.5.27.5. rdsINIReadInt – получить целое значение параметра.....	477
A.5.27.6. rdsINIReadString – получить текст значения параметра.....	477
A.5.27.7. rdsINIWriteDouble – установить вещественное значение параметра.....	478
A.5.27.8. rdsINIWriteInt – установить целое значение параметра.....	479
A.5.27.9. rdsINIWriteString – установить текстовое значение параметра.....	480
A.5.27.10. Команда RDS_HINI_CREATESECTION – создать секцию.....	480
A.5.27.11. Команда RDS_HINI_DELETEKEYLAST – удалить параметр из текущей секции.....	481
A.5.27.12. Команда RDS_HINI_DELETESECTION – удалить секцию.....	481
A.5.27.13. Команда RDS_HINI_GETLASTERROR – получить результат последней операции.....	482
A.5.27.14. Команда RDS_HINI_LOADFILE – загрузить текст из файла.....	482
A.5.27.15. Команда RDS_HINI_RESET – очистка текста.....	483
A.5.27.16. Команда RDS_HINI_SAVEBLOCKTEXT – передать текст параметров блока в РДС.....	483
A.5.27.17. Команда RDS_HINI_SAVEFILE – записать текст в файл.....	484
A.5.27.18. Команда RDS_HINI_SETTEXT – занести текст в объект.....	485
A.5.27.19. Макрос rdsINIClearText – очистка объекта.....	485
A.5.27.20. Макрос rdsINICreateSection – создать секцию.....	486
A.5.27.21. Макрос rdsINIDeleteSection – удалить секцию.....	486
A.5.27.22. Макрос rdsINIDeleteValue – удалить параметр из текущей секции.....	487
A.5.27.23. Макрос rdsINILoadFile – загрузить текст из файла.....	487
A.5.27.24. Макрос rdsINIReadBool – получить логическое значение параметра.....	488
A.5.27.25. Макрос rdsINISaveBlockText – передать текст параметров блока в РДС.	489
A.5.27.26. Макрос rdsINISaveFile – записать текст в файл.....	489
A.5.27.27. Макрос rdsINISetText – занести текст в объект.....	490
A.5.27.28. Макрос rdsINIWriteBool – установить логическое значение параметра...	490

A.5.28. Вспомогательный объект для работы с модальными окнами.....	491
A.5.28.1. rdsFORMCreate – создать объект для работы с окном.....	491
A.5.28.2. rdsFORMAddEdit – добавить поле ввода.....	492
A.5.28.3. Типы и флаги полей ввода.....	494
A.5.28.4. rdsFORMAddTab – добавить вкладку.....	501
A.5.28.5. rdsFORMEnableSidePanel – включить боковую панель.....	502
A.5.28.6. rdsFORMShowModalEx – открыть окно с функцией обратного вызова.....	503
A.5.28.7. rdsFORMShowModalServ – открыть окно с расширенной функцией обратного вызова.....	504
A.5.28.8. Команда RDS_FORM_CLEAR – очистить объект.....	505
A.5.28.9. Команда RDS_FORM_INVALIDATE – обновить окно.....	505
A.5.28.10. Команда RDS_FORM_SHOWMODAL – открыть окно.....	506
A.5.28.11. Команда RDS_FORMVAL_2NDEDITENABLED – разрешение основного поля в двойном поле ввода со списком.....	506
A.5.28.12. Команда RDS_FORMVAL_AUXLISTITEM – номер варианта выпадающего списка в двойном поле ввода со списком.....	508
A.5.28.13. Команда RDS_FORMVAL_AUXLISTWIDTH – ширина выпадающего списка в двойном поле ввода со списком.....	509
A.5.28.14. Команда RDS_FORMVAL_CHECK – управление разрешающим флагом поля ввода.....	510
A.5.28.15. Команда RDS_FORMVAL_ENABLED – разрешение и запрещение всего поля ввода.....	511
A.5.28.16. Команда RDS_FORMVAL_HKSHIFTS – состояние Ctrl, Alt и Shift в поле ввода кода клавиши.....	512
A.5.28.17. Команда RDS_FORMVAL_ITEMINDEX – номер варианта, выбранного в выпадающем списке с фиксированными вариантами.....	513
A.5.28.18. Команда RDS_FORMVAL_LIST – установка списка вариантов.....	515
A.5.28.19. Команда RDS_FORMVAL_MLHEIGHT – высота многострочного поля ввода.....	516
A.5.28.20. Команда RDS_FORMVAL_MLRETURNS – управление реакцией многострочного поля ввода на клавишу Enter.....	517
A.5.28.21. Команда RDS_FORMVAL_PBBEVEL – установка рамки вокруг области рисования.....	517
A.5.28.22. Команда RDS_FORMVAL_PBHEIGHT – высота области программного рисования.....	518
A.5.28.23. Команда RDS_FORMVAL_RANGEMAX – значение поля ввода конца диапазона.....	520
A.5.28.24. Команда RDS_FORMVAL_UPDOWNINC – шаг изменения поля ввода со стрелками.....	521
A.5.28.25. Команда RDS_FORMVAL_UPDOWNMAX – максимальное значение поля ввода со стрелками.....	522
A.5.28.26. Команда RDS_FORMVAL_UPDOWNMIN – минимальное значение поля ввода со стрелками.....	524
A.5.28.27. Команда RDS_FORMVAL_VALUE – значение поля.....	525
A.5.28.28. Макрос rdsFORMClear – очистка объекта.....	527
A.5.28.29. Макрос rdsFORMEnableControl – разрешение и запрещение всего поля ввода.....	528
A.5.28.30. Макрос rdsFORMGetBool – получение целого значения поля ввода и преобразование его в логическое.....	528
A.5.28.31. Макрос rdsFORMGetDouble – получение вещественного значения поля ввода.....	529

A.5.28.32. Макрос rdsFORMGetEnableCheck – получение значения дополнительного разрешающего флага поля ввода.....	530
A.5.28.33. Макрос rdsFORMGetInt – получение целого значения поля ввода.....	530
A.5.28.34. Макрос rdsFORMGetString – получение значения поля ввода в виде строки.....	531
A.5.28.35. Макрос rdsFORMSetBool – установка целого значения поля ввода как логического.....	532
A.5.28.36. Макрос rdsFORMSetComboList – установка списка вариантов.....	532
A.5.28.37. Макрос rdsFORMSetDouble – установка вещественного значения поля ввода.....	533
A.5.28.38. Макрос rdsFORMSetEnableCheck – установка значения дополнительного разрешающего флага поля ввода.....	534
A.5.28.39. Макрос rdsFORMSetInt – установка целого значения поля ввода.....	534
A.5.28.40. Макрос rdsFORMSetMultilineHeight – установка высоты многострочного поля ввода.....	535
A.5.28.41. Макрос rdsFORMShowModal – открытие модального окна.....	536
A.5.29. Вспомогательный объект для отмены редактирования параметров блока.....	536
A.5.29.1. rdsBEUCreate – создать объект для отмены редактирования параметров блока.....	536
A.5.29.2. Команда RDS_BEU_STORECHANGED – запись измененных параметров блока.....	537
A.5.29.3. Макрос rdsBEUStore – запись измененных параметров блока.....	538
A.5.30. Вспомогательный объект для вывода индикатора выполнения.....	538
A.5.30.1. rdsPBARCreate – создать объект для вывода индикатора выполнения.....	538
A.5.30.2. Команда RDS_PBAR_ADDTOPOS – добавить число к текущей позиции индикатора.....	540
A.5.30.3. Команда RDS_PBAR_HIDE – убрать индикатор с экрана.....	540
A.5.30.4. Команда RDS_PBAR_MAX – максимальное значение индикатора.....	541
A.5.30.5. Команда RDS_PBAR_POSITION – текущая позиция индикатора.....	541
A.5.30.6. Команда RDS_PBAR_RESET – сбросить индикатор.....	542
A.5.30.7. Команда RDS_PBAR_SETCAPTION – установить заголовок индикатора.....	543
A.5.30.8. Команда RDS_PBAR_SHOW – показать индикатор.....	543
A.5.30.9. Макрос rdsPBARIncrement – увеличить текущую позицию индикатора.....	543
A.5.30.10. Макрос rdsPBARSetPos – установить текущую позицию индикатора.....	544
A.5.30.11. Макрос rdsPBARShow – показать или скрыть индикатор.....	544
A.5.31. Вспомогательный объект для панелей в окне подсистемы.....	545
A.5.31.1. rdsPANCreate – создать объект для работы с панелью.....	545
A.5.31.2. rdsPANGetDescr – получить описание панели.....	547
A.5.31.3. Команда RDS_PAN_CAPTION – заголовок панели.....	548
A.5.31.4. Команда RDS_PAN_CLIENTHEIGHT – высота внутренней, доступной для модели, части панели.....	548
A.5.31.5. Команда RDS_PAN_CLIENTWIDTH – ширина внутренней, доступной для модели, части панели.....	549
A.5.31.6. Команда RDS_PAN_FLAGS – флаги панели.....	549
A.5.31.7. Команда RDS_PAN_HEIGHT – общая высота панели.....	550
A.5.31.8. Команда RDS_PAN_LEFT – горизонтальная координата панели.....	550
A.5.31.9. Команда RDS_PAN_MAXCLHEIGHT – максимальная высота внутренней части панели.....	551
A.5.31.10. Команда RDS_PAN_MAXCLWIDTH – максимальная ширина внутренней части панели.....	552

A.5.31.11. Команда RDS_PAN_MINCLHEIGHT – минимальная высота внутренней части панели.....	552
A.5.31.12. Команда RDS_PAN_MINCLWIDTH – минимальная ширина внутренней части панели.....	553
A.5.31.13. Команда RDS_PAN_TOP – вертикальная координата панели.....	554
A.5.31.14. Команда RDS_PAN_VISIBLE – видимость панели.....	554
A.5.31.15. Команда RDS_PAN_WIDTH – общая ширина панели.....	555
A.5.32. Вспомогательный объект для работы с форматом CSV.....	556
A.5.32.1. rdsCSVCreate – создать объект для работы с текстом в формате CSV.....	556
A.5.32.2. rdsCSVGetItem – получить элемент текста.....	557
A.5.32.3. rdsCSVSetItem – установить элемент текста.....	558
A.5.32.4. Команда RDS_CSV_CLEAR – очистка текста.....	559
A.5.32.5. Команда RDS_CSV_CLOSEFILE – закрыть файл.....	559
A.5.32.6. Команда RDS_CSV_DELIMITERCHAR – символ-разделитель.....	560
A.5.32.7. Команда RDS_CSV_FILEERROR – проверка ошибки в последней операции с файлом.....	560
A.5.32.8. Команда RDS_CSV_FILEISOPEN – проверка успешности открытия файла для чтения или записи.....	561
A.5.32.9. Команда RDS_CSV_LINE – одна строка текста.....	561
A.5.32.10. Команда RDS_CSV_LINECOLUMNS – число элементов в строке.....	562
A.5.32.11. Команда RDS_CSV_LINECOUNT – число строк в тексте.....	563
A.5.32.12. Команда RDS_CSV_LOADFROMFILE – загрузить текст из файла.....	563
A.5.32.13. Команда RDS_CSV_MAXCOLUMNS – число элементов в самой длинной строке.....	564
A.5.32.14. Команда RDS_CSV_OPENFILEREAD – открыть файл для чтения.....	564
A.5.32.15. Команда RDS_CSV_OPENFILEWRITE – открыть файл для записи.....	565
A.5.32.16. Команда RDS_CSV_QUOTECHAR – символ-ограничитель строки.....	566
A.5.32.17. Команда RDS_CSV_SAVETOFILE – записать текст в файл.....	566
A.5.32.18. Команда RDS_CSV_STRFROMFILE – считать строку из файла.....	567
A.5.32.19. Команда RDS_CSV_STRTOFILE – записать строку в файл.....	567
A.5.32.20. Команда RDS_CSV_TEXT – весь текст объекта.....	568
A.5.33. Отладочные функции.....	569
A.5.33.1. rdsBlockMessageBox – вывести окно сообщения с указанием имени блока....	569
A.5.33.2. rdsdebugBlockInfo – информация о блоке.....	570
A.5.33.3. rdsdebugLogString – добавить строку в текстовый файл.....	571
A.5.33.4. rdsSetDebugText – установить отладочный текст.....	572
A.5.34. Функции поддержки автоматической компиляции моделей.....	572
A.5.34.1. rdscompAttachDifferentModel – замена имени подключаемой модели.....	572
A.5.34.2. rdscompCompileModel – компилировать модель.....	573
A.5.34.3. rdscompGetBlockModelData – получить данные модели блока.....	574
A.5.34.4. rdscompGetModelBlock – обслуживаемый моделью блок по номеру.....	574
A.5.34.5. rdscompGetModelData – обслуживаемая модулем модель по номеру.....	575
A.5.34.6. rdscompGetModelDataByName – модель по имени.....	576
A.5.34.7. rdscompOpenBlockModelEditor – вызвать редактор модели блока.....	577
A.5.34.8. rdscompRenameModel – переименовать модель.....	577
A.5.34.9. rdscompReturnModelName – возврат имени модели из функции пользовательского интерфейса.....	579
A.5.34.10. rdscompReturnModelNameLabel – заголовок поля ввода имени модели.....	579
A.5.34.11. rdscompSetAltModelName – установить альтернативное имя модели.....	580
A.5.34.12. rdscompSetBlockModel – подключить модель к блоку.....	580

A.5.34.13. rdscompSetModelFunction – установить имена DLL и функции скомпилированной модели.....	582
Приложение Б. Функции, константы и структуры библиотеки RdsCtrl.dll.....	583
Б.1. Структуры библиотеки RdsCtrl.dll.....	583
Б.1.1. RDSCTRL_BLOCKMSGDATA – сообщение от блока.....	583
Б.1.2. RDSCTRL_MENUITEM – описание пункта меню РДС.....	584
Б.1.3. RDSCTRL_MSGEVENTDATA – сообщение от РДС.....	586
Б.1.4. RDSCTRL_NEWFILEDATA – описание события смены загруженной схемы....	587
Б.1.5. RDSCTRL_PROGRESSDATA – описание события хода загрузки или сохранения схемы.....	588
Б.1.6. RDSCTRL_SAVEFILEDATA – описание события сохранения схемы.....	589
Б.1.7. RDSCTRL_SETTINGS – общие параметры РДС.....	590
Б.1.8. RDSCTRL_ZOOMRECT – параметры прямоугольника масштабирования.....	591
Б.2. События, на которые может реагировать программа.....	592
Б.2.1. Способы реакции на события.....	592
Б.2.2. RDSCTRLEVENT_BLOCKMSG – сообщение от блока.....	593
Б.2.3. RDSCTRLEVENT_CALCMODE – режим моделирования.....	594
Б.2.4. RDSCTRLEVENT_CALCSTART – запуск расчета.....	594
Б.2.5. RDSCTRLEVENT_CALCSTOP – остановка расчета.....	595
Б.2.6. RDSCTRLEVENT_CONNCLOSED – завершение РДС.....	595
Б.2.7. RDSCTRLEVENT_EDITMODE – режим редактирования.....	596
Б.2.8. RDSCTRLEVENT_LOADREQ – запрос загрузки схемы.....	596
Б.2.9. RDSCTRLEVENT_NEWFILE – загрузка или создание схемы.....	597
Б.2.10. RDSCTRLEVENT_PROGRESS – ход загрузки или сохранения схемы.....	597
Б.2.11. RDSCTRLEVENT_SAVEFILE – сохранение схемы.....	598
Б.3. Функции библиотеки RdsCtrl.dll.....	598
Б.3.1. Доступ к функциям RdsCtrl.dll.....	598
Б.3.2. Функции управления связью с РДС.....	599
Б.3.2.1. rdscrtlClose – завершить РДС.....	599
Б.3.2.2. rdscrtlConnect – запустить РДС.....	600
Б.3.2.3. rdscrtlCreateLink – создать связь с РДС.....	600
Б.3.2.4. rdscrtlDeleteLink – уничтожить связь с РДС.....	601
Б.3.2.5. rdscrtlDisconnect – завершить РДС.....	602
Б.3.2.6. rdscrtlIsConnected – проверить связь с РДС.....	602
Б.3.2.7. rdscrtlLeave – прекратить управление РДС.....	603
Б.3.2.8. rdscrtlRestoreConnection – перезапустить РДС при необходимости.....	603
Б.3.2.9. rdscrtlSetPath – установить путь к РДС.....	604
Б.3.2.10. rdscrtlSetStringCallback – регистрация функции возврата строки.....	604
Б.3.3. Функции управления интерфейсом пользователя РДС.....	606
Б.3.3.1. rdscrtlEnableCalcMode – разрешение режима моделирования.....	606
Б.3.3.2. rdscrtlEnableEditMode – разрешение режима редактирования.....	606
Б.3.3.3. rdscrtlEnableOptions – разрешение настройки РДС.....	607
Б.3.3.4. rdscrtlEnablePropEdit – разрешение изменения параметров блоков РДС.....	608
Б.3.3.5. rdscrtlEnableRun – разрешение запуска расчета.....	609
Б.3.3.6. rdscrtlEnableRunInterface – разрешение переключения режимов.....	609
Б.3.3.7. rdscrtlEnableSubsystemWindows – разрешение открытия окон подсистем....	610
Б.3.3.8. rdscrtlEnableUI – разрешение интерфейса пользователя РДС.....	611
Б.3.3.9. rdscrtlGetGeneralSettings – получить настройки интерфейса пользователя РДС	612
Б.3.3.10. rdscrtlSetAutoSave – установить автосохранение при выходе.....	612
Б.3.3.11. rdscrtlSetExitMode – установить предупреждение о выходе из РДС.....	613

Б.3.3.12. rdsetrlSetGeneralSettings – установить настройки интерфейса пользователя РДС.....	614
Б.3.3.13. rdsetrlShowMainWindow – видимость главного окна РДС.....	614
Б.3.4. Функции общего назначения.....	615
Б.3.4.1. rdsetrlBlockExtIdByName – внешний идентификатор по имени блока.....	615
Б.3.4.2. rdsetrlBlockMenuClick – имитация выбора пункта меню блока.....	616
Б.3.4.3. rdsetrlBlockMenuClickEx – имитация выбора пункта меню блока.....	617
Б.3.4.4. rdsetrlBlockNameByExtId – имя блока по внешнему идентификатору.....	618
Б.3.4.5. rdsetrlBringAppToFront – переместить РДС на передний план.....	619
Б.3.4.6. rdsetrlCallBlockFunction – передача блоку числа и строки (устаревшая).....	619
Б.3.4.7. rdsetrlCallBlockFunctionEx – передача блоку числа и строки.....	621
Б.3.4.8. rdsetrlCheckModalWindows – проверить наличие модальных окон.....	622
Б.3.4.9. rdsetrlClearSystem – очистить схему.....	623
Б.3.4.10. rdsetrlCloseAllSysExceptRoot – закрыть окна всех подсистем, открыть окно корневой.....	623
Б.3.4.11. rdsetrlCloseAllWindows – закрыть все окна.....	624
Б.3.4.12. rdsetrlCloseModalWindows – закрыть все модальные окна.....	624
Б.3.4.13. rdsetrlCloseSysWindow – закрыть окно подсистемы.....	625
Б.3.4.14. rdsetrlEnableWinRefresh – разрешение/запрет обновления окон.....	625
Б.3.4.15. rdsetrlFindBlock – параметры блока по имени.....	626
Б.3.4.16. rdsetrlFindOpSetProviders – поиск блоков, выполняющих операцию.....	628
Б.3.4.17. rdsetrlGetBlockVarValue – получить значение переменной блока.....	629
Б.3.4.18. rdsetrlGetMenuItemData – получить данные пункта меню.....	630
Б.3.4.19. rdsetrlGetMode – получить режим работы РДС.....	631
Б.3.4.20. rdsetrlGetModFlag – проверить наличие изменений в схеме.....	632
Б.3.4.21. rdsetrlListBlocks – получить список блоков.....	632
Б.3.4.22. rdsetrlMinimizeApp – свернуть приложение РДС.....	634
Б.3.4.23. rdsetrlReadBlockMenuItems – считать пункты меню блока в память.....	635
Б.3.4.24. rdsetrlResetCalc – сбросить расчет.....	636
Б.3.4.25. rdsetrlRestoreApp – восстановить свернутое приложение РДС.....	637
Б.3.4.26. rdsetrlSetBlockVarValue – установить значение переменной блока.....	637
Б.3.4.27. rdsetrlSetCalcMode – включить режим моделирования.....	638
Б.3.4.28. rdsetrlSetControllerName – установить имя управляющей программы.....	639
Б.3.4.29. rdsetrlSetEditMode – включить режим редактирования.....	640
Б.3.4.30. rdsetrlSetModFlag – управление флагом изменений в схеме.....	640
Б.3.4.31. rdsetrlSetString – установить глобальную строку.....	641
Б.3.4.32. rdsetrlStartCalc – запустить расчет.....	642
Б.3.4.33. rdsetrlStopCalc – остановить расчет.....	642
Б.3.5. Функции загрузки и сохранения схемы.....	643
Б.3.5.1. rdsetrlDeleteExchangeMemory – удалить разделяемую память.....	643
Б.3.5.2. rdsetrlEndBlockByBlockLoad – завершить поблочную загрузку схемы.....	643
Б.3.5.3. rdsetrlEndBlockByBlockSave – завершить поблочное сохранение схемы.....	644
Б.3.5.4. rdsetrlGetBlockByBlockSavePiece – получить очередной объект при поблочном сохранении.....	645
Б.3.5.5. rdsetrlGetSystemContent – получить полный текст схемы.....	647
Б.3.5.6. rdsetrlLoadSystemFromFile – загрузить схему из файла.....	648
Б.3.5.7. rdsetrlLoadSystemFromMem – загрузить схему из области памяти.....	649
Б.3.5.8. rdsetrlLoadSystemTagged – загрузить схему из файла в специальном двоичном формате.....	650
Б.3.5.9. rdsetrlLoadSystemTaggedEx – загрузить схему из файла или памяти в специальном двоичном формате.....	651

Б.3.5.10. rdscrtlNoDirectLoad – запрет загрузки схемы пользователем.....	652
Б.3.5.11. rdscrtlNoDirectSave – запрет сохранения схемы пользователем.....	653
Б.3.5.12. rdscrtlSaveSystemTagged – записать схему в файл в специальном двоичном формате.....	654
Б.3.5.13. rdscrtlSaveSystemTaggedEx – записать схему в файл или разделяемую память в специальном двоичном формате.....	656
Б.3.5.14. rdscrtlSaveSystemTaggedMem – записать схему в разделяемую память в специальном двоичном формате.....	658
Б.3.5.15. rdscrtlSaveSystemToFile – сохранить схему в файл.....	660
Б.3.5.16. rdscrtlSetBlockByBlockLoadPiece – передать очередной объект при поблочной загрузке.....	661
Б.3.5.17. rdscrtlSetProgressDelay – установка интервала между сообщениями о ходе загрузки/сохранения.....	662
Б.3.5.18. rdscrtlStartBlockByBlockLoad – начать поблочную загрузку схемы.....	663
Б.3.5.19. rdscrtlStartBlockByBlockSave – начать поблочное сохранение схемы.....	664
Б.3.6. Функции реакции на события.....	665
Б.3.6.1. rdscrtlEnableEvents – разрешение реакции на события.....	665
Б.3.6.2. rdscrtlRegisterBlockMsgCallback – регистрация функции для реакции на сообщение от блока.....	665
Б.3.6.3. rdscrtlRegisterEventMessage – регистрация оконного сообщения для реакции на событие.....	667
Б.3.6.4. rdscrtlRegisterEventStdCallback – регистрация функции для реакции на событие.....	668
Б.3.6.5. rdscrtlUnregisterEvent – отмена регистрации реакции на событие.....	669
Б.3.7. Функции для работы с портами вывода.....	670
Б.3.7.1. rdscrtlGetViewportParams – получить масштаб и сдвиг подсистемы в порте вывода.....	670
Б.3.7.2. rdscrtlGetViewportSysArea – размеры рабочего поля подсистемы в порте вывода.....	671
Б.3.7.3. rdscrtlGetVPMouseLevel – тип реакции на мышшь у подсистемы в порте вывода.....	672
Б.3.7.4. rdscrtlReleaseViewport – уничтожить порт вывода.....	673
Б.3.7.5. rdscrtlSetViewport – создать порт вывода.....	674
Б.3.7.6. rdscrtlSetViewportParams – установить масштаб и сдвиг подсистемы в порте вывода.....	675
Б.3.7.7. rdscrtlSetViewportRect – задать положение порта вывода в окне.....	676
Б.3.7.8. rdscrtlSetViewportZoomRect – настроить масштаб и сдвиг подсистемы по прямоугольнику.....	676
Б.3.7.9. rdscrtlSetViewportZoomRectEx – настроить масштаб и сдвиг подсистемы по прямоугольнику (расширенная).....	678
Б.3.7.10. rdscrtlUpdateViewport – обновить порт вывода.....	678
Б.3.7.11. rdscrtlViewportBlockAtPos – получить имя блока в указанной точке порта вывода.....	679
Б.3.7.12. rdscrtlViewportFit – настроить порт вывода на изображение всей подсистемы.....	680
Б.3.7.13. rdscrtlViewportKeyboard – вызвать в РДС реакцию на клавиатуру.....	681
Б.3.7.14. rdscrtlViewportMouse – вызвать в РДС реакцию на мышшь.....	682
Б.3.7.15. rdscrtlViewportSystem – получить имя подсистемы в порте вывода.....	684
Б.3.7.16. rdscrtlVPPopupHint – получить текст всплывающей подсказки.....	685
Б.3.8. Отладочные функции.....	686
Б.3.8.1. rdscrtlClearLog – очистить журнал.....	686

Б.3.8.2. rdscrtlEnableLog – включить/выключить журнал.....	687
Б.3.8.3. rdscrtlLogString – записать текст в журнал.....	687
Б.3.8.4. rdscrtlSetLogFile – задать имя файла журнала.....	688
Приложение В. Параметры командной строки РДС.....	689
В.1. Передача параметров в командной строке и указание имени файла схемы.....	689
В.2. “/calcmode” – перейти в режим моделирования.....	689
В.3. “/end” – прекратить разбор параметров.....	689
В.4. “/hide” – скрыть главное окно РДС.....	690
В.5. “/ini” – задать путь к INI-файлам РДС.....	690
В.6. “/nosplash” – не выводить заставку РДС.....	690
В.7. “/run” – перейти в режим расчета.....	690
В.8. “/server” – запустить выделенный сервер РДС.....	691
В.9. “/skip” – не разбирать следующий параметр.....	691
В.10. “/temp” – задать путь к папке временных файлов РДС.....	691
В.11. “/wintemp” – использовать папку временных файлов Windows.....	691
Алфавитный указатель.....	693

Приложение А. Функции, константы и структуры РДС

Описываются сервисные функции, экспортированные из модуля rds.exe, используемые в них структуры и константы. Применение этих функций для создания моделей блоков описывается в главе 2, для создания модулей автоматической компиляции моделей – в главе 4.

А.1. Идентификаторы объектов и вспомогательные типы

Схема в РДС состоит из множества различных объектов: блоков, связей, переменных, таймеров и т.п. Каждый из таких объектов имеет уникальный идентификатор, который нужно указывать в сервисных функциях для ссылки на конкретный объект, с которым производится то или иное действие. Большинство этих идентификаторов представляют собой указатели на какие-либо данные (тип LPVOID Windows API), но, для лучшей читаемости исходных текстов программ, в файле “RdsDef.h” для них оператором typedef введены специальные типы:

<i>Tun</i>	<i>Объект</i>
RDS_BHANDLE	Блок в схеме, независимо от его типа (простой блок, подсистема, внешний вход или выход, ввод шины). Используется во всех сервисных функциях, получающих или устанавливающих параметры блоков. Модель блока всегда имеет доступ к идентификатору своего блока и его родительской подсистемы через структуру данных блока RDS_BLOCKDATA (см. стр. 28).
RDS_CHANDLE	Связь или шина в схеме. Используется во всех сервисных функциях, получающих или устанавливающих параметры связей и шин.
RDS_TIMERID	Программируемый таймер блока. Используется в функциях работы с таймерами (см. А.5.12), в реакциях на событие RDS_BFM_TIMER (стр. 47) и RDS_BFM_WINREFRESH (стр. 77).
RDS_MENUITEM	Пункт системного или контекстного меню, добавленный моделью блока. Используется в сервисных функциях работы с меню (см. А.5.17).
RDS_VHANDLE	Статическая или динамическая переменная блока. Используется в некоторых сервисных функциях для получения описания переменных и их редактирования. Для доступа к значениям переменных, как правило, не используется (см. §2.5 и §2.6).
RDS_HOBJECT	Вспомогательный объект РДС (см. А.5.22). Вспомогательные объекты служат для различных целей: редактирования связей, разбора текста, открытия окон и т.п., но все они имеют этот тип.
RDS_NETSTATION	Одна из машин с запущенной копией РДС, подключенная к серверу. Используется для указания машины при отправке данных по сети конкретному блоку функцией rdsNetSendData (стр. 392) и в реакциях на события RDS_BFM_NETDATA RECEIVED (стр. 87) и RDS_BFM_NETERROR (стр. 89).

<i>Tun</i>	<i>Объект</i>
RDS_NETBLOCK	Блок на одной из машин с запущенной копией РДС, подключенной к серверу. Используется для указания блока при отправке данных по сети конкретному блоку функцией <code>rdsNetSendData</code> (стр. 392) и в реакциях на события <code>RDS_BFM_NETDATARECEIVED</code> (стр. 87) и <code>RDS_BFM_NETERROR</code> (стр. 89). Не взаимозаменяем с <code>RDS_BHANDLE</code> – последний может использоваться только в качестве идентификатора блока в пределах одной схемы на одной машине.
RDS_COMPHANDLE	Модуль автоматической компиляции. Используется в сервисных функциях, работающих с автокомпилируемой моделью (А.5.34), для указания конкретного модуля.
RDS_MODELHANDLE	Автоматически компилируемая модель. Используется в сервисных функциях, работающих с автокомпилируемой моделью, для указания конкретной модели.

Для взаимодействия с Windows API многие сервисные функции и структуры РДС работают со значениями следующих, стандартных для Windows, типов:

<i>Tun</i>	<i>Объект</i>
BOOL	Логическое значение, может принимать значения TRUE (истина) или FALSE (ложь). Этот тип эквивалентен стандартному типу C “int”.
COLORREF	Тридцатидвухбитное целое число, описывающее цвет. Младший байт числа содержит интенсивность красного канала (в диапазоне от 0 до 255), второй байт – интенсивность зеленого, третий байт – интенсивность синего, четвертый (старший) байт всегда нулевой. Таким образом, число 0xFF0000 соответствует синему цвету, 0xFFFF00 – белому, 0 – черному.
DWORD	Тридцатидвухбитное беззнаковое целое число (unsigned long). Может принимать значения от 0 до 4294967295. В РДС достаточно часто используется для хранения различных битовых флагов или размеров областей памяти.
HBITMAP	Дескриптор загруженного в память растрового рисунка. В РДС используется только в функции <code>rdsRegisterWindow</code> (стр. 166).
HDC	Контекст устройства (device context) Windows. Контексты устройств используются в графических функциях Windows API для указания объекта (например, окна), на котором производится построение изображения.
HINSTANCE	Дескриптор загруженного в память модуля, например, DLL. Фактически, это базовый адрес модуля, но в функциях Windows API он часто используется просто как идентификатор этого модуля.
HWND	Дескриптор окна Windows. Такие дескрипторы используются в функциях Windows API, выполняющих различные действия с окнами.
LOGFONT	Структура, содержащая описание шрифта Windows. Ее поля подробно описаны в руководстве по Windows API. В РДС она используется не очень широко.
LPSTR	Указатель на строку восьмибитных символов, завершающуюся нулевым байтом. Этот тип эквивалентен стандартному типу C “char*”.

<i>Тип</i>	<i>Объект</i>
LPVOID	Указатель на некоторый объект в памяти без уточнения типа этого объекта. Этот тип эквивалентен стандартному типу C “void*”.
POINT	Структура, описывающая точку с двумя целыми координатами (поля x и y).
RECT	Структура, описывающая прямоугольник через целые координаты его левой верхней (поля left и top) и правой нижней (поля right и bottom) точек.

В сервисных функциях и структурах РДС также используются следующие обозначения-синонимы для стандартных типов Windows и языка C:

<i>Описание в РДС</i>	<i>Стандартное описание</i>	<i>Назначение</i>
RDSCALL	CALLBACK (в Windows API), __stdcall (в Borland C++)	Тип вызова всех сервисных функций РДС и всех функций обратного вызова, используемых в моделях блоков и модулях автокомпиляции. Аргументы функции передаются в стеке справа налево, стек освобождается вызванной функцией.
RDS_SHORT	short int	Двухбайтовое целое число со знаком. В РДС предусмотрен такой тип переменных блока, больше этот тип нигде не используется.
RDS_PCOLORREF	COLORREF*, LPCOLORREF	Указатель на тридцатидвухбитное число, описывающее цвет (COLORREF).

А.2. События блока и связанные с ними описания

Описываются все события, на которые может реагировать модель блока, а также структуры данных, связанные с этими событиями.

А.2.1. Функция модели блока

Функция модели блока – это связанная с блоком функция, которую РДС вызывает в ответ на различные события, происходящие с этим блоком. Функции моделей должны размещаться в динамически подключаемых библиотеках (DLL), в параметрах блока указывается имя такой библиотеки и имя экспортированной из нее функции. Все функции моделей блоков имеют тип вызова `RDSCALL`, описанный в файле “`RdsDef.h`” – ему полностью соответствует тип `CALLBACK` в Windows API: аргументы функции передаются в стеке справа налево, стек освобождается вызванной функцией. Функции моделей вызываются как в главном потоке РДС, обслуживающем интерфейс пользователя, так и в потоке расчета, выполняющем такты моделирования: вызвавший поток зависит от конкретного события, для реакции на которое вызывается функция. Функция модели имеет следующий вид:

```
int RDSCALL имя_функции_модели(  
    int CallMode,           // Режим вызова (событие)  
    RDS_PBLOCKDATA BlockData, // Данные блока  
    LPVOID ExtParam        // Дополнительные параметры  
);
```

Параметры функции:

`CallMode`

Событие, для реакции на которое вызывается функция модели блока. Это одна из целых констант `RDS_BFM_*`, описанных ниже.

`BlockData`

Указатель на структуру данных блока `RDS_BLOCKDATA` (см. А.2.3). В этой структуре хранятся общие данные блока, модель которого вызывается.

`ExtParam`

Указатель на дополнительные параметры события. Формат данных, на которые ссылается это указатель общего вида, зависит от конкретного события, то есть от значения параметра `CallMode`.

Возвращаемые значения:

Функция возвращает целое число, указывающее РДС на результат реакции на событие. В зависимости от события эти числа интерпретируются по-разному, возможные возвращаемые значения указаны в описании конкретных событий. В большинстве случаев возврат константы `RDS_BFR_DONE`, равной нулю, информирует РДС о том, что реакция на событие выполнена успешно.

Чаще всего функция модели содержит внутри оператор `switch`, в котором, в зависимости от произошедшего события (то есть от значения параметра `CallMode`), выполняются различные действия:

```
extern "C" __declspec(dllexport)  
int RDSCALL имя_модели(  
    int CallMode,  
    RDS_PBLOCKDATA BlockData,  
    LPVOID ExtParam)  
{
```

```

switch(CallMode)
{
    case <событие_1>:
        <действия>
        return <результат>;
    case <событие_2>:
        <действия>
        break;
    ...
}
return RDS_BFR_DONE;
}

```

Если блок имеет личную область данных, то есть область памяти, в которой хранятся его внутренние параметры, не обрабатываемые РДС, эта область обычно отводится в реакции на событие инициализации RDS_BFM_INIT (стр. 38) и освобождается в реакции на событие RDS_BFM_CLEANUP (стр. 33).

А.2.2. Главная функция DLL

Динамически подключаемая библиотека (DLL) с функциями моделей блоков РДС, как и любая другая DLL в Windows, должна иметь главную функцию (точку входа), которая вызывается при загрузке библиотеки в память и ее выгрузке. Чаще всего эта функция имеет имя `DllEntryPoint` (имя главной функции DLL обычно указывается в описании используемого для создания библиотек компилятора). В главной функции обычно выполняются действия, общие для всех моделей блоков в данной библиотеке:

- получение доступа к сервисным функциям РДС;
- регистрация функций блоков;
- проверка соответствия версии РДС версии библиотеки, если это необходимо.

Эти действия не обязательно выполнять именно в главной функции DLL – она просто предоставляет для них удобное место вызова. Написание главной функции DLL подробно рассмотрено в §2.2.

Главная функция DLL имеет следующий вид:

```

int WINAPI DllEntryPoint(
    HINSTANCE hinst,           // Дескриптор модуля DLL
    unsigned long reason,      // Причина вызова главной функции
    void *lpReserved           // Зарезервировано
);

```

Параметры функции:

`hinst`

Дескриптор модуля загруженной DLL. Его можно использовать для доступа к различным ресурсам (изображениям, строкам и т.п.), хранящимся в файле DLL вместе с функциями, а также для определения пути к файлу DLL функцией Windows API `GetModuleFileName`.

`reason`

Причина вызова главной функции DLL (одна из четырех констант Windows API):

DLL_PROCESS_ATTACH	Процесс (в данном случае – <code>rds.exe</code>) загрузил данную DLL в память. Обычно в этот момент выполняются все действия по инициализации глобальных переменных, с которыми работают функции этой DLL.
DLL_THREAD_ATTACH	Процесс создал новый поток (в DLL с моделями блоков обычно не используется).

DLL_THREAD_DETACH	Поток завершился (в DLL с моделями блоков обычно не используется).
DLL_PROCESS_DETACH	Процесс выгружает данную DLL из памяти. В этот момент выполняются действия по очистке глобальных данных, если это необходимо.

lpReserved

Указывает на способ загрузки или выгрузки библиотеки. При создании DLL с моделями блоков или модулями автоматической компиляции для РДС этот параметр не используется.

Возвращаемые значения:

При вызове главной функции при загрузке DLL в память процесса (при этом параметр reason равен DLL_PROCESS_ATTACH) возврат ненулевого значения сигнализирует об успешности инициализации, возврат нулевого – об ошибках. В остальных случаях возвращаемое значение не используется.

Пример:

Главная функция, обеспечивающая моделям блоков доступ ко всем сервисным функциям РДС с помощью макросов из файла “RdsFunc.h” (см. стр. 141).

```
#include <windows.h>
#include <RdsDef.h>

// Подготовка описаний сервисных функций
#define RDS_SERV_FUNC_BODY GetInterfaceFunctions
#include <RdsFunc.h>

// Главная функция DLL
int WINAPI DllEntryPoint(HINSTANCE /*hinst*/,
                        unsigned long reason,
                        void* /*lpReserved*/)
{ if(reason==DLL_PROCESS_ATTACH) // Загрузка DLL
  { // Получение доступа к функциям
    if(!GetInterfaceFunctions())
      MessageBox(NULL, "Нет доступа к функциям", "Ошибка", MB_OK);
  }
  return 1;
}
```

A.2.3. RDS_BLOCKDATA – структура данных блока

Структура данных блока RDS_BLOCKDATA содержит основные параметры блока, которые обычно используются функцией его модели (A.2.1) для выполнения реакции на событие. Эта структура создается РДС одновременно с блоком и хранится в памяти в течение всего времени существования этого блока.

```
typedef struct {
    LPVOID VarData;      // Адрес дерева переменных блока
    LPVOID BlockData;    // Адрес личной области данных блока
    RDS_BHANDLE Block;   // Идентификатор блока
    LPSTR BlockName;     // Имя блока
    RDS_BHANDLE Parent;  // Идентификатор подсистемы
    DWORD Flags;         // Флаги
    int Width, Height;   // Размеры прямоугольника блока
}
```

```

    int Tag;                // Пользовательское поле
} RDS_BLOCKDATA;
typedef RDS_BLOCKDATA *RDS_PBLOCKDATA;

```

Поля структуры:

VarData

Указатель на дерево переменных блока. В дереве переменных хранятся все значения статических переменных блока, функция модели может считывать их оттуда и записывать туда новые значения. Для переменных сложной структуры (строк, массивов, структур, переменных произвольного типа) в дереве переменных хранятся указатели на области памяти, занимаемые этими переменными, в которых, в свою очередь, могут содержаться указатели на другие области и т.д., поэтому данные переменных блока в общем случае имеют древовидную структуру. Для изменения размера массивов и длины строк, изменения фактического типа переменных произвольного типа и операций со сложными переменными предусмотрены специальные сервисные функции РДС, модель блока не должна самостоятельно пытаться отводить память в дереве переменных или изменять значение самого поля VarData. Работа со статическими переменными блока подробно рассмотрена в §2.5.

BlockData

Указатель на личную область данных блока. Перед самым первым для данного блока вызовом модели РДС записывает в это поле значение NULL и больше к нему не обращается. Если модели блока требуется отводить память под свои нужды, она может записать указатель на отведенную область памяти в это поле. В этом случае освободить отведенную область тоже должна будет функция модели – обычно это делается в реакции на событие RDS_BFM_CLEANUP (стр. 33), вызываемой непосредственно перед отключением модели от блока. Фактически, поскольку РДС никак не вмешивается в работу функции модели с этим полем, разработчик может использовать его для хранения любого нужного ему указателя.

Block

Уникальный идентификатор данного блока. Для идентификаторов блоков в РДС введен специальный тип RDS_BHANDLE (см. стр. 23), такие идентификаторы используются в сервисных функциях для указания блока, с которым нужно произвести то или иное действие. Из этого поля функция модели может считать идентификатор “своего” блока, менять значение поля она не должна.

BlockName

Указатель на строку во внутренней памяти РДС, в которой хранится имя данного блока. Функция модели не должна как-либо изменять эту строку, для переименования блоков существует специальная сервисная функция rdsRenameBlock (стр. 241).

Parent

Идентификатор родительской подсистемы блока, то есть подсистемы, внутри которой находится данный блок. Функция модели не должна изменять значение этого поля.

Flags

Битовые флаги, определяющие состояние и поведение блока – некоторые из них модель может и читать, и писать, некоторые – только читать (их изменение игнорируется РДС):

RDS_VARCHECKFAILED	Блок имеет не совместимую с моделью структуру статических переменных. Этот флаг взводится РДС по результатам вызова модели блока для проверки типов переменных (событие RDS_BFM_VARCHECK, стр. 48). Модель блока может только читать этот флаг, его
--------------------	---

	<p>установка игнорируется. На самом деле, модель сможет обнаружить этот флаг взведенным только в реакции на событие отключения модели <code>RDS_BFM_CLEANUP</code>, поскольку при несовместимом типе переменных все остальные вызовы модели блокируются.</p>
<code>RDS_NEEDSDLLREDRAW</code>	<p>Действия, выполненные функцией модели, привели к изменению внешнего вида блока, и он должен быть перерисован при следующем обновлении окна подсистемы. Этот флаг используется только в блоках, изображения которых рисуются функцией модели программно в реакции на событие <code>RDS_BFM_DRAW</code> (стр. 57), для всех остальных блоков он игнорируется. Перед любым вызовом функции модели РДС автоматически взводит этот флаг, поэтому модели не обязательно работать с ним – если она ничего не предпримет, блок будет перерисован. Если внешний вид блока в результате вызова функции модели не изменился, модель может сбросить этот флаг чтобы избежать затрат времени на лишнюю перерисовку.</p>
<code>RDS_MOUSECAPTURE</code>	<p>Блок “захватил” мышь: если модель взведет этот флаг, информация обо всех манипуляциях мышью в окне родительской подсистемы блока будет поступать только в этот блок, независимо от того, над каким блоком находится курсор. Сброс флага восстановит нормальный порядок работы. Установка и сброс этого флага возможны только в реакциях на перемещение мыши или нажатие и отпускание ее кнопок, во всех остальных реакциях РДС игнорирует изменения этого флага и возвращает его в исходное состояние.</p>
<code>RDS_NOWINREFRESH</code>	<p>При взведении этого флага перерисовка немодальных окон данного блока (если они есть) или окна подсистемы (если данный блок – подсистема) будет временно запрещена. Временное запрещение обновления окон используется в тех случаях, когда выполняется какая-либо длительная операция, занимающая несколько тактов расчета, и в середине этой операции, когда готовы еще не все данные, перерисовывать окна нежелательно. Для установки и сброса этого флага обычно применяется сервисная функция <code>rdsEnableWindowRefresh</code> (стр. 255), но, при желании, разработчик модели может управлять им непосредственно.</p>
<code>RDS_WINREFRESHWAITING</code>	<p>Установка этого флага сигнализирует о том, что немодальные окна блока или окно подсистемы необходимо перерисовать, как только обновление окон будет разрешено. Если обновление окон запрещено (взведен флаг <code>RDS_NOWINREFRESH</code>) и поступает команда обновления (по таймеру или от сервисной функции <code>rdsRefreshBlockWindows</code>, стр. 263), этот флаг взводится автоматически. Как только обновление</p>

	<p>окон снова будет разрешено сервисной функцией <code>rdsEnableWindowRefresh</code> (стр. 255), РДС проверит флаг <code>RDS_WINREFRESHWAITING</code> и, если он установлен, даст повторную команду на обновление окон. При необходимости, разработчик модели может взводить и сбрасывать этот флаг вручную.</p>
<p><code>RDS_DISABLED</code></p>	<p>Блок не реагирует на действия пользователя: пользователь не сможет удалить этот блок, изменить его параметры и выделить его в режиме редактирования. РДС не взводит этот флаг, модель блока может управлять им самостоятельно. По умолчанию флаг сброшен.</p>
<p><code>RDS_CTRLCALC</code></p>	<p>Если этот флаг взведен, перед началом каждого такта расчета модель этого блока вызывается для реакции на событие <code>RDS_BFM_PREMODEL</code> (стр. 41) в том случае, если в этом такте блок должен быть запущен в режиме <code>RDS_BFM_MODEL</code> (стр. 40), то есть если он запускается каждый такт или его первая статическая переменная (“Start”) не равна нулю. По умолчанию флаг сброшен, модель может взвести его, если ей нужно вызываться непосредственно перед каждым тактом расчета. Взвести флаг нужно до перехода в режим моделирования или расчета, иначе его фактическая установка будет отложена до перехода в режим редактирования (это связано с внутренней логикой работы РДС).</p>
<p><code>Width, Height</code></p>	<p>Ширина (<code>Width</code>) и высота (<code>Height</code>) описывающего прямоугольника блока в точках экрана в масштабе 100% без учета возможной связи с переменными, то есть размер блока, заданный пользователем в режиме редактирования. Эти поля можно использовать только в том случае, если модель блока рисует его внешний вид программно. Если изображение блока задано прямоугольником с текстом или векторной картинкой, эти поля не используются.</p>
<p><code>Tag</code></p>	<p>Целое значение, никак не обрабатываемое РДС. Разработчик модели может использовать его по своему усмотрению – например, для хранения каких-либо флагов. РДС не инициализирует это поле при подключении к блоку новой модели, поэтому функция модели должна заниматься этим самостоятельно.</p>

А.2.4. События общего назначения

К событиям общего назначения относятся переключения режимов РДС, обмен данными между блоками, вызовы блоков по таймеру и для выполнения тактов моделирования и т.п.

А.2.4.1. `RDS_BFM_CALCMODE` – переход из режима редактирования в режим моделирования

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (`int CallMode`):

Константа `RDS_BFM_CALCMODE`.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_CALCMODE вызывается при переходе из режима редактирования в режим моделирования или режим расчета (технически в РДС переход из режима редактирования в режим расчета производится не непосредственно, а через режим моделирования, хотя пользователь этого и не видит). Режимы работы РДС подробно описаны в §1.3. Вне режима редактирования пользователь не может изменять схему или вызывать функцию настройки блока, поэтому эту реакцию можно использовать для подготовки каких-либо вспомогательных данных, на которые влияет структура схемы или параметры блока, которые не изменятся до возврата в режим редактирования. Например, модель блока может составить список других блоков, соединенных с ним связями, для того, чтобы вызывать у них какие-либо функции, или выполнить подписку на динамическую переменную, имя которой задается пользователем в параметрах блока (см. §2.6.3).

Следует помнить, что если в РДС запрещен режим редактирования (например, при управлении из другого приложения, см. §3.1), сразу после загрузки схемы включится режим моделирования, а не режим редактирования, то есть будет немедленно вызвана реакция на событие RDS_BFM_CALCMODE.

См. также:

RDS_BFM_EDITMODE (стр. 35), RDS_BFM_STARTCALC (стр. 46),
RDS_BFM_STOPCALC (стр. 46), rdsSystemInEditMode (стр. 175),
rdsCalcProcessIsRunning (стр. 148), rdsExecuteCommand (стр. 151).

A.2.4.2. RDS_BFM_CHECKFUNCSUPPORT – проверка поддержки блоком функции с заданным идентификатором**Поток, в котором вызывается функция модели:**

Главный поток РДС или поток расчета (тот же поток, в котором вызвана сервисная функция, запросившая проверку поддержки функции).

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_CHECKFUNCSUPPORT.

Третий параметр функции модели (void *ExtParam):

Указатель на целый идентификатор функции (int*).

Возвращаемое функцией модели значение:

RDS_BFM_DONE или 0	Функция не поддерживается блоком, вызывать ее нельзя.
Любое ненулевое значение	Функция поддерживается блоком.

Это устаревшее событие в настоящее время практически не используется. При вызове одним блоком функции другого блока (см. §2.13) вызвавший блок может передать в сервисную функцию флаг RDS_BCALL_CHECKSUPPORT, потребовав тем самым от РДС выполнения проверки поддержки данной функции вызванным блоком. Функция модели должна вернуть ненулевое значение, если функция блока с данным идентификатором поддерживается. Если будет возвращено нулевое значение, вызов функции блока не будет произведен.

Крайне желательно писать реакции на вызов функции блока (событие RDS_BFM_FUNCTIONCALL, см. стр. 35) таким образом, чтобы при вызове функции, которую модель блока не поддерживает, не выполнялось никаких действий. В этом случае необходимость в предварительной проверке поддержки функции отпадает.

Пример:

Информирование РДС о поддержке вызова функции с именем “MyFunctionName” в модели блока.

```
// Глобальная переменная для идентификатора функции
int MyFuncId;

...

// Регистрация функции блока (например, в главной функции DLL)
MyFuncId=rdsRegisterFunction("MyFunctionName");

...

// Реакция на событие в функции модели блока
switch(CallMode)
{ case RDS_BFM_CHECKFUNCSUPPORT:
    if (*((int*)ExtParam)==MyFuncId)
        return 1; // Функция поддерживается
    return 0; // Остальные не поддерживаются
...
}
```

См. также:

RDS_BFM_FUNCTIONCALL (стр. 35),
rdsCheckBlockFunctionSupport (стр. 313),
rdsCallBlockFunction (стр. 310), rdsQueueCallBlockFunction (стр. 313),
rdsBroadcastFunctionCallsEx (стр. 308).

A.2.4.3. RDS_BFM_CLEANUP – очистка данных блока

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (**int** CallMode):

Константа RDS_BFM_CLEANUP.

Третий параметр функции модели (**void** *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие RDS_BFM_CLEANUP – последнее событие перед отключением модели от блока из-за его удаления, выгрузки схемы из памяти, подключения к блоку новой модели и т.п. Если в реакции на событие RDS_BFM_INIT модель отвела память под личную область данных блока, в реакции на RDS_BFM_CLEANUP она должна освободить ее. События инициализации и очистки подробно рассмотрены в §2.4.

Пример:

Отведение памяти под личную область данных блока оператором C++ new и ее освобождение оператором delete.

```

// Структура личной области данных блока
typedef struct
{ int IntData;
  double DoubleData;
} TMyBlockData;

// Модель блока
extern "C" __declspec(dllexport) int RDSCALL Model1(
    int CallMode,           // Событие
    RDS_PBLOCKDATA BlockData, // Данные блока
    LPVOID ExtParam)        // Дополнительные параметры
{ TMyBlockData *data;
  switch(CallMode)
  { case RDS_BFM_INIT:      // Отведение памяти
    data=new TMyBlockData;
    BlockData->BlockData=data;
    data->IntData=0;
    data->DoubleData=1.0;
    break;
    case RDS_BFM_CLEANUP:   // Освобождение памяти
    data=(TMyBlockData*) (BlockData->BlockData);
    delete data;
    break;
  }
  return RDS_BFR_DONE;
}

```

См. также:

RDS_BFM_INIT (стр. 38).

A.2.4.4. RDS_BFM_DYNVARCHANGE – изменение динамической переменной

Поток, в котором вызывается функция модели:

Главный поток РДС или поток расчета – тот же поток, в котором выполнено создание или удаление переменной или вызвана функция `rdsNotifyDynVarSubscribers` (стр. 352).

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_DYNVARCHANGE.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру подписки данного блока на изменившуюся переменную (RDS_DYNVARLINK*, стр. 121).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие RDS_BFM_DYNVARCHANGE возникает при создании и удалении динамической переменной, на которую подписался блок, а также при вызове моделью какого-либо другого блока сервисной функции `rdsNotifyDynVarSubscribers` (стр. 352) для этой переменной. Следует помнить, что это событие не возникает автоматически при записи в динамическую переменную нового значения – записавший значение блок должен явно вызвать функцию `rdsNotifyDynVarSubscribers`, чтобы уведомить всех подписчиков на переменную о произведенном изменении. Работа с динамическими переменными и реакция на это событие подробно рассмотрены в §2.6.

См. также:

`rdsNotifyDynVarSubscribers` (стр. 352), `RDS_DYNVARLINK` (стр. 121).

A.2.4.5. `RDS_BFM_EDITMODE` – переход в режим редактирования

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (`int CallMode`):

Константа `RDS_BFM_EDITMODE`.

Третий параметр функции модели (`void *ExtParam`):

Не используется (`NULL`).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие `RDS_BFM_EDITMODE` вызывается при переходе в режим редактирования из режимов моделирования или расчета (технически в РДС переход в режима редактирования из режима расчета производится не непосредственно, а через режим моделирования, хотя пользователь этого и не видит). Режимы работы РДС подробно описаны в §1.3. Если в РДС разрешен режим редактирования (он может быть запрещен при управлении из другого приложения, см. §3.1), это событие, кроме того, возникает сразу после загрузки схемы, поскольку после загрузки РДС находится именно в режиме редактирования.

См. также:

`RDS_BFM_CALCMODE` (стр. 31), `RDS_BFM_STARTCALC` (стр. 46),
`RDS_BFM_STOPCALC` (стр. 46), `rdsSystemInEditMode` (стр. 175),
`rdsCalcProcessIsRunning` (стр. 148), `rdsExecuteCommand` (стр. 151).

A.2.4.6. `RDS_BFM_FUNCTIONCALL` – вызов функции блока

Поток, в котором вызывается функция модели:

Главный поток РДС или поток расчета (вызов выполняется в одном потоке с моделью вызывающего функцию блока).

Первый параметр функции модели (`int CallMode`):

Константа `RDS_BFM_FUNCTIONCALL`.

Третий параметр функции модели (`void *ExtParam`):

Указатель на структуру `RDS_FUNCTIONCALldata`, в которой содержатся идентификатор вызванной функции, ее параметры, а также другая информация, связанная с вызовом

Возвращаемое функцией модели значение:

Возвращенное значение передается вызвавшему функцию блоку, если функция вызвана непосредственно. При отложенном вызове возвращенное значение игнорируется.

Вызов функций позволяет блокам передавать друг другу информацию непосредственно, без участия связей или динамических переменных (этот механизм подробно рассмотрен в §2.13). Функция блока может вызываться как немедленно

(выполнение функции модели вызвавшего блока приостанавливается до завершения вызванной функции), так и отложенным вызовом (выполнение функции модели вызвавшего блока продолжается, а после ее завершения вызывается модель другого блока). При вызове функции какого-либо блока его модель реагирует на событие RDS_BFM_FUNCTIONCALL, при этом в третьем параметре функции модели передается указатель на структуру RDS_FUNCTIONCALldata:

```
typedef struct {
    int Function;           // Идентификатор функции
    LPVOID Data;           // Параметры функции
    int Reserved;          // Зарезервировано
    RDS_BHANDLE Caller;    // Вызвавший блок
    BOOL Broadcast;        // Вызов не одного блока, а нескольких
    int BroadcastCnt;      // Номер блока среди всех вызванных
    BOOL Stop;             // Прекратить вызов блоков
    BOOL Delayed;          // Отложенный вызов
    DWORD DataBufSize;     // Размер буфера при отложенном вызове
} RDS_FUNCTIONCALldata;
typedef RDS_FUNCTIONCALldata *RDS_PFUNCTIONCALldata;
```

Поля структуры:

Function

Уникальный целый идентификатор вызванной функции. Прежде чем функцию блока можно будет вызывать или реагировать на ее вызов, имя этой функции необходимо зарегистрировать вызовом `rdsRegisterFunction` (стр. 316), который даст функции уникальный целый идентификатор. Именно он используется во всех сервисных функциях РДС, и с ним модель блока должна сравнить поле `Function`, чтобы определить, какая именно функция вызвана.

Data

Указатель на область памяти, в которой находятся параметры вызванной функции. Обычно эту область оформляют в виде структуры или массива. При немедленном (прямом) вызове функции вызванная модель может не только получать какие-либо параметры через эту область памяти, но и возвращать через нее значения, поскольку функция модели вызвавшего блока получит управление обратно только после завершения модели вызванного, и сможет считать данные из этой области памяти. При отложенном вызове (см. §2.13.5) функция вызвавшего блока завершится до самого вызова, поэтому в поле `Data` в этом случае передается указатель на созданную РДС копию области памяти, хранящей параметры функции – модель вызвавшего блока уже не сможет ничего оттуда считать.

Reserved

Значение поля не используется в современных версиях РДС.

Caller

Идентификатор (RDS_BHANDLE) вызвавшего функцию блока. Модель вызванного блока может использовать его для того чтобы, в свою очередь, вызвать у него какую-либо функцию в ответ.

Broadcast

Значение TRUE в этом поле сигнализирует о том, что данная функция была вызвана сразу у нескольких блоков (см. §2.13.3), значение FALSE – что она вызвана только у данного блока.

BroadcastCnt

При Broadcast==TRUE в этом поле будет находиться порядковый (начинающийся с нуля) номер данного блока среди всех вызванных. Например, если функция вызвана у всех блоков какой-либо подсистемы, и в этой подсистеме находится три блока, при вызове первого из них в BroadcastCnt будет передан ноль, при вызове второго – 1, при вызове третьего – 2.

Stop

Исходно в этом поле записано значение FALSE. При Broadcast==TRUE модель вызванного блока может записать в Stop значение TRUE, запретив тем самым вызов функции для оставшихся блоков. Таким образом можно организовать, например, поиск в подсистеме блока, выполняющего какие-либо действия: можно вызвать функцию у всех блоков подсистемы и написать реакцию на вызов этой функции так, чтобы первый же выполняющий ее блок произвел необходимые действия или сообщил вызвавшему блоку свой идентификатор, а затем прервал дальнейший перебор блоков и вызов их функций, присвоив Stop значение TRUE.

Delayed

Значение FALSE в этом поле указывает на прямой вызов функции, значение TRUE – на отложенный.

DataBufSize

При отложенном вызове функции в этом поле передается размер области данных, указатель на которую находится в поле Data. Этот размер всегда известен, поскольку размер области данных с параметрами функции передается в сервисные функции, выполняющие отложенный вызов, вместе с указателем на ее начало. РДС делает копию этой области, чтобы после завершения модели вызвавшего функцию блока можно было передать эти данные вызванному.

Пример:

Типичный пример непосредственного вызова функции какого-либо блока. Параметры функции передаются в структуре, в которой предусмотрено поле для проверки размера. Описание этой структуры должно быть доступно и вызывающей функции модели (для вызова), и вызываемой (для реакции).

```
// Структура параметров функции
typedef struct {
    DWORD servSize;    // Размер этой структуры
    int IParam;        // Целый параметр
    double DParam;     // Вещественный параметр
} TMyFuncParam;

...

// Глобальная переменная для идентификатора функции
int MyFuncId;

...

// Регистрация функции блока (например, в главной функции DLL)
MyFuncId=rdsRegisterFunction("MyFunctionName");
```

Вызов функции блока:

```
RDS_BHANDLE block=...;    // Идентификатор вызываемого блока
TMyFuncParam param;       // Структура параметров функции
int retvalue;             // Результат возврата функции

param.servSize=sizeof(param);
param.IParam=100;
```

```

param.DParam=3.14;
// Вызов функции
retval=rdsCallBlockFunction(block,MyFuncId,&param);

```

Реакция на вызов функции в модели блока:

```

TMyFuncParam *pparam;          // Структура параметров функции
RDS_PFFUNCTIONCALLDATA pfuncdata; // Данные вызова

switch(CallMode)
{
  case RDS_BFM_FUNCTIONCALL:
    pfuncdata=(RDS_PFFUNCTIONCALLDATA)ExtParam;
    if(pfuncdata->Function==MyFuncId)
    {
      pparam=(TMyFuncParam*)(pfuncdata->Data);
      if(pparam->servSize==sizeof(TMyFuncParam))
      {
        // Выполнение функции
        ...
      }
    }
    break;
}

```

См. также:

RDS_BFM_CHECKFUNCSUPPORT (стр. 32),
 rdsCheckBlockFunctionSupport (стр. 313),
 rdsCallBlockFunction (стр. 310), rdsQueueCallBlockFunction (стр. 313),
 rdsBroadcastFunctionCallsEx (стр. 308).

A.2.4.7. RDS_BFM_INIT – инициализация блока

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (**int CallMode**):

Константа RDS_BFM_INIT.

Третий параметр функции модели (**void *ExtParam**):

Не используется (NULL).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие RDS_BFM_INIT – самое первое событие в “жизни” блока РДС. Реакция на него вызывается сразу после подключения к блоку какой-либо новой модели (вручную пользователем, при добавлении нового блока в схему, при загрузке схемы из файла и т.п.). В этой реакции модели блоков обычно создают какие-либо структуры данных, необходимые для работы блока. События инициализации и очистки подробно рассмотрены в §2.4.

На момент вызова реакции RDS_BFM_INIT структура статических переменных блока еще не проверена, поэтому обращаться к ним не следует. Все действия, зависящие от структуры статических переменных, следует выполнять в реакции на событие проверки структуры переменных RDS_BFM_VARCHECK.

Пример:

Пример реакции на событие RDS_BFM_INIT приведен в описании события RDS_BFM_CLEANUP (стр. 33).

См. также:

RDS_BFM_CLEANUP (стр. 33), RDS_BFM_VARCHHECK (стр. 48).

A.2.4.8. RDS_BFM_LOADSTATE – загрузка состояния блока

Поток, в котором вызывается функция модели:

Главный поток РДС или поток расчета (тот же поток, в котором вызвана сервисная функция загрузки состояния блока `rdsLoadSystemState`, см. стр. 251).

Первый параметр функции модели (`int CallMode`):

Константа `RDS_BFM_LOADSTATE`.

Третий параметр функции модели (`void *ExtParam`):

Не используется (`NULL`).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие `RDS_BFM_LOADSTATE` вызывается в процессе работы сервисной функции `rdsLoadSystemState` для восстановления состояния каждого блока, ранее сохраненного функцией `rdsSaveSystemState` (стр. 252). В этой реакции модель должна загрузить из памяти все данные, сохраненные в реакции на событие `RDS_BFM_SAVESTATE` (стр. 45).

Сохранение и загрузка состояния одного или нескольких блоков используется в РДС для управления расчетом: вызовом сервисной функции `rdsSaveSystemState` можно сохранить в памяти текущее состояние одного или нескольких блоков, а затем, вызвав `rdsLoadSystemState`, вернуть эти блоки в запомненное состояние. Таких сохраненных состояний может быть сколько угодно – при сохранении каждое получает уникальный целый идентификатор, который используется в функции `rdsLoadSystemState` для указания конкретного восстанавливаемого состояния. При сохранении состояния значения статических переменных блока записываются в память автоматически, а все остальные данные должны быть сохранены моделью при помощи вызова `rdsWriteBlockData` (стр. 280) в реакции на событие `RDS_BFM_SAVESTATE`. Загрузка сохраненных моделью данных в реакции на событие `RDS_BFM_LOADSTATE` осуществляется вызовом `rdsReadBlockData` (стр. 277). Сохранение и загрузка состояний блоков и подсистем подробно описана в §2.14.3.

Пример:

Пример сохранения и загрузки переменных, хранящихся в личной области данных блока.

```
// Структура личной области данных блока
typedef struct
{ // Переменные состояния
    int IntState;
    double DoubleState;
} TMyBlockData2;

// Модель блока
extern "C" __declspec(dllexport) int RDSCALL Model2(
    int CallMode, // Событие
    RDS_PBLOCKDATA BlockData, // Данные блока
```

```

LPVOID ExtParam) // Дополнительные параметры
{ TMyBlockData2 *data=(TMyBlockData*)(BlockData->BlockData);
  switch(CallMode)
  {
    ...
    case RDS_BFM_SAVESTATE: // Сохранение состояния
      rdsWriteBlockData(&IntState,sizeof(IntState));
      rdsWriteBlockData(&DoubleState,sizeof(DoubleState));
      break;
    case RDS_BFM_LOADSTATE: // Загрузка состояния
      rdsReadBlockData(&IntState,sizeof(IntState));
      rdsReadBlockData(&DoubleState,sizeof(DoubleState));
      break;
    ...
  }
  return RDS_BFR_DONE;
}

```

См. также:

RDS_BFM_SAVESTATE (стр. 45), rdsLoadSystemState (стр. 251),
 rdsDeleteSystemState (стр. 250), rdsSaveSystemState (стр. 252),
 rdsReadBlockData (стр. 277), rdsWriteBlockData (стр. 280).

A.2.4.9. RDS_BFM_MODEL – выполнение такта расчета

Поток, в котором вызывается функция модели:

Поток расчета.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_MODEL.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

В режиме расчета реакция на событие RDS_BFM_MODEL циклически вызывается у простых блоков в каждом такте. Именно в ней обычно выполняется основной объем вычислений блока, производимых им в режиме расчета. Эта реакция вызывается для блока только в том случае, если запуск модели разрешен, то есть если одновременно выполняются следующие условия:

1. Данный блок – простой. У подсистем, внешних входов/выходов и вводов шин эта реакция не вызывается, поскольку они не участвуют в расчете.
2. В параметрах блока установлен запуск каждый такт *или* его первая статическая переменная (сигнальный вход запуска модели, чаще всего называющийся “Start”) имеет ненулевое значение.
3. Данный блок не находится за пределами подсистемы, для которой в данный момент включен режим отдельного расчета (см. §2.14.4) функцией rdsSetExclusiveCalc (стр. 169).

РДС в режиме расчета производит (как правило, в отдельном потоке) следующие действия (см. §1.3):

- Для всех простых блоков, модели которых взвели флаг `RDS_CTRLCALC` (стр. 31), вызывается реакция на событие `RDS_BFM_PREMODEL` (стр. 41), если запуск модели данного блока разрешен (см. выше).
- Для всех простых блоков, запуск моделей которых все еще разрешен (предыдущий вызов `RDS_BFM_PREMODEL` мог запретить вызов модели блока), вызывается реакция на событие `RDS_BFM_MODEL`. Перед вызовом реакции сигналу запуска модели (первой статической переменной блока) присваивается нулевое значение – по умолчанию в следующем такте расчета модель блока не запустится, если только она не взведет свой сигнал запуска самостоятельно внутри реакции `RDS_BFM_MODEL`, или на него не поступит единица по связи от другого блока, или в параметрах блока не указан запуск каждый такт (в этом случае значение сигнала запуска игнорируется). Кроме того, перед вызовом реакции `RDS_BFM_MODEL` второй статической переменной блока (сигналу готовности блока “Ready”) присваивается единица – по завершении реакции блок будет считаться сработавшим и его выходы передадутся по связям, если только модель блока принудительно не обнулит сигнал готовности внутри реакции. После передачи данных выходов блока по связям сигнал готовности обнулится автоматически.
- Начинается следующий такт расчета с новым вызовом реакций `RDS_BFM_PREMODEL` и `RDS_BFM_MODEL`.

В большинстве случаев вычисление выходов блока производится именно в реакции на событие `RDS_BFM_MODEL`, поскольку этот вызов постоянно производится в режиме расчета, чередуясь с передачей данных по связям. Самые простые модели (например, модели алгебраических блоков) состоят всего из двух реакций: `RDS_BFM_MODEL` и `RDS_BFM_VARCHECK` (стр. 48). Для управления запуском модели в такте расчета используется сигнал запуска (первый сигнальный вход, “Start”) и флаги запуска входов блока (см. §1.5). Внутри реакции модель блока может принудительно перезапустить себя в следующем такте (взведя свой собственный флаг запуска), а также запретить передачу по связям всех своих выходов (сбросив свой сигнал готовности).

Следует помнить, что реакция на событие `RDS_BFM_MODEL` обычно вызывается в цикле в отдельном потоке расчета, поэтому в ней нельзя открывать модальные окна, поскольку это приведет к остановке всего потока расчета до закрытия окна (см. §1.8).

Многочисленные примеры реакций на событие `RDS_BFM_MODEL` приведены в §2.5 и следующих за ним.

См. также:

`RDS_BFM_PREMODEL` (стр. 41), `RDS_BFM_VARCHECK` (стр. 48),
`rdsSetExclusiveCalc` (стр. 169).

A.2.4.10. `RDS_BFM_PREMODEL` – вызов модели перед тактом расчета

Поток, в котором вызывается функция модели:

Поток расчета.

Первый параметр функции модели (`int CallMode`):

Константа `RDS_BFM_PREMODEL`.

Третий параметр функции модели (`void *ExtParam`):

Не используется (`NULL`).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

В режиме расчета реакция на событие RDS_BFM_PREMODEL вызывается в начале каждого такта расчета у простых блоков, модели которых взвели флаг RDS_CTRLCALC (стр. 31). Эта реакция, как и реакция на событие RDS_BFM_MODEL (стр. 40), вызывается для блока только в том случае, если запуск модели разрешен, то есть если одновременно выполняются следующие условия:

1. Данный блок – простой. У подсистем, внешних входов/выходов и вводов шин эта реакция не вызывается, поскольку они не участвуют в расчете.
2. В параметрах блока установлен запуск каждый такт *или* его первая статическая переменная (сигнальный вход запуска модели, чаще всего называющийся “Start”) имеет ненулевое значение.
3. Данный блок не находится за пределами подсистемы, для которой в данный момент включен режим отдельного расчета (см. §2.14.4) функцией rdsSetExclusiveCalc (стр. 169).

Чаще всего эта реакция используется для управления расчетом, когда действия моделей в такте необходимо разделить на два этапа: сначала все блоки вызываются для реакции на RDS_BFM_PREMODEL, затем – для реакции на RDS_BFM_MODEL.

Пример использования этого события приведен в §2.14.5.

См. также:

RDS_BFM_MODEL (стр. 40), RDS_CTRLCALC (стр. 31),
RDS_BFM_VARCHECK (стр. 48), rdsSetExclusiveCalc (стр. 169).

A.2.4.11. RDS_BFM_REMOTEMSG – вызов от внешнего приложения

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_REMOTEMSG.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру RDS_REMOTEMSGDATA, в которой содержится переданная внешним приложением информация.

Возвращаемое функцией модели значение:

Передается вызвавшему приложению.

Событие RDS_BFM_REMOTEMSG возникает при управлении РДС из внешнего приложения (см. главу 3) в момент вызова этим приложением данного блока схемы с помощью предназначенных для этого функций библиотеки RdsCtrl.dll. Вызывая блок, приложение передает ему целое число и строку, которые модель блока может считать из полей структуры RDS_REMOTEMSGDATA, указатель на которую передается в третьем параметре функции модели.

```
typedef struct {  
    LPSTR String;           // Переданная приложением строка  
    int Value;              // Переданное приложением число  
    LPSTR ControllerName;   // Имя приложения  
} RDS_REMOTEMSGDATA;  
typedef RDS_REMOTEMSGDATA *RDS_PREMOTEMSGDATA;
```

Поля структуры:

String

Указатель на строку, переданную вызвавшим приложением (сама строка находится во внутренней памяти РДС и будет удалена после завершения этой реакции).

Value

Целое число, переданное вызвавшим приложением.

ControllerName

Указатель на строку с названием управляющего приложения во внутренней памяти РДС, если это приложение сообщило РДС свое название вызовом функции `rdscrtlSetControllerName` (стр. 639). Эта строка обычно никак не связана с путем к исполняемому файлу приложения или другими его параметрами, это просто некий уникальный идентификатор, закладываемый одновременно в приложение и модель блока, и позволяющий вызванному блоку определить, какое именно приложение его в данный момент вызывает. В большинстве случаев модели блоков, реагирующие на событие `RDS_BFM_REMOTEMSG`, разрабатываются специально под конкретное управляющее приложение, поэтому анализ этой строки чаще всего не производится.

Получив от вызвавшего приложения целое число и строку, модель блока может также вернуть приложению целое число и строку. Для возврата строки вызывается сервисная функция `rdsRemoteReply` (стр. 398), а в качестве возвращаемого целого числа используется результат возврата самой функции модели блока. Прием возвращенной блоком строки, как правило, требует от управляющего приложения регистрации специальной функции обратного вызова, через которую библиотека `RdsCtrl.dll` будет возвращать полученные от блоков строки.

Примеры вызова блока внешним приложением и реакции на событие `RDS_BFM_REMOTEMSG` в модели подробно рассмотрены в §3.3.

Пример:

1) Служебная функция управляющего приложения, записывающая строку в буфер. Здесь используется фиксированный размер буфера, что, на самом деле, не очень хорошо для полноценной программы. Примеры подобных функций для произвольных длин строк приведены в §3.1.

```
// Функция обратного вызова, записывающая полученную
// из РДС строку в буфер
#define STRBUFSIZE 1000 // Размер буфера для возврата строк
void RDSCALL RetStrCallback(LPVOID ptr, LPSTR str)
{ // Определяем длину строки
  int len=strlen(str);
  // Приводим указатель ptr к типу "указатель на char"
  char *pStr=(char*)ptr;
  if(pStr==NULL) return; // Ошибка - нет указателя на буфер
  if(len>=STRBUFSIZE)
    len=STRBUFSIZE-1;
  strncpy(pStr,str,len); // Копируем строку в буфер
  // Дописываем 0 в конец строки
  pStr[len]=0;
}
```

2) Регистрация этой функции – выполняется один раз после загрузки `RdsCtrl.dll`:

```
rdscrtlSetStringCallback(RetStrCallback);
```

3) Передача блоку с полным именем “:Sys1:Block10” числа 99 и строки “Строка 1”, запись возвращенного блоком числа в переменную `retcode` и строки в буфер `buffer` (используется зарегистрированная ранее служебная функция). В примере `link` – идентификатор связи управляющего приложения с РДС.

```
char buffer[STRBUFSIZE];    // Буфер для возврата строки
retcode=rdsctrlCallBlockFunctionEx(
    link,                    // Идентификатор связи с РДС
    ":Sys1:Block10",        // Полное имя вызываемого блока
    99,                     // Передаваемое блоку число
    "Строка 1",             // Передаваемая блоку строка
    &buffer);               // Буфер для возвращаемой строки
```

4) Реакция на вызов от управляющего приложения в функции модели блока. Здесь полученная строка выводится в сообщении пользователю, а вызвавшему приложению передается строка “Возвращаемая строка” (она запишется в массив `buffer` приведенного выше фрагмента текста программы) и число, равное увеличенному на единицу принятому числу (в данном случае – 100, это число будет записано в переменную `retcode` приведенного выше фрагмента).

```
// Указатель на структуру с параметрами вызова
RDS_REMOTEMSGDATA *pmsgdata;
switch(CallMode)
{ case RDS_BFM_REMOTEMSG: // Вызов от внешнего приложения
    pmsgdata=(RDS_REMOTEMSGDATA*)ExtParam;
    rdsMessageBox(pmsgdata->String,"ДУ",MB_OK);
    // Возврат строки
    rdsRemoteReply("Возвращаемая строка");
    // Возврат целого числа
    return pmsgdata->Value+1;
    ...
}
```

См. также:

`rdsctrlCallBlockFunctionEx` (стр. 621),
`rdsctrlSetControllerName` (стр. 639), `rdsRemoteReply` (стр. 398),
`rdsctrlSetStringCallback` (стр. 604).

A.2.4.12. `RDS_BFM_RESETCALC` – сброс расчета

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (`int CallMode`):

Константа `RDS_BFM_RESETCALC`.

Третий параметр функции модели (`void *ExtParam`):

Не используется (`NULL`).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие `RDS_BFM_RESETCALC` возникает при сбросе расчета, то есть возврате всей схемы или ее отдельной подсистемы в исходное состояние. Всем статическим переменным блока при этом автоматически присваиваются значения по умолчанию, а все остальные меняющиеся со временем параметры блока должны, при необходимости, сбрасываться самой

функцией модели в реакции на это событие. Сброс расчета может производиться либо пользователем при помощи соответствующей кнопки или пункта меню РДС, либо одной из моделей блока программно при помощи сервисной функции `rdsResetSystemState` (стр. 252). Программный сброс расчета подробно рассматривается в §2.14.2.

См. также:

`rdsResetSystemState` (стр. 252).

A.2.4.13. RDS_BFM_SAVESTATE – запись состояния блока

Поток, в котором вызывается функция модели:

Главный поток РДС или поток расчета (тот же поток, в котором вызвана сервисная функция записи состояния).

Первый параметр функции модели (int CallMode):

Константа `RDS_BFM_SAVESTATE`.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие `RDS_BFM_SAVESTATE` возникает при вызове сервисной функции `rdsSaveSystemState` (стр. 252) для сохранения состояния одного или нескольких блоков. В этой реакции модель должна записать в память все текущие данные блока, которые позже могут быть восстановлены при вызове функции `rdsLoadSystemState` (стр. 251), то есть при реакции на парное к данному событие `RDS_BFM_LOADSTATE` (стр. 39).

Сохранение и загрузка состояния одного или нескольких блоков используется в РДС для управления расчетом: вызовом сервисной функции `rdsSaveSystemState` можно сохранить в памяти текущее состояние одного или нескольких блоков, а затем, вызвав `rdsLoadSystemState`, вернуть эти блоки в запомненное состояние. Таких сохраненных состояний может быть сколько угодно – при сохранении каждое получает уникальный целый идентификатор, который используется в функции `rdsLoadSystemState` для указания конкретного восстанавливаемого состояния. При сохранении состояния значения статических переменных блока записываются в память автоматически, а все остальные данные должны быть сохранены моделью при помощи вызова `rdsWriteBlockData` (стр. 280) в реакции на событие `RDS_BFM_SAVESTATE`. Загрузка сохраненных моделью данных осуществляется вызовом `rdsReadBlockData` (стр. 277) в реакции на событие `RDS_BFM_LOADSTATE`. Сохранение и загрузка состояний блоков и подсистем подробно описана в §2.14.3.

Пример:

Пример сохранения и загрузки переменных, хранящихся в личной области данных блока, приведен на стр. 39.

См. также:

`RDS_BFM_LOADSTATE` (стр. 39), `rdsLoadSystemState` (стр. 251),
`rdsDeleteSystemState` (стр. 250), `rdsSaveSystemState` (стр. 252),
`rdsReadBlockData` (стр. 277), `rdsWriteBlockData` (стр. 280).

A.2.4.14. RDS_BFM_STARTCALC – запуск расчета

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_STARTCALC.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру RDS_STARTSTOPDATA, в которой содержатся параметры события.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие RDS_BFM_STARTCALC возникает при запуске расчета, то есть при переходе в режим расчета из режима моделирования (переход в режим расчета из режима редактирования производится через режим моделирования, хотя пользователь этого и не видит). В третьем параметре функции модели передается указатель на структуру RDS_STARTSTOPDATA, содержащую параметры запуска расчета:

```
typedef struct {  
    BOOL FirstStart;    // Расчет запущен с самого начала  
    BOOL Loop;          // Расчет будет работать непрерывно  
} RDS_STARTSTOPDATA;  
typedef RDS_STARTSTOPDATA *RDS_PSTARTSTOPDATA;
```

Поля структуры:

FirstStart

TRUE, если расчет запущен с самого начала (сразу после загрузки схемы или сброса расчета), или FALSE, если расчет повторно запущен после остановки.

Loop

TRUE, если расчет запущен в нормальном, циклическом, режиме, или FALSE, если после выполнения одного такта он будет автоматически остановлен (пользователь нажал кнопку “выполнить один такт”).

См. также:

RDS_BFM_STOPCALC (стр. 46), rdsCalcProcessNeverStarted (стр. 149),
rdsCalcProcessIsRunning (стр. 148), rdsStartCalc (стр. 174).

A.2.4.15. RDS_BFM_STOPCALC – остановка расчета

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_STOPCALC.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру RDS_STARTSTOPDATA, в которой содержатся параметры события (см. стр. 46).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие RDS_BFM_STOPCALC возникает после остановки расчета, то есть при переходе из режима расчета в режим моделирования (переход в режим редактирования из режима расчета производится через режим моделирования, хотя пользователь этого и не видит). В третьем параметре функции модели передается указатель на структуру RDS_STARTSTOPDATA содержащую параметры, аналогичные параметрам события запуска расчета.

См. также:

RDS_BFM_STARTCALC (стр. 46), rdsStopCalc (стр. 175).

A.2.4.16. RDS_BFM_TIMER – срабатывание таймера блока**Поток, в котором вызывается функция модели:**

Главный поток РДС или поток расчета – в зависимости от режима работы таймера.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_TIMER.

Третий параметр функции модели (void *ExtParam):

Идентификатор сработавшего таймера RDS_TIMERID, приведенный к типу void*.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие RDS_BFM_TIMER возникает при срабатывании таймера блока, созданного сервисной функцией rdsSetBlockTimer (стр. 300) в режимах RDS_TIMERS_TIMER или RDS_TIMERS_SYSTIMER. При срабатывании таймера в режиме RDS_TIMERS_TIMER модель блока будет вызвана в потоке расчета, в режиме RDS_TIMERS_SYSTIMER – в главном потоке. В третьем параметре функции модели передается идентификатор сработавшего таймера (идентификатор таймера возвращается сервисной функцией, создавшей его). Если модель блока одновременно использует несколько таймеров, сравнение этого идентификатора с идентификаторами созданных ей таймеров позволит выяснить, какой из них сработал.

Использование таймеров позволяет блокам выполнять различные действия периодически через заданный интервал времени или однократно с заданной задержкой. Примеры работы с таймерами блоков приведены в §2.9.

См. также:

rdsSetBlockTimer (стр. 300), RDS_BFM_WINREFRESH (стр. 77),
RDS_TIMERDESCRIPTION (стр. 136).

A.2.4.17. RDS_BFM_UNLOADSYSTEM – схема будет выгружена из памяти**Поток, в котором вызывается функция модели:**

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_UNLOADSYSTEM.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие RDS_BFM_UNLOADSYSTEM возникает во всех блоках непосредственно перед тем, как вся загруженная в данный момент схема будет удалена из памяти из-за загрузки другой схемы, создания новой или завершения РДС. Реагируя на это событие, модель блока может очистить какие-либо внешние вспомогательные данные, связанные со схемой. Например, если при работе со схемой были созданы временные файлы, в этой реакции их можно удалить.

Данная реакция вызывается у всех блоков до выгрузки схемы, то есть перед тем, как начнут вызываться реакции RDS_BFM_CLEANUP (стр. 33). На момент вызова RDS_BFM_UNLOADSYSTEM все блоки схемы еще находятся в памяти и их модели еще не отключены.

См. также:

RDS_BFM_AFTERLOAD (стр. 50), RDS_BFM_CLEANUP (стр. 33).

A.2.4.18. RDS_BFM_VARCHECK – проверка допустимости структуры статических переменных блока**Поток, в котором вызывается функция модели:**

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_VARCHECK.

Третий параметр функции модели (void *ExtParam):

Указатель на строку типа (char*, см. §1.5) статических переменных блока.

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Структура переменных блока удовлетворяет требованиям модели.
RDS_BFR_ERROR	Структура переменных блока не удовлетворяет требованиям модели, все дальнейшие реакции, кроме RDS_BFM_CLEANUP и новых RDS_BFM_VARCHECK, будут заблокированы.
RDS_BFR_BADVARMSG	Структура переменных блока не удовлетворяет требованиям модели, все дальнейшие реакции, кроме RDS_BFM_CLEANUP и новых RDS_BFM_VARCHECK, будут заблокированы, пользователю будет выведено сообщение.

Событие RDS_BFM_VARCHECK возникает при подключении к простому блоку новой модели после события RDS_BFM_INIT (стр. 38), а также при любом изменении структуры статических переменных блока. У подсистем, внешних входов/выходов и вводов шин это событие не возникает. Реагируя на это событие, функция модели должна сообщить РДС, подходит ли ей данная структура переменных. Если функция вернет константу RDS_BFR_DONE, структура будет считаться подходящей, и модель блока будет вызываться РДС для реакции на все остальные события. Если же она вернет RDS_BFR_ERROR или

RDS_BFR_BADVARSMSG, все события, кроме очистки данных блока (RDS_BFM_CLEANUP) и повторных проверок структуры переменных не будут передаваться в модель блока. Таким образом, написав реакцию на событие RDS_BFM_VARCHECK, программист может быть уверен, что структура статических переменных блока соответствует его ожиданиям, что даст ему возможность обращаться к этим переменным по фиксированным смещениям относительно начала дерева переменных. Работа со статическими переменными блока и примеры реакций на событие RDS_BFM_VARCHECK подробно рассмотрены в §2.5.

Текущая структура переменных блока, которую модель должна проанализировать, передается в третьем параметре функции в виде строки типа, в которой каждой переменной соответствует символ, обозначающий ее тип (все символы относятся к латинскому алфавиту):

Символ	Константа в РДС	Обозначаемый тип
S	RDS_VARTYPE_SIGNAL	Сигнал (1 байт)
L	RDS_VARTYPE_LOGICAL	Логический (1 байт)
C	RDS_VARTYPE_CHAR	char (целый, 1 байт)
H	RDS_VARTYPE_SHORT	short int (целый, 2 байта)
I	RDS_VARTYPE_INT	int (целый, 4 байта)
F	RDS_VARTYPE_FLOAT	float (вещественный, 4 байта)
D	RDS_VARTYPE_DOUBLE	double (вещественный, 8 байт)
A	RDS_VARTYPE_STRING	Строка
V	RDS_VARTYPE_RUNTIME	Произвольный тип (изменяется в процессе работы)
M	RDS_VARTYPE_ARRAY	Массив или матрица (следующий символ – тип элемента)
{	RDS_VARTYPE_STRUCT	Начало структуры (начиная со следующего символа описываются типы полей структуры)
}	RDS_VARTYPE_STRUCTEND	Конец структуры (описание типов полей структуры закончилось)

Последовательность таких символов однозначно описывает структуру статических переменных блока и их размещение в памяти. Поскольку вся совокупность переменных блока с точки зрения РДС является структурой, строка типа всегда начинается с символа “{” и заканчивается символом “}”. Ниже приведены некоторые примеры строк типа переменных блоков:

“{SSDD}”	Две первых переменных блока – сигналы (“S”), за ними следуют две вещественных переменных двойной точности (“D”).
“{SSMIA}”	Две первых переменных блока – сигналы (“S”), за ними следует массив или матрица целых чисел (“M”), за ней – строка (“A”).
“{SSVMMD}”	Два сигнала (“S”), переменная произвольного типа (“V”), матрица матриц вещественных чисел (“MMD”).
“{SS{DD}I{DD}}”	Два сигнала (“S”), структура с двумя вещественными полями двойной точности (“{DD}”), целое число (“I”), снова структура с двумя вещественными полями двойной точности (“{DD}”).
“{SSM{IDMA}}”	Два сигнала (“S”) и матрица (“M”) структур, содержащих три поля: целое (“I”), вещественное (“D”) и матрицу строк (“MA”).

Строка типа переменных простого блока всегда начинается с двух букв “S”, поскольку две первых переменных любого простого блока – это сигнал запуска и сигнал готовности. В

строке типа не содержится никакой информации о том, является ли переменная входом или выходом, о ее имени и об именах используемых структур (для структур указываются только типы их полей). Вся эта информация не влияет на собственно структуру дерева переменных, то есть на смещения к данным каждой переменной от начала дерева. По той же причине в строке типа не делается различия между матрицами и массивами – в РДС они хранятся в памяти одинаково.

Модель блока должна сравнить полученную ей в реакции на событие RDS_BFM_VARCHECK строку типа с одной или несколькими заложенными в нее строками, и вернуть РДС константу, указывающую на допустимость данной структуры переменных для данной модели. Возврат RDS_BFR_ERROR отличается от возврата RDS_BFR_BADVARSMSG только тем, что в последнем случае РДС выведет пользователю сообщение о недопустимости структуры переменных блока.

Пример:

Реакция на событие RDS_BFM_VARCHECK в модели, которая, помимо двух обязательных сигналов, должна иметь три вещественных переменных двойной точности:

```
switch(CallMode)
{
    case RDS_BFM_VARCHECK:
        if(strcmp((char*)ExtParam, "{SSDDD}")==0)
            return RDS_BFR_DONE;
        return RDS_BFR_BADVARSMSG;
```

См. также:

RDS_BFM_CLEANUP (стр. 33), RDS_BLOCKDATA (стр. 28).

А.2.5. События загрузки и сохранения схемы и отдельных блоков

В реакциях на эти события модель блока может выполнить какие-либо действия до, во время и после загрузки и сохранения схемы или отдельного блока. Сохранение и загрузка личных параметров блока, с которыми модель работает без участия РДС, также должна выполняться в этих реакциях.

А.2.5.1. RDS_BFM_AFTERLOAD – завершена загрузка схемы

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_AFTERLOAD.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_AFTERLOAD вызывается у всех блоков схемы сразу после окончания загрузки схемы из файла. На момент ее вызова все блоки и связи схемы уже загружены и модель блока может, при необходимости, считывать их параметры, вызывать функции других блоков и т.п. В этой реакции также можно производить инициализацию объектов, относящихся ко всей схеме в целом, например, стирать или создавать временные и вспомогательные файлы. Пример использования этой реакции приведен в §2.13.6.

См. также:

RDS_BFM_LOADBIN (стр. 52), RDS_BFM_LOADTXT (стр. 52).

A.2.5.2. RDS_BFM_AFTERSAVE – завершено сохранение схемы

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_AFTERSAVE.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_AFTERSAVE вызывается у всех блоков схемы сразу после окончания сохранения схемы в файл или передачи ее полного содержимого другому приложению через библиотеку дистанционного управления RdsCtrl.dll (см. §3.5). В этой реакции можно, например, сохранить параметры, относящиеся ко всей схеме в отдельный файл, или завершить какие-либо действия, начатые в реакции на событие начала сохранения схемы RDS_BFM_BEFORESAVE (стр. 51). Файл сохраненной схемы на момент вызова этой реакции уже закрыт, поэтому вызывать сервисные функции для сохранения данных блока в ней не следует (эти вызовы будут проигнорированы).

См. также:

RDS_BFM_BEFORESAVE (стр. 51).

A.2.5.3. RDS_BFM_BEFORESAVE – начато сохранение схемы

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_BEFORESAVE.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на это событие вызывается у всех блоков схемы непосредственно перед началом сохранения схемы в файл или передачей ее полного содержимого другому приложению через библиотеку дистанционного управления RdsCtrl.dll (см. §3.5). В этой реакции можно подготовить к сохранению какие-либо параметры, относящиеся ко всей схеме в целом, или привести схему в состояние, пригодное для сохранения. Например, если блоки по командам пользователя вносили в схему какие-либо изменения, которые не должны сохраняться, в реакции на событие RDS_BFM_BEFORESAVE можно привести схему в исходный вид. Файл для сохранения схемы на момент вызова этой реакции еще не открыт,

поэтому вызывать сервисные функции для сохранения данных блока в ней не следует (эти вызовы будут проигнорированы).

См. также:

RDS_BFM_AFTERSAVE (стр. 51).

A.2.5.4. RDS_BFM_LOADBIN – загрузка данных блока в двоичном формате

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_LOADBIN.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Данные загружены успешно.
RDS_BFR_ERROR	При загрузке произошла ошибка.

Реакция на событие RDS_BFM_LOADBIN вызывается для загрузки данных, сохраненных моделью в двоичном формате при сохранении схемы или отдельного блока в момент реакции на событие RDS_BFM_SAVEBIN (стр. 53). Модель должна загрузить в память параметры блока, хранящиеся в его личной области данных, не обрабатываемой РДС, причем размеры и последовательность загружаемых блоков данных должны совпадать с размерами и последовательностью сохраненных.

Загрузка данных блока выполняется в следующих случаях:

- при загрузке блока в составе схемы;
- при загрузке отдельного блока из файла на диске (например, при вставке его в схему из библиотеки блоков);
- при вставке блока из буфера обмена.

Событие RDS_BFM_LOADBIN возникает только в том случае, если данные блока были сохранены в двоичном формате. Если данные были сохранены в текстовом, вместо него возникнет событие RDS_BFM_LOADTXT (стр. 52). Формат сохранения может выбираться моделью блока независимо от того, в каком формате сохраняется вся схема или блок, поэтому в функции модели блока достаточно предусмотреть реакцию либо на пару событий, работающих с двоичным форматом (RDS_BFM_SAVEBIN/RDS_BFM_LOADBIN), либо на пару, работающую с текстовым (RDS_BFM_SAVETXT/RDS_BFM_LOADTXT). Сохранение и загрузка данных блока подробно рассматриваются в §2.8.

См. также:

RDS_BFM_SAVEBIN (стр. 53), RDS_BFM_LOADTXT (стр. 52),
rdsReadBlockData (стр. 277).

A.2.5.5. RDS_BFM_LOADTXT – загрузка данных блока в текстовом формате

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_LOADTXT.

Третий параметр функции модели (void *ExtParam):

Указатель на строку (char*), в которой содержится весь блок текста с сохраненными параметрами блока. Функция модели должна разобрать этот текст и записать параметры в личную область данных блока.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_LOADTXT вызывается для загрузки данных, сохраненных в текстовом формате при сохранении схемы или отдельного блока в момент реакции на событие RDS_BFM_SAVETXT (стр. 54). Модель должна загрузить в память параметры блока, хранящиеся в его личной области данных, не обрабатываемой РДС, анализируя текст, записанный ей же в момент сохранения. В РДС встроены различные дополнительные функции и объекты, использование которых позволяет упростить создание и разбор текста, описывающего параметры блока.

Загрузка данных блока выполняется в следующих случаях:

- при загрузке блока в составе схемы;
- при загрузке отдельного блока из файла на диске (например, при вставке его в схему из библиотеки блоков);
- при вставке блока из буфера обмена.

Событие RDS_BFM_LOADTXT возникает только в том случае, если данные блока были сохранены в текстовом формате. Если данные были сохранены в двоичном, вместо него возникнет событие RDS_BFM_LOADBIN (стр. 52). Формат сохранения может выбираться моделью блока независимо от того, в каком формате сохраняется вся схема или блок, поэтому в функции модели блока достаточно предусмотреть реакцию либо на пару событий, работающих с двоичным форматом (RDS_BFM_SAVEBIN/RDS_BFM_LOADBIN), либо на пару, работающую с текстовым (RDS_BFM_SAVETXT/RDS_BFM_LOADTXT). Сохранение и загрузка данных блока подробно рассматриваются в §2.8.

Результат возврата функции модели при реакции на событие не анализируется РДС. Если в процессе загрузки параметров произошла ошибка, функция модели должна вызвать сервисную функцию rdsReportTextLoadError (стр. 278), передав в ее параметре текстовое описание ошибки, которое будет показано пользователю.

См. также:

RDS_BFM_SAVETXT (стр. 54), RDS_BFM_LOADBIN (стр. 52),
rdsReportTextLoadError (стр. 278), rdsSTRCreateTextReader (стр. 460),
rdsINICreateTextHolder (стр. 473), rdsCSVCreate (стр. 556).

A.2.5.6. RDS_BFM_SAVEBIN – запись данных блока в двоичном формате

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_SAVEBIN.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

RDS_BFR_DONE Данные записаны успешно.

RDS_BFR_ERROR При записи произошла ошибка.

Реакция на событие RDS_BFM_SAVEBIN вызывается для записи параметров блока в двоичном формате при сохранении схемы или отдельного блока. В современных версиях РДС событие RDS_BFM_SAVEBIN возникает в одном из двух случаев:

- при записи блока в буфер обмена;
- при сохранении схемы или отдельного блока, если модель отказалась записывать данные в текстовом формате.

Если модель блока, вызванная для реакции на событие RDS_BFM_SAVEBIN, не сохранит никаких данных (то есть ни разу не вызовет функцию rdsWriteBlockData, стр. 280), она будет повторно вызвана для сохранения данных, но уже в текстовом формате. Таким образом, в функции модели достаточно предусмотреть запись (и загрузку) параметров только в одном из форматов. Сохранение и загрузка данных блока подробно рассматриваются в §2.8.

См. также:

RDS_BFM_SAVETXT (стр. 54), RDS_BFM_LOADBIN (стр. 52),
rdsWriteBlockData (стр. 280).

A.2.5.7. RDS_BFM_SAVETXT – запись данных блока в текстовом формате

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_SAVETXT.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_SAVETXT вызывается для записи параметров блока в текстовом формате при сохранении схемы или отдельного блока. В современных версиях РДС это событие возникает в одном из двух случаев:

- при сохранении всей схемы или отдельного блока;
- при записи блока в буфер обмена, если модель отказалась записывать данные в двоичном формате.

Если модель блока, вызванная для реакции на событие RDS_BFM_SAVETXT, не сохранит никаких данных (то есть ни разу не вызовет ни одну из сервисных функций записи текстовых данных), она будет повторно вызвана для сохранения данных, но уже в двоичном формате. Таким образом, в функции модели достаточно предусмотреть запись (и загрузку) параметров только в одном из форматов. Сохранение и загрузка данных блока подробно рассматриваются в §2.8.

Результат возврата функции модели при реакции на событие не анализируется. В реакции на событие RDS_BFM_SAVETXT функция модели не осуществляет фактической записи на диск, она только формирует текст, который записывается РДС позже. По этой причине никаких способов сообщить РДС об ошибках записи в этой реакции не предусмотрено.

См. также:

RDS_BFM_LOADTXT (стр. 52), RDS_BFM_SAVEBIN (стр. 53), функции загрузки и сохранения параметров (стр. 277), rdsSTRCreateTextReader (стр. 460), rdsINICreateTextHolder (стр. 473), rdsCSVCreate (стр. 556).

А.2.6. События пользовательского интерфейса и рисования внешнего вида блоков

В реакциях на эти события осуществляется взаимодействие с пользователем и настройка параметров блока, а также программное рисование внешнего вида блока, если это необходимо.

А.2.6.1. RDS_BFM_BLOCKPANEL – уведомление от панели блока в подсистеме

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_BLOCKPANEL.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру RDS_PANOPERATION, в которой содержится причина вызова (что именно произошло с панелью блока) и указатель на структуру описания панели.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на это событие вызывается при любых действиях с панелью блока в подсистеме: при ее создании, удалении, перемещении, изменении размера и т.п. (см. §2.10.4). В третьем параметре функции модели при этом передается указатель на структуру RDS_PANOPERATION, описывающую произошедшее с панелью:

```
typedef struct {  
    int Operation;                // Действие с панелью  
    RDS_PPANDESCRIPTION Panel;    // Описание панели  
} RDS_PANOPERATION;  
typedef RDS_PANOPERATION *RDS_PPANOPERATION;
```

Поля структуры:

Operation

Действие с панелью, на которое может отреагировать модель блока. В этом поле может содержаться одна из следующих констант:

RDS_PANOP_CREATE	Для панели создан объект-окно Windows. Модель блока теперь может использовать его дескриптор (например, для добавления внутрь панели различных полей ввода).
RDS_PANOP_DESTROY	Объект-окно Windows, связанный с панелью, будет уничтожен. Модель должна удалить все поля ввода и другие объекты, которые она создала в этом окне.
RDS_PANOP_RESIZED	Размер панели только что изменен пользователем или одной из сервисных функций РДС.
RDS_PANOP_MOVED	Панель перемещена пользователем.
RDS_PANOP_PAINT	Модель должна перерисовать содержимое панели.

Panel

Указатель на структуру описания панели RDS_PANDESCRIPTION (стр. 131). Эта структура создается только на время вызова модели блока для реакции на событие RDS_BFM_BLOCKPANEL. После завершения реакции структура уничтожается, поэтому переданный указатель нельзя запоминать для дальнейшего использования. Если модели блока понадобится описание панели вне реакции на событие RDS_BFM_BLOCKPANEL, она может вызвать сервисную функцию rdsPANGetDescr (стр. 547).

См. также:

RDS_PANDESCRIPTION (стр. 131), rdsPANCreate (стр. 545),
rdsPANGetDescr (стр. 547).

A.2.6.2. RDS_BFM_CONTEXTPOPUP – вызов контекстного меню блока

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_CONTEXTPOPUP.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_CONTEXTPOPUPDATA.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_CONTEXTPOPUP вызывается непосредственно перед вызовом контекстного меню блока или свободного места окна подсистемы, то есть при щелчке правой кнопкой мыши на блоке или в окне, если этот щелчок не был обработан одним из блоков подсистемы или самой подсистемой. Обычно в этой реакции модель добавляет в стандартное контекстное меню свои собственные пункты или управляет видимостью или разрешенностью ранее добавленных пунктов (см. §2.12.6). В третьем параметре функции модели при этом передается указатель на структуру RDS_CONTEXTPOPUPDATA:

```
typedef struct {  
    BOOL EditMode;           // Включен режим редактирования  
    BOOL FreeSpace;          // Щелчок на свободном месте окна  
} RDS_CONTEXTPOPUPDATA;  
typedef RDS_CONTEXTPOPUPDATA *RDS_PCONTEXTPOPUPDATA;
```

Поля структуры:

EditMode

TRUE, если РДС в данный момент находится в режиме редактирования, и FALSE, если РДС находится в режимах моделирования или расчета. Обычно пункты контекстного меню в режиме редактирования отличаются от пунктов в двух других режимах, и модель может использовать значение этого поля для добавления в меню нужных в данный момент пунктов.

FreeSpace

FALSE, если вызвано контекстное меню самого блока, и TRUE, если контекстное меню вызвано щелчком не на изображении блока, а на свободном месте окна подсистемы. Это поле может принимать значение TRUE только в том случае, если

данный блок – подсистема, и ее окно в данный момент открыто. Анализируя это значение, модель подсистемы может отличить контекстное меню, вызванное внутри ее собственного окна, от контекстного меню, вызванного в окне ее родительской подсистемы для этой подсистемы как для обычного блока.

См. также:

RDS_BFM_MENUFUNCTION (стр. 63), RDS_BFR_SHOWMENU (стр. 67),
rdsRegisterContextMenuEx (стр. 361),
rdsAdditionalContextMenuEx (стр. 355),
rdsChangeMenuItem (стр. 356), rdsEnableMenuItem (стр. 358),
rdsSetMenuItemOptions (стр. 363).

A.2.6.3. RDS_BFM_DRAW – рисование внешнего вида блока

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_DRAW.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_DRAWDATA.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_DRAW вызывается для блока, модель которого рисует внешний вид блока программно, в момент обновления окна подсистемы, в которой находится этот блок. Для включения программного рисования в окне параметров блока должен быть установлен флаг “Внешний вид блока – Определяется функцией DLL”, в противном случае блок будет изображаться либо векторной картинкой, либо прямоугольником с текстом, и реакция RDS_BFM_DRAW не будет вызвана. В ответ на вызов модель должна нарисовать в окне подсистемы изображение, используя для этого сервисные функции РДС или графические функции API Windows (программное рисование внешнего вида блоков подробно рассмотрено в §2.10). В третьем параметре функции модели при этом передается указатель на структуру RDS_DRAWDATA, содержащую оконные координаты, по которым необходимо нарисовать изображение, а также другие необходимые для рисования параметры:

```
typedef struct {  
    HDC dc;                // Контекст устройства рисования  
    BOOL CalcMode;         // Текущий режим РДС  
    int BlockX,            // Координаты точки привязки  
        BlockY;           // блока в окне  
    double DoubleZoom;     // Масштаб окна (в долях единицы)  
    BOOL RectValid;        // Флаг изменения размеров блока  
    int Left,Top;          // Верхний левый угол блока  
    int Width,Height;      // Ширина и высота блока  
    RECT *VisibleRect;     // Видимая в окне часть рабочего поля  
    BOOL FullDraw;         // Необходимо перерисовать весь блок  
} RDS_DRAWDATA;  
typedef RDS_DRAWDATA *RDS_PDRAWDATA;
```

Поля структуры:

dc

Контекст устройства Windows (device context, HDC), на котором функция модели должна нарисовать изображение. Его можно использовать в вызовах графических функций Windows API: Rectangle, LineTo, TextOut и т.п. Если для рисования используются сервисные функции РДС, контекст устройства указывать не нужно.

CalcMode

Текущий режим РДС: TRUE для режимов моделирования и расчета и FALSE для режима редактирования. В режимах моделирования и расчета на изображении блока, как правило, отражаются текущие значения переменных этого блока – если переменные не влияют на его внешний вид, в программном рисовании, чаще всего, нет необходимости.

BlockX, BlockY

Координаты точки привязки блока в окне (в данном случае, для программно рисуемых блоков, это левый верхний угол изображения) с учетом возможной связи положения блока с его статическими переменными. В этих координатах уже учтен текущий масштаб окна, то есть их можно использовать непосредственно в графических функциях. Горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля.

DoubleZoom

Текущий масштаб окна в долях единицы: 1 для 100%, 2 для 200%, 0.5 для 50% и т.п.

RectValid

Логический флаг изменения размеров прямоугольника блока. Перед вызовом реакции РДС записывает в это поле значение FALSE и заполняет следующие четыре поля структуры (Left, Top, Width и Height) согласно положению блока в окне, значениям его переменных, если положение с ними связано, и масштабу окна. Если по каким-либо причинам модель блока не устраивает вычисленный таким образом размер (например, если реальный размер блока как-то связан с его состоянием, и по результатам вычислений при рисовании он не совпал с размером, исходно определенным РДС), модель может записать в поле RectValid значение TRUE и заменить значения полей Left, Top, Width и Height на новые, вычисленные при рисовании. В этом случае РДС запомнит новый, вычисленный моделью, размер блока и будет использовать его для определения попадания курсора мыши в изображение этого блока вплоть до следующего рисования.

Left, Top

Левый верхний угол (Left – горизонтальная координата, Top – вертикальная) прямоугольной области, занимаемой блоком в окне в текущем масштабе, с учетом возможной связи положения блока с переменными. Эти значения можно непосредственно использовать в функциях рисования. В этих же полях модель может вернуть новое, вычисленное при рисовании, положение левого верхнего угла блока, если присвоит полю RectValid значение TRUE.

Width, Height

Ширина (Width) и высота (Height) прямоугольной области, занимаемой блоком в окне в текущем масштабе, с учетом возможной связи размеров блока с переменными. Эти значения можно непосредственно использовать в функциях рисования. В этих же полях модель может вернуть новые, вычисленные при рисовании, размеры блока, если присвоит полю RectValid значение TRUE.

VisibleRect

Указатель на структуру типа RECT (стандартный тип, используемый в Windows API для описания прямоугольника), в которой находятся координаты видимой в данный момент в окне части рабочего поля подсистемы. Ориентируясь на размеры этой области, модель может уменьшить потери времени на обновление окна, не рисуя части блока, не попавшие в видимую в данный момент область.

FullDraw

Логическое поле, указывающее на необходимость полной перерисовки всего изображения блока (TRUE) или только тех его частей, которые изменились с момента последнего рисования (FALSE). Если в параметрах подсистемы включен флаг “В системе только неподвижные не перекрывающиеся блоки”, при обновлении окна такой подсистемы по таймеру РДС будет вызывать рисование всех ее блоков с FullDraw равным FALSE, а во всех остальных случаях – с FullDraw равным TRUE. Таким образом, если окно не было перекрыто другими окнами и его изображение сохранилось на экране с момента последнего рисования, модель блока сможет избежать потерь времени на перерисовку не изменившихся частей изображения. Подобная оптимизация рисования подробно рассмотрена в §2.10.2.

Пример:

Модель блока, рисующего себя в виде белого прямоугольника с черной рамкой. Толщина рамки равна двум точкам экрана в масштабе 100%, в других масштабах она пропорционально уменьшается или увеличивается. Размер прямоугольника совпадает с размером всего блока, заданным пользователем.

```
extern "C" __declspec(dllexport) int RDSCALL ModelDraw(  
    int CallMode,                // Событие  
    RDS_PBLOCKDATA BlockData,    // Данные блока  
    LPVOID ExtParam)             // Дополнительные параметры  
{  
    RDS_PDRAWDATA DrawData;  
    switch(CallMode)  
    {  
        case RDS_BFM_DRAW:        // Рисование  
            DrawData=(RDS_PDRAWDATA)ExtParam;  
            // Цвет и стиль рамки  
            rdsXGSetPenStyle(0,PS_SOLID,  
                2*DrawData->DoubleZoom,0,R2_COPYPEN);  
            // Цвет фона  
            rdsXGSetBrushStyle(0,RDS_GFS_SOLID,0xffffffff);  
            // Прямоугольник  
            rdsXGRectangle(DrawData->Left,  
                DrawData->Top,  
                DrawData->Left+DrawData->Width,  
                DrawData->Top+DrawData->Height);  
  
            break;  
    }  
    return RDS_BFR_DONE;  
}
```

См. также:

RDS_BFM_DRAWADDITIONAL (стр. 59), графические функции РДС (стр. 365).

A.2.6.4. RDS_BFM_DRAWADDITIONAL – дополнительное рисование

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_DRAWADDITIONAL.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_DRAWDATA (см. стр. 57).

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_DRAWADDITIONAL вызывается для всех блоков подсистемы при обновлении ее окна. В отличие от события рисования RDS_BFM_DRAW, которое возникает только для блоков, внешний вид которых рисуется программно, событие RDS_BFM_DRAWADDITIONAL возникает у всех блоков независимо от их внешнего вида: у рисуемых программно, у имеющих векторную картинку и у изображаемых прямоугольником с текстом. Реакция на это событие позволяет выводить поверх изображения блока различную дополнительную информацию, например, иконки, сигнализирующие о каких-либо ошибках в блоке. Пример использования дополнительного рисования приведен в §2.10.3.

См. также:

RDS_BFM_DRAW (стр. 57), графические функции РДС (стр. 365),
rdsXGDrawStdIcon (стр. 368).

A.2.6.5. RDS_BFM_KEYDOWN – нажатие клавиши**Поток, в котором вызывается функция модели:**

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_KEYDOWN.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_KEYDATA.

Возвращаемое функцией модели значение:

RDS_BFR_DONE Нажатие клавиши не обработано блоком.

RDS_BFR_STOP Нажатие клавиши обработано, его не нужно передавать в остальные блоки подсистемы и вызывать соответствующий ей пункт главного меню РДС, если такой имеется.

Реакция на событие RDS_BFM_KEYDOWN вызывается при нажатии какой-либо клавиши клавиатуры в том случае, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;
- окно подсистемы, в которой находится данный блок, имеет фокус ввода (то есть это самое верхнее окно и РДС – активное приложение);
- в параметрах блока разрешена реакция на клавиатуру.

В режимах моделирования и расчета при нажатии клавиши поочередно (в произвольном порядке) вызываются модели всех блоков активного окна подсистемы, для которых разрешена реакция на клавиатуру. В параметре ExtParam при этом передается указатель на структуру RDS_KEYDATA, в которой содержится описание нажатой клавиши. Если функция модели вернет константу RDS_BFR_DONE, нажатие клавиши будет считаться не обработанным, и вызовется модель следующего блока; если же она вернет RDS_BFR_STOP, перебор моделей блоков будет прекращен, кроме того, если нажатая клавиша соответствует

какому-либо пункту меню РДС, этот пункт не будет вызван. Если ни одна из моделей блоков подсистемы не вернула RDS_BFR_STOP, будет вызвана модель самой подсистемы для реакции на событие RDS_BFM_WINDOWKEYDOWN (стр. 72). Если и модель подсистемы не сообщит об успешной обработке нажатия клавиши, нажатие будет обработано РДС (вызовется пункт главного меню, которому соответствует эта клавиша, если такой пункт есть). Реакция блоков на нажатие и отпускание клавиш подробно рассмотрена в §2.12.4.

В режиме редактирования модели блоков не могут реагировать на нажатия клавиш. Единственный способ связать вызов модели блока с нажатием клавиши или сочетания клавиш в режиме редактирования – это зарегистрировать дополнительный пункт системного меню (см. §2.12.7) функцией rdsRegisterMenuItem (стр. 362) и связать с ним “горячую клавишу”.

Для описания нажатой клавиши служит структура RDS_KEYDATA (она используется и в других событиях реакции на клавиатуру):

```
typedef struct {
    int KeyCode;           // Виртуальный код клавиши
    BOOL Repeat;           // Признак автоповтора
    int RepeatCount;       // Число повторений
    DWORD Shift;           // Флаги клавиатуры (RDS_M*, RDS_K*)
    int KeyEvent;          // Событие – RDS_BFM_KEYDOWN или
                          // RDS_BFM_KEYUP
    BOOL Handled;          // Событие обработано (возврат)
    int Viewport;          // Номер порта вывода или -1
} RDS_KEYDATA;
typedef RDS_KEYDATA *RDS_PKEYDATA;
```

Поля структуры:

KeyCode

Виртуальный код клавиши (VK_*) согласно описаниям Windows API.

Repeat

Только для события нажатия клавиши: TRUE, если нажатие клавиши сгенерировано автоповтором клавиатуры, и FALSE в противном случае. Таким образом, при одиночных нажатиях клавиш в этом поле всегда находится FALSE.

RepeatCount

Только при автоповторе нажатия клавиши – число повторов с момента прошлого вызова реакции на событие.

Shift

Битовые флаги, описывающие состояние специальных клавиш клавиатуры и кнопок мыши в момент нажатия данной клавиши:

RDS_MLEFTBUTTON	Нажата левая кнопка мыши.
RDS_MRIGHTBUTTON	Нажата правая кнопка мыши.
RDS_MMIDDLEBUTTON	Нажата средняя кнопка мыши.
RDS_KSHIFT	Нажата клавиша “Shift”.
RDS_KALT	Нажата клавиша “Alt”.
RDS_KCTRL	Нажата клавиша “Ctrl”.

Для выделения флагов, относящихся только к кнопкам мыши или только к специальным клавишам, можно использовать специальные константы-маски:

RDS_MOUSEFLAGS	Все флаги кнопок мыши (RDS_MLEFTBUTTON RDS_MRIGHTBUTTON RDS_MMIDDLEBUTTON).
RDS_KBDFLAGS	Все флаги специальных клавиш (RDS_KSHIFT RDS_KALT RDS_KCTRL).

KeyEvent

Произошедшее событие: RDS_BFM_KEYDOWN для нажатия клавиши и RDS_BFM_KEYUP для отпускания. В реакциях на события RDS_BFM_KEY* это поле дублирует параметр CallMode функции модели. В реакциях подсистем на нажатия и отпускания клавиш в окне RDS_BFM_WINDOWKEY* в это поле записывается идентификатор события, близкого по смыслу к произошедшему событию: для RDS_BFM_WINDOWKEYDOWN (стр. 72) в это поле записывается RDS_BFM_KEYDOWN, для RDS_BFM_WINDOWKEYUP (стр. 73) – RDS_BFM_KEYUP:

<i>Событие</i>	<i>CallMode</i>	<i>Поле KeyEvent</i>
Нажатие клавиши, обрабатывается блоком	RDS_BFM_KEYDOWN	RDS_BFM_KEYDOWN
Отпускание клавиши, обрабатывается блоком	RDS_BFM_KEYUP	RDS_BFM_KEYUP
Не обработанное блоками нажатие клавиши, обрабатывается подсистемой	RDS_BFM_WINDOWKEYDOWN	RDS_BFM_KEYDOWN
Не обработанное блоками отпускание клавиши, обрабатывается подсистемой	RDS_BFM_WINDOWKEYUP	RDS_BFM_KEYUP

Handled

Перед вызовом самого первого блока подсистемы это поле устанавливается в FALSE. Модель блока может присвоить ему TRUE для того, чтобы сообщить РДС об успешной обработке клавиши – это равносильно возврату RDS_BFR_STOP.

Viewport

Номер порта вывода (см. §3.6), из которого пришла информация о нажатии клавиши, или –1, если клавиша нажата в обычном окне подсистемы.

См. также:

RDS_BFM_KEYUP (стр. 62), RDS_BFM_WINDOWKEYDOWN (стр. 72), RDS_BFM_WINDOWKEYUP (стр. 73).

A.2.6.6. RDS_BFM_KEYUP – отпускание клавиши

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_KEYUP.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_KEYDATA (см. стр. 61).

Возвращаемое функцией модели значение:

RDS_BFR_DONE Отпускание клавиши не обработано блоком.

RDS_BFR_STOP Отпускание клавиши обработано, его не нужно передавать в остальные блоки подсистемы.

Реакция на событие RDS_BFM_KEYUP вызывается при отпускании ранее нажатой клавиши клавиатуры в том случае, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;

- окно подсистемы, в которой находится данный блок, имеет фокус ввода (то есть это самое верхнее окно и РДС – активное приложение);
- в параметрах блока разрешена реакция на клавиатуру.

Как и при нажатии клавиши, при ее отпускании в режимах моделирования и расчета поочередно (в произвольном порядке) вызываются модели всех блоков активного окна подсистемы, для которых разрешена реакция на клавиатуру. В параметре `ExtParam` при этом передается указатель на структуру `RDS_KEYDATA`, в которой содержится описание отпущенной клавиши. Если функция модели вернет константу `RDS_BFR_DONE`, отпускание клавиши будет считаться не обработанным, и вызовется модель следующего блока; если же она вернет `RDS_BFR_STOP`, перебор моделей блоков будет прекращен. Если ни одна из моделей блоков подсистемы не вернула `RDS_BFR_STOP`, будет вызвана модель самой подсистемы для реакции на событие `RDS_BFM_WINDOWKEYUP` (стр. 73). Реакция блоков на нажатие и отпускание клавиш подробно рассмотрена в §2.12.4.

См. также:

`RDS_KEYDATA` (стр. 61), `RDS_BFM_KEYDOWN` (стр. 60),
`RDS_BFM_WINDOWKEYDOWN` (стр. 72), `RDS_BFM_WINDOWKEYUP` (стр. 73).

A.2.6.7. `RDS_BFM_MENUFUNCTION` – выбор пользователем пункта меню

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (`int CallMode`):

Константа `RDS_BFM_MENUFUNCTION`.

Третий параметр функции модели (`void *ExtParam`):

Указатель на структуру описания пункта меню `RDS_MENUFUNCDATA`.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие `RDS_BFM_MENUFUNCTION` вызывается при выборе пользователем либо постоянного пункта контекстного или системного меню, зарегистрированного моделью блока, либо временного пункта контекстного меню, добавленного в него моделью при реакции на событие `RDS_BFM_CONTEXTPOPUP` (стр. 56). Постоянные пункты системного меню создаются сервисной функцией `rdsRegisterMenuItem` (стр. 362) и размещаются в меню “Система | Дополнительно” главного окна РДС, с ними можно связывать “горячие клавиши” (см. §2.12.7). Постоянные пункты контекстного (то есть выводимого при щелчке правой кнопкой мыши на блоке или свободном месте окна подсистемы, см. §2.12.6) меню создаются сервисными функциями `rdsRegisterContextMenuItem` (стр. 360) и `rdsRegisterContextMenuItemEx` (стр. 361), временные пункты контекстного меню – сервисными функциями `rdsAdditionalContextMenuItem` (стр. 354) и `rdsAdditionalContextMenuItemEx` (стр. 355). Вызывая модель блока для реакции на выбор пункта меню, РДС не делает разницы между системным и контекстным меню или между постоянными и временными пунктами – во всех случаях в модель передается константа `RDS_BFM_MENUFUNCTION`.

Для описания выбранного пункта служит структура `RDS_MENUFUNCDATA`:

```
typedef struct {
    int Function; // Номер функции меню
```

```

    int MenuData; // Данные меню
} RDS_MENUFUNCDDATA;
typedef RDS_MENUFUNCDDATA *RDS_PMENUFUNCDDATA;

```

Поля структуры:

Function

Идентификатор пункта меню, указанный при его создании.

MenuData

Дополнительное целое число, указанное при создании выбранного пункта меню.

Пара чисел Function / MenuData указывается при создании пункта меню и должна быть уникальной для данного блока, чтобы модель могла отличить один пункт от другого в реакции на событие RDS_BFM_MENUFUNCTION. Разные блоки могут иметь совпадающие идентификаторы пунктов меню, поскольку при выборе пункта всегда вызывается модель именно того блока, который создал этот пункт.

См. также:

RDS_BFM_CONTEXTPOPUP (стр. 56), rdsRegisterContextMenuItem (стр. 360),
 rdsRegisterContextMenuItemEx (стр. 361),
 rdsAdditionalContextMenuItem (стр. 354),
 rdsAdditionalContextMenuItemEx (стр. 355),
 rdsRegisterMenuItem (стр. 362).

A.2.6.8. RDS_BFM_MOUSEDBLCLICK – двойной щелчок мыши

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_MOUSEDBLCLICK.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_MOUSEDATA.

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Двойной щелчок обработан блоком.
RDS_BFR_NOTPROCESSED	Двойной щелчок не обработан блоком, он будет передан в родительскую подсистему, а затем, если и она его не обработает, в РДС.

Реакция на событие RDS_BFM_MOUSEDBLCLICK вызывается при двойном щелчке на изображении блока в окне подсистемы, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;
- изображение блока находится на видимом слое окна подсистемы, для которого разрешено редактирование;
- в параметрах блока разрешена реакция на мышь.

Блоки могут обрабатывать щелчки кнопок мыши только в режимах моделирования и расчета: в режиме редактирования мышь используется для редактирования схемы. Если изображения блоков перекрываются, двойной щелчок мыши сначала будет передан в блок, изображение которого находится ближе всего к переднему плану. Если его модель не обработает двойной щелчок, будет вызвана модель блока, находящегося дальше, и т.д. Если ни один блок не обработает двойной щелчок, будет вызвана модель подсистемы для реакции

на событие RDS_BFM_WINDOWMOUSEDCLICK (стр. 73). Если же и подсистема не среагирует на двойной щелчок, он будет обработан РДС (будет включен режим редактирования и, если щелчок пришелся на блок, будет открыто окно его параметров или вызвана функция настройки). Реакция блоков на мышь подробно рассматривается в §2.12. Следует помнить, что, в отличие от большинства других событий, при реакции на мышь возврат стандартной константы RDS_BFR_DONE сообщает РДС о том, что блок обработал действие пользователя. Таким образом, если разрешить в параметрах блока реакцию на мышь, блок начнет перехватывать все щелчки, пришедшие на его изображение, даже если не включать в модель реакции на события RDS_BFM_MOUSE*. Чтобы блок, которому разрешена реакция на мышь, был “прозрачным” для щелчков, его модель должна вернуть константу RDS_BFR_NOTPROCESSED (пример использования этой константы приведен в §2.12.3).

Для описания произошедшего события служит структура RDS_MOUSEDATA (она используется и в других событиях реакции на мышь):

```
typedef struct {
    int x,y;           // Координаты курсора мыши
    int BlockX,BlockY; // Координаты точки привязки блока
    int Left,Top;      // Верхний левый угол изображения блока
    int Width,Height;  // Размеры изображения блока
    int IntZoom;        // Масштаб окна подсистемы в %
    DWORD Button;       // Кнопка мыши (RDS_M*)
    DWORD Shift;        // Флаги мыши и клавиатуры (RDS_M*, RDS_K*)
    double DoubleZoom;  // Масштабный к-т окна (в долях единицы)
    int MouseEvent;     // Событие
    int Viewport;       // Номер порта вывода или -1
} RDS_MOUSEDATA;
typedef RDS_MOUSEDATA *RDS_PMOUSEDATA;
```

Поля структуры:

x, y

Координаты курсора мыши в окне подсистемы на момент возникновения события.

BlockX,BlockY

Координаты точки привязки изображения блока в текущем масштабе. Для блоков с векторной картинкой это положение начала координат этой картинки, для программно рисуемых блоков и блоков, изображаемых прямоугольником с текстом, это координаты левого верхнего угла изображения (горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля). Если положение блока связано с его переменными, значения BlockX и BlockY в этой структуре указываются уже с учетом значений этих переменных. При реакции на события RDS_BFM_WINDOWMOUSE* эти поля не используются.

Left,Top

Горизонтальная (Left) и вертикальная (Top) фактические координаты верхнего левого угла изображения блока в текущем масштабе на момент последней перерисовки окна подсистемы. При реакции на события RDS_BFM_WINDOWMOUSE* эти поля не используются.

Width,Height

Фактические ширина (Width) и высота (Height) изображения блока в текущем масштабе на момент последней перерисовки окна подсистемы. При реакции на события RDS_BFM_WINDOWMOUSE* эти поля не используются.

IntZoom

Текущий масштаб окна подсистемы в процентах.

Button

Кнопка мыши, нажатие или отпускание которой вызвало событие:

RDS_MLEFTBUTTON	Левая кнопка мыши.
RDS_MRIGHTBUTTON	Правая кнопка мыши.
RDS_MMIDDLEBUTTON	Средняя кнопка мыши.

Shift

Битовые флаги, описывающие состояние специальных клавиш клавиатуры и кнопок мыши в момент возникновения события (см. стр. 61).

DoubleZoom

Текущий масштаб окна подсистемы в долях единицы: 1 для 100%, 2 для 200%, 0.5 для 50% и т.п.

MouseEvent

Действие пользователя, которое привело к возникновению события. В реакциях на события RDS_BFM_MOUSE* дублирует параметр CallMode функции модели. В реакциях подсистем RDS_BFM_WINDOWMOUSE* в это поле записывается идентификатор события, близкого по смыслу к произошедшему событию:

<i>Событие</i>	<i>CallMode</i>	<i>Поле MouseEvent</i>
Нажатие кнопки мыши на изображении блока	RDS_BFM_MOUSEDOWN	RDS_BFM_MOUSEDOWN
Отпускание кнопки мыши на изображении блока	RDS_BFM_MOUSEUP	RDS_BFM_MOUSEUP
Перемещение курсора мыши над изображением блока	RDS_BFM_MOUSEMOVE	RDS_BFM_MOUSEMOVE
Двойной щелчок на изображении блока	RDS_BFM_MOUSEDBLCLICK	RDS_BFM_MOUSEDBLCLICK
Нажатие кнопки мыши на свободном месте окна подсистемы	RDS_BFM_WINDOWMOUSEDOWN	RDS_BFM_MOUSEDOWN
Отпускание кнопки мыши на свободном месте окна подсистемы	RDS_BFM_WINDOWMOUSEUP	RDS_BFM_MOUSEUP
Перемещение курсора мыши в окне подсистемы	RDS_BFM_WINDOWMOUSEMOVE	RDS_BFM_MOUSEMOVE
Двойной щелчок на свободном месте окна подсистемы	RDS_BFM_WINDOWMOUSEDBLCLICK	RDS_BFM_MOUSEDBLCLICK

<i>Событие</i>	<i>CallMode</i>	<i>Поле MouseEvent</i>
Проверка возможности выбора блока щелчком мыши (стр. 80)	RDS_BFM_MOUSESELECT	RDS_BFM_MOUSESELECT

Viewport

Номер порта вывода (см. §3.6), из которого пришла информация о событии (нажатии, отпускании кнопки, перемещении курсора и т.п.), или -1, если событие произошло в обычном окне подсистемы.

См. также:

RDS_BFM_MOUSEDOWN (стр. 67), RDS_BFM_MOUSEUP (стр. 69),
RDS_BFM_MOUSEMOVE (стр. 68), RDS_BFM_WINDOWMOUSEDOWN (стр. 74),
RDS_BFM_WINDOWMOUSEUP (стр. 76), RDS_BFM_WINDOWMOUSEMOVE (стр. 75),
RDS_BFM_WINDOWMOUSEDCLICK (стр. 73).

A.2.6.9. RDS_BFM_MOUSEDOWN – нажатие кнопки мыши

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_MOUSEDOWN.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_MOUSEDATA (стр. 65).

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Нажатие обработано блоком.
RDS_BFR_NOTPROCESSED	Нажатие не обработано блоком, оно будет передано в родительскую подсистему, а затем, если и она его не обработает, в РДС.
RDS_BFR_SHOWMENU	Нажатие правой кнопки мыши обработано блоком, но, несмотря на это, нужно вывести контекстное меню.

Реакция на событие RDS_BFM_MOUSEDOWN вызывается при нажатии любой кнопки мыши на изображении блока в окне подсистемы, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;
- изображение блока находится на видимом слое окна подсистемы, для которого разрешено редактирование;
- в параметрах блока разрешена реакция на мышшь.

Кроме того, если блок захватил мышшь (взведен флаг RDS_MOUSECAPTURE, см. стр. 30), реакция на это событие будет вызываться даже если курсор мыши покинет пределы изображения блока (захват мыши подробно описан в §2.12.2).

Блоки могут обрабатывать щелчки кнопок мыши только в режимах моделирования и расчета: в режиме редактирования мышшь используется для редактирования схемы. Если изображения блоков перекрываются, нажатие кнопки мыши сначала будет передано в блок, изображение которого находится ближе всего к переднему плану. Если его модель не обработает нажатие, будет вызвана модель блока, находящегося дальше, и т.д. Если ни один блок не обработает нажатие, будет вызвана модель подсистемы для реакции на событие

RDS_BFM_WINDOWMOUSEDOWN (стр. 74). Если же и подсистема не среагирует, нажатие будет обработано РДС (если нажата правая кнопка, будет показано контекстное меню). Реакция блоков на мышь подробно рассматривается в §2.12. Следует помнить, что, в отличие от большинства других событий, при реакции на мышь возврат стандартной константы RDS_BFR_DONE сообщает РДС о том, что блок обработал действие пользователя. Таким образом, если разрешить в параметрах блока реакцию на мышь, блок начнет перехватывать все щелчки, пришедшие на его изображение, даже если не включать в модель реакции на события RDS_BFM_MOUSE*. Чтобы блок, которому разрешена реакция на мышь, был “прозрачным” для щелчков, его модель должна вернуть константу RDS_BFR_NOTPROCESSED (пример использования этой константы приведен в §2.12.3). В реакции на нажатие правой кнопки модель может также вернуть константу RDS_BFR_SHOWMENU, информируя РДС о том, что, хотя нажатие обработано, необходимо все равно вывести контекстное меню блока. При нажатиях других кнопок возврат RDS_BFR_SHOWMENU аналогичен возврату RDS_BFR_NOTPROCESSED.

См. также:

RDS_MOUSEDATA (стр. 65), RDS_BFM_MOUSEDBLCLICK (стр. 64),
RDS_BFM_MOUSEUP (стр. 69), RDS_BFM_MOUSEMOVE (стр. 68),
RDS_BFM_WINDOWMOUSEDOWN (стр. 74), RDS_BFM_WINDOWMOUSEUP (стр. 76),
RDS_BFM_WINDOWMOUSEMOVE (стр. 75),
RDS_BFM_WINDOWMOUSEDBLCLICK (стр. 73), RDS_BFM_MOUSESELECT (стр. 80).

A.2.6.10. RDS_BFM_MOUSEMOVE – перемещение курсора мыши

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_MOUSEMOVE.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_MOUSEDATA (стр. 65).

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Перемещение курсора обработано блоком.
RDS_BFR_NOTPROCESSED	Перемещение курсора не обработано блоком, оно будет передано в родительскую подсистему.

Реакция на событие RDS_BFM_MOUSEMOVE вызывается при перемещении курсора мыши в пределах изображения блока в окне подсистемы, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;
- изображение блока находится на видимом слое окна подсистемы, для которого разрешено редактирование;
- в параметрах блока разрешена реакция на мышь;
- нажата хотя бы одна кнопка мыши, *или* в параметрах блока разрешена реакция на перемещения курсора при не нажатых кнопках.

Кроме того, если блок захватил мышь (взведен флаг RDS_MOUSECAPTURE, см. стр. 30), реакция на это событие будет вызываться даже если курсор мыши покинет пределы изображения блока (захват мыши подробно описан в §2.12.2).

Блоки могут обрабатывать перемещения курсора мыши только в режимах моделирования и расчета: в режиме редактирования мышь используется для редактирования

схемы. Если изображения блоков перекрываются, перемещение курсора сначала будет передано в блок, изображение которого находится ближе всего к переднему плану. Если его модель не обработает событие, будет вызвана модель блока, находящегося дальше, и т.д. Если ни один блок не обработает перемещение курсора, будет вызвана модель подсистемы для реакции на событие `RDS_BFM_WINDOWMOUSEMOVE` (стр. 75). Реакция блоков на мышшь подробно рассматривается в §2.12.

См. также:

`RDS_MOUSEDATA` (стр. 65), `RDS_BFM_MOUSEDBLCLICK` (стр. 64),
`RDS_BFM_MOUSEDOWN` (стр. 67), `RDS_BFM_MOUSEUP` (стр. 69),
`RDS_BFM_WINDOWMOUSEDOWN` (стр. 74), `RDS_BFM_WINDOWMOUSEUP` (стр. 76),
`RDS_BFM_WINDOWMOUSEMOVE` (стр. 75),
`RDS_BFM_WINDOWMOUSEDBLCLICK` (стр. 73).

A.2.6.11. `RDS_BFM_MOUSEUP` – отпускание кнопки мыши

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (`int CallMode`):

Константа `RDS_BFM_MOUSEUP`.

Третий параметр функции модели (`void *ExtParam`):

Указатель на структуру описания события `RDS_MOUSEDATA` (стр. 65).

Возвращаемое функцией модели значение:

<code>RDS_BFR_DONE</code>	Отпускание кнопки обработано блоком.
<code>RDS_BFR_NOTPROCESSED</code>	Отпускание кнопки не обработано блоком, оно будет передано в родительскую подсистему.

Реакция на событие `RDS_BFM_MOUSEUP` вызывается при отпускании любой кнопки мыши на изображении блока в окне подсистемы, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;
- изображение блока находится на видимом слое окна подсистемы, для которого разрешено редактирование;
- в параметрах блока разрешена реакция на мышшь.

Кроме того, если блок захватил мышшь (взведен флаг `RDS_MOUSECAPTURE`, см. стр. 30), реакция на это событие будет вызываться даже если курсор мыши покинет пределы изображения блока (захват мыши подробно описан в §2.12.2).

Блоки могут обрабатывать щелчки кнопок мыши только в режимах моделирования и расчета: в режиме редактирования мышшь используется для редактирования схемы. Если изображения блоков перекрываются, отпускание кнопки мыши сначала будет передано в блок, изображение которого находится ближе всего к переднему плану. Если его модель не обработает отпускание кнопки, будет вызвана модель блока, находящегося дальше, и т.д. Если ни один блок не обработает событие, будет вызвана модель подсистемы для реакции на событие `RDS_BFM_WINDOWMOUSEUP` (стр. 76). Реакция блоков на мышшь подробно рассматривается в §2.12.

См. также:

`RDS_MOUSEDATA` (стр. 65), `RDS_BFM_MOUSEDBLCLICK` (стр. 64),
`RDS_BFM_MOUSEDOWN` (стр. 67), `RDS_BFM_MOUSEMOVE` (стр. 68),

RDS_BFM_WINDOWMOUSEDOWN (стр. 74), RDS_BFM_WINDOWMOUSEUP (стр. 76),
RDS_BFM_WINDOWMOUSEMOVE (стр. 75),
RDS_BFM_WINDOWMOUSEDLCLICK (стр. 73), RDS_BFM_MOUSESELECT (стр. 80).

A.2.6.12. RDS_BFM_POPUPHINT – вывод всплывающей подсказки

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_POPUPHINT.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_POPUPHINTDATA.

Возвращаемое функцией модели значение:

RDS_BFR_DONE Необходимо вывести всплывающую подсказку (игнорируется, если строка подсказки не передана в РДС).

RDS_BFR_NOTPROCESSED Всплывающую подсказку выводить не нужно.

Реакция на событие RDS_BFM_POPUPHINT вызывается у блока, в параметрах которого разрешен вывод всплывающей подсказки, если курсор мыши задержался в пределах его изображения (время задержки определяется операционной системой). В этой реакции модель может передать РДС текст подсказки, которую необходимо вывести, при помощи сервисной функции rdsSetHintText (стр. 249), а также записать в поля переданной структуры RDS_POPUPHINTDATA параметры подсказки – размеры зоны, интервал гашения и т.д. Если функция модели не установит текст подсказки или передаст в функцию rdsSetHintText пустую строку, подсказка выведена не будет. Вывод всплывающих подсказок к блокам подробно рассмотрен в §2.11.

В параметре ExtParam передается указатель на структуру RDS_POPUPHINTDATA, в которой содержатся текущие параметры изображения блока и через которую модель возвращает параметры показа подсказки, если это необходимо:

```
typedef struct {  
    int x,y;                      // Координаты курсора мыши  
    int BlockX,BlockY;    // Координаты точки привязки блока  
    int Left,Top;                // Верхний левый угол изображения блока  
    int Width,Height;    // Размеры изображения блока  
    int HZLeft,HZTop,    // Возвращаемый размер зоны  
                                 HZWidth,HZHeight; // действия подсказки  
    int ReshowTimeout;    // Возвращаемый интервал повторного вывода  
    int HideTimeout;        // Возвращаемый интервал гашения  
    int IntZoom;                // Масштаб окна в %  
    double DoubleZoom;    // Масштабный к-т окна (в долях единицы)  
} RDS_POPUPHINTDATA;  
typedef RDS_POPUPHINTDATA *RDS_PPUPUPHINTDATA;
```

Поля структуры:

x, y

Текущие координаты курсора мыши в окне подсистемы на момент возможного вывода подсказки.

BlockX,BlockY

Координаты точки привязки изображения блока в текущем масштабе. Для блоков с векторной картинкой это положение начала координат этой картинки, для программно рисуемых блоков и блоков, изображаемых прямоугольником с текстом, это координаты левого верхнего угла изображения (горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля). Если положение блока связано с его переменными, значения BlockX и BlockY в этой структуре указываются уже с учетом значений этих переменных.

Left,Top

Горизонтальная (Left) и вертикальная (Top) фактические координаты верхнего левого угла изображения блока в текущем масштабе на момент последней перерисовки окна подсистемы.

Width,Height

Фактические ширина (Width) и высота (Height) изображения блока в текущем масштабе на момент последней перерисовки окна подсистемы.

HZLeft, HZTop, HZWidth, HZHeight

Возвращаемая моделью зона действия подсказки: горизонтальная (HZLeft) и вертикальная (HZTop) координаты верхнего левого угла зоны, ее ширина (HZWidth) и высота (HZHeight). При выходе курсора мыши за пределы этой прямоугольной зоны будет запрошен повторный вывод подсказки – так можно разбить всю площадь блока на несколько участков и выводить в них разные подсказки. На момент вызова функции модели в этих полях записаны координаты и размеры всего изображения блока, поэтому, если модель ничего в них не изменит, весь блок будет считаться одной зоной.

ReshowTimeout

Возвращаемое моделью время в миллисекундах после гашения подсказки, по истечении которого подсказку необходимо вывести снова. На момент вызова функции модели это поле имеет нулевое значение, что означает, что подсказка не будет выведена повторно до тех пор, пока курсор мыши не покинет зону ее действия.

HideTimeout

Возвращаемое моделью время в миллисекундах после вывода подсказки, по истечении которого ее необходимо убрать с экрана. На момент вызова функции модели в это поле записано стандартное для операционной системы значение.

IntZoom

Текущий масштаб окна подсистемы в процентах.

DoubleZoom

Текущий масштаб окна подсистемы в долях единицы: 1 для 100%, 2 для 200%, 0.5 для 50% и т.п.

См. также:

rdsSetHintText (стр. 249).

A.2.6.13. RDS_BFM_SETUP – вызов функции настройки блока

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_SETUP.

Третий параметр функции модели (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модели значение:

0 или RDS_BFR_DONE	Параметры блока не изменились.
Любое ненулевое значение	Параметры блока изменены.

Событие RDS_BFM_SETUP наступает при вызове функции настройки блока пользователем (см. §2.7). Реагируя на него, функция модели должна самостоятельно открыть окно, в котором пользователь сможет изменить параметры блока, если, конечно, это необходимо. Возврат функцией модели нулевого значения информирует о том, что пользователь не внес никаких изменений в параметры блока. Возврат любого ненулевого сообщит о том, что параметры блока изменились: теперь при попытке закрыть текущую схему или загрузить другую РДС предложит пользователю сохранить измененную схему. Явно вызывать сервисную функцию rdsSetModifiedFlag (стр. 171) для указания наличия изменений в схеме в этом случае не требуется.

Вызов функции настройки возможен только в режиме редактирования.

См. также:

rdsSetModifiedFlag (стр. 171), вспомогательные объекты для модальных окон (стр. 491).

А.2.6.14. RDS_BFM_WINDOWKEYDOWN – реакция подсистемы на нажатие клавиши в своем окне**Поток, в котором вызывается функция модели:**

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_WINDOWKEYDOWN.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_KEYDATA (стр. 61).

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Нажатие клавиши не обработано подсистемой и должно быть обработано РДС (будет вызван соответствующий этой клавише пункт главного меню РДС, если такой имеется).
RDS_BFR_STOP	Нажатие клавиши обработано.

Реакция на событие RDS_BFM_WINDOWKEYDOWN вызывается в модели подсистемы при нажатии какой-либо клавиши клавиатуры, в том случае, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;
- окно данной подсистемы имеет фокус ввода (то есть это самое верхнее окно и РДС – активное приложение);
- ни один из блоков подсистемы, реагируя на событие RDS_BFM_KEYDOWN (см. стр. 60), не заявил о том, что нажатие клавиши им обработано;
- в параметрах данной подсистемы разрешена реакция на клавиатуру в ее собственном окне.

Подсистема реагирует на нажатия клавиш в своем окне после всех находящихся в этом окне блоков. Если модель какого-либо блока, реагируя на событие RDS_BFM_KEYDOWN, вернет

константу RDS_BFR_STOP, обработка нажатия клавиши на этом прекратится и модель подсистемы вызвана не будет. Реакция подсистем на не перехваченные блоками события клавиатуры и мыши рассмотрена в §2.12.5.

См. также:

RDS_KEYDATA (стр. 61), RDS_BFM_KEYDOWN (стр. 60),
RDS_BFM_WINDOWKEYUP (стр. 73).

A.2.6.15. RDS_BFM_WINDOWKEYUP – реакция подсистемы на отпускание клавиши в своем окне

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_WINDOWKEYUP.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_KEYDATA (стр. 61).

Возвращаемое функцией модели значение:

RDS_BFR_DONE Отпускание клавиши не обработано подсистемой и должно быть обработано РДС.

RDS_BFR_STOP Отпускание клавиши обработано.

Реакция на событие RDS_BFM_WINDOWKEYUP вызывается в модели подсистемы при отпускании какой-либо клавиши клавиатуры, в том случае, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;
- окно данной подсистемы имеет фокус ввода (то есть это самое верхнее окно и РДС – активное приложение);
- ни один из блоков подсистемы, реагируя на событие RDS_BFM_KEYUP (стр. 62), не заявил о том, что отпускание клавиши им обработано;
- в параметрах данной подсистемы разрешена реакция на клавиатуру в ее собственном окне.

Подсистема реагирует на отпускание клавиш в своем окне после всех находящихся в этом окне блоков. Если модель какого-либо блока, реагируя на событие RDS_BFM_KEYUP, вернет константу RDS_BFR_STOP, обработка отпускания клавиши на этом прекратится и модель подсистемы вызвана не будет. Реакция подсистем на не перехваченные блоками события клавиатуры и мыши рассмотрена в §2.12.5.

См. также:

RDS_KEYDATA (стр. 61), RDS_BFM_KEYUP (стр. 62),
RDS_BFM_WINDOWKEYDOWN (стр. 72).

A.2.6.16. RDS_BFM_WINDOWMOUSEDBLCLICK – реакция подсистемы на двойной щелчок мыши в своем окне

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_WINDOWMOUSEDBLCLICK.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_MOUSEDATA (стр. 65).

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Двойной щелчок обработан подсистемой.
RDS_BFR_NOTPROCESSED	Двойной щелчок не обработан подсистемой, он будет передан в РДС.

Реакция на событие RDS_BFM_WINDOWMOUSEDBLCLICK вызывается при двойном щелчке в окне подсистемы, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;
- щелчок пришелся на свободное (не занятое изображениями блоков) место рабочего поля окна, *или* блок, на изображение которого пришелся щелчок, не среагировал на него, то есть его модель вернула RDS_BFR_NOTPROCESSED;
- в параметрах подсистемы разрешена реакция на действия мышью в окне.

Если двойной щелчок мыши пришелся на изображение блока в окне подсистемы, сначала будет вызвана его модель. Только если модель блока не обработала щелчок, он будет передан в модель подсистемы. Если подсистема тоже не среагирует на двойной щелчок, он будет обработан РДС (будет включен режим редактирования и, если щелчок пришелся на блок, будет открыто окно его параметров или вызвана функция настройки). Реакция подсистем на не перехваченные блоками события клавиатуры и мыши рассмотрена в §2.12.5.

См. также:

RDS_MOUSEDATA (стр. 65), RDS_BFM_MOUSEDBLCLICK (стр. 64),
RDS_BFM_WINDOWMOUSEDOWN (стр. 74), RDS_BFM_WINDOWMOUSEUP (стр. 76),
RDS_BFM_WINDOWMOUSEMOVE (стр. 75).

А.2.6.17. RDS_BFM_WINDOWMOUSEDOWN – реакция подсистемы на нажатие кнопки мыши в своем окне**Поток, в котором вызывается функция модели:**

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_WINDOWMOUSEDOWN.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события (RDS_MOUSEDATA*, стр. 65).

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Нажатие обработано подсистемой.
RDS_BFR_NOTPROCESSED	Нажатие не обработано подсистемой, оно будет передано в РДС.

Реакция на событие RDS_BFM_WINDOWMOUSEDOWN вызывается при нажатии кнопки мыши в окне подсистемы, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;

- нажатие пришлось на свободное (не занятое изображениями блоков) место рабочего поля окна, *или* блок, на изображение которого пришлось нажатие, не среагировал на него, то есть его модель вернула RDS_BFR_NOTPROCESSED;
- в параметрах подсистемы разрешена реакция на действия мышью в окне.

Если нажатие кнопки мыши пришлось на изображение блока в окне подсистемы, сначала будет вызвана его модель. Только если модель не обработала щелчок, он будет передан в модель подсистемы. Реакция подсистем на не перехваченные блоками события клавиатуры и мыши рассмотрена в §2.12.5.

См. также:

RDS_MOUSEDATA (стр. 65), RDS_BFM_MOUSEDOWN (стр. 67),
RDS_BFM_WINDOWMOUSEDOWN (стр. 74), RDS_BFM_WINDOWMOUSEUP (стр. 76),
RDS_BFM_WINDOWMOUSEMOVE (стр. 75),
RDS_BFM_WINDOWMOUSEDCLICK (стр. 73).

A.2.6.18. RDS_BFM_WINDOWMOUSEMOVE – реакция подсистемы на перемещение курсора мыши в своем окне

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_WINDOWMOUSEMOVE.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_MOUSEDATA (стр. 65).

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Перемещение курсора обработано подсистемой.
RDS_BFR_NOTPROCESSED	Перемещение курсора не обработано подсистемой, оно будет передано в РДС.

Реакция на событие RDS_BFM_WINDOWMOUSEMOVE вызывается при перемещении курсора мыши в окне подсистемы, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;
- перемещение происходит над свободным (не занятым изображениями блоков) местом рабочего поля окна, *или* блок, над изображением которого перемещается курсор, не среагировал на перемещение, то есть его модель вернула RDS_BFR_NOTPROCESSED;
- в параметрах подсистемы разрешена реакция на действия мышью в окне;
- нажата хотя бы одна кнопка мыши, *или* в параметрах подсистемы разрешена реакция на перемещения курсора в окне при не нажатых кнопках.

Кроме того, если подсистема захватила мышшь (введен флаг RDS_MOUSECAPTURE, см. стр. 30), реакция ее модели будет вызываться при любом перемещении курсора по рабочему полю окна, независимо от того, находится ли под курсором какой-либо блок (захват мыши подробно описан в §2.12.2).

Если перемещение курсора мыши пришлось на изображение блока в окне подсистемы, сначала будет вызвана его модель. Только если модель не обработала перемещение, оно будет передано в модель подсистемы. Реакция подсистем на не перехваченные блоками события клавиатуры и мыши рассмотрена в §2.12.5.

См. также:

RDS_MOUSEDATA (стр. 65), RDS_BFM_MOUSEMOVE (стр. 68),
RDS_BFM_WINDOWMOUSEDOWN (стр. 74), RDS_BFM_WINDOWMOUSEUP (стр. 76),
RDS_BFM_WINDOWMOUSEDBLCLICK (стр. 73).

A.2.6.19. RDS_BFM_WINDOWMOUSEUP – реакция подсистемы на отпускание кнопки мыши в своем окне

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_WINDOWMOUSEUP.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_MOUSEDATA (стр. 65).

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Отпускание кнопки обработано подсистемой.
RDS_BFR_NOTPROCESSED	Отпускание кнопки не обработано подсистемой, оно будет передано в РДС.

Реакция на событие RDS_BFM_WINDOWMOUSEUP вызывается при отпускании кнопки мыши в окне подсистемы, если одновременно выполняются следующие условия:

- РДС находится в режиме моделирования или расчета;
- отпускание происходит над свободным (не занятым изображениями блоков) местом рабочего поля окна, *или* блок, над изображением которого отпущена кнопка, не среагировал на это событие, то есть его модель вернула RDS_BFR_NOTPROCESSED;
- в параметрах подсистемы разрешена реакция на действия мышью в окне.

Кроме того, если подсистема захватила мышь (взведен флаг RDS_MOUSECAPTURE, см. стр. 30), реакция ее модели будет вызываться независимо от того, находится ли под курсором какой-либо блок (захват мыши подробно описан в §2.12.2).

Если отпускание кнопки мыши пришлось на изображение блока в окне подсистемы, сначала будет вызвана его модель. Только если модель не обработала отпускание кнопки, оно будет передано в модель подсистемы. Реакция подсистем на не перехваченные блоками события клавиатуры и мыши рассмотрена в §2.12.5.

См. также:

RDS_MOUSEDATA (стр. 65), RDS_BFM_MOUSEUP (стр. 69),
RDS_BFM_WINDOWMOUSEDOWN (стр. 74), RDS_BFM_WINDOWMOUSEMOVE (стр. 75),
RDS_BFM_WINDOWMOUSEDBLCLICK (стр. 73).

A.2.6.20. RDS_BFM_WINDOWOPERATION – открытие и закрытие окна подсистемы

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_WINDOWOPERATION.

Третий параметр функции модели (**void *ExtParam**):

Указатель на структуру описания события RDS_WINOPERATIONDATA.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_WINDOWOPERATION вызывается у подсистемы и у всех непосредственно находящихся в ней блоков при открытии или закрытии окна подсистемы. Модели глубоко вложенных блоков (блоков, находящихся в подсистемах этой подсистемы) не вызываются.

В параметре ExtParam передается указатель на структуру RDS_WINOPERATIONDATA, описывающую событие:

```
typedef struct {  
    int Operation;           // Операция с окном  
    HWND Handle;            // Дескриптор окна  
    BOOL EditMode;          // Включен режим редактирования  
    BOOL Running;           // Идет расчет  
    BOOL OwnWindow;         // Операция с окном данной подсистемы  
} RDS_WINOPERATIONDATA;  
typedef RDS_WINOPERATIONDATA *RDS_PWINOPERATIONDATA;
```

Поля структуры:

Operation

Одна из констант RDS_SWO_*, указывающая, что именно произошло с окном:

RDS_SWO_OPEN открытие окна;
RDS_SWO_CLOSE закрытие окна.

Handle

Дескриптор окна подсистемы. Его можно использовать в вызовах Windows API.

EditMode

TRUE – РДС находится в режиме редактирования, FALSE – в режимах моделирования или расчета.

Running

TRUE – РДС находится в режиме расчета (то есть работает поток расчета), FALSE – в режимах редактирования или моделирования.

OwnWindow

TRUE – открыто или закрыто окно подсистемы, модель которой вызывается, FALSE – окно подсистемы, родительской по отношению к вызываемой подсистеме или блоку.

См. также:

rdsCheckSystemWindow (стр. 255), rdsCloseSystemWindow (стр. 255),
rdsOpenSystemWindow (стр. 261), rdsOpenSystemWindowEx (стр. 262).

A.2.6.21. RDS_BFM_WINREFRESH – обновление окон блока

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (**int CallMode**):

Константа RDS_BFM_WINREFRESH.

Третий параметр функции модели (**void *ExtParam**):

Указатель на структуру описания события RDS_WINREFRESHDATA.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_WINREFRESH вызывается у подсистемы или блока при необходимости обновить принадлежащие им окна по таймеру или по команде от пользователя, другого блока или внешнего управляющего приложения. Действия по обновлению стандартных окон подсистем производятся автоматически, в их модели не обязательно добавлять реакцию на это событие. Если же блоку или подсистеме принадлежат какие-либо немодальные окна, открытые моделью самостоятельно, процедурой их обновления должна заниматься модель. Следует помнить, что при работе с немодальными окнами в РДС необходимо блокировать данные, как описывается в §1.8.

В параметре ExtParam при реакции на это событие передается указатель на структуру RDS_WINREFRESHDATA:

```
typedef struct {  
    RDS_TIMERID Timer; // Таймер, вызвавший обновление, или NULL  
    double Delay;      // Возвращаемое время обновления окна, мс  
} RDS_WINREFRESHDATA;  
typedef RDS_WINREFRESHDATA *RDS_PWINREFRESHDATA;
```

Поля структуры:

Timer

Идентификатор сработавшего таймера, созданного с флагом RDS_TIMERS_WINREF (такие таймеры служат для обновления окон, см. стр. 136 и 300) или NULL, если обновление окна вызвано не таймером, а сервисной функцией или командой пользователя.

Delay

Возвращаемое моделью полное время обновления окна в миллисекундах. Исходно в этом поле находится значение –1. Модель может не вычислять это время и оставить в поле отрицательное значение, в этом случае РДС вычислит время обновления окна по задержке между вызовом модели и возвратом из нее. Времена обновления окон используются в адаптивном алгоритме вычисления частоты обновления для недопущения слишком больших потерь времени на обновление при высокой загрузке процессора.

См. также:

rdsRefreshBlockWindows (стр. 263), rdsSetBlockTimer (стр. 300),
RDS_TIMERDESCRIPTION (стр. 136).

А.2.7. События, связанные с изменением схемы пользователем

Эти события возникают при перемещении блоков, изменении их размеров, добавлении и удалении блоков в схему и т.п. Хотя по смыслу они и близки к событиям пользовательского интерфейса (А.2.6), они выделены в отдельную группу, поскольку реакции на них требуются гораздо реже.

А.2.7.1. RDS_BFM_MANUALDELETE – удаление блока пользователем

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_MANUALDELETE.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_MANUALDELETEDATA.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие RDS_BFM_MANUALDELETE возникает непосредственно перед тем, как данный блок или содержащая его подсистема будут удалены из схемы пользователем. При удалении блока сервисной функцией или при очистке памяти перед загрузкой новой схемы это событие не возникает. Реакция на это событие может использоваться, например, для информирования пользователя о возможных последствиях удаления важных для работы блоков (см. §2.12.8). Реагируя на событие RDS_BFM_MANUALDELETE, модель не может отменить удаление блока – он будет удален в любом случае.

В параметре ExtParam при реакции на это событие передается указатель на структуру RDS_MANUALDELETEDATA:

```
typedef struct {  
    BOOL Single;    // Удаляется один блок  
    BOOL WithSys;   // Удаляется блок внутри удаляемой подсистемы  
} RDS_MANUALDELETEDATA;  
typedef RDS_MANUALDELETEDATA *RDS_PMANUALDELETEDATA;
```

Поля структуры:**Single**

TRUE, если данный блок – единственный удаляемый, и FALSE, если удаляется группа блоков, в которую входит данный.

WithSys

TRUE, если пользователь удаляет подсистему, внутри которой (на любом уровне вложенности) находится этот блок, и FALSE, если этот блок удаляется непосредственно (пользователь выделил данный блок или группу из блоков, в которую вошел данный, и выбрал пункт меню “Удалить”).

См. также:

RDS_BFM_CLEANUP (стр. 33).

A.2.7.2. RDS_BFM_MANUALINSERT – вставка блока пользователем**Поток, в котором вызывается функция модели:**

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_MANUALINSERT.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_MANUALINSERTDATA.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие `RDS_BFM_MANUALINSERT` возникает после того, как данный блок помещен в схему пользователем при помощи вставки из буфера обмена, загрузки из отдельного файла, добавления из библиотеки или с панели блоков. Если блок добавлен в схему сервисной функцией, это событие не возникает. Если в схему вставляется подсистема, содержащая другие блоки, реакция на событие `RDS_BFM_MANUALINSERT` вызывается только у самой подсистемы, а внутренние блоки информацию о событии не получают. Реакция на это событие может использоваться, например, для автоматического открытия окна настройки блока при его вставке из библиотеки (см. §2.12.8).

В параметре `ExtParam` при реакции на это событие передается указатель на структуру `RDS_MANUALINSERTDATA`:

```
typedef struct {
    int Reason;      // Способ добавления (константа RDS_LS_*)
    BOOL Single;     // Добавлен только один блок
} RDS_MANUALINSERTDATA;
typedef RDS_MANUALINSERTDATA *RDS_PMANUALINSERTDATA;
```

Поля структуры:

Reason

Одна из двух стандартных констант, указывающих на способ добавления блока в схему (эти же константы возвращаются при вызове сервисной функции `rdsGetSystemInt` с параметром `RDS_GSISAVELOADACTION`, см. стр. 159):

`RDS_LS_LOADCLIPBRD` Блок вставлен из буфера обмена.

`RDS_LS_LOADFROMFILE` Блок вставлен из библиотеки, с панели блоков или загружен из отдельного файла.

Single

`TRUE`, если в схему вставлен только один блок, и `FALSE`, если вставлено сразу несколько блоков (например, группа блоков из буфера обмена).

См. также:

`RDS_BFM_INIT` (стр. 38), `RDS_BFM_LOADTXT` (стр. 52),
`RDS_BFM_LOADBIN` (стр. 52).

A.2.7.3. `RDS_BFM_MOUSESELECT` – возможность выбора блока мышью

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (`int CallMode`):

Константа `RDS_BFM_MOUSESELECT`.

Третий параметр функции модели (`void *ExtParam`):

Указатель на структуру описания события `RDS_MOUSEDATA` (стр. 65).

Возвращаемое функцией модели значение:

`RDS_BFR_DONE` Указанная в параметрах события точка принадлежит изображению блока и этот блок должен быть выделен.

`RDS_BFR_NOTPROCESSED` Указанная точка не принадлежит блоку, нужно выделить один из ниже лежащих блоков, если такие есть.

Реакция на событие `RDS_BFM_MOUSESELECT` вызывается при попытке выделить блок в режиме редактирования щелчком левой кнопки мыши, либо вызвать его контекстное меню щелчком правой. Чаще всего это событие используется в блоках сложной формы (см.

§2.12.3) для проверки, принадлежит ли точка под курсором мыши данному блоку, или она приходится на “прозрачную” часть занятой им прямоугольной области, сквозь которую может быть виден другой, лежащий ниже, блок. Если точка, координаты которой переданы в структуре RDS_MOUSEDATA, принадлежит данному блоку, функция модели должна вернуть константу RDS_BFR_DONE, если не принадлежит – константу RDS_BFR_NOTPROCESSED. Поскольку по умолчанию функция модели всегда возвращает RDS_BFR_DONE, блоки, в которых отсутствует реакция на событие RDS_BFM_MOUSESELECT, считаются занимающими всю область их описывающего прямоугольника.

См. также:

RDS_MOUSEDATA (стр. 65), RDS_BFM_MOUSEDOWN (стр. 67).

A.2.7.4. RDS_BFM_MOVED – перемещение блока

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_MOVED.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_MOVEDATA.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Событие RDS_BFM_MOVED возникает при любом изменении положения блока на рабочем поле подсистемы: при его перетаскивании пользователем, при задании его координат, при вызове сервисной функции для перемещения блока и т.п. Реакция на это событие может применяться, например, для синхронного перемещения группы блоков при перемещении одного из них, или для сообщения новых координат блока управляющей программе. В параметре ExtParam передается указатель на структуру RDS_MOVEDATA:

```
typedef struct {
    int MoveReason;      // Причина перемещения
    int OldX, OldY;      // Старые координаты точки привязки
    int NewX, NewY;      // Новые координаты точки привязки
} RDS_MOVEDATA;
typedef RDS_MOVEDATA *RDS_PMOVEDATA;
```

Поля структуры:

MoveReason

Одна из стандартных констант, указывающих на причину изменения координат блока:

RDS_MR_SET	Координаты блока установлены непосредственно: заданы пользователем в окне параметров, установлены сервисной функцией rdsMoveBlock (стр. 240) и т.п.
RDS_MR_DRAG	Блок “переташен” мышью.
RDS_MR_KEYBOARD	Выделенный в окне подсистемы блок перемещен нажатием курсорных клавиш.
RDS_MR_UNDOREDO	Ранее выполненное перемещение блока отменено или повторено пользователем.

OldX, OldY

Координаты точки привязки блока до перемещения в масштабе 100%. Для блоков с векторной картинкой точка привязки – это положение начала координат этой картинки, для программно рисуемых блоков и блоков, изображаемых прямоугольником с текстом, это координаты левого верхнего угла изображения. Горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля.

NewX, NewY

Координаты точки привязки блока после перемещения в масштабе 100%.

См. также:

rdsMoveBlock (стр. 240), rdsSetBlockRect (стр. 246).

A.2.7.5. RDS_BFM_RENAME – переименование блока

Поток, в котором вызывается функция модели:

Главный поток РДС или поток расчета – в зависимости от причины переименования.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_RENAME.

Третий параметр функции модели (void *ExtParam):

Указатель на строку (char*), в которой содержится предыдущее (до переименования) имя блока.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_RENAME вызывается у блока, переименованного пользователем или сервисной функцией rdsRenameBlock (стр. 241). В ней можно, например, изменить имена каких-либо связанных с блоком объектов, которые формируются из имени блока. Если блок переименован пользователем, функция модели вызывается в главном потоке расчета, если же он переименован в результате вызова rdsRenameBlock, функция модели будет вызвана в потоке, вызвавшем rdsRenameBlock.

См. также:

rdsRenameBlock (стр. 241).

A.2.7.6. RDS_BFM_RESIZE – размер блока изменен пользователем

Поток, в котором вызывается функция модели:

Главный поток РДС или поток расчета (только если размер изменен сервисной функцией, вызванной в потоке расчета).

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_RESIZE.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_RESIZEDATA.

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Изменение размеров блока подтверждено моделью (размеры могут быть скорректированы).
RDS_BFR_STOP	Изменение размеров блока отменено моделью.

Событие RDS_BFM_RESIZE возникает у блока, внешний вид которого рисуется программно, сразу после изменения его размеров пользователем. На момент вызова реакции на это событие размер блока еще не изменен, он изменится только после нее. Чаще всего в этой реакции подстраиваются размеры каких-либо внутренних элементов изображения блока (см. §2.10.2) или корректируются изменения размера, сделанные пользователем, если по каким-либо причинам они не подходят для блока.

В параметре ExtParam передается указатель на структуру RDS_RESIZEDATA, описывающую изменение размеров:

```
typedef struct {  
    BOOL HorzResize;    // Изменение горизонтального размера  
    BOOL VertResize;    // Изменение вертикального размера  
    int newWidth,newHeight; // Новые значения ширины и высоты  
    int GridDx,GridDy; // Шаги сетки редактора  
    BOOL SnapToGrid;    // Привязка к сетке  
} RDS_RESIZEDATA;  
typedef RDS_RESIZEDATA *RDS_PRE resizedata;
```

Поля структуры:

HorzResize

Изменена ширина блока.

VertResize

Изменена высота блока. Вместе с полем HorzResize это поле определяет, каким именно образом пользователь изменил размер блока:

<i>HorzResize</i>	<i>VertResize</i>	<i>Действия пользователя</i>
FALSE	FALSE	Новые размеры блока заданы числами в окне параметров.
FALSE	TRUE	Пользователь перетащил верхнюю или нижнюю метку масштабирования вверх или вниз (изменена только высота).
TRUE	FALSE	Пользователь перетащил левую или правую метку масштабирования влево или вправо (изменена только ширина).
TRUE	TRUE	Пользователь перетащил одну из угловых меток масштабирования (изменены и ширина, и высота).

newWidth,newHeight

Новые значения ширины (newWidth) и высоты (newHeight) блока в масштабе 100%. Модель может вмешаться в изменение размеров, заменив значения в этих полях. Например, можно записать вместо newHeight старое значение высоты блока, тогда изменение высоты, сделанное пользователем, будет проигнорировано. Текущие (старые) размеры блока могут быть получены при помощи сервисных функций rdsGetBlockDimensions (стр. 221) и rdsGetBlockDimensionsEx (стр. 221).

GridDx,GridDy

Горизонтальный (GridDx) и вертикальный (GridDy) шаги сетки в окне подсистемы.

SnapToGrid

TRUE, если в окне подсистемы включена привязка к сетке.

См. также:

RDS_BFM_RESIZING (стр. 84), rdsGetBlockDimensions (стр. 221),
rdsGetBlockDimensionsEx (стр. 221).

A.2.7.7. RDS_BFM_RESIZING – изменение размеров блока пользователем

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_RESIZING.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру описания события RDS_RESIZEDATA (стр. 83).

Возвращаемое функцией модели значение:

RDS_BFR_DONE	Изменение размеров блока подтверждено моделью (размеры могут быть скорректированы).
RDS_BFR_STOP	Изменение размеров блока отменено моделью.

Событие RDS_BFM_RESIZING возникает у блока, внешний вид которого рисуется программно, в процессе изменения его размеров пользователем. В отличие от события RDS_BFM_RESIZE (стр. 82), которое возникает по окончании изменения размеров, это событие возникает постоянно при каждом движении курсора мыши, когда пользователь перетаскивает одну из меток масштабирования блока. Реакция на событие RDS_BFM_RESIZING может использоваться для создания визуальной обратной связи при изменении размеров: если модель в реакции на это событие как-то вмешается в изменение размеров блока, скорректировав поля newWidth и newHeight структуры RDS_RESIZEDATA, это немедленно отразится на размере прямоугольника, который пользователь видит при перетаскивании меток масштабирования (см. §2.12.8).

См. также:

RDS_RESIZEDATA (стр. 83), RDS_BFM_RESIZE (стр. 82),
rdsGetBlockDimensions (стр. 221), rdsGetBlockDimensionsEx (стр. 221).

A.2.8. События обмена данными по сети

События обмена данными по сети возникают при получении блоком порции данных от блока на другой машине, а также, в некоторых случаях, после отправки им такой порции.

A.2.8.1. RDS_BFM_NETCONNECT – установка соединения

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_NETCONNECT.

Третий параметр функции модели (**void *ExtParam**):

Указатель на структуру RDS_NETCONNDATA, содержащую идентификатор установленного соединения, имя канала и другую информацию о соединении.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_NETCONNECT вызывается у блока, запросившего сетевое соединение с каким-либо каналом передачи данных сервера вызовом rdsNetConnect (стр. 391), после успешной установки этого соединения. Чаще всего отправку данных на сервер не производят до наступления этого события: хотя РДС позволяет ставить данные в очередь на передачу до фактической установки соединения, обычно этого не делают, чтобы не перегружать очередь. Процедуры обмена данными по сети, используемые в РДС, подробно описаны в §2.15.

В параметре ExtParam передается указатель на структуру RDS_NETCONNDATA, содержащую параметры установленного соединения:

```
typedef struct {  
    int ConnId;           // Идентификатор соединения  
    LPSTR Host;           // URL или IP-адрес сервера  
    int Port;             // Порт  
    LPSTR Channel;        // Имя канала  
    BOOL ByServer;        // Соединение разорвано сервером  
} RDS_NETCONNDATA;  
typedef RDS_NETCONNDATA *RDS_PNETCONNDATA;
```

Поля структуры:

ConnId

Идентификатор соединения, возвращенный функцией rdsNetConnect при его создании.

Host

Имя (URL) или IP-адрес сервера, с которым установлено соединение (только в том случае, если в качестве сервера не используется та же копия rds.exe, в которую загружена схема с данным блоком).

Port

Номер сетевого порта, через который идет обмен данными.

Channel

Имя канала передачи данных.

ByServer

TRUE, если соединение разорвано сервером (только в реакции на событие RDS_BFM_NETDISCONNECT, в реакции на RDS_BFM_NETCONNECT это поле не используется).

См. также:

RDS_BFM_NETDISCONNECT (стр. 88), rdsNetConnect (стр. 391).

A.2.8.2. RDS_BFM_NETDATAACCEPTED – получение данных сервером

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (**int CallMode**):

Константа RDS_BFM_NETDATAACCEPTED.

Третий параметр функции модели (**void *ExtParam**):

Указатель на структуру RDS_NETACCEPTDATA, содержащую информацию о принятом сервером блоке данных.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_NETDATAACCEPTED вызывается у блока, передавшего данные на сервер, после получения сервером этих данных. Это событие возникает только в том случае, если, передавая данные, блок запросил у сервера подтверждение их приема, указав флаг RDS_NETSEND_SERVREPLY (стр. 390) при вызове функции передачи данных rdsNetBroadcastData (стр. 389) или rdsNetSendData (стр. 392). Чаще всего реакция на событие RDS_BFM_NETDATAACCEPTED используется для организации в блоке собственной очереди передачи данных, не связанной с очередью РДС: в этом случае блок не передает очередную порцию данных, пока не получит от сервера подтверждение приема предыдущей. Следует помнить, что подтверждение выдается при получении данных сервером, а не блоком-получателем данных – серверу еще только предстоит отправить данные получателю. Процедуры обмена данными по сети, используемые в РДС, подробно описаны в §2.15.

В параметре ExtParam передается указатель на структуру RDS_NETACCEPTDATA:

```
typedef struct {  
    int ConnId;           // Идентификатор соединения  
    LPSTR Host;           // URL или IP-адрес сервера  
    int Port;             // Порт  
    LPSTR Channel;        // Имя канала  
    int Id;               // Идентификатор полученного блока данных  
} RDS_NETACCEPTDATA;  
typedef RDS_NETACCEPTDATA *RDS_PNETACCEPTDATA;
```

Поля структуры:

ConnId

Идентификатор соединения, возвращенный функцией rdsNetConnect (стр. 391) при его создании.

Host

Имя (URL) или IP-адрес сервера, с которым установлено соединение (только в том случае, если в качестве сервера не используется та же копия rds.exe, в которую загружена схема с данным блоком).

Port

Номер сетевого порта, через который идет обмен данными.

Channel

Имя канала передачи данных.

Id

Идентификатор полученных сервером данных (целое число, указанное в вызове одной из функций передачи данных). По нему модель блока может определить, какая именно порция данных принята сервером.

См. также:

RDS_BFM_NETDATARECEIVED (стр. 87), rdsNetBroadcastData (стр. 389), rdsNetSendData (стр. 392).

A.2.8.3. RDS_BFM_NETDATARECEIVED – получение данных блоком

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_NETDATARECEIVED.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру RDS_NETRECEIVEDDATA, содержащую информацию о принятых данных и указатели на них.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_NETDATARECEIVED вызывается у блока, для которого поступили данные с сервера. Для того, чтобы принимать данные по сети, блок должен сначала вызовом rdsNetConnect (стр. 391) установить соединение с конкретным каналом передачи данных конкретного сервера, и указать при этом, что он будет получать данные из этого канала. Процедуры обмена данными по сети, используемые в РДС, подробно описаны в §2.15.

В параметре ExtParam передается указатель на структуру RDS_NETRECEIVEDDATA, в которой содержатся указатели на принятые данные и информация об их отправителе:

```
typedef struct {  
    int ConnId;           // Идентификатор соединения  
    LPSTR Host;           // URL или IP-адрес сервера  
    int Port;             // Порт  
    LPSTR Channel;        // Имя канала  
    int Id;               // Принятое целое число  
    LPSTR Str;            // Принятая строка  
    LPVOID Buffer;         // Принятый буфер или NULL  
    DWORD BufferSize;     // Размер принятого буфера  
    RDS_NETSTATION SenderStation; // Идентификатор  
                                // машины-отправителя  
    RDS_NETBLOCK SenderBlock; // Идентификатор блока  
                                // на машине-отправителе  
} RDS_NETRECEIVEDDATA;  
typedef RDS_NETRECEIVEDDATA *RDS_PNETRECEIVEDDATA;
```

Поля структуры:

ConnId

Идентификатор соединения, возвращенный функцией rdsNetConnect при его создании.

Host

Имя (URL) или IP-адрес сервера, с которым установлено соединение (только в том случае, если в качестве сервера не используется та же копия rds.exe, в которую загружена схема с данным блоком).

Port

Номер сетевого порта, через который идет обмен данными.

Channel

Имя канала передачи данных.

Id

Целый идентификатор полученных данных (произвольное целое число, указанное передавшим данные блоком в вызове одной из функций передачи).

Str

Указатель на принятую строку во внутренней памяти РДС. В этом поле никогда не находится константа NULL: даже если передавший блок не отправлял строку в составе порции данных, в поле Str будет содержаться указатель на пустую строку (то есть на нулевой байт).

Buffer

Указатель на начало принятых двоичных данных во внутренней памяти РДС. Если передававший данные блок не отправлял двоичные данные, в этом поле будет находиться NULL.

BufferSize

Размер (в байтах) принятых двоичных данных, на которые указывает поле Buffer.

SenderStation

Идентификатор машины-передатчика, то есть машины, на которой запущена копия РДС, в которую загружена схема, блок которой отправил эти данные. Идентификаторы типа RDS_NETSTATION используются в РДС для передачи данных конкретному блоку на конкретной машине.

SenderBlock

Идентификатор блока на машине-передатчике. Идентификаторы блоков типа RDS_NETBLOCK не взаимозаменяемы с идентификаторами RDS_BHANDLE: первые используются только в сетевых функциях для передачи данных конкретному блоку, вторые – в сервисных функциях для работы с блоками в пределах одной схемы. Пара (SenderStation, SenderBlock) однозначно определяет блок-отправитель данных и может использоваться в функции rdsNetSendData (стр. 392) для передачи ответа пославшему данные блоку.

См. также:

rdsNetBroadcastData (стр. 389), rdsNetSendData (стр. 392).

A.2.8.4. RDS_BFM_NETDISCONNECT – разрыв соединения

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_NETDISCONNECT.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру RDS_NETCONNDATA (стр. 85), содержащую идентификатор соединения, имя канала и другую информацию о соединении.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_NETDISCONNECT вызывается у блока, ранее запрашивавшего сетевое соединение с каким-либо каналом передачи данных сервера вызовом rdsNetConnect (стр. 391), при разрыве этого соединения. Если соединение разорвано по инициативе сервера, в поле ByServer структуры RDS_NETCONNDATA, указатель на которую передается в параметре ExtParam, будет находиться значение TRUE. В этом случае РДС будет пытаться самостоятельно восстановить разорванное соединение, никаких действий в модели блока для этого предпринимать не нужно. Если соединение разорвано моделью блока при помощи функции rdsNetCloseConnection (стр. 391), в поле ByServer будет находиться значение FALSE. Процедуры обмена данными по сети, используемые в РДС, подробно описаны в §2.15.

См. также:

RDS_BFM_NETCONNECT (стр. 84), rdsNetCloseConnection (стр. 391).

A.2.8.5. RDS_BFM_NETERROR – ошибка при работе с сетью

Поток, в котором вызывается функция модели:

Главный поток РДС.

Первый параметр функции модели (int CallMode):

Константа RDS_BFM_NETERROR.

Третий параметр функции модели (void *ExtParam):

Указатель на структуру RDS_NETERRORDATA, содержащую код и описание ошибки.

Возвращаемое функцией модели значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_BFM_NETERROR вызывается у блоков, участвующих в приеме или передаче данных, при возникновении различных ошибок в сетевом соединении. Большую часть таких ошибок РДС обрабатывает самостоятельно, поэтому, как правило, в моделях блоков реакция на это событие не требуется. Процедуры обмена данными по сети, используемые в РДС, подробно описаны в §2.15.

При реакции на событие RDS_BFM_NETERROR в параметре ExtParam передается указатель на структуру RDS_NETERRORDATA:

```
typedef struct {
    int ConnId;           // Идентификатор соединения
    LPSTR Host;           // URL или IP-адрес сервера
    int Port;             // Порт
    LPSTR Channel;        // Имя канала
    int ErrorCode;        // Код ошибки (RDS_NETERR_*)
    RDS_NETSTATION Station; // Идентификатор машины
    RDS_NETBLOCK Block;   // Идентификатор блока
} RDS_NETERRORDATA;
typedef RDS_NETERRORDATA *RDS_PNETERRORDATA;
```

Поля структуры:

ConnId

Идентификатор соединения, возвращенный функцией rdsNetConnect (стр. 391) при его создании.

Host

Имя (URL) или IP-адрес сервера, с которым установлено соединение (только в том случае, если в качестве сервера не используется та же копия rds.exe, в которую загружена схема с данным блоком).

Port

Номер сетевого порта, через который идет обмен данными.

Channel

Имя канала передачи данных.

ErrorCode

Код возникшей ошибки – одна из следующих констант:

RDS_NETERR_NOBLOCK	Блок, которому переданы данные функцией rdsNetSendData (стр. 392), не существует.
RDS_NETERR_SEND	Ошибка при передаче данных.
RDS_NETERR_RECEIVE	Ошибка при приеме данных.
RDS_NETERR_DISCONNECT	Ошибка при попытке разорвать связь.
RDS_NETERR_ACCEPT	Ошибка при установке соединения сервера с клиентом.
RDS_NETERR_CLIENTCONN	Для блоков схемы, работающей на сервере: ошибка при установке соединения с клиентом.
RDS_NETERR_GENERAL	Неизвестная ошибка.

Station

Сетевой идентификатор “проблемной” машины типа RDS_NETSTATION. При коде ошибки RDS_NETERR_NOBLOCK в этом поле указывается сетевой идентификатор машины, переданный при вызове данным блоком функции rdsNetSendData, если пара идентификаторов “машина, блок”, использованная в этом вызове, не соответствует реально существующему блоку (то есть если такой машины нет, или если идентификатор машины правильный, но на ней нет блока с указанным в вызове идентификатором).

Block

Сетевой идентификатор “проблемного” блока типа RDS_NETBLOCK. При коде ошибки RDS_NETERR_NOBLOCK в этом поле указывается сетевой идентификатор блока, переданный при вызове данным блоком функции rdsNetSendData, если пара идентификаторов “машина, блок”, использованная в этом вызове, не соответствует реально существующему блоку (то есть если такой машины нет, или если идентификатор машины правильный, но на ней нет блока с указанным в вызове идентификатором).

См. также:

RDS_BFM_NETCONNECT (стр. 84), RDS_BFM_NETDISCONNECT (стр. 88),
rdsNetBroadcastData (стр. 389), rdsNetSendData (стр. 392).

А.3. События модуля автокомпиляции и связанные с ними структуры

Описываются все события, на которые может реагировать модуль автоматической компиляции моделей блоков (см. главу 4), а также структуры данных, связанные с этими событиями.

А.3.1. Функция модуля автокомпиляции

Функция модуля автокомпиляции – это функция, которая вызывается РДС в ответ на различные события, относящиеся к блокам, для которых включена автоматическая компиляция моделей при помощи данного модуля. Принципы и способы автоматической компиляции моделей подробно рассмотрены в главе 4. Функции модулей автокомпиляции должны размещаться в динамически подключаемых библиотеках (DLL) и регистрироваться в РДС в окне настроек автокомпиляции (см. §4.1). Все функции модулей автокомпиляции имеют тип вызова **RDSCALL** (стр. 25), описанный в файле “RdsDef.h” – ему полностью соответствует тип **CALLBACK** в Windows API: аргументы функции передаются в стеке справа налево, стек освобождается вызванной функцией. В отличие от моделей блоков, функции модулей автокомпиляции всегда вызываются в главном потоке РДС, обслуживающем интерфейс пользователя. Функция модуля имеет следующий вид:

```
int RDSCALL имя_функции_модуля(  
    int CallMode,                                // Режим вызова (событие)  
    RDS_COMPMODULEDATA ModuleData,              // Данные модуля  
    LPVOID ExtParam                             // Дополнительные параметры  
);
```

Параметры функции:

CallMode

Событие, для реакции на которое вызывается функция модуля. Это одна из целых констант **RDS_COMP_***, описанных ниже.

ModuleData

Указатель на структуру данных модуля (**RDS_COMPMODULEDATA**, стр. 92). В этой структуре хранятся общие данные модуля и его настроечные параметры.

ExtParam

Указатель на дополнительные параметры события. Формат данных, на которые ссылается этот указатель общего вида, зависит от конкретного события, то есть от значения параметра **CallMode**.

Возвращаемые значения:

Функция возвращает целое число, указывающее РДС на результат реакции на событие. В зависимости от события эти числа интерпретируются по-разному.

Как и функция модели блока, функция модуля автокомпиляции чаще всего содержит внутри оператор **switch**, в котором, в зависимости от произошедшего события (то есть от значения параметра **CallMode**), выполняются различные действия:

```
extern "C" __declspec(dllexport)  
    int RDSCALL имя_модуля(  
        int CallMode,  
        RDS_COMPMODULEDATA ModuleData,  
        LPVOID ExtParam)  
{  
    switch(CallMode)  
    { case <событие_1>:  
        <действия>  
        return <результат>;
```

```

        case <событие_2>:
            <действия>
            return <результат>;
        ...
    }
}

```

Если модуль имеет личную область данных, то есть область памяти, в которой хранятся его внутренние параметры, не обрабатываемые РДС, эта область обычно отводится в реакции на событие инициализации RDS_COMPM_INIT (стр. 103) и освобождается в реакции на событие RDS_COMPM_CLEANUP (стр. 99).

A.3.2. RDS_COMPMODULEDATA – структура данных модуля

Структура данных модуля автокомпиляции RDS_COMPMODULEDATA содержит основные параметры этого модуля, которые нужны его функции (A.3.1) для реакции на различные события. Эта структура создается РДС перед первым вызовом функции модуля автокомпиляции и хранится в памяти все время его существования, то есть до тех пор, пока последняя модель, обслуживаемая этим модулем, не будет отключена от последнего связанного с ней блока схемы. Общая структура данных, используемых для автоматической компиляции моделей, подробно рассмотрена в главе 4.

```

typedef struct {
    RDS_COMPHANDLE Module;    // Идентификатор модуля
    LPVOID ModuleData;        // Адрес личной области данных модуля
    LPSTR DllFullPath;        // Полный путь к DLL модуля
    LPSTR DllFuncName;        // Имя функции DLL модуля
    int NModels;              // Число моделей, связанных в данный
                             // момент с этим модулем
} RDS_COMPMODULEDATA;
typedef RDS_COMPMODULEDATA *RDS_PCOMPMODULEDATA;

```

Поля структуры:

Module

Уникальный идентификатор данного модуля. Тип RDS_COMPHANDLE используется в РДС для указания на конкретный модуль автокомпиляции.

ModuleData

Указатель на личную область данных модуля. Перед самым первым вызовом функции данного модуля РДС записывает в это поле значение NULL и больше к нему не обращается. Если модулю требуется отводить память под свои нужды, он может записать указатель на отведенную область памяти в это поле. В этом случае освободить отведенную область тоже должна будет функция модуля – обычно это делается в реакции на событие RDS_COMPM_CLEANUP (стр. 99).

DllFullPath

Полный путь к DLL, в которой находится функция данного модуля автокомпиляции. Может использоваться для считывания каких-либо параметров, хранящихся в файлах в одной папке с DLL модуля. Функция модуля не должна изменять эту строку.

DllFuncName

Экспортированное имя функции данного модуля автокомпиляции. Функция модуля не должна изменять эту строку.

NModels

Общее число моделей, подключенных в данный момент к этому модулю автокомпиляции. Вместе с сервисной функцией `rdscompGetModelData` (стр. 575) может использоваться для организации цикла по всем моделям, обслуживаемым данным модулем. Функция модуля не должна изменять это число.

См. также:

`RDS_COMPM_INIT` (стр. 103), `RDS_COMPM_CLEANUP` (стр. 99),
`rdscompGetModelData` (стр. 575).

A.3.3. `RDS_COMPMODELDATA` – структура данных модели

Структура данных автокомпилируемой модели `RDS_COMPMODELDATA` содержит основные параметры этой модели, которые нужны функции обслуживающего ее модуля (A.3.1) для реакции на различные события. Эта структура создается перед первым подключением данной модели к какому-либо блоку схемы и хранится в памяти до тех пор, пока эта модель не будет отключена от последнего связанного с ней блока. Общая структура данных, используемых для автоматической компиляции моделей, подробно рассмотрена в главе 4.

```
typedef struct {
    RDS_MODELHANDLE Model;      // Идентификатор модели
    LPSTR ModelName;           // Имя модели
    LPSTR ModelNameUC;         // Имя модели в верхнем регистре
    LPVOID ModelData;          // Адрес области данных модели
    int NBlocks;               // Число блоков, связанных
                               // с данной моделью
    RDS_COMPHANDLE Module;     // Идентификатор обслуживающего
                               // модуля автокомпиляции

    LPSTR CompDllName;         // DLL скомпилированной модели
    LPSTR CompDllFunc;         // Имя функции скомпилированной модели
    BOOL Valid;                // Признак необходимости перекомпиляции

    LPSTR AltModelName;        // Альтернативное имя модели
    int Tag;                   // Пользовательское поле
} RDS_COMPMODELDATA;
typedef RDS_COMPMODELDATA *RDS_PCOMPMODELDATA;
```

Поля структуры:

`Model`

Уникальный идентификатор данной модели. Тип `RDS_MODELHANDLE` используется в РДС для указания конкретной автоматически компилируемой модели, с которой производится то или иное действие.

`ModelName`

Указатель на строку с именем данной модели во внутренней памяти РДС. Имя модели должно однозначно идентифицировать эту модель с точки зрения пользователя – именно это имя он указывает в окне параметров блока, когда подключает к нему автоматически компилируемую модель. Имена моделей, отличающиеся только регистром символов, считаются в РДС одинаковыми. Если модуль автокомпиляции хранит тексты своих моделей в отдельных файлах, в качестве имени модели может использоваться имя файла с ее текстом. Функция модуля не должна изменять эту строку, для переименования модели следует использовать сервисную функцию `rdscompRenameModel` (стр. 577).

ModelNameUC

Указатель на строку с именем данной модели, приведенным к верхнему регистру, во внутренней памяти РДС. Поскольку в РДС имена моделей, отличающиеся только регистром, считаются одинаковыми, эту строку можно использовать для поиска модели по имени. Функция модуля не должна изменять эту строку, для переименования модели следует использовать сервисную функцию `rdscmpRenameModel` (стр. 577).

ModelData

Указатель на личную область данных этой модели. Перед подключением данной модели к самому первому блоку схемы РДС записывает в это поле значение `NULL` и больше к нему не обращается. Если модулю требуется отводить память для хранения параметров каждой из своих моделей, он может записать указатель на отведенную область памяти в это поле при инициализации модели, то есть в момент реакции на событие `RDS_COMPM_MODELINIT` (стр. 105). В этом случае освободить отведенную область тоже должна будет функция модуля – обычно это делается перед отключением модели от последнего связанного с ней блока в реакции на событие `RDS_COMPM_MODELCLEANUP` (стр. 104).

NBlocks

Число блоков, к которым в данный момент подключена эта модель. Вместе с сервисной функцией `rdscmpGetModelBlock` (стр. 574) может использоваться для перебора всех таких блоков. Функция модуля автокомпиляции не должна изменять это число.

Module

Идентификатор модуля автокомпиляции, к которому относится данная модель. Функция модуля автокомпиляции не должна изменять это поле.

CompDllName

Путь к файлу DLL, в котором будет находиться функция данной модели после компиляции. Строка пути размещается во внутренней памяти РДС. Функция модуля не должна изменять это поле непосредственно: путь к файлу DLL устанавливается сервисной функцией `rdscmpSetModelFunction` (стр. 582) при вызове модуля автокомпиляции для реакции на событие `RDS_COMPM_PREPARE` (стр. 107).

CompDllFunc

Имя экспортированной функции модели блока, которая будет создана в результате компиляции данной модели. Строка имени размещается во внутренней памяти РДС. Функция модуля не должна изменять это поле непосредственно: как и имя файла DLL (`CompDllName`), имя функции устанавливается сервисной функцией `rdscmpSetModelFunction` при вызове модуля автокомпиляции для реакции на событие `RDS_COMPM_PREPARE`.

Valid

Признак необходимости компиляции данной модели: функция модуля, реагируя на событие `RDS_COMPM_PREPARE`, записывает в это поле `TRUE`, если модель не изменилась с момента прошлой компиляции и повторная компиляция не требуется. Если модель изменилась, в это поле необходимо записать `FALSE`. Чаще всего необходимость компиляции устанавливают, сравнивая время последнего изменения текста модели с временем создания файла DLL.

AltModelName

Указатель на строку с альтернативным именем модели во внутренней памяти РДС. Функция модуля не должна изменять это поле непосредственно: альтернативное имя

модели устанавливается сервисной функцией `rdscompSetAltModelName` (стр. 580) и запоминается в файле схемы вместе с остальными параметрами блока. В отличие от основного имени модели, на которое указывает поле `ModelName`, альтернативное имя модели пользователь не видит и установить или изменить его вручную не может. Фактически, альтернативное имя модели представляет собой хранимую строку, которую модуль автокомпиляции может использовать в своих целях любым удобным программисту образом.

Tag

Целое поле, никак не обрабатываемое РДС. Программист может использовать его любым удобным способом.

См. также:

`RDS_COMPM_MODELINIT` (стр. 105), `RDS_COMPM_MODELCLEANUP` (стр. 104),
`RDS_COMPM_PREPARE` (стр. 107), `rdscompSetAltModelName` (стр. 580),
`rdscompSetModelFunction` (стр. 582), `rdscompRenameModel` (стр. 577).

А.3.4. События модуля автокомпиляции

А.3.4.1. `RDS_COMPM_ATTACHBLOCK` – подключение модели к блоку

Первый параметр функции модуля (`int CallMode`):

Константа `RDS_COMPM_ATTACHBLOCK`.

Третий параметр функции модуля (`void *ExtParam`):

Указатель на структуру `RDS_COMPBLOCKOPDATA`, содержащую идентификатор блока, идентификатор подключаемой к нему модели и причину подключения.

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие `RDS_COMPM_ATTACHBLOCK` возникает при подключении очередного блока к модели, то есть при включении в параметрах этого блока автоматической компиляции его модели с указанием данного модуля автокомпиляции в качестве обслуживающего модель. Подключение блока к модели происходит при указании имени модели в окне параметров, при загрузке схемы с блоками, модели которых компилируются автоматически, при вставке таких блоков из буфера обмена и т.п, то есть во всех случаях, когда РДС устанавливает связь между автокомпилируемой моделью и блоком. Обычно в реакции на это событие параметры блока приводятся в соответствие с моделью, например, блоку назначается указанная в модели структура статических переменных. Пример реакции модуля на событие `RDS_COMPM_ATTACHBLOCK` приведен в §4.4.

В параметре `ExtParam` при реакции на это событие передается указатель на структуру `RDS_COMPBLOCKOPDATA` (эта же структура используется и в реакции на событие `RDS_COMPM_DETACHBLOCK`, стр. 101):

```
typedef struct {  
    RDS_PCOMPMODELDATA Model;           // Данные модели  
    RDS_BHANDLE Block;                  // Идентификатор блока  
    int AttachReason;                   // Причина подключения блока  
                                        // (константа RDS_COMP_AR_*)  
} RDS_COMPBLOCKOPDATA;  
typedef RDS_COMPBLOCKOPDATA *RDS_PCOMPBLOCKOPDATA;
```

Поля структуры:

Model

Указатель на данные модели, подключаемой к блоку в реакции на RDS_COMPM_ATTACHBLOCK или отключаемой от него в реакции на RDS_COMPM_DETACHBLOCK.

Block

Уникальный идентификатор блока, изменение параметров которого вызвало это событие.

AttachReason

Причина подключения модели к блоку (только в реакции на RDS_COMPM_ATTACHBLOCK). В этом поле может находиться одна из следующих констант:

RDS_COMP_AR_LOADSYSTEM	Модель подключается к блоку в процессе загрузки всей схемы.
RDS_COMP_AR_LOADCLIPBRD	Модель подключается к новому блоку, вставляемому из буфера обмена.
RDS_COMP_AR_LOADFROMFILE	Модель подключается к новому блоку, вставляемому из файла или библиотеки.
RDS_COMP_AR_LOADUNDO	Имя подключенной к блоку модели изменилось из-за отмены изменений пользователем или возврата ранее отмененных изменений.
RDS_COMP_AR_MANUALSET	Новая модель задана для блока пользователем в окне параметров или вызовом одной из сервисных функций – например, <code>rdscmpSetBlockModel</code> (стр. 580).
RDS_COMP_AR_RENAMEMODEL	Модель переименована – производится подключение всех обслуживавшихся ей блоков к модели с новым именем.
RDS_COMP_AR_UNKNOWN	Прочие причины изменения имени автокомпилируемой модели блока.

См. также:

RDS_COMPM_DETACHBLOCK (стр. 101), `rdscmpSetBlockModel` (стр. 580), `rdscmpAttachDifferentModel` (стр. 572), `rdscmpRenameModel` (стр. 577).

A.3.4.2. RDS_COMPM_CANATTACHBLK – проверка возможности подключения модели к блоку

Первый параметр функции модуля (**int CallMode**):

Константа RDS_COMPM_CANATTACHBLK.

Третий параметр функции модуля (**void *ExtParam**):

Указатель на структуру RDS_COMPCANATTACHBLKDATA, содержащую идентификатор блока, имя модели, причину подключения и другую информацию.

Возвращаемое функцией модуля значение:

RDS_COMPR_DONE	Подключение данной модели к данному блоку возможно.
RDS_COMPR_ERROR	Подключение данной модели к данному блоку невозможно, выводится сообщение об ошибке.

RDS_COMPR_ERRORNOMSG Подключение данной модели к данному блоку невозможно, сообщение об ошибке не выводится.

Событие RDS_COMPM_CANATTACHBLK возникает непосредственно перед подключением автокомпилируемой модели к блоку. Данные модели в этот момент еще не созданы и идентификатора для нее еще нет, поэтому в структуре параметров события RDS_COMPCANATTACHBLKDATA передается только имя модели. Реагируя на этот вызов, функция модуля должна проверить принципиальную возможность подключения данной модели к данному блоку. Например, модель, использующая статические переменные, не может подключаться к подсистемам, внешним входам/выходам и вводам шин, поскольку их структура переменных устроена не так, как у простых блоков. Если подключение модели к блоку невозможно, функция либо возвращает константу RDS_COMPR_ERROR или RDS_COMPR_ERRORNOMSG, либо вызывает сервисную функцию rdscompAttachDifferentModel (стр. 572), сообщая РДС, что, хотя модель с данными именем подключить нельзя, вместо нее можно подключить модель с другим именем. Пример реакции модуля на это событие приведен в §4.3.

В параметре ExtParam при реакции на событие RDS_COMPM_CANATTACHBLOCK передается указатель на структуру RDS_COMPCANATTACHBLKDATA:

```
typedef struct {
    LPSTR ModelName;      // Имя модели
    LPSTR ModelNameUC;    // Имя модели в верхнем регистре
    LPSTR AltModelName;   // Альтернативное имя модели
    RDS_BHANDLE Block;    // Идентификатор блока
    int AttachReason;     // Причина подключения (RDS_COMP_AR_*)
    BOOL ChangeModel;     // Возврат: подключить другую
                        // модель вместо этой
} RDS_COMPCANATTACHBLKDATA;
typedef RDS_COMPCANATTACHBLKDATA *RDS_PCOMPCANATTACHBLKDATA;
```

Поля структуры:

ModelName

Указатель на строку во внутренней памяти РДС, содержащую имя подключаемой к блоку модели в том виде, в каком его указал пользователь. Функция модуля не должна изменять это поле.

ModelNameUC

Указатель на строку во внутренней памяти РДС, содержащую имя подключаемой к блоку модели в верхнем регистре. Функция модуля не должна изменять это поле.

AltModelName

Указатель на строку во внутренней памяти РДС, содержащую альтернативное имя подключаемой к блоку модели. Функция модуля не должна изменять это поле.

Block

Идентификатор блока, к которому подключается указанная модель.

AttachReason

Причина подключения модели к блоку – одна из констант RDS_COMP_AR_* (см. стр. 96).

ChangeModel

Возвращаемый признак подключения другой модели. Перед вызовом функции модуля для реакции на событие RDS_COMPM_CANATTACHBLK РДС записывает в это поле значение FALSE. Если функция модуля хочет подключить к указанному блоку другую модель вместо той, имя которой передано в поле ModelName, она должна записать в

ChangeModel значение TRUE, указать новое имя подключаемой модели вызовом сервисной функции rdscompAttachDifferentModel, и вернуть значение RDS_COMPR_DONE.

См. также:

RDS_COMPM_ATTACHBLOCK (стр. 95),
rdscompAttachDifferentModel (стр. 572).

A.3.4.3. RDS_COMPM_CANRENAMEMODEL – проверка возможности переименования модели

Первый параметр функции модуля (int CallMode):

Константа RDS_COMPM_CANRENAMEMODEL.

Третий параметр функции модуля (void *ExtParam):

Указатель на структуру RDS_COMPMODELRENAMEDATA, содержащую старое и новое имя модели.

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие RDS_COMPM_CANRENAMEMODEL возникает перед переименованием модели, то есть перед тем, как имя модели будет изменено сервисной функцией rdscompRenameModel (стр. 577). Реагируя на это событие, модуль автокомпиляции может разрешить или запретить переименование модели. Если новое имя модели совпадает с каким-либо уже используемым именем модели в этом же модуле, при определенных параметрах функции rdscompRenameModel переименованная модель может заменить собой модель с совпавшим именем, то есть все блоки, ранее обслуживавшиеся той моделью, теперь будут обслуживаться переименованной.

В параметре ExtParam при реакции на событие RDS_COMPM_CANRENAMEMODEL передается указатель на структуру RDS_COMPMODELRENAMEDATA (эта же структура используется в реакции на событие RDS_COMPM_MODELRENAMED):

```
typedef struct {  
    RDS_PCOMPMODELDATA Model;           // Данные переименовываемой  
                                         // (переименованной) модели  
    LPSTR OldModelName;                 // Старое имя модели  
    LPSTR NewModelName;                 // Новое имя модели  
    BOOL AllowRename;                   // Переименование разрешено  
                                         // (возврат для RDS_COMPM_CANRENAMEMODEL)  
} RDS_COMPMODELRENAMEDATA;  
typedef RDS_COMPMODELRENAMEDATA *RDS_PCOMPMODELRENAMEDATA;
```

Поля структуры:

Model

Указатель на структуру данных модели RDS_PCOMPMODELDATA (см. стр. 93), которая переименовывается (в реакции на RDS_COMPM_CANRENAMEMODEL) или уже переименована (в реакции на RDS_COMPM_MODELRENAMED).

OldModelName

Указатель на строку во внутренней памяти РДС, содержащую имя модели до переименования. Функция модуля не должна изменять это поле.

NewModelName

Указатель на строку во внутренней памяти РДС, содержащую имя модели после переименования. Функция модуля не должна изменять это поле.

AllowRename

Возвращаемый флаг разрешения переименования (только в реакции на RDS_COMPM_CANRENAMEMODEL). По умолчанию в этом поле находится значение TRUE, означающее, что переименование разрешено. Если переименование запрещено, функция модуля должна записать в это поле значение FALSE.

См. также:

RDS_COMPM_MODELRENAMED (стр. 105), rdscompRenameModel (стр. 577).

A.3.4.4. RDS_COMPM_CLEANUP – очистка данных модуля

Первый параметр функции модуля (int CallMode):

Константа RDS_COMPM_CLEANUP.

Третий параметр функции модуля (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие RDS_COMPM_CLEANUP возникает перед выгрузкой модуля автокомпиляции из памяти, то есть после того, как последняя модель, обслуживаемая этим модулем, отключена от последнего использовавшего ее блока. В этой реакции обычно удаляется личная область данных модуля, если она была создана в реакции на событие RDS_COMPM_INIT (стр. 103). Пример реакции модуля на это событие приведен в §4.2.

См. также:

RDS_COMPM_INIT (стр. 103), RDS_COMPMODULEDATA (стр. 92).

A.3.4.5. RDS_COMPM_CLOSEALLWIN – закрытие всех окон

Первый параметр функции модуля (int CallMode):

Константа RDS_COMPM_CLOSEALLWIN.

Третий параметр функции модуля (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модуля значение:

Число окон модуля, оставшихся открытыми.

Событие RDS_COMPM_CLOSEALLWIN возникает при необходимости закрыть все немодальные окна, принадлежащие модулю автокомпиляции (например, перед выгрузкой из памяти всей схемы). Функция модуля должна либо сразу закрыть все такие окна, если это возможно, либо послать этим окнам сообщение, которое должно их закрыть, и вернуть число окон, которые еще открыты. Если функция вернет ненулевое значение, РДС обработает все сообщения, накопившиеся в очереди приложения (включая сообщения, которые должны закрыть окна), и повторит вызов функции модуля с параметром RDS_COMPM_CLOSEALLWIN. Вызов будет повторяться до тех пор, пока функция не вернет нулевое значение, сообщая тем самым об отсутствии открытых окон.

A.3.4.6. RDS_COMPM_COMPILE – компиляция моделей

Первый параметр функции модуля (**int CallMode**):

Константа RDS_COMPM_COMPILE.

Третий параметр функции модуля (**void *ExtParam**):

Указатель на структуру RDS_COMPILEDATA, содержащую список моделей, которые должны быть скомпилированы.

Возвращаемое функцией модуля значение:

RDS_COMPR_DONE Модели скомпилированы.
RDS_COMPR_ERROR Модели не скомпилированы.

Событие RDS_COMPM_COMPILE возникает в тот момент, когда нужно скомпилировать все обслуживаемые модулем модели. Необходимость компиляции проверяется при вызове реакции на другое событие, RDS_COMPM_PREPARE (стр. 107), которая вызывается для каждой модели в отдельности. В отличие от RDS_COMPM_PREPARE, событие RDS_COMPM_COMPILE возникает не для каждой модели, а один раз для всего модуля. Именно в реакции на него обычно выполняются все основные действия модуля: формирование исходного текста программы, вызов компилятора, обработка ошибок компиляции и т.п. Кроме того, если в параметрах модели задается структура статических переменных блока (а, как правило, это необходимо), в этой реакции всем блокам, которым назначена эта модель, присваивается указанная структура переменных. Пример реакции на событие RDS_COMPM_COMPILE приведен в §4.4.

В параметре ExtParam при реакции на событие RDS_COMPM_COMPILE передается указатель на структуру RDS_COMPILEDATA:

```
typedef struct {  
    RDS_PCOMPMODELDATA *InvalidModels; // Модели, которые нужно  
                                         // компилировать  
    int IMCount; // Размер массива InvalidModels  
    BOOL Rebuild; // Принудительная перекомпиляция всех моделей  
} RDS_COMPILEDATA;  
typedef RDS_COMPILEDATA *RDS_PCOMPILEDATA;
```

Поля структуры:

InvalidModels

Массив указателей на структуры данных RDS_PCOMPMODELDATA (см. стр. 93) тех моделей, которые должны быть скомпилированы. Этот массив формируется РДС, функция модуля не должна менять его элементы. Модуль должен скомпилировать модели InvalidModels[0] ... InvalidModels[IMCount-1].

IMCount

Число моделей, которые необходимо скомпилировать, то есть размер массива InvalidModels.

Rebuild

TRUE, если пользователь выбрал принудительную компиляцию всех моделей схемы, FALSE в противном случае. Независимо от значения этого поля, функция модуля должна скомпилировать только те модели, указатели на данные которых находятся в массиве InvalidModels. При принудительной компиляции этот массив будет содержать все модели блоков, обслуживаемые данным модулем.

См. также:

RDS_COMPM_PREPARE (стр. 107), rdscompCompileModel (стр. 573).

A.3.4.7. RDS_COMPM_DETACHBLOCK – отключение модели от блока

Первый параметр функции модуля (int CallMode):

Константа RDS_COMPM_DETACHBLOCK.

Третий параметр функции модуля (void *ExtParam):

Указатель на структуру RDS_COMPBLOCKOPDATA (стр. 95), содержащую идентификатор блока и идентификатор отключаемой от него модели.

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие RDS_COMPM_DETACHBLOCK возникает при отключении очередного блока от модели. Такое отключение происходит при выключении в параметрах этого блока автоматической компиляции, перед подключением к нему другой модели, при удалении блока и т.п., то есть во всех случаях, когда РДС разрывает связь между автокомпилируемой моделью и блоком.

Как и при реакции на событие RDS_COMPM_ATTACHBLOCK (стр. 95), в параметре ExtParam при реакции на это событие передается указатель на структуру RDS_COMPBLOCKOPDATA, однако в данной реакции поле структуры AttachReason не используется.

См. также:

RDS_COMPM_ATTACHBLOCK (стр. 95), RDS_COMPBLOCKOPDATA (стр. 95),
rdscompSetBlockModel (стр. 580),
rdscompAttachDifferentModel (стр. 572), rdscompRenameModel (стр. 577).

A.3.4.8. RDS_COMPM_EXECFUNCTION – реакция на действия пользователя

Первый параметр функции модуля (int CallMode):

Константа RDS_COMPM_EXECFUNCTION.

Третий параметр функции модуля (void *ExtParam):

Указатель на структуру RDS_COMPEXECFUNCDATA, содержащую команду пользователя и дополнительные данные.

Возвращаемое функцией модуля значение:

RDS_COMPR_DONE Команда пользователя выполнена.
RDS_COMPR_ERROR При выполнении команды возникли ошибки.

Событие RDS_COMPM_EXECFUNCTION возникает при различных действиях пользователя на вкладке “Компиляция” окна параметров блока. На момент этого события модель может быть еще не подключена к блоку, и действия пользователя, вызвавшие это событие, могут быть направлены на подключение новой модели. Эта реакция вызывается при нажатии пользователем кнопок “Обзор”, “Сохранить как” и т.п., в результате функция модуля может вернуть в РДС измененное имя модели вызовом rdscompReturnModelName (стр. 579). Пример реакции на это событие приведен в §4.3.

В параметре ExtParam при реакции на событие RDS_COMPM_EXECFUNCTION передается указатель на структуру RDS_COMPEXECFUNCDATA:

```
typedef struct {
    int Function;           // Действие (RDS_COMPFLAG_FUNC*)
    LPSTR ModelName;        // Содержимое строки имени модели
    RDS_HOBJECT BlockVars;  // Переменные блока
    int BlockType;          // Тип блока (RDS_BT* или RDS_TUNKNOWN)
} RDS_COMPEXECFUNCDATA;
typedef RDS_COMPEXECFUNCDATA *RDS_PCOMPEXECFUNCDATA;
```

Поля структуры:

Function

Одна из констант, описывающих действие, совершенное пользователем (они частично совпадают с флагами, возвращаемыми функцией модуля при реакции на событие RDS_COMPM_GETOPTIONS, см. стр. 103):

RDS_COMPFLAG_FUNCMODEL BrowSE	Пользователь нажал кнопку “Обзор...”
RDS_COMPFLAG_FUNCMODEL CREATE	Пользователь нажал кнопку “Новый...”
RDS_COMPFLAG_FUNCMODEL SAVEAS	Пользователь нажал кнопку “Сохранить как...”
RDS_COMPFLAG_FUNCMODEL USERINPUT	Пользователь ввел текст имени модели вручную (вызывается в момент закрытия окна параметров блока кнопкой “ОК”).

ModelName

Указатель на строку с именем модели на вкладке “Компиляция” окна параметров блока. Это текст во внутренней памяти РДС, функция модуля не может изменять его значение. Если в результате реакции необходимо изменить этот текст (например, если пользователь выбрал новое имя модели кнопкой “Обзор”), необходимо вызвать сервисную функцию rdscompReturnModelName.

BlockVars

Вспомогательный объект (см. стр. 430), содержащий структуру переменных текущего блока. Это поле может использоваться только в реакции на нажатие кнопки “Новый”, то есть при Function==RDS_COMPFLAG_FUNCMODELCREATE).

BlockType

Тип блока, в окне параметров которого производятся действия (см. стр. 113):

RDS_BTSYSTEM	Подсистема.
RDS_BTSIMPLEBLOCK	Простой блок.
RDS_BTINPUTBLOCK	Внешний вход.
RDS_BTOUTPUTBLOCK	Внешний выход.
RDS_BTBUSPORT	Ввод шины.

Для всех действий, кроме ввода имени модели вручную (RDS_COMPFLAG_FUNCMODELUSERINPUT), функция модуля вызывается немедленно при нажатии пользователем соответствующей кнопки. При ручном редактировании строки имени модели этот факт запоминается, а затем функция модуля вызывается при закрытии окна параметров блока кнопкой “ОК” – в этот момент модуль может проверить, есть ли модель с введенным вручную именем, и создать ее, если это необходимо.

См. также:

RDS_COMPM_GETOPTIONS (стр. 103), RDS_BLOCKDESCRIPTION (стр. 113), rdscompReturnModelName (стр. 579).

A.3.4.9. RDS_COMPM_GETOPTIONS – описание возможностей модуля

Первый параметр функции модуля (int CallMode):

Константа RDS_COMPM_GETOPTIONS.

Третий параметр функции модуля (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модуля значение:

Набор битовых флагов, описывающих возможности модуля (частично совпадают с константами, используемыми в реакции на событие RDS_COMPM_EXECFUNCTION, см. стр. 101):

RDS_COMPFLAG_FUNCMODEL BrowSE	На вкладке “Компиляция” есть кнопка “Обзор...”
RDS_COMPFLAG_FUNCMODEL CREATE	На вкладке “Компиляция” есть кнопка “Новый...”
RDS_COMPFLAG_FUNCMODEL SAVEAS	На вкладке “Компиляция” есть кнопка “Сохранить как...”
RDS_COMPFLAG_FUNCMODEL USERINPUT	На вкладке “Компиляция” пользователю разрешено вводить имя модели вручную.
RDS_COMPFLAG_CANCHANGE STRUCT	При изменении структур в РДС модуль автокомпиляции может изменять структуры в моделях.

Реакция на событие RDS_COMPM_GETOPTIONS вызывается в момент запроса описания возможностей модуля автокомпиляции. В ответ функция модуля должна вернуть набор битовых флагов, каждый из которых соответствует той или иной поддерживаемой функции. Все функции, кроме RDS_COMPFLAG_CANCHANGE STRUCT, относятся к внешнему виду вкладки “Компиляция” окна параметров блока. Флаг RDS_COMPFLAG_CANCHANGE STRUCT сообщает РДС о том, что если пользователь изменит состав какой-либо из используемых блоками структурных переменных, необходимо вызвать данный модуль, чтобы он внес соответствующие изменения в обслуживаемые им модели, использующие эти структуры.

Кроме возврата флагов, разрешающих кнопки в окне параметров блока, функция модуля может установить текст перед полем ввода имени модели, вызвав для этого сервисную функцию rdscompReturnModelNameLabel (стр. 579).

Пример реакции на это событие приведен в §4.3.

См. также:

RDS_COMPM_EXECFUNCTION (стр. 101),
rdscompReturnModelNameLabel (стр. 579).

A.3.4.10. RDS_COMPM_INIT – инициализация модуля

Первый параметр функции модуля (int CallMode):

Константа RDS_COMPM_INIT.

Третий параметр функции модуля (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие `RDS_COMPM_INIT` возникает перед первым использованием данного модуля автокомпиляции: перед подключением первой обслуживаемой им модели к первому блоку схемы или, если в схеме нет моделей, обслуживаемых этим модулем, перед вызовом его функции настройки. Это всегда самое первое событие, на которое реагирует модуль.

Если модулю нужна личная область памяти для хранения своих параметров (путей к компилятору и библиотекам, настроек и т.п.), обычно она отводится именно в реакции на это событие, и указатель на отведенную область памяти записывается в поле `ModuleData` структуры данных модуля `RDS_COMPMODULEDATA` (стр. 92) – на момент вызова реакции на событие `RDS_COMPM_INIT` эта структура уже создана. Пример реакции функции модуля на это событие и отведения личной области памяти модуля приведен в §4.2.

См. также:

`RDS_COMPM_CLEANUP` (стр. 99), `RDS_COMPMODULEDATA` (стр. 92).

А.3.4.11. `RDS_COMPM_MODECHANGE` – изменение режима РДС**Первый параметр функции модуля (`int CallMode`):**

Константа `RDS_COMPM_MODECHANGE`.

Третий параметр функции модуля (`void *ExtParam`):

Не используется (`NULL`).

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие `RDS_COMPM_MODECHANGE` возникает при любом изменении режима работы РДС (редактирование, моделирование, расчет – см. §1.3). Обычно реакция на это событие используется для разрешения и запрещения различных органов управления в немодальных окнах редактора модели, если модуль создает такие окна (например, можно запретить изменение модели, если РДС не находится в режиме редактирования). Текущий режим работы РДС не передается в параметрах этого события, его можно узнать при помощи вызова сервисных функций `rdsSystemInEditMode` (стр. 175) и `rdsCalcProcessIsRunning` (стр. 148).

См. также:

`RDS_COMPM_OPENEDITOR` (стр. 106), `rdsSystemInEditMode` (стр. 175), `rdsCalcProcessIsRunning` (стр. 148).

А.3.4.12. `RDS_COMPM_MODELCLEANUP` – очистка данных модели**Первый параметр функции модуля (`int CallMode`):**

Константа `RDS_COMPM_MODELCLEANUP`.

Третий параметр функции модуля (`void *ExtParam`):

Указатель на структуру данных модели `RDS_COMPMODELDATA` (стр. 93).

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие `RDS_COMPM_MODELCLEANUP` возникает перед отключением какой-либо модели, обслуживаемой данным модулем, от последнего использовавшего ее блока. После вызова этой реакции структура данных модели уничтожается и модель удаляется из списка моделей модуля. Обычно в реакции на это событие функция модуля освобождает память, отведенную под нужды модели в реакции на событие `RDS_COMPM_MODELINIT` (стр. 105).

В параметре `ExtParam` при реакции на это событие передается указатель на структуру данных отключаемой модели.

См. также:

`RDS_COMPM_MODELINIT` (стр. 105), `RDS_COMPMODELDATA` (стр. 93).

A.3.4.13. `RDS_COMPM_MODELINIT` – инициализация модели

Первый параметр функции модуля (`int CallMode`):

Константа `RDS_COMPM_MODELINIT`.

Третий параметр функции модуля (`void *ExtParam`):

Указатель на структуру данных модели `RDS_COMPMODELDATA` (стр. 93).

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие `RDS_COMPM_MODELINIT` возникает перед подключением какой-либо модели, обслуживаемой данным модулем, к первому блоку схемы. Модуль автокомпиляции на момент этого события уже инициализирован. Обычно в этой реакции функция модуля, если это необходимо, отводит память под нужды модели и записывает указатель на эту отведенную область памяти в поле `ModelData` структуры данных модели `RDS_COMPMODELDATA`, указатель на которую передается в параметре `ExtParam`.

См. также:

`RDS_COMPM_MODELCLEANUP` (стр. 104), `RDS_COMPMODELDATA` (стр. 93).

A.3.4.14. `RDS_COMPM_MODELRENAMED` – модель переименована

Первый параметр функции модуля (`int CallMode`):

Константа `RDS_COMPM_MODELRENAMED`.

Третий параметр функции модуля (`void *ExtParam`):

Указатель на структуру `RDS_COMPMODELRENAMEDATA` (стр. 98), содержащую старое и новое имя модели.

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие `RDS_COMPM_MODELRENAMED` возникает после переименования модели, если оно было разрешено в реакции на событие `RDS_COMPM_CANRENAMEMODEL`. В данной реакции функция модуля уже не может никак вмешаться в переименование, поэтому поле `AllowRename` структуры `RDS_COMPMODELRENAMEDATA`, указатель на которую передается в параметре `ExtParam`, здесь не используется. Реакция на это событие может использоваться для того, чтобы, например, отразить изменение имени модели в окне редактора, или сформировать полный путь к файлу модели по ее имени, если модели хранятся в отдельных файлах.

См. также:

RDS_COMPM_CANRENMODEL (стр. 98), rdscompRenameModel (стр. 577).

A.3.4.15. RDS_COMPM_OPENEDITOR – вызов редактора модели

Первый параметр функции модуля (int CallMode):

Константа RDS_COMPM_OPENEDITOR.

Третий параметр функции модуля (void *ExtParam):

Указатель на структуру RDS_OPENEDITORDATA, содержащую указатель на данные модели и идентификатор блока, через параметры которого дана команда открыть редактор.

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие RDS_COMPM_OPENEDITOR возникает тогда, когда пользователь командует открыть редактор какой-либо модели. Обычно для этого используется контекстное меню блока с автокомпилируемой моделью или кнопка на вкладке “Компиляция” окна параметров такого блока. Реагируя на это событие, функция модуля должна каким-либо образом предоставить пользователю возможность редактировать модель, связанную с выбранным блоком – не важно, будет это модальное или немодальное окно или вызов внешнего приложения. Пример редактора модели, оформленного в виде модального окна, приведен в §4.3.

В параметре ExtParam при реакции на событие RDS_COMPM_OPENEDITOR передается указатель на структуру RDS_OPENEDITORDATA:

```
typedef struct {  
    RDS_PCOMPMODELDATA Model;           // Данные модели  
    RDS_BHANDLE Block;                  // Блок, для которого  
                                        // вызван редактор  
} RDS_OPENEDITORDATA;  
typedef RDS_OPENEDITORDATA *RDS_POPENEDITORDATA;
```

Поля структуры:

Model

Указатель на структуру данных модели RDS_PCOMPMODELDATA (стр. 93), для которой нужно вызвать редактор.

Block

Идентификатор блока, из контекстного меню или окна параметров которого вызван редактор модели. В большинстве случаев модулю автокомпиляции не требуется знать этот идентификатор, поскольку одна модель может быть подключена к нескольким блокам, и не так важно, через параметры какого именно блока вызывается редактор модели. Тем не менее, идентификатор блока может использоваться, например, если в самом редакторе предусмотрена функция подключения другой модели к данному блоку.

См. также:

rdscompOpenBlockModelEditor (стр. 577).

A.3.4.16. RDS_COMPM_PREPARE – подготовка модели к компиляции

Первый параметр функции модуля (**int CallMode**):

Константа RDS_COMPM_PREPARE.

Третий параметр функции модуля (**void *ExtParam**):

Указатель на структуру RDS_COMPPREPAREDATA, содержащую указатель на данные модели.

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Реакция на событие RDS_COMPM_PREPARE вызывается для каждой модели, обслуживаемой данным модулем, для проверки необходимости компиляции этой модели и для подготовки ее к компиляции. Если модель не требуется компилировать (то есть файл DLL, соответствующий этой модели, существует, и текст модели не изменялся с момента последней компиляции), функция модуля должна в этой реакции присвоить полю Valid структуры данных модели RDS_COMPMODELDATA (стр. 93) значение TRUE (исходно туда заносится значение FALSE). Если после реакции на все события RDS_COMPM_PREPARE хотя бы у одной модели поле Valid будет иметь значение FALSE, возникнет событие RDS_COMPM_COMPILE (стр. 100), в реакции на которое функция модуля должна будет скомпилировать модель. Если компиляция моделей производится по команде пользователя “Система | Перекомпилировать все модели”, установка поля Valid будет игнорироваться: независимо от него в параметрах события RDS_COMPM_COMPILE будут указаны все модели, обслуживаемые данным модулем.

Реагируя на событие RDS_COMPM_PREPARE, функция модуля должна, помимо установки поля Valid в структуре данных модели, указать имя файла DLL, который будет создан в результате компиляции, и имя экспортированной функции модели блока в этом файле. Для этого она должна вызвать сервисную функцию rdscompSetModelFunction (стр. 582). Пример реакции на это событие приведен в §4.4.

В параметре ExtParam при реакции на событие RDS_COMPM_PREPARE передается указатель на структуру RDS_COMPPREPAREDATA:

```
typedef struct {  
    RDS_PCOMPMODELDATA Model;           // Данные модели  
    BOOL Rebuild;                       // Принудительная компиляция  
} RDS_COMPPREPAREDATA;  
typedef RDS_COMPPREPAREDATA *RDS_COMPPREPAREDATA;
```

Поля структуры:

Model

Указатель на структуру данных модели RDS_COMPMODELDATA (стр. 93), которую нужно подготовить к компиляции. В этой структуре функция модуля должна присвоить полю Valid значение TRUE, если модель компилировать не нужно.

Rebuild

TRUE, если пользователь выбрал принудительную компиляцию всех моделей схемы, FALSE в противном случае.

См. также:

RDS_COMPM_COMPILE (стр. 100), RDS_COMPMODELDATA (стр. 93),
rdscompSetModelFunction (стр. 582).

A.3.4.17. RDS_COMPM_SAVEBLOCK – сохранение блока

Первый параметр функции модуля (**int CallMode**):

Константа RDS_COMPM_SAVEBLOCK.

Третий параметр функции модуля (**void *ExtParam**):

Указатель на структуру RDS_COMPSAVEBLOCKDATA, содержащую указатель на данные модели, идентификатор блока и другие параметры события.

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие RDS_COMPM_SAVEBLOCK возникает при сохранении каждого блока, которому назначена автоматически компилируемая модель, независимо от причины сохранения блока: сохранение в отдельный файл, сохранение в составе схемы, копирование в буфер обмена и т.д. В реакции на него функция модуля может, при необходимости, изменить имя модели этого блока, присвоив полю ChangeName структуры RDS_COMPSAVEBLOCKDATA, указатель на которую передан в параметре ExtParam, значение TRUE, и вызвав сервисную функцию rdscompAttachDifferentModel (стр. 572). Например, если модели хранятся в отдельных файлах, реакцию на событие RDS_COMPM_SAVEBLOCK можно использовать для замены полного пути к файлу модели на путь относительно файла схемы – это упростит перенос схемы в другую папку, поскольку при этом уже не нужно будет корректировать имена моделей.

В параметре ExtParam при реакции на это событие передается указатель на структуру RDS_COMPSAVEBLOCKDATA:

```
typedef struct {
    RDS_PCOMPMODELDATA Model;           // Данные модели
    RDS_BHANDLE Block;                  // Сохраняемый блок
    int SaveAction;                     // Причина сохранения (RDS_LS_SAVE*)
    BOOL ChangeName;                   // Возврат: изменить имя модели
} RDS_COMPSAVEBLOCKDATA;
typedef RDS_COMPSAVEBLOCKDATA *RDS_PCOMPSAVEBLOCKDATA;
```

Поля структуры:

Model

Указатель на структуру данных модели RDS_COMPMODELDATA (стр. 93), которая связана с сохраняемым блоком.

Block

Уникальный идентификатор сохраняемого блока.

SaveAction

Одна из стандартных констант, указывающих на причину сохранения блока (эти же константы возвращаются при вызове сервисной функции rdsGetSystemInt с параметром RDS_GSISAVELOADACTION, см. стр. 159):

RDS_LS_SAVECONTENT	Запись параметров блока при сохранении схемы.
RDS_LS_SAVEROOT	Запись параметров корневой подсистемы при сохранении схемы.
RDS_LS_SAVECLIPBRD	Копирование блока в буфер обмена.
RDS_LS_SAVETOFILE	Запись одиночного блока в файл или в библиотеку.
RDS_LS_SAVEUNDO	Запись параметров блока в буфер отмены изменений.

RDS_LS_SAVEAUTOCOMP	Запись параметров блока перед автоматической компиляцией его модели (после компиляции они будут загружены обратно).
RDS_LS_SAVETAGGED	Запись блока при поблочной записи схемы (см. §3.5).

ChangeName

Возвращаемый признак изменения имени модели. Исходно в этом поле записано значение FALSE. Если имя модели для сохраняемого блока необходимо изменить, функция модуля должна присвоить этому полю значение TRUE и вызвать сервисную функцию `rdscompAttachDifferentModel` (стр. 572), передав в ее параметрах новое имя модели и, при необходимости, ее новое альтернативное имя (см. стр. 94).

См. также:

RDS_COMPM_SAVESYSTEM (стр. 109), RDS_COMPMODELDATA (стр. 93),
RDS_GSISAVELOADACTION (стр. 159),
`rdscompAttachDifferentModel` (стр. 572).

A.3.4.18. RDS_COMPM_SAVESYSTEM – сохранение схемы

Первый параметр функции модуля (**int CallMode**):

Константа RDS_COMPM_SAVESYSTEM.

Третий параметр функции модуля (**void *ExtParam**):

Указатель на структуру RDS_COMPSAVESYSTEMDATA, содержащую новое и прежнее имена файла сохраняемой схемы.

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие RDS_COMPM_SAVESYSTEM возникает перед сохранением всей схемы, реакция на него вызывается один раз для каждого модуля автокомпиляции. Эта реакция может использоваться, например, для предложения пользователю сделать копии всех моделей для новой схемы, если он сохраняет схему под другим именем.

В параметре ExtParam при реакции на это событие передается указатель на структуру RDS_COMPSAVESYSTEMDATA:

```
typedef struct {
    LPSTR FileName;           // Имя сохраняемого файла
    LPSTR OldFileName;        // Имя файла до сохранения или NULL
                             // при первом сохранении
} RDS_COMPSAVESYSTEMDATA;
typedef RDS_COMPSAVESYSTEMDATA *RDS_PCOMPSAVESYSTEMDATA;
```

Поля структуры:

FileName

Указатель на строку с полным путем к сохраняемому файлу схемы во внутренней памяти РДС. Функция модуля не должна изменять эту строку.

OldFileName

Указатель на строку с прежним именем файла (при прошлом сохранении) или NULL, если только что созданная схема сохраняется в первый раз. Если схема сохраняется в тот же самый файл, что и в прошлый раз, значение поля будет совпадать с полем FileName. Эта строка находится во внутренней памяти РДС, функция модуля не должна изменять ее.

См. также:

RDS_COMPM_SAVEBLOCK (стр. 108).

A.3.4.19. RDS_COMPM_SETUP – настройка модуля автокомпиляции

Первый параметр функции модуля (int CallMode):

Константа RDS_COMPM_SETUP.

Третий параметр функции модуля (void *ExtParam):

Не используется (NULL).

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие RDS_COMPM_SETUP возникает при вызове пользователем настроек модуля автокомпиляции в окне списка модулей (пункт меню “Сервис | Автокомпиляция...”). Обычно в этих настройках указывается путь к внешнему компилятору, способы формирования исходного текста программ и т.п. Реагируя на это событие, функция модуля должна самостоятельно открыть окно, обеспечить пользователю возможность задать все необходимые настройки, сохранить их так, чтобы они считывались при последующих подключениях данного модуля к другим схемам и т.д. Пример использования этого события для настройки модуля приведен в §4.2.

См. также:

RDS_COMPM_INIT (стр. 103).

A.3.4.20. RDS_COMPM_STRUCTCHANGE – изменение структуры

Первый параметр функции модуля (int CallMode):

Константа RDS_COMPM_STRUCTCHANGE.

Третий параметр функции модуля (void *ExtParam):

Указатель на структуру RDS_COMPSTRUCTCHGDATA, содержащую имя изменившейся структуры.

Возвращаемое функцией модуля значение:

Не используется, можно возвращать любое значение.

Событие RDS_COMPM_STRUCTCHANGE возникает при любом изменении типа одной из зарегистрированных в РДС структур, используемых в составе переменных блоков, а также при переименовании или удалении такой структуры. Реакция на это событие вызывается независимо от того, вернула ли функция модуля флаг RDS_COMPFLAG_CANCHANGESTRUCT в реакции на событие RDS_COMPM_GETOPTIONS (стр. 103). Обычно эта реакция используется для корректировки модели блока, если она использует измененную структуру, или для вывода сообщения пользователю о том, что модель стала неработоспособной и о действиях, которые он должен предпринять для того, чтобы эту модель можно было снова использовать.

В параметре ExtParam при реакции на это событие передается указатель на структуру RDS_COMPSTRUCTCHGDATA:

```
typedef struct {  
    LPSTR OldStructType;    // Имя типа структуры до изменения  
    LPSTR NewStructType;    // Имя типа структуры после изменения
```

```

    BOOL Renamed;                // Структура переименована
} RDS_COMPSTRUCTCHGDATA;
typedef RDS_COMPSTRUCTCHGDATA *RDS_PCOMPSTRUCTCHGDATA;

```

Поля структуры:

OldStructType

Указатель на строку с именем типа структуры до редактирования. Под этим именем данная структура была известна модулю до ее изменения. Строка находится во внутренней памяти РДС, функция модуля не должна изменять ее.

NewStructType

Указатель на строку с именем типа структуры после редактирования. Строка находится во внутренней памяти РДС, функция модуля не должна изменять ее.

Renamed

TRUE, если структура была переименована (то есть если OldStructType не совпадает с NewStructType), FALSE в противном случае.

См. также:

RDS_COMPM_GETOPTIONS (стр. 103).

А.4. Структуры РДС

Описываются структуры, используемые в различных сервисных функциях РДС.

А.4.1. RDS_ARRAYACCESSDATA – описание матрицы/массива

Структура RDS_ARRAYACCESSDATA может использоваться для упрощения доступа к массиву или матрице в составе статических или динамических переменных блока. Массивы и матрицы в РДС устроены одинаково (массив – это матрица с единственной строкой), поэтому эта структура описывает и массивы, и матрицы. Пример ее использования приведен в §2.5.3.

Работать с массивами и матрицами можно и без этой структуры, разбирая дерево переменных блока вручную или при помощи специальных макросов (стр. 337).

```
typedef struct {  
    BOOL Exists;           // TRUE - массив существует (то есть не 0x0)  
    int Rows, Cols;        // Размерность массива  
    int ItemSize;          // Размер элемента массива  
    LPVOID Data;           // Начало области данных массива  
} RDS_ARRAYACCESSDATA;  
typedef RDS_ARRAYACCESSDATA *RDS_PARRAYACCESSDATA;
```

Поля структуры:

Exists

TRUE, если матрица (массив) существует, то есть имеет хотя бы один элемент, и FALSE, если она пуста.

Rows

Число строк в матрице (массиве), если она не пуста. Для непустого массива в этом поле всегда записана единица.

Cols

Число столбцов в матрице (массиве).

ItemSize

Размер одного элемента матрицы или массива в байтах. Элементы массивов и матриц хранятся в памяти последовательно, поэтому это поле можно использовать для определения указателя на элемент с нужным номером (следует помнить, что в РДС нумерация элементов массивов и матриц начинается с нуля). Например, чтобы получить смещение в байтах от начала массива к элементу с индексом 8 (то есть к его девятому элементу), нужно умножить ItemSize на восемь.

Data

Указатель на нулевой (самый первый) элемент массива или матрицы. Начиная с этого указателя, элементы массива хранятся последовательно, элементы матрицы – последовательно по строкам.

Поле Data структуры RDS_ARRAYACCESSDATA имеет тип void*, то есть “указатель общего вида”, поэтому для работы с элементами массива или матрицы его нужно привести к нужному типу. Например, для матрицы вещественных чисел двойной точности (double) обращение к элементу с индексом строки 2 и индексом столбца 3 будет выглядеть так:

```
RDS_ARRAYACCESSDATA adata;  
// ... здесь должно быть заполнение структуры adata  
// какой-либо сервисной функцией ...  
double a_2_3=((double*)(adata.Data))[2*adata.Cols+3];
```


См. также:

rdsGetVarArrayAccessData (стр. 343), RDS_ARRAYITEMADDR (стр. 340),
RDS_ARRAYITEM (стр. 339).

A.4.2. RDS_BLOCKDESCRIPTION – описание блока

Структура RDS_BLOCKDESCRIPTION используется для получения описания общих параметров блока. Для описания размеров и положения блока используется другая структура – RDS_BLOCKDIMENSIONS (стр. 117). Чаще всего для заполнения структуры RDS_BLOCKDESCRIPTION вызывается функция rdsGetBlockDescription (стр. 220), но и многие другие сервисные функции, возвращающие идентификаторы блоков, позволяют одновременно заполнить и структуру описания возвращаемого блока. Примеры использования этой структуры приведен в §2.6.3, §2.12.1 и др.

```
typedef struct {
    DWORD servSize;      // Размер этой структуры в байтах
    RDS_BHANDLE Block;   // Идентификатор блока
    int BlockType;       // Тип блока
    LPSTR BlockName;     // Имя блока
    RDS_BHANDLE Parent;  // Идентификатор родительской подсистемы
    LPSTR ParentName;    // Имя родительской подсистемы
    LPSTR BlockComment;  // Комментарий блока
    int LayerId;         // Идентификатор слоя
    LPSTR DllFile;       // Имя файла DLL или NULL
    LPSTR DllFunc;       // Имя функции DLL или NULL
    HINSTANCE Module;    // Загруженный модуль DLL или NULL
    LPSTR ParentVar;     // Имя переменной подсистемы, которая
                        // соответствует этому блоку-входу или
                        // выходу
    BOOL Selected;       // Блок выделен в редакторе
    int NumberOfVars;    // Число переменных блока
    DWORD Flags;         // флаги описания (RDS_BDF_*)
    DWORD ExtId;         // Внешний уникальный идентификатор
    int NamePos;         // Положение имени блока (RDS_BDNP_*)
    int NameDx,          // Смещение левого верхнего угла имени от
    NameDy;              // точки привязки блока (в масштабе 100%)
    int AltNameAlignment; // Выравнивание текста, выводимого
                        // вместо имени (если он есть)
} RDS_BLOCKDESCRIPTION;
typedef RDS_BLOCKDESCRIPTION *RDS_PBLOCKDESCRIPTION;
```

Поля структуры:

servSize

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю servSize необходимо присвоить значение sizeof(RDS_BLOCKDESCRIPTION).

Block

Уникальный идентификатор блока.

BlockType

Тип блока – одна из констант RDS_BT* или RDS_TUNKNOWN:

RDS_TUNKNOWN Неизвестно (при вызове функции, заполняющей эту структуру, возникла ошибка).

RDS_BTSYSTEM Подсистема (включая корневую).

RDS_BTSIMPLEBLOCK	Простой блок.
RDS_BTINPUTBLOCK	Внешний вход.
RDS_BTOUTPUTBLOCK	Внешний выход.
RDS_BTBUSPORT	Ввод шины.
RDS_BTDLLBLOCK	Простой блок (синоним RDS_BTSIMPLEBLOCK).

BlockName

Указатель на строку с именем блока в его родительской подсистеме. Строка находится во внутренней памяти РДС, ее нельзя изменять. Для переименования блоков используется сервисная функция `rdsRenameBlock` (стр. 241).

Parent

Идентификатор подсистемы, родительской по отношению к данному блоку, то есть подсистемы, непосредственно внутри которой находится данный блок. У корневой подсистемы, которая не имеет родительской, в этом поле находится значение `NULL`.

ParentName

Указатель на строку с именем родительской подсистемы. Строка находится во внутренней памяти РДС, ее нельзя изменять. У корневой подсистемы, которая не имеет родительской, это поле указывает на пустую строку.

BlockComment

Указатель на строку с комментарием блока. Строка находится во внутренней памяти РДС, ее нельзя изменять. Комментарий блока можно изменить функцией `rdsSetBlockComment` (стр. 243).

LayerId

Идентификатор слоя подсистемы, на котором находится данный блок. Для корневой подсистемы, которая не находится в какой-либо другой подсистеме, значение не определено. Для перемещения блока на другой слой можно использовать функцию `rdsSetBlockLayer` (стр. 244).

DllFile

Указатель на строку с именем файла DLL, в котором находится модель блока, или `NULL`, если у блока нет модели. Строка находится во внутренней памяти РДС, ее нельзя изменять. Для подключения к блоку другой модели следует использовать функцию `rdsSetBlockModel` (стр. 245).

DllFunc

Указатель на строку с именем экспортированной из DLL функции модели блока, или `NULL`, если у блока нет модели. Строка находится во внутренней памяти РДС, ее нельзя изменять. Для подключения к блоку другой модели следует использовать функцию `rdsSetBlockModel`.

Module

Дескриптор загруженного модуля DLL с моделью блока или `NULL`, если у блока нет модели.

ParentVar

Для внешних входов и выходов – указатель на строку с именем статической переменной родительской подсистемы, которая соответствует данному блоку. Для всех остальных типов блоков – `NULL`. Эта строка находится во внутренней памяти РДС, ее нельзя изменять.

Selected

`TRUE`, если данный блок выделен в подсистеме (только в режиме редактирования). `FALSE`, если блок не выделен, или если РДС находится в режимах моделирования или

расчета (выделение блоков при переходе в эти режимы сбрасывается), или если окно подсистемы с этим блоком закрыто. Для программного выделения блока или для снятия выделения можно использовать функцию `rdsSelectBlock` (стр. 241).

NumberOfVars

Число статических переменных блока.

Flags

Один или несколько объединенных битовым ИЛИ флагов описания параметров блока (большая их часть задается различными флагами в окне параметров блока):

RDS_BDF_ALLOWRESIZE	Пользователю разрешено изменение размеров блока путем перетаскивания мышью одного из восьми маркеров выделения.
RDS_BDF_FREEMOUSEMOVE	Функция модели блока вызывается при перемещении курсора мыши над изображением блока (событие RDS_BFM_MOUSEMOVE, стр. 68) даже тогда, когда ни одна из кнопок не нажата (только при установленном флаге RDS_BDF_MOUSEEVENTS).
RDS_BDF_HASPICTURE	У блока есть векторная картинка. Наличие этого флага никак не связано со способом изображения внешнего вида блока – для блока может быть задана векторная картинка, но изображаться он при этом может программно. Способ изображения внешнего вида блока определяются флагами RDS_BDF_SELFDRAW и RDS_BDF_TEXTRECT.
RDS_BDF_KBDEVENTS	Функции блока разрешена реакция на клавиатуру (события RDS_BFM_KEYDOWN – стр. 60 и RDS_BFM_KEYUP – стр. 62).
RDS_BDF_LOCKHEIGHT	При изменении размеров блока пользователем запрещено изменение его высоты (только при установленном флаге RDS_BDF_ALLOWRESIZE).
RDS_BDF_LOCKWIDTH	При изменении размеров блока пользователем запрещено изменение его ширины (только при установленном флаге RDS_BDF_ALLOWRESIZE).
RDS_BDF_MOUSEEVENTS	Функция блока вызывается при нажатии и отпускании кнопок мыши и перемещении курсора с нажатыми кнопками, если курсор находится над изображением блока (события RDS_BFM_MOUSEDOWN – стр. 67, RDS_BFM_MOUSEUP – стр. 69, RDS_BFM_MOUSEDBLCLICK – стр. 64, RDS_BFM_MOUSEMOVE – стр. 68).
RDS_BDF_NAMEOFF	Изображение имени блока в окне подсистемы отключено. Если изображение имен блоков отключено для всей подсистемы, они не будут изображаться независимо от состояния этого флага.
RDS_BDF_POPUPHINT	Функция блока вызывается для вывода всплывающей подсказки (событие RDS_BFM_POPUPHINT, стр. 70).
RDS_BDF_RUNEVERYCYCLE	Функция модели блока (событие RDS_BFM_MODEL, стр. 40) вызывается в каждом такте расчета независимо от состояния сигнала запуска (первого сигнального входа) этого блока.
RDS_BDF_SELFDRAW	Изображение блока рисуется его функцией модели при

	реакции на событие RDS_BFM_DRAW (стр. 57). Этот флаг не может быть установлен одновременно с флагом RDS_BDF_TEXTRECT.
RDS_BDF_SETUPBYDCLICK	Двойной щелчок на изображении блока вызывает функцию его настройки (событие RDS_BFM_SETUP, стр. 71). Если этот флаг сброшен, двойной щелчок будет обрабатываться РДС и приводить к открытию либо окна параметров блока, либо редактора модели (для блоков с автокомпилируемой моделью, см. стр. 106).
RDS_BDF_SETUPFUNC	У блока есть функция настройки (его функция может реагировать на событие RDS_BFM_SETUP, стр. 71).
RDS_BDF_SHOWMAINPOINT	Для блока с векторной картинкой в окне подсистемы изображается точка привязки (начало координат этой картинки). Если изображение точек привязки отключено для всей подсистемы, они не будут изображаться независимо от состояния этого флага.
RDS_BDF_TEXTRECT	Блок изображается прямоугольником с текстом. Этот флаг не может быть установлен одновременно с флагом RDS_BDF_SELFDRAW.

ExtId

Уникальный в пределах схемы целый идентификатор данного блока, не изменяющийся при сохранении и последующей загрузке этой схемы. Такие идентификаторы используются при управлении РДС из другого приложения (см. §3.5).

NamePos

Положение имени блока, если оно отображается в окне подсистемы (одна из констант RDS_BDNP_*):

RDS_BDNP_BELOW	Имя изображается под блоком по центру.
RDS_BDNP_ABOVE	Имя изображается над блоком по центру.
RDS_BDNP_CUSTOM	Нестандартное положение имени — оно было перетасщено пользователем вручную.

NameDx, NameDy

Горизонтальное (NameDx) и вертикальное (NameDy) смещения левого верхнего угла отображаемого имени блока относительно точки привязки изображения этого блока в масштабе 100%. Точкой привязки изображения блока считается начало координат его векторной картинки, если он изображается картинкой, или левый верхний угол занимаемой им прямоугольной области во всех остальных случаях. Горизонтальная ось координат направлена вправо, вертикальная — вниз, начало координат — левый верхний угол рабочего поля.

AltNameAlignment

Выравнивание текста, выводимого вместо имени блока, если он был задан вызовом функции rdsSetBlockAltNameText (стр. 242):

RDS_ALTBLKNAME_LEFT (-1)	Выравнивание по левому краю.
RDS_ALTBLKNAME_CENTER (0)	Выравнивание по центру.
RDS_ALTBLKNAME_RIGHT (1)	Выравнивание по правому краю.

См. также:

`rdsGetBlockDescription` (стр. 220), `rdsRenameBlock` (стр. 241),
`rdsSetBlockComment` (стр. 243), `rdsSetBlockLayer` (стр. 244),
`rdsSetBlockModel` (стр. 245), `rdsSelectBlock` (стр. 241),
`rdsSetBlockAltNameText` (стр. 242), события блоков (стр. 26).

A.4.3. RDS_BLOCKDIMENSIONS – размеры и положение блока или связи

Структура `RDS_BLOCKDIMENSIONS` используется для получения координат и размеров описывающего прямоугольника блока или связи. Положение и размеры блока можно получить при помощи функций `rdsGetBlockDimensions` (стр. 221) и `rdsGetBlockDimensionsEx` (стр. 221), описывающий прямоугольник связи – при помощи функции `rdsGetConnDimensions` (стр. 226). В параметрах этих функций можно указать, нужно ли при вычислении размеров учитывать текущий масштаб подсистемы и значения переменных блока, если они влияют на его размеры и координаты. Примеры использования этой структуры приведены в §2.13.4 и §2.16.2.

```
typedef struct {  
    DWORD servSize;      // Размер этой структуры в байтах  
    int BlockX,BlockY;    // Положение точки привязки  
    int Left,Top;         // Верхний левый угол прямоугольника блока  
    int Width,Height;     // Размеры прямоугольника блока  
} RDS_BLOCKDIMENSIONS;  
typedef RDS_BLOCKDIMENSIONS *RDS_PBLOCKDIMENSIONS;
```

Поля структуры:

`servSize`

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю `servSize` необходимо присвоить значение `sizeof(RDS_BLOCKDIMENSIONS)`.

`BlockX,BlockY`

Горизонтальная (`BlockX`) и вертикальная (`BlockY`) координаты точки привязки изображения блока на рабочем поле подсистемы. Для блоков, изображаемых векторной картинкой, точкой привязки является начало координат этой картинке, для блоков, изображаемых программно или прямоугольником с текстом – левый верхний угол описывающего прямоугольника блока. Горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля.

`Left,Top`

Горизонтальная (`Left`) и вертикальная (`Top`) координаты описывающего прямоугольника блока, то есть прямоугольника минимального размера с горизонтальными и вертикальными сторонами, который целиком покрывает изображение блока. Ни одна точка изображения блока не находится левее координаты `Left` и выше координаты `Top`.

`Width,Height`

Ширина (`Width`) и высота (`Height`) описывающего прямоугольника блока. Ни одна точка изображения блока не находится правее координаты (`Left+Width`) и ниже координаты (`Top+Height`).

См. также:

rdsGetBlockDimensions (стр. 221), rdsGetBlockDimensionsEx (стр. 221),
rdsGetConnDimensions (стр. 226).

A.4.4. RDS_CONNAPPEARANCE – внешний вид связи или шины

Структура RDS_CONNAPPEARANCE используется для получения и установки внешнего вида связей и шин – их цвета, толщины линий, размеров стрелок и т.п. Координаты точек связи или шины, соединяемые связью переменные и каналы шин в этой структуре не отражаются, для их получения и установки следует использовать структуру RDS_CONNDESCRIPTION (стр. 119) и сервисные функции работы со связями (стр. 203). Пример работы со структурой RDS_CONNAPPEARANCE приведен в §2.13.4.

```
typedef struct {  
    DWORD servSize;      // Размер этой структуры в байтах  
    COLORREF LineColor;  // Цвет  
    int LineWidth;       // Толщина линии  
    int LineStyle;       // Стилль линии (WinAPI)  
    int ArrowLength;     // Длина стрелки (0...255)  
    int ArrowWidth;      // Выступ стрелки (0...255)  
    int DotSize;         // Диаметр точки соединения  
} RDS_CONNAPPEARANCE;  
typedef RDS_CONNAPPEARANCE *RDS_PCONNAPPEARANCE;
```

Поля структуры:

servSize

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю servSize необходимо присвоить значение sizeof(RDS_CONNAPPEARANCE).

LineColor

Цвет связи (тип COLORREF, см. стр. 24).

LineWidth

Толщина линии связи в точках экрана в масштабе 100%.

LineStyle

Стилль линии связи – одна из констант, используемых в графических функциях Windows API:

PS_DASH	Пунктирная линия.
PS_DASHDOT	Линия из чередующихся отрезков и точек.
PS_DASHDOTDOT	Линия из повторяющихся групп “отрезок-точка-точка”.
PS_DOT	Линия, состоящая из точек.
PS_INSIDEFRAME	Специальный стилль сплошной линии, разрешающий Windows скорректировать размеры геометрической фигуры, ограниченной этой линией так, чтобы она уместилась в заданный прямоугольник. В РДС этот стилль не используется для рисования связей и не может быть задан пользователем для связи или шины.
PS_NULL	Невидимая линия (этот стилль не может быть задан пользователем для связи или шины).
PS_SOLID	Сплошная линия.

ArrowLength

Длина стрелки на конце связи, соединенном с входом блока, или на конце шины, соединенном с вводом шины. Указывается в точках экрана для масштаба 100%. Принимает значения от 0 (нет стрелки) до 255. Стрелки длиннее 255 точек не поддерживаются.

ArrowWidth

Боковой выступ стрелки на конце связи, соединенном с входом блока, или на конце шины, соединенном с вводом шины. Указывается в точках экрана для масштаба 100%. Принимает значения от 0 (нет стрелки) до 255. Стрелки с выступом больше 255 точек (то есть общей шириной более 510 точек) не поддерживаются.

DotSize

Размер узла связи, то есть диаметр круга, рисуемого в точках разветвления. Указывается в точках экрана для масштаба 100%. Принимает значения от 0 (нет узла) до 255. Узлы диаметром более 255 точек не поддерживаются.

См. также:

rdsGetConnAppearance (стр. 225), rdsSetConnAppearance (стр. 247),
rdsGetConnStyleAppearance (стр. 227),
rdsAltConnAppearanceOp (стр. 204), RDS_CONNDESCRIPTION (стр. 119).

A.4.5. RDS_CONNDESCRIPTION – описание связи или шины

Структура RDS_CONNDESCRIPTION используется для получения описания связи или шины – ее идентификатора, числа точек и линий, составляющих геометрию связи и т.п. Описания внешнего вида, толщины линий и других параметров внешнего вида связи или шины в этой структуре не содержится, для них используется структура RDS_CONNAPPEARANCE (стр. 118). Для анализа соединяемых связей переменных или получения списка каналов шины нужно, как правило, сначала заполнить эту структуру вызовом функции rdsGetConnDescription (стр. 225), а потом уже, зная число точек, линий и каналов, перебирать их по одному при помощи других функций. Пример использования структуры RDS_CONNDESCRIPTION приведен в §2.13.4.

Эта структура используется только для получения описания связей и шин, изменить их параметры с ее помощью нельзя. Для изменения параметров существующих связей или шин и создания новых используется вспомогательный объект, создаваемый функцией rdsCECreateEditor (стр. 408).

```
typedef struct {  
    DWORD servSize;           // Размер этой структуры в байтах  
    RDS_CHANDLE Conn;         // Идентификатор связи  
    int ConnType;             // Тип связи  
    RDS_BHANDLE Parent;       // Идентификатор родительской подсистемы  
    LPSTR ParentName;         // Имя родительской подсистемы  
    BOOL Active;              // Состояние связи (включена/отключена)  
    int LayerId;              // Идентификатор слоя  
    int NumPoints;            // Число точек в связи  
    LPSTR BusName;            // Имя (только для шины)  
    int NumChannels;          // Число каналов (только для шины)  
    int NumLines;             // Число линий  
    DWORD ExtId;              // Внешний уникальный идентификатор  
} RDS_CONNDESCRIPTION;  
typedef RDS_CONNDESCRIPTION *RDS_PCONNDESCRIPTION;
```

Поля структуры:

`servSize`

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю `servSize` необходимо присвоить значение `sizeof(RDS_CONNDESCRIPTION)`.

`Conn`

Уникальный идентификатор данной связи или шины. Для идентификаторов связей и шин в РДС введен специальный тип `RDS_CHANDLE` (см. стр. 23), такие идентификаторы используются в сервисных функциях для указания связи или шины, с которыми нужно произвести то или иное действие.

`ConnType`

Тип объекта: связь или шина. В РДС шины являются подвидом связей, поэтому большинство сервисных функций у шин и связей общие. В этом поле содержится одна из следующих констант:

<code>RDS_TUNKNOWN</code>	Неизвестный тип (при вызове функции, заполняющей эту структуру, возникла ошибка).
<code>RDS_CTCONNECTION</code>	Связь.
<code>RDS_CTBUS</code>	Шина.

`Parent`

Идентификатор родительской подсистемы, то есть подсистемы, непосредственно внутри которой находится данная связь или шина.

`ParentName`

Указатель на строку с именем родительской подсистемы. Строка находится во внутренней памяти РДС, ее нельзя изменять.

`Active`

`TRUE`, если связь включена, и `FALSE`, если выключена. Включение/выключение связей обычно производится пользователем с клавиатуры или через контекстное меню.

`LayerId`

Идентификатор слоя подсистемы, на котором находится данная связь или шина. Для ее перемещения на другой слой можно использовать функцию `rdsSetConnLayer` (стр. 248).

`NumPoints`

Общее число узловых точек в геометрическом изображении связи или шины. В это число входят и точки соединения с блоками, и промежуточные точки (точки излома), и точки разветвления. Получить описание конкретной точки с заданным номером (ее координаты, тип и т.п.) можно при помощи функции `rdsGetPointDescription` (стр. 237).

`BusName`

Указатель на строку с именем шины (только для шин, для связей не используется). Строка находится во внутренней памяти РДС, ее нельзя изменять.

`NumChannels`

Общее число каналов передачи данных в шине (для связей не используется).

`NumLines`

Общее число линий (прямых и кривых Безье), соединяющих пары узловых точек в геометрическом изображении связи или шины. Получить описание линии с заданным

номером (ее тип, координаты касательных кривой и т.п.) можно при помощи функции `rdsGetLineDescription` (стр. 231).

ExtId

Уникальный в пределах схемы целый идентификатор данной связи или шины, не изменяющийся при сохранении и последующей загрузке этой схемы. Такие идентификаторы используются при управлении РДС из другого приложения (см. §3.5).

См. также:

`rdsGetConnDescription` (стр. 225), `rdsGetFirstConn` (стр. 228),
`rdsGetNextConn` (стр. 234), `RDS_CONNAPPEARANCE` (стр. 118),
`rdsSetConnLayer` (стр. 248), `rdsGetPointDescription` (стр. 237),
`rdsGetLineDescription` (стр. 231), `rdsCECreateEditor` (стр. 408).

A.4.6. RDS_DYNVARLINK – подписка на динамическую переменную

Структура `RDS_DYNVARLINK` создается во внутренней памяти РДС при подписке блока на динамическую переменную (см. §2.6) и используется моделью блока для обращения к такой переменной. Модели не требуется самостоятельно создавать эти структуры – она всегда работает с указателями на них, возвращенными сервисными функциями подписки. Структура уничтожается РДС автоматически при прекращении подписки на переменную или удалении подписавшегося блока.

```
typedef struct {  
    LPVOID Data;           // Адрес области данных переменной  
    LPSTR  VarName;        // Имя переменной  
    LPSTR  VarType;        // Тип переменной  
    RDS_BHANDLE Provider;  // Блок-владелец переменной  
    LPVOID UID;           // Служебный идентификатор переменной  
                           // (служебное поле, изменять нельзя)  
    RDS_VHANDLE Var;       // Идентификатор переменной для  
                           // использования в сервисных функциях  
} RDS_DYNVARLINK;  
typedef RDS_DYNVARLINK *RDS_PDYNVARLINK;
```

Поля структуры:

Data

Указатель на область данных переменной. Структура этой области данных в точности соответствует данным статической переменной такого же типа в дереве переменных блока (см. §2.5). Если динамическая переменная не найдена, в этом поле будет находиться значение `NULL`.

VarName

Указатель на строку с именем динамической переменной. Строка находится во внутренней памяти РДС, ее нельзя изменять.

VarType

Указатель на строку типа динамической переменной (она устроена так же, как и строка типа статических переменных). Строка находится во внутренней памяти РДС, ее нельзя изменять.

Provider

Идентификатор блока, в котором находится данная динамическая переменная.

UID

Служебный идентификатор данной переменной, используемый внутри РДС. Значение этого поля нельзя изменять.

Var

Идентификатор переменной (тип RDS_VHANDLE, см. стр. 23), используемый в некоторых сервисных функциях для работы с переменными.

См. также:

rdsSubscribeToDynamicVar (стр. 353),
rdsUnsubscribeFromDynamicVar (стр. 354),
rdsCreateAndSubscribeDV (стр. 347).

A.4.7. RDS_EDITORPARAMETERS – параметры окна подсистемы

Структура RDS_EDITORPARAMETERS используется для получения параметров окна подсистемы – размеров, положения, числа слоев и т.п. – при помощи сервисной функции rdsGetEditorParameters (стр. 258). Примеры использования этой структуры описаны в §2.10.4 и §2.12.5.

```
typedef struct {  
    DWORD servSize;           // Размер этой структуры  
    int GridDx, GridDy;       // Шаг сетки  
    BOOL SnapToGrid;         // Включена привязка к сетке  
    BOOL DisplayGrid;        // Изображается сетка  
    COLORREF GridColor;      // Цвет точек сетки  
    BOOL Visible;            // Окно редактора открыто  
    int WinLeft, WinTop,      // Размеры и положение окна  
        WinWidth, WinHeight;  
    int WorkWidth, WorkHeight; // Размеры рабочей области  
    int ScrollX, ScrollY;     // Позиция полос прокрутки  
    double Zoom;             // Масштаб  
    COLORREF MainPointColor;  // Цвет точек привязки блоков  
    BOOL ShowBlockNames;     // Показывать имена блоков  
    BOOL ShowVarNames;       // Показывать имена переменных  
    int NumLayers;           // Число слоев  
    int CurLayerNum;         // Номер текущего слоя  
    int CurLayerId;          // Идентификатор текущего слоя  
    int NumConfigs;          // Число конфигураций слоев  
    int CurConfig;           // Номер текущей конфигурации  
    BOOL PrintZoneActive;     // Зона печати включена  
    BOOL DisplayPrintZone;    // Зона печати отображается  
    int PZLeft, PZTop,       // Размеры и положение зоны печати  
        PZWidth, PZHeight;  
    BOOL WinMaximized;       // Окно развернуто на весь экран  
    BOOL WinMinimized;       // Окно свернуто  
    DWORD RefreshDelay;      // Задержка автоматического обновления  
                                // окна в режиме расчета  
    COLORREF BlockNameColor;  // Цвет имен блоков  
    int BlockNameDistance;    // Расстояние от блока до имени  
    COLORREF BackgroundColor; // Цвет фона окна  
    BOOL DefBackground;       // В качестве цвета фона  
                                // окна выбран цвет окна Windows  
    DWORD WindowReactions;    // Битовые флаги реакции окна на  
                                // "мышь" и клавиатуру  
    BOOL Wallpaper;          // Включены обои
```

```

    BOOL WallpaperTile;           // Обоями заполнено все рабочее поле
    int WallpaperWidth, WallpaperHeight; // Размеры обоев
    BOOL Dashboard;              // В системе только неподвижные
                                // неперекрывающиеся блоки
} RDS_EDITORPARAMETERS;
typedef RDS_EDITORPARAMETERS *RDS_PEDITORPARAMETERS;

```

Поля структуры:

servSize

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю **servSize** необходимо присвоить значение `sizeof(RDS_EDITORPARAMETERS)`.

GridDx, GridDy

Шаг сетки окна по горизонтали (**GridDx**) и вертикали (**GridDy**). При включенной привязке к сетке влияет на перемещение блоков мышью при редактировании схемы. Шаг сетки указывается без учета масштаба – это именно шаг изменения координат блоков.

SnapToGrid

TRUE, если включена привязка к сетке, и **FALSE**, если выключена.

DisplayGrid

TRUE, если точки сетки изображаются в окне, и **FALSE**, если сетка невидима.

GridColor

Цвет точек сетки (тип **COLORREF**, см. стр. 24).

Visible

TRUE, если окно подсистемы открыто, и **FALSE**, если оно закрыто. Параметры окна подсистемы можно получить и при закрытом окне – они хранятся в РДС и будут применены к окну в момент открытия.

WinLeft, WinTop, WinWidth, WinHeight

Горизонтальная (**WinLeft**) и вертикальная (**WinTop**) координаты левого верхнего угла окна подсистемы, а также его ширина (**WinWidth**) и высота (**WinHeight**) в точках экрана. Положение и размеры окна указываются с учетом его рамки и полосы заголовка, то есть это его внешние размеры. Эти поля можно использовать для привязки каких-либо вспомогательных окон к границам окна подсистемы.

WorkWidth, WorkHeight

Ширина (**WorkWidth**) и высота (**WorkHeight**) рабочего поля подсистемы, то есть пространства, на котором можно размещать блоки и связи. Размер рабочего поля указывается без учета масштаба. Если размеры рабочего поля в текущем масштабе больше размеров внутренней части окна, в окне будут отображаться полосы прокрутки.

ScrollX, ScrollY

Положение полос прокрутки окна в масштабе 100%. Фактически, это координата левой верхней точки рабочего поля, видимой в окне.

Zoom

Текущий масштаб окна в долях единицы: 1 для 100%, 2 для 200%, 0.5 для 50% и т.п.

MainPointColor

Цвет, которым изображается точка привязки картинки блока, если ее отображение разрешено в параметрах блока.

ShowBlockNames

TRUE, если в окне разрешено отображение имен блоков, и FALSE, если оно запрещено. Если отображение имени запрещено в параметрах самого блока, имя не будет показываться в окне независимо от значения этого поля.

ShowVarNames

TRUE, если в окне разрешено отображение имен переменных на концах связей, и FALSE, если оно запрещено. Если отображение имени запрещено в параметрах точки связи, имя не будет показываться в окне независимо от значения этого поля.

NumLayers

Общее число слоев в подсистеме.

CurLayerNum

Порядковый номер текущего слоя в текущей конфигурации слоев – чем меньше номер, тем ближе слой к переднему плану. В разных конфигурациях слоев одни и те же слои могут иметь разные порядковые номера. Это поле может принимать значения от 0 до NumLayers-1.

CurLayerId

Идентификатор текущего слоя. Слой имеет один и тот же идентификатор во всех конфигурациях слоев. Именно эти идентификаторы указываются в сервисных функциях работы со слоями (стр. 270), структуре описания блока RDS_BLOCKDESCRIPTION (стр. 113) и структуре описания связи RDS_CONNDESCRIPTION (стр. 119).

NumConfigs

Общее число конфигураций слоев в подсистеме.

CurConfig

Порядковый номер текущей конфигурации в списке конфигураций слоев. Это поле может принимать значения от 0 до NumConfigs-1.

PrintZoneActive

TRUE, если в подсистеме активна зона печати (прямоугольная область, которой ограничивается выводимая на печать или сохраняемая в виде растрового рисунка часть рабочего поля подсистемы), FALSE в противном случае.

DisplayPrintZone

TRUE, если зона печати отображается в окне подсистемы, и FALSE, если ее отображение выключено. Зона печати может не отображаться, но быть при этом активной.

PZLeft, PZTop, PZWidth, PZHeight

Горизонтальная (PZLeft) и вертикальная (PZTop) координаты левого верхнего угла зоны печати, ее ширина (PZWidth) и высота (PZHeight). Положение и размеры зоны указываются в координатах рабочего поля, то есть в масштабе 100%.

WinMaximized

TRUE, если окно подсистемы развернуто на весь экран, FALSE в противном случае.

WinMinimized

TRUE, если окно подсистемы свернуто, FALSE в противном случае.

RefreshDelay

Интервал (в миллисекундах) автоматического обновления окна в режиме расчета, или 0, если автоматическое обновление выключено.

BlockNameColor

Цвет, которым в окне отображаются имена блоков.

BlockNameDistance

Интервал в точках экрана в масштабе 100% между изображением блока и его именем. Это значение используется по умолчанию, пользователь может перетащить имя блока вручную в произвольное место окна (параметры отображения имени конкретного блока указываются в структуре его описания, см. стр. 113).

BackgroundColor

Цвет фона рабочего поля подсистемы.

DefBackground

TRUE, если в качестве цвета фона рабочего поля установлен стандартный цвет окна Windows, и FALSE, если цвет задан пользователем вручную. Подсистемы, для которых установлен стандартный цвет рабочего поля, могут выглядеть по-разному на разных машинах.

WindowReactions

Битовые флаги, определяющие возможность реакции окна подсистемы на мышь и клавиатуру, если соответствующие события не были перехвачены внутренними блоками подсистемы:

Флаг	Событие
RDS_BDF_MOUSEEVENTS	Функция модели подсистемы будет вызывается при нажатии и отпускании кнопок мыши и перемещении курсора с нажатыми кнопками в пределах рабочего поля (события RDS_BFM_WINDOWMOUSEDOWN – стр. 74, RDS_BFM_WINDOWMOUSEUP – стр. 76, RDS_BFM_WINDOWMOUSEDBLCLICK – стр. 73, RDS_BFM_WINDOWMOUSEMOVE – стр. 75).
RDS_BDF_FREEMOUSEMOVE	Функция модели подсистемы вызывается при перемещении курсора мыши в пределах рабочего поля (событие RDS_BFM_WINDOWMOUSEMOVE, стр. 75) даже тогда, когда ни одна из кнопок не нажата (только при установленном флаге RDS_BDF_MOUSEEVENTS).
RDS_BDF_KBDEVENTS	Функции модели подсистемы разрешена реакция на клавиатуру (события RDS_BFM_WINDOWKEYDOWN – стр. 72 и RDS_BFM_WINDOWKEYUP – стр. 73).

Wallpaper

TRUE, если на рабочем поле изображаются растровые “обои”, и FALSE в противном случае.

WallpaperTile

TRUE, если “обоями” заполнено все рабочее поле подсистемы, FALSE, если они изображаются только в его левом верхнем углу.

WallpaperWidth, WallpaperHeight

Ширина (WallpaperWidth) и высота (WallpaperHeight) растрового изображения “обоев”. Следует помнить, что при изменении масштаба подсистемы размер растрового изображения обоев не изменяется.

Dashboard

TRUE, если для подсистемы установлен флаг “в подсистеме только неподвижные не перекрывающиеся блоки” (то есть включена оптимизация рисования, см. §2.10.2), и FALSE, если этот флаг не установлен.

См. также:

`rdsGetEditorParameters` (стр. 258).

A.4.8. RDS_EDITORTOOLBARS – описание панелей окна подсистемы

Структура `RDS_EDITORTOOLBARS` раньше использовалась для получения и установки видимости различных панелей в окне подсистемы. Сейчас для этого используются сервисные функции `rdsGetEditorWindowFlags` (стр. 259) и `rdsSetEditorWindowFlags` (стр. 265), работающие с битовыми флагами.

```
typedef struct {  
    BOOL LayersBar;      // Видимость панели слоев  
    BOOL ZoomBar;        // Видимость панели масштаба  
    BOOL DisplayBar;     // Видимость панели изображения  
    BOOL PrintBar;       // Видимость панели печати  
    BOOL StatusBar;      // Видимость строки состояния  
} RDS_EDITORTOOLBARS;  
typedef RDS_EDITORTOOLBARS *RDS_PEDITORTOOLBARS;
```

Поля структуры:

`LayersBar`

TRUE, если включена панель с именем текущего слоя, текущей конфигурации и кнопкой вызова редактора слоев.

`ZoomBar`

TRUE, если включена панель с полем ввода текущего масштаба, кнопкой увеличения и кнопкой перетаскивания.

`DisplayBar`

TRUE, если включена панель с кнопками разрешения отображения сетки, имен блоков, имен переменных и кнопкой привязки к сетке.

`PrintBar`

TRUE, если включена панель с кнопками печати и задания зоны печати.

`StatusBar`

TRUE, если включена строка состояния.

В этой структуре нет поля, отражающего включение панели расчета (с кнопками переключения режимов и главного окна). Она используется только для сохранения совместимости со старыми моделями блоков.

См. также:

`rdsGetEditorToolBars` (стр. 259), `rdsSetEditorToolBars` (стр. 265),
`rdsGetEditorWindowFlags` (стр. 259), `rdsSetEditorWindowFlags` (стр. 265).

A.4.9. RDS_FINDBYEXTIDDATA – результаты поиска по идентификатору

Структура `RDS_FINDBYEXTIDDATA` используется в параметрах функции `rdsBlockOrConnByExtId` (стр. 206), которая ищет в схеме блок или связь по его или ее уникальному целому идентификатору. Такие целые идентификаторы используются при управлении РДС из другого приложения (см. §3.5), их можно получить через поле `ExtId` структур описания блока `RDS_BLOCKDESCRIPTION` (стр. 113) и связи `RDS_CONNDESCRIPTION` (стр. 119).

```
typedef struct {  
    DWORD servSize;      // Размер этой структуры
```

```

    BOOL Found;           // Что-то найдено
    RDS_BHANDLE Block;    // Найденный блок
    RDS_CHANDLE Conn;     // Найденная связь или шина
    int Type;             // Тип найденного объекта
} RDS_FINDBYEXTIDDATA;
typedef RDS_FINDBYEXTIDDATA *RDS_PFOUNDBYEXTIDDATA;

```

Поля структуры:

servSize

Размер этой структуры в байтах. Перед вызовом сервисной функции `rdsBlockOrConnByExtId` полю `servSize` необходимо присвоить значение `sizeof(RDS_FINDBYEXTIDDATA)`.

Found

Найдены блок, связь или шина с указанным в параметре функции целым идентификатором.

Block

Внутренний идентификатор (тип `RDS_BHANDLE`, см. стр. 23) блока, которому принадлежит переданный в параметрах функции целый идентификатор. Практически все сервисные функции РДС работают именно с идентификаторами типа `RDS_BHANDLE`.

Conn

Внутренний идентификатор (тип `RDS_CHANDLE`, см. стр. 23) связи или шины, которой принадлежит переданный в параметрах функции целый идентификатор. Практически все сервисные функции РДС работают именно с идентификаторами типа `RDS_CHANDLE`.

Type

Тип найденного объекта (одна из стандартных констант типа, см. также стр. 113 и 120):

<code>RDS_TUNKNOWN</code>	объект не найден
<code>RDS_BTSYSTEM</code>	подсистема
<code>RDS_BTSIMPLEBLOCK</code>	простой блок
<code>RDS_BTINPUTBLOCK</code>	внешний вход
<code>RDS_BTOUTPUTBLOCK</code>	внешний выход
<code>RDS_BTBUSPORT</code>	ввод шины
<code>RDS_CTCONNECTION</code>	связь
<code>RDS_CTBUS</code>	шина

В результате вызова функции `rdsBlockOrConnByExtId` в этой структуре будет заполнено либо поле `Block`, если переданный целый идентификатор принадлежит блоку (в поле `Conn` при этом будет записано значение `NULL`), либо поле `Conn`, если идентификатор принадлежит связи или шине (при этом `NULL` будет записано в `Block`). В поле `Found` будет записано `TRUE`, если переданный идентификатор принадлежит какому-либо объекту, то есть если либо `Conn`, либо `Block` не равны `NULL`.

См. также:

`rdsBlockOrConnByExtId` (стр. 206), `RDS_BLOCKDESCRIPTION` (стр. 113),
`RDS_CONNDESCRIPTION` (стр. 119).

A.4.10. RDS_FORMSERVFUNCDATA – параметр функции обратного вызова модального окна

Структура RDS_FORMSERVFUNCDATA используется в качестве параметра пользовательской функции обратного вызова, указатель на которую передается в функцию открытия модального окна rdsFORMShowModalServ (стр. 504). Такая функция обратного вызова применяется для организации реакции в модальном окне, созданном функцией rdsFORMCreate (стр. 491), на изменение полей ввода, нажатие кнопок и т.п., а также для программного рисования в этом окне. В структуре RDS_FORMSERVFUNCDATA в пользовательскую функцию передается идентификатор события, возникшего в окне, идентификатор поля ввода, с которым связано событие, а также другая информация об этом событии. Пример использования функции обратного вызова в модальном окне и этой структуры приведен в §2.7.3.

```
typedef struct {  
    int Event;      // Событие (RDS_FORMSERVEVENT_*)  
    int CtrlId;     // Идентификатор органа управления  
    // Для RDS_FORMSERVEVENT_DRAW  
    HDC dc;        // Контекст устройства для рисования (WinAPI)  
    int Left,Top;   // Верхний левый угол зоны рисования  
    int Width,Height; // Размеры зоны рисования  
} RDS_FORMSERVFUNCDATA;  
typedef RDS_FORMSERVFUNCDATA *RDS_PFORMSERVFUNCDATA;
```

Поля структуры:

Event

Идентификатор произошедшего в окне события. В этом поле находится одна из следующих констант:

RDS_FORMSERVEVENT_CHANGE	Изменение одного или нескольких полей ввода. Если изменено одно поле ввода, в CtrlId будет находиться его идентификатор, если несколько (при первом открытии окна, когда сразу все поля получают начальные значения), в CtrlId будет записано значение -1.
RDS_FORMSERVEVENT_DRAW	Программное рисование в одной из специальных областей окна (в CtrlId – идентификатор области).
RDS_FORMSERVEVENT_CLICK	Нажатие кнопки (в CtrlId – идентификатор кнопки).

CtrlId

Идентификатор поля ввода, кнопки или другого объекта модального окна, с которым связано произошедшее событие. Эти идентификаторы присваиваются объектам при их создании функцией rdsFORMAddEdit (стр. 492).

dc

Контекст устройства Windows (device context, HDC), на котором функция обратного вызова должна нарисовать изображение (только при Event, равном RDS_FORMSERVEVENT_DRAW). Это значение можно использовать в вызовах графических функций Windows API.

Left, Top

Горизонтальная (Left) и вертикальная (Top) координаты верхнего левого угла специальной области рисования, созданной в окне (только при Event, равном RDS_FORMSERVEVENT_DRAW).

Width, Height

Ширина (Width) и высота (Height) специальной области рисования, созданной в окне (только при Event, равном RDS_FORMSERVEVENT_DRAW).

См. также:

rdsFORMShowModalServ (стр. 504), rdsFORMCreate (стр. 491),
rdsFORMAddEdit (стр. 492).

A.4.11. RDS_FUNCPROVIDERLINK – подписка на исполнителя функции

Структура RDS_FUNCPROVIDERLINK создается во внутренней памяти РДС при подписке блока на информацию об блоке-исполнителе какой-либо функции (см. §2.13.6) и используется моделью блока для вызова функции этого блока. Модели не требуется самостоятельно создавать эти структуры, она всегда работает с указателями на них, возвращенными сервисной функцией rdsSubscribeToFuncProvider (стр. 317). Структура уничтожается РДС автоматически при прекращении подписки на исполнителя или удалении подписавшегося блока.

```
typedef struct {  
    RDS_BHANDLE Block; // Блок-исполнитель  
    int FuncId;        // Функция  
} RDS_FUNCPROVIDERLINK;  
typedef RDS_FUNCPROVIDERLINK *RDS_PFUNCPROVIDERLINK;
```

Поля структуры:

Block

Идентификатор блока, зарегистрировавшегося как исполнитель функции с запрошенным идентификатором. Если такого блока в схеме нет, в этом поле будет находиться NULL. РДС автоматически обновляет это поле в созданных структурах по мере регистрации новых блоков и удалении старых.

FuncId

Идентификатор функции, подписка на исполнителя которой запрашивалась.

Если поле Block этой структуры не равно NULL, пару Block, FuncId можно использовать для вызова функции при помощи rdsCallBlockFunction (стр. 310) или rdsQueueCallBlockFunction (стр. 313).

См. также:

rdsRegisterFunction (стр. 316), rdsCallBlockFunction (стр. 310),
rdsQueueCallBlockFunction (стр. 313),
rdsSubscribeToFuncProvider (стр. 317),
rdsUnsubscribeFromFuncProvider (стр. 318),
rdsRegisterFuncProvider (стр. 315),
rdsUnregisterFuncProvider (стр. 317).

A.4.12. RDS_LINEDESCRIPTION – описание отрезка внутри связи или шины

Структура RDS_LINEDESCRIPTION используется для получения описания отдельного отрезка, соединяющего пару точек связи или шины в ее геометрическом

изображении. Эта структура заполняется сервисными функциями `rdsGetLineDescription` (стр. 231) и `rdsFindNextConnectedLine` (стр. 218). Пример ее использования приведен в §2.13.4.

Структура используется только для получения описания отрезка, изменить параметры этого отрезка с ее помощью нельзя. Для изменения параметров существующих связей или шин и создания новых используется вспомогательный объект, создаваемый функцией `rdsCECreateEditor` (стр. 408).

```
typedef struct {
    DWORD servSize;           // Размер этой структуры в байтах
    int LineType;             // Тип отрезка (RDS_LN*)
    int nPoint1, nPoint2;     // Номера соединяемых точек
    int x1, y1;               // Абсолютные координаты точки nPoint1
    int x2, y2;               // Абсолютные координаты точки nPoint2
    int dx1, dy1;             // Смещения управляющей точки для nPoint1
    int dx2, dy2;             // Смещения управляющей точки для nPoint2
    RDS_CHANDLE Owner;        // Связь-владелец отрезка
} RDS_LINEDESCRIPTION;
typedef RDS_LINEDESCRIPTION *RDS_PLINEDESCRIPTION;
```

Поля структуры:

`servSize`

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю `servSize` необходимо присвоить значение `sizeof(RDS_LINEDESCRIPTION)`.

`LineType`

Тип отрезка: отрезок прямой или кривая Безье В этом поле может находиться одна из двух констант:

<code>RDS_LNLINE</code>	Отрезок прямой.
<code>RDS_LNBEZIER</code>	Кривая Безье.

`nPoint1, nPoint2`

Порядковые номера соединяемых отрезком точек связи или шины. Обращение к точкам всегда идет по их номеру, общее число точек можно получить через поле `NumPoints` структуры `RDS_CONNDESCRIPTION` (стр. 119).

`x1, y1`

Горизонтальная (`x1`) и вертикальная (`y1`) координаты точки связи с номером `nPoint1`. В отличие от структуры описания точки `RDS_POINTDESCRIPTION` (стр. 133), в этой структуре всегда указываются абсолютные координаты точки на рабочем поле подсистемы в масштабе 100%.

`x2, y2`

Горизонтальная (`x2`) и вертикальная (`y2`) координаты точки связи с номером `nPoint2`. В отличие от структуры описания точки `RDS_POINTDESCRIPTION`, в этой структуре всегда указываются абсолютные координаты точки на рабочем поле подсистемы в масштабе 100%.

`dx1, dy1`

Горизонтальное (`dx1`) и вертикальное (`dy1`) смещение управляющей точки (точки касательной) для точки связи с номером `nPoint1`. Эти поля используются только для кривых Безье, то есть если `LineType == RDS_LNBEZIER`.

dx2, dy2

Горизонтальное (dx2) и вертикальное (dy2) смещение управляющей точки (точки касательной) для точки связи с номером nPoint2. Эти поля используются только для кривых Безье, то есть если LineType == RDS_LNBEZIER.

Owner

Идентификатор связи или шины, которой принадлежит данный отрезок.

См. также:

rdsGetLineDescription (стр. 231), rdsFindNextConnectedLine (стр. 218), RDS_CONNDESCRIPTION (стр. 119), rdsGetConnDescription (стр. 225), rdsCECreateEditor (стр. 408).

A.4.13. RDS_PANDESCRIPTION – описание панели блока в окне подсистемы

Структура RDS_PANDESCRIPTION используется для получения описания панели, созданной блоком в окне подсистемы. Эта структура заполняется сервисной функцией rdsPANGetDescr (стр. 547), а также создается и автоматически заполняется РДС при вызове реакции блока на действия с панелью (событие RDS_BFM_BLOCKPANEL, см. стр. 55). Пример использования структуры RDS_PANDESCRIPTION приведен в §2.10.4.

Эта структура используется только для получения описания панели. Для изменения параметров и внешнего вида панели предусмотрены специальные команды (стр. 545).

```
typedef struct {
    DWORD servSize;      // Размер этой структуры в байтах
    RDS_HOBJECT Object;  // Объект, связанный с этой панелью
    RDS_BHANDLE Block;   // Блок-владелец панели
    int PLeft, PTop;     // Левый верхний угол панели
    int PWidth, PHeight; // Размер панели
    int Order;           // Номер панели в данном блоке
    BOOL Visible;        // Видимость панели
    HWND Handle;         // Оконный объект или NULL
    int Width, Height;   // Размеры оконного объекта
    BOOL Border;         // Наличие рамки
    BOOL CloseButton;    // Наличие кнопки закрытия панели
    BOOL Scalable;       // Панель меняет размер при
                        // изменении масштаба системы
    BOOL Sizeable;       // Размер панели может
                        // изменяться пользователем
    BOOL Moveable;       // Положение панели может
                        // изменяться пользователем
    BOOL CaptionBar;     // Есть полоса заголовка
    LPSTR Caption;       // Текущий заголовок
} RDS_PANDESCRIPTION;
typedef RDS_PANDESCRIPTION *RDS_PPANDESCRIPTION;
```

Поля структуры:

servSize

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю servSize необходимо присвоить значение sizeof(RDS_PANDESCRIPTION).

Object

Идентификатор вспомогательного объекта РДС (см. стр. 23), связанного с данной панелью. Этот идентификатор указывается во всех сервисных функциях, которые работают с панелями.

Block

Идентификатор блока, которому принадлежит данная панель.

PLeft, PTop

Горизонтальная (PLeft) и вертикальная (PTop) координаты левого верхнего угла панели на рабочем поле окна подсистемы. Эти координаты указываются для масштаба 100%.

PWidth, PHeight

Ширина (PWidth) и высота (PHeight) панели на рабочем поле окна подсистемы в масштабе 100%. Это внешние размеры панели – размеры оконного объекта Windows, находящегося на этой панели, могут быть меньше за счет рамки панели и полосы ее заголовка.

Order

Порядковый номер панели в блоке, определяющий ее близость к переднему плану. Панели одного и того же блока с большим значением поля Order будут перекрывать панели с меньшим его значением. Перекрывание панелей разных блоков определяется взаимным расположением самих блоков.

Visible

TRUE, если панель должна быть видима, FALSE, если она должна быть скрыта. Это поле определяет не фактическую видимость панели, а ее желаемое состояние. При закрытом окне подсистемы ни одна панель в ней не показывается на экране, но на значении поля Visible это никак не отражается. Те панели, для которых поле Visible установлено в TRUE, будут автоматически показаны при открытии окна подсистемы.

Handle

Дескриптор оконного объекта Windows, находящегося внутри панели. Его можно использовать в вызовах Windows API. Если оконный объект не создан (если панель невидима, или если окно ее подсистемы закрыто), в этом поле будет находиться значение NULL.

Width, Height

Ширина (Width) и высота (Height) оконного объекта Windows, находящегося внутри панели, если он существует.

Border

TRUE, если панель имеет рельефную рамку, FALSE в противном случае.

CloseButton

TRUE, если панель имеет кнопку закрытия в правом верхнем углу полосы заголовка, FALSE в противном случае. Кнопка закрытия позволяет пользователю убрать панель с экрана, она может выводиться только на панелях с рамкой и заголовком.

Scalable

TRUE, если панель автоматически изменяет размеры при изменении масштаба подсистемы, и FALSE, если ее размеры остаются постоянными независимо от масштаба.

Sizeable

TRUE, если пользователь может изменять размеры панели, перетаскивая ее границы и углы, FALSE в противном случае. Изменение размеров возможно только у панелей с включенной рамкой.

Moveable

TRUE, если пользователь может перемещать панель, перетаскивая ее за полосу заголовка, FALSE в противном случае. Панели без заголовков перетаскивать нельзя.

CaptionBar

TRUE, если панель имеет полосу заголовка, FALSE в противном случае.

Caption

Указатель на строку во внутренней памяти РДС, в которой хранится текст, выводимый в заголовке панели (если включена полоса заголовка). Функция модели не должна как-либо изменять эту строку, для изменения текста заголовка используется команда RDS_PAN_CAPTION (стр. 548).

См. также:

rdsPANGetDescr (стр. 547), rdsPANCreate (стр. 545),
RDS_BFM_BLOCKPANEL (стр. 55), RDS_PAN_CAPTION (стр. 548).

A.4.14. RDS_POINTDESCRIPTION – описание точки связи или шины

Структура RDS_POINTDESCRIPTION используется для получения описания отдельной точки внутри связи или шины. Эта структура заполняется сервисной функцией rdsGetPointDescription (стр. 237) и некоторыми другими функциями. Примеры ее использования приведены в §§2.7.4, 2.13.2 и 2.13.4.

Структура используется только для получения описания точки, изменить параметры этой точки с ее помощью нельзя. Для изменения параметров существующих связей или шин и создания новых используется вспомогательный объект, создаваемый функцией rdsCECreateEditor (стр. 408).

```
typedef struct {  
    DWORD servSize;           // Размер этой структуры в байтах  
    int PointType;            // Тип точки (RDS_PT*)  
    BOOL Source;              // TRUE - источник данных,  
                             // FALSE - получатель  
    int Status;               // Состояние (RDS_PS*)  
    LPSTR ObjectName;         // Имя соединенного блока или шины  
    LPSTR VarName;            // Имя соединенной переменной  
    RDS_BHANDLE Block;        // Соединенный блок или NULL  
    RDS_CHANDLE Bus;          // Соединенная шина или NULL  
    RDS_CHANDLE Owner;        // Связь-владелец точки  
    int x,y;                  // Координаты точки  
    int PointNum;             // Номер этой точки внутри связи/шины  
} RDS_POINTDESCRIPTION;  
typedef RDS_POINTDESCRIPTION *RDS_PPOINTDESCRIPTION;
```

Поля структуры:

servSize

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю servSize необходимо присвоить значение sizeof(RDS_POINTDESCRIPTION).

PointType

Тип точки. В этом поле может находиться одна из четырех констант:

RDS_PTINTERNAL	Промежуточная точка связи или шины (излом или разветвление).
RDS_PTBLOCK	Точка соединения связи с блоком.
RDS_PTBUS	Точка соединения связи с шиной.
RDS_PTBUSTOBLOCK	Точка соединения шины с подсистемой или шинным вводом.

Source

TRUE, если данная точка связи является источником данных (подключена к выходу блока или шины), и FALSE, если она получает данные (подключена ко входу блока или шины). Используется только для точек соединения с блоками и шинами (в поле PointType – RDS_PTBLOCK или RDS_PTBUS).

Status

Состояние точки (только для точек, не являющихся промежуточными). В этом поле может находиться одна из следующих констант, указывающих на ошибки соединения:

RDS_PSNORMAL	Нет ошибок соединения.
RDS_PSBADVAR	Отсутствует либо указанный в параметрах точки блок, либо указанная переменная в блоке.
RDS_PSBADTYPE	Вход блока, с которой соединена точка, имеет тип, несовместимый с типом выхода, к которому присоединена связь.

ObjectName

Указатель на строку во внутренней памяти РДС, в которой хранится имя соединенного с точкой блока (для точек соединения с блоками, типы RDS_PTBLOCK и RDS_PTBUSTOBLOCK) или соединенной шины (для точек соединения связей с шинами, тип RDS_PTBUS). Для промежуточных точек в этом поле записано значение NULL.

VarName

Указатель на строку во внутренней памяти РДС, в которой хранится имя соединенной с точкой переменной (для точек соединения связей с блоками, тип RDS_PTBLOCK) или канала шины (для точек соединения связей с шинами, тип RDS_PTBUS). Для всех остальных типов точек в этом поле записано значение NULL.

Block

Идентификатор соединенного с точкой блока для точек соединения с блоками (типы RDS_PTBLOCK и RDS_PTBUSTOBLOCK). Для всех остальных типов точек в этом поле записано значение NULL.

Bus

Идентификатор соединенной с точкой шины для точек соединения связей с шинами (тип RDS_PTBUS). Для всех остальных типов точек в этом поле записано значение NULL.

Owner

Идентификатор связи или шины, которой принадлежит данная точка.

x, y

Горизонтальная (x) и вертикальная (y) координаты точки. Для промежуточных точек и точек соединения связей с шинами (типы RDS_PTINTERNAL и RDS_PTBUS) указываются абсолютные координаты на рабочем поле, то есть смещения относительно его левого верхнего угла. Для точек соединения связей и шин с блоками

(типы `RDS_PTBLOCK` и `RDS_PTBUSTOBLOCK`) указываются относительные координаты, то есть смещения относительно координат точки привязки соединенного блока. Все координаты указаны для масштаба 100%, горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля (для абсолютных координат) или точка привязки блока (для относительных).

PointNum

Порядковый номер данной точки в связи или шине. Обращение к точкам всегда идет по их номеру, общее число точек можно получить через поле `NumPoints` структуры `RDS_CONNDESCRIPTION` (стр. 119).

См. также:

`rdsGetPointDescription` (стр. 237), `RDS_CONNDESCRIPTION` (стр. 119),
`rdsGetConnDescription` (стр. 225), `rdsCECreateEditor` (стр. 408).

A.4.15. `RDS_SERVFONTPARAMS` – описание шрифта

Структура `RDS_SERVFONTPARAMS` используется в некоторых сервисных функциях для описания параметров шрифта при программном рисовании внешнего вида блока. Пример ее применения приведен в §2.10.1.

```
#define RDS_SERVFONTPARAMSNAMESIZE 256 // Размер массива имени
typedef struct {
    DWORD servSize; // Размер этой структуры в байтах
    char Name[RDS_SERVFONTPARAMSNAMESIZE]; // Имя шрифта
    int CharSet; // Набор символов
    int Height; // Высота (в точках экрана)
    int Size; // Высота (в типографских точках)
    BOOL SizePriority; // TRUE - использовать Size,
                    // FALSE - использовать Height
    COLORREF Color; // Цвет шрифта
    BOOL Bold; // Жирный
    BOOL Italic; // Курсив
    BOOL Underline; // Подчеркнутый
    BOOL StrikeOut; // Зачеркнутый
} RDS_SERVFONTPARAMS;
typedef RDS_SERVFONTPARAMS *RDS_PSERVFONTPARAMS;
```

Поля структуры:

servSize

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю `servSize` необходимо присвоить значение `sizeof(RDS_SERVFONTPARAMS)`.

Name

Массив, в котором находится строка названия шрифта. Размер этого массива задан `define`-константой `RDS_SERVFONTPARAMSNAMESIZE`, равной 256 – таким образом, длина строки названия шрифта не может превышать 255 символов.

CharSet

Стандартная константа Windows API, определяющая набор символов (язык) шрифта. Чаще всего употребляются следующие константы:

`ANSI_CHARSET` Шрифт ANSI.
`DEFAULT_CHARSET` Символы определяется названием и размером шрифта.

RUSSIAN_CHARSET Русские символы.
SYMBOL_CHARSET Различные символические обозначения.
Полный список возможных наборов символов приведен в описании Windows API.

Height

Высота шрифта в точках экрана.

Size

Высота шрифта в типографских точках (стандартная единица задания размера шрифта во всех диалогах Windows).

SizePriority

Это поле используется только в том случае, когда структура RDS_SERVFONTPARAMS используется для установки, а не для получения параметров шрифта. Если в нем будет записано значение TRUE, высота шрифта будет считана из поля Size и будет задана в типографских точках. Если же в нем будет записано FALSE, высота шрифта будет считана из поля Height и будет задана в точках экрана. Если структура используется для получения параметров, в ней всегда заполняются и поле Height, и поле Size, и работающая со структурой модель блока может использовать значения обоих этих полей.

Color

Цвет шрифта (тип COLORREF, см. стр. 24).

Bold

TRUE, если шрифт имеет полужирное начертание.

Italic

TRUE, если шрифт имеет курсивное начертание.

Underline

TRUE, если включено подчеркивание.

StrikeOut

TRUE, если шрифт перечеркнут.

См. также:

rdsStructToFontText (стр. 279), rdsFontTextToStruct (стр. 290),
rdsXGSetFontByParStr (стр. 381).

A.4.16. RDS_TIMERDESCRIPTION – описание таймера

Структура RDS_TIMERDESCRIPTION используется для получения описания программируемого таймера блока. Эта структура заполняется сервисной функцией rdsGetBlockTimerDescr (стр. 299), пример ее использования приведен в §2.9.2.

Структура используется только для получения описания таймера, изменить его параметры с ее помощью нельзя. Для изменения параметров таймеров используются специальные сервисные функции (стр. 298).

```
typedef struct {  
    DWORD servSize;      // Размер этой структуры в байтах  
    DWORD Delay;         // Задержка таймера (мс)  
    DWORD StartTime;     // Время начала отсчета (мс)  
    BOOL On;             // Таймер включен  
    DWORD Mode;          // Флаги режима  
} RDS_TIMERDESCRIPTION;  
typedef RDS_TIMERDESCRIPTION *RDS_PTIMERDESCRIPTION;
```


Поля структуры:

servSize

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю servSize необходимо присвоить значение sizeof(RDS_TIMERDESCRIPTION).

Delay

Интервал в миллисекундах перед срабатыванием таймера (между срабатываниями для циклических таймеров).

StartTime

Время начала отсчета таймера в миллисекундах. Это значение функции Windows API GetTickCount() на момент запуска таймера. Таким образом, таймер сработает тогда, когда значение GetTickCount() станет большим или равным (StartTime + Delay).

On

TRUE, если данный таймер включен, и FALSE, если выключен.

Mode

Режим работы таймера. Формируется как объединение битовым ИЛИ константы режима (RDS_TIMERM_*), константы способа срабатывания (RDS_TIMERS_*) и флагов режима (RDS_TIMERF_*). Константа режима может быть одной из следующих:

RDS_TIMERM_LOOP	Таймер циклический, после срабатывания он автоматически перезапускается с той же задержкой.
RDS_TIMERM_STOP	Таймер однократный, после срабатывания он будет остановлен. Его можно перезапустить сервисными функциями rdsSetBlockTimer (стр. 300) или rdsRestartBlockTimer (стр. 299).
RDS_TIMERM_DELETE	Таймер однократный, после срабатывания он будет автоматически удален.

Константа способа срабатывания может быть одной из следующих:

RDS_TIMERS_SIGNAL	При срабатывании таймера у блока-владельца взводится сигнал запуска, то есть его первой сигнальной переменной присваивается значение 1. Таймер работает только в режиме расчета.
RDS_TIMERS_TIMER	При срабатывании таймера модель блока-владельца вызывается в потоке расчета для реакции на событие RDS_BFM_TIMER (см. стр. 47). Таймер работает только в режиме расчета.
RDS_TIMERS_WINREF	При срабатывании таймера модель блока-владельца вызывается в главном потоке для реакции на событие RDS_BFM_WINREFRESH (см. стр. 77). Таймер работает только в режиме расчета.
RDS_TIMERS_SYSTIMER	При срабатывании таймера модель блока-владельца вызывается в главном потоке для реакции на событие RDS_BFM_TIMER (см. стр. 47). Таймер работает во всех режимах.

На данный момент поддерживается единственный флаг режима:

RDS_TIMERF_FIXFREQ	При способе срабатывания RDS_TIMERS_WINREF к этому таймеру не будет применяться автоматическое снижение частоты при чрезмерной загрузке процессора процедурами обновления окон. При остальных способах срабатывания флаг игнорируется.
--------------------	--

Для выделения из поля Mode трех его компонентов (режима, способа срабатывания и флагов) можно использовать следующие битовые маски:

RDS_TIMERMASK_M	Маска режима работы.
RDS_TIMERMASK_S	Маска способа срабатывания.
RDS_TIMERMASK_F	Маска флагов.

См. также:

rdsGetBlockTimerDescr (стр. 299).

A.4.17. RDS_VARDESCRIPTION – описание переменной блока

Структура RDS_VARDESCRIPTION используется для получения описания переменной блока и заполняется различными сервисными функциями. Примеры ее использования приведены в §§2.7.4 и 2.16.1.

```
typedef struct {
    DWORD servSize;      // Размер этой структуры в байтах
    char Type;           // Тип переменной (RDS_VARTYPE_*)
    DWORD Flags;         // Флаги (RDS_VARFLAG_*)
    LPSTR Name;          // Имя переменной
    LPSTR StructType;    // Имя типа структуры или NULL
    int StructFields;    // Число полей (только для структур)
    int DataSize;        // Размер блока данных
    RDS_VHANDLE Var;     // Идентификатор переменной
    int Rank;            // Уровень переменной
} RDS_VARDESCRIPTION;
typedef RDS_VARDESCRIPTION *RDS_PVARDESCRIPTION;
```

Поля структуры:

servSize

Размер этой структуры в байтах. Перед вызовом любой сервисной функции, работающей с этой структурой, полю servSize необходимо присвоить значение sizeof(RDS_VARDESCRIPTION).

Type

Символ типа переменной (одна из констант RDS_VARTYPE_*, см. стр. 49).

Flags

Набор битовых флагов переменной:

RDS_VARFLAG_INPUT	Переменная является входом блока (не может быть установлен одновременно с RDS_VARFLAG_OUTPUT).
RDS_VARFLAG_OUTPUT	Переменная является выходом блока (не может быть установлен одновременно с RDS_VARFLAG_INPUT).
RDS_VARFLAG_RUN	При срабатывании данного входа автоматически запускается модель (только для переменных – входов блока).
RDS_VARFLAG_MENU	Имя переменной присутствует в меню присоединения связи к блоку (только для входов и выходов блоков).

RDS_VARFLAG_SHOWNAME Имя переменной отображается рядом с конечной точкой присоединенной к ней связи (только для входов и выходов блоков, считывается только в момент создания новой связи – далее отображение имени уже определяется параметрами точки этой связи).

RDS_VARFLAG_ONEINDEX Переменная – массив, а не матрица, то есть для ее элементов используется один индекс, а не два (только для переменных типа **RDS_VARTYPE_ARRAY**).

Для входов и выходов блока флаги **RDS_VARFLAG_INPUT**, **RDS_VARFLAG_OUTPUT** и **RDS_VARFLAG_RUN** устанавливаются не только для переменных, непосредственно находящихся в структуре переменных блока, но и для их элементов. Например, если какая-либо структура является входом, срабатывание связи которого вызывает запуск модели, флаги **RDS_VARFLAG_INPUT** и **RDS_VARFLAG_RUN** будут установлены не только у самой этой структуры, но и у всех ее полей.

Name

Указатель на строку во внутренней памяти РДС, в которой хранится имя данной переменной. Функция модели не должна как-либо изменять эту строку, для изменения и переименования переменных блока следует использовать вспомогательный объект, создаваемый функцией `rdsVSCreateEditor` (стр. 430).

StructType

Указатель на строку во внутренней памяти РДС, в которой хранится название типа структуры, если данная переменная является структурой (тип **RDS_VARTYPE_STRUCT**). Функция модели не должна изменять эту строку. Если переменная структурой не является, или у структуры нет имени, в этом поле находится значение **NULL**.

StructFields

Число полей в переменной, если она является структурой (тип **RDS_VARTYPE_STRUCT**). Для всех остальных переменных в это поле записывается ноль.

DataSize

Размер в байтах блока данных этой переменной (см. §1.5 и §2.5), то есть объем памяти, который занимает эта переменная в дереве переменных блока. Например, вещественная переменная двойной точности (`double`, тип **RDS_VARTYPE_DOUBLE**) занимает в памяти восемь байтов, строка (тип **RDS_VARTYPE_STRING**) – четыре байта, поскольку в дереве переменных хранится только указатель на строку, и т.п.

Var

Уникальный идентификатор данной переменной для использования в различных сервисных функциях.

Rank

Уровень данной переменной, то есть максимальная вложенность элементов в этой переменной. Например, для любой простой переменной уровень будет равен нулю, для массива или матрицы вещественных чисел – единице (один вложенный элемент), для матрицы матриц целых – двум (внутри этой матрицы – еще одна матрица, внутри которой – целое число, то есть в переменной два элемента, вложенных один в другой). Для структуры уровень будет числом, на единицу большим максимального уровня всех ее полей.

См. также:

RDS_BFM_VARCHECK (стр. 48), rdsGetBlockVar (стр. 328),
rdsFindStructVar (стр. 327), rdsGetStructVar (стр. 331),
rdsGetVarField (стр. 332), rdsFindBlockVar (стр. 326),
rdsSetBlockVarFlags (стр. 335), rdsVSCreateEditor (стр. 430),
rdsVSGetVarDescription (стр. 441).

А.5. Сервисные функции РДС

Описываются функции, экспортированные из главного модуля РДС (rds.exe), которые могут вызываться моделями блоков для взаимодействия с РДС и между собой.

А.5.1. Доступ к сервисным функциям РДС

Сервисные функции РДС – это функции, экспортированные из программы rds.exe. Все они имеют тип RDSCALL (см. стр. 25): аргументы функции передаются в стеке справа налево, стек освобождается вызванной функцией. Для доступа к сервисной функции следует получить указатель на нее при помощи функции Windows API GetProcAddress, привести этот указатель к правильному типу, учитывающему типы параметров и возвращаемого значения этой функции (тип указателя на каждую функцию приводится в описании этой функции в данном приложении), после чего можно будет вызывать эту функцию непосредственно по указателю на нее. Поскольку получить указатели на сервисные функции достаточно один раз в момент загрузки DLL с моделью блока, удобнее всего сделать это в главной функции этой DLL (стр. 27).

Чтобы не получать вручную все указатели на все сервисные функции, можно воспользоваться файлом “RdsFunc.h”. Для этого следует включить в исходный текст программы следующий фрагмент:

```
#define RDS_SERV_FUNC_BODY имя_функции  
#include <RdsFunc.h>
```

В результате в это место текста будет вставлен полный набор глобальных переменных-указателей на сервисные функции, одноименных этим функциям, а также дополнительная функция с именем *имя_функции* для заполнения всех этих переменных. Останется только вызвать эту дополнительную функцию из главной функции DLL, после чего все сервисные функции можно будет вызывать непосредственно по их именам. Имена глобальных переменных совпадают с именами сервисных функций, поэтому запись вида

```
rdsMessageBox ("Сообщение", "Заголовок", MB_OK);
```

будет означать, что функция, указатель на которую находится в глобальной переменной rdsMessageBox, вызывается с параметрами “Сообщение”, “Заголовок” и MB_OK. Поскольку переменная с именем rdsMessageBox имеет правильный тип указателя на функцию с нужными типами параметров, ее можно использовать вместо имени функции без всякого приведения типов.

Если перед включением файла “RdsFunc.h” не будет определен макрос RDS_SERV_FUNC_BODY, все описания глобальных переменных будут включены с ключевым словом extern, то есть они будут описаны как внешние, находящиеся в другом модуле. Таким образом, если проект DLL состоит из нескольких модулей, в каждом из них нужно включить файл “RdsFunc.h”, и в одном и только одном из них (обычно в том, в котором находится главная функция DLL) перед этим включением определить макрос RDS_SERV_FUNC_BODY. В результате в одном модуле будут описаны глобальные переменные-указатели на функции и функция их заполнения, а во всех остальных эти же переменные будут описаны как внешние. Это даст возможность вызывать сервисные функции по имени в любом из модулей проекта.

Для управления описаниями в файле “RdsFunc.h” можно использовать еще две дополнительных константы. Если перед включением этого файла объявить define-константу RDS_NOEXTERNFUNCPTRS (константа может не иметь значения, важен сам факт описания), внешние описания глобальных переменных-указателей на функции с ключевым словом extern не будут включены в исходный текст модуля. Если перед включением файла описать константу RDS_NOOBJMACROS, не будут определены макросы работы со

вспомогательными объектами, описанные в А.5.23–А.5.32. Эти две константы позволяют исключать из программы описания, которые по каким-либо причинам не нужны разработчику моделей.

Примеры доступа к сервисным функциям РДС приведены в §2.2 и А.2.2.

А.5.2. Управление работой РДС и функции общего назначения

Описываются различные функции и макроопределения общего назначения, используемые для управления приложением РДС, переключения режимов, получения общесистемной информации и т.п.

А.5.2.1. Макрос RDS_GETFLAG – получение битового флага

Макрос RDS_GETFLAG предназначен для проверки взведенности битового флага в беззнаковом целом (DWORD).

```
RDS_GETFLAG(  
    storage,    // Набор флагов  
    mask       // Маска  
)
```

Определение:

```
#define RDS_GETFLAG(storage, mask) \  
    (((storage) & (mask)) != 0)
```

Параметры:

storage

Беззнаковое целое (DWORD), содержащее в себе набор битовых флагов.

mask

Проверяемый битовый флаг (число, в котором взведен бит в позиции проверяемого флага).

Возвращаемое значение:

Если флаг, соответствующий параметру mask, взведен в числе storage, макрос возвращает истинное значение, если флаг не взведен – ложное.

Примечания:

Не следует использовать этот макрос для проверки одновременной взведенности нескольких флагов. Если в параметре mask содержится несколько единичных битов, RDS_GETFLAG вернет истинное значение, если в storage взведен хотя бы один из них.

См. также:

RDS_SETFLAG (стр. 145).

А.5.2.2. Макрос RDS_INITSERVSIZE – инициализация поля размера в стандартных структурах РДС

Макрос RDS_INITSERVSIZE присваивает полю servSize переданной в его параметре структуры размер этой структуры в байтах.

```
RDS_INITSERVSIZE(  
    structvar    // Переменная-структура  
)
```

Определение:

```
#define RDS_INITSERVSIZE(structvar) \
    structvar.servSize=sizeof(structvar)
```

Параметр:

structvar

Переменная, представляющая собой структуру одного из стандартных типов РДС, в которой есть поле servSize.

Примечания:

Присваивать полю servSize размер содержащей его структуры необходимо перед вызовом большинства сервисных функций, работающих со стандартными структурами РДС (см. А.4). Сервисные функции, в которые передаются указатели на эти структуры, прежде всего сравнивают беззнаковое целое число (DWORD), содержащееся в самом первом поле структуры, с ожидаемым размером структуры данного типа. Если это число окажется меньше ожидаемого размера, функция либо сообщит вызвавшей программе об ошибке, либо будет работать только с полями структуры, укладывающимися в переданный размер.

Пример:

```
// Структура описания блока
RDS_BLOCKDESCRIPTION descr;
// Присвоение размера структуры первому полю
RDS_INITSERVSIZE(descr);
// Получение описания блока
rdsGetBlockDescription(Block, &descr);
```

А.5.2.3. Макрос RDS_INTVERSION – преобразование версии РДС в целое число

Макрос RDS_INTVERSION преобразует три компонента версии РДС (старший номер, младший номер и номер сборки) в одно целое число, которое можно использовать для сравнения версий.

```
RDS_INTVERSION(
    high,          // Старший номер
    low,           // Младший номер
    build          // Номер сборки
)
```

Определение:

```
#define RDS_INTVERSION(high, low, build) \
    (((high)<<24) | ((low)<<16) | (build))
```

Параметры:

high

Старший номер версии – самое первое число в трехкомпонентном названии версии (например, в названии версии “1.0.261” старший номер равен 1). Это число занимает один байт и может принимать значения от 0 до 127. Более поздние версии РДС имеют больший старший номер.

low

Младший номер версии – второе число в трехкомпонентном названии версии (например, в названии версии “1.0.261” младший номер равен 0). Это число занимает один байт и может принимать значения от 0 до 255. При одинаковых старших номерах версий более поздние версии РДС имеют больший младший номер.

build

Номер сборки – последнее число в трехкомпонентном названии версии (например, в названии версии “1.0.261” номер сборки равен 261). Это число занимает два байта и может принимать значения от 0 до 65535. При одинаковых старших и младших номерах версий более поздние версии РДС имеют больший номер сборки.

Возвращаемое значение:

Целое тридцатидвухбитное число, соответствующее переданным компонентам номера версии.

Примечания:

Макрос RDS_INTVERSION формирует из трех компонентов номера версии одно целое число сдвигом влево старшего номера на 24 бита, младшего – на 16 бит, и объединения получившихся чисел операцией битового ИЛИ с номером сборки. Именно в таком формате номер версии РДС возвращается сервисной функцией `rdsServiceVersion` (стр. 169). Числа в этом формате можно сравнивать: если число, полученное при помощи макроса RDS_INTVERSION из одной версии больше числа, полученного из другой, значит, первая версия старше второй.

Помимо макроса RDS_INTVERSION в файле “RdsDef.h” определено четыре константы, описывающих текущую, то есть соответствующую времени создания “RdsDef.h”, версию РДС:

RDS_FVERSIONHIGH	Старший номер текущей версии РДС.
RDS_FVERSIONLOW	Младший номер текущей версии РДС.
RDS_FVERSIONBUILD	Номер сборки текущей версии РДС.
RDS_FVERSION	Полный номер текущей версии РДС (число, полученное из трех компонентов версии указанным выше образом).

Описания этих констант можно, при необходимости, подавить, описав константу RDS_NOVERSIONDEFINES перед включением файла “RdsDef.h”.

Пример:

```
#include <windows.h>
#include <RdsDef.h>
// Подготовка описаний сервисных функций
#define RDS_SERV_FUNC_BODY GetInterfaceFunctions
#include <RdsFunc.h>
// Главная функция DLL
int WINAPI DllEntryPoint(HINSTANCE /*hinst*/,
                        unsigned long reason,
                        void* /*lpReserved*/)
{ if (reason==DLL_PROCESS_ATTACH) // Загрузка DLL
  { int version;
    // Получение доступа к функциям
    GetInterfaceFunctions();
    // Проверка наличия функции получения версии
    if (rdsServiceVersion==NULL) // Нет функции
      { MessageBox(NULL, "Эта DLL не работает "
                    "без РДС", "Ошибка", MB_OK);
        return 1;
      }
    // Получение версии РДС
    version=rdsServiceVersion();
    if (version<RDS_INTVERSION(1,0,120))
      MessageBox(NULL, "Нужна версия РДС не ниже 1.0.120",
                  "Ошибка", MB_OK);
```



```

    }
    return 1;
}

```

См. также:

`rdsServiceVersion` (стр. 169),

A.5.2.4. Макрос `RDS_SETFLAG` – установка битового флага

Макрос `RDS_SETFLAG` предназначен для установки битового флага в беззнаковом целом (`DWORD`) согласно переданному логическому значению.

```

RDS_SETFLAG(
    storage,    // Переменная с набором флагов
    mask,       // Маска (битовый флаг)
    value       // Состояние флага
)

```

Определение:

```

#define RDS_SETFLAG(storage,mask,value) \
    ((storage) = (value)?((storage) | (mask)): \
    ((DWORD)((storage) & (~(mask))))))

```

Параметры:

`storage`

Переменная типа `DWORD`, в которой взводится или сбрасывается выбранный флаг.

`mask`

Устанавливаемый битовый флаг (число, в котором взведен бит в позиции флага).

`value`

`TRUE`, если флаг `mask` нужно взвести в переменной `storage`, и `FALSE`, если его нужно сбросить.

Примечания:

Макрос разворачивается в операцию присваивания значения `storage`, поэтому параметр `storage` должен быть чем-то, чему можно присваивать значения (*lvalue* в терминах языка C).

Пример:

```

DWORD flags=0;
RDS_SETFLAG(flags,RDS_CTRLCALC,TRUE);

```

См. также:

`RDS_GETFLAG` (стр. 142).

A.5.2.5. `rdsApplicationIsActive` – активность приложения РДС

Функция `rdsApplicationIsActive` проверяет, находится ли РДС на переднем плане, то есть активно ли приложение РДС, в котором вызвана эта функция.

```

BOOL RDSCALL rdsApplicationIsActive(void);

```

Тип указателя на эту функцию:

`RDS_BV`

Возвращаемое значение:

TRUE, если приложение на переднем плане, и FALSE, если нет.

A.5.2.6. rdsBadSystemTime – ошибка в системных часах

Функция rdsBadSystemTime грубо проверяет правильность работы системных часов.

```
BOOL RDSCALL rdsBadSystemTime(void);
```

Тип указателя на эту функцию:

RDS_BV

Возвращаемое значение:

TRUE, если в показаниях системных часов есть явная ошибка, FALSE в противном случае.

Примечания:

Правильность работы системных часов важна в программах, ориентирующихся в своей работе на время изменения каких-либо файлов. Например, модули автоматической компиляции часто вызывают внешний компилятор для сборки DLL с моделью блока только в том случае, если время изменения файла модели больше времени изменения файла DLL, то есть DLL устарела (см. §4.4). Если системные часы идут неправильно (например, сбрасываются при выключении питания), такая проверка будет давать непредсказуемые результаты. Чтобы избежать возможных ошибок, следует в таких случаях проверять работу часов вызовом rdsBadSystemTime.

Функция rdsBadSystemTime проверяет часы следующим образом. РДС при каждом своем завершении записывает текущее время и дату в файл rds.ini. При вызове rdsBadSystemTime текущее системное время сравнивается с этим записанным значением. Если текущее время окажется меньше, значит, с момента последнего завершения РДС часы сбросились. Такая проверка не дает полной гарантии правильности работы часов, тем не менее, в большинстве случаев ее достаточно для определения сброса часов при разряде их батарейки.

A.5.2.7. rdsBlockModalWinClose – сообщение о закрытии модального окна

Функция rdsBlockModalWinClose уведомляет РДС о закрытии модального окна, открытого средствами Windows API. Она всегда используется вместе с парной функцией rdsBlockModalWinOpen (стр. 147).

```
void RDSCALL rdsBlockModalWinClose(  
    RDS_BHANDLE Block           // Блок, открывший окно  
);
```

Тип указателя на эту функцию:

RDS_VBh

Параметр:

Block

Идентификатор блока, с которым было связано открытое модальное окно, или NULL, если оно было связано с блоком, из модели которого вызвана эта функция.

Примечания:

Если модель блока или модуль автокомпиляции открывают модальные окна не средствами РДС, а самостоятельно, при помощи функций Windows API, они должны сообщать РДС о каждом открытии и закрытии окна функциями `rdsBlockModalWinOpen` (стр. 147) и `rdsBlockModalWinClose` соответственно (см. §1.8). На время открытия модального окна РДС запрещает удаление блока, модель которого его открыла, поскольку это может привести к выгрузке DLL с моделью блока до того, как завершится процедура окна, которая тоже, как правило, находится в этой DLL. По этой причине в параметре функции `rdsBlockModalWinClose` следует передавать идентификатор блока, которому принадлежит окно, за исключением случаев, когда эта функция вызывается из модели блока – при этом РДС может самостоятельно определить блок-владелец окна и в параметре `Block` можно передать значение `NULL`.

РДС считает число вызовов функций `rdsBlockModalWinOpen` и `rdsBlockModalWinClose` для каждого блока в отдельности, поэтому если, например, из модели одного блока (или откуда-то еще с указанием идентификатора этого блока) функция `rdsBlockModalWinOpen` была вызвана три раза, этот блок можно будет удалять только после третьего вызова `rdsBlockModalWinClose`.

Пример использования функции `rdsBlockModalWinClose` приведен в §2.7.5.

См. также:

`rdsBlockModalWinOpen` (стр. 147), `rdsModalWindowExists` (стр. 164),
`rdsModalWindowMustClose` (стр. 165).

A.5.2.8. `rdsBlockModalWinOpen` – сообщение об открытии модального окна

Функция `rdsBlockModalWinOpen` уведомляет РДС об открытии модального окна средствами Windows API. Она всегда используется вместе с парной функцией `rdsBlockModalWinClose` (стр. 146).

```
void RDSCALL rdsBlockModalWinOpen(  
    RDS_BHANDLE Block           // Блок, открывший окно  
);
```

Тип указателя на эту функцию:

`RDS_VBh`

Параметр:

`Block`

Идентификатор блока, с которым связано открытое модальное окно, или `NULL`, если оно связано с блоком, из модели которого вызвана эта функция.

Примечания:

Если модель блока или модуль автокомпиляции открывают модальные окна не средствами РДС, а самостоятельно, при помощи функций Windows API, они должны сообщать РДС о каждом открытии и закрытии окна функциями `rdsBlockModalWinOpen` и `rdsBlockModalWinClose` (стр. 146) соответственно (см. §1.8). На время открытия модального окна РДС запрещает удаление блока, модель которого его открыла, поскольку это может привести к выгрузке DLL с моделью блока до того, как завершится процедура окна, которая тоже, как правило, находится в этой DLL. По этой причине в параметре функции `rdsBlockModalWinOpen` следует передавать идентификатор блока, которому принадлежит окно, за исключением случаев, когда эта функция вызывается из модели блока

– при этом РДС может самостоятельно определить блок-владелец окна и в параметре `Block` можно передать значение `NULL`.

РДС считает число вызовов функций `rdsBlockModalWinOpen` и `rdsBlockModalWinClose` для каждого блока в отдельности, поэтому если, например, из модели одного блока (или откуда-то еще с указанием идентификатора этого блока) функция `rdsBlockModalWinOpen` была вызвана три раза, этот блок можно будет удалять только после третьего вызова `rdsBlockModalWinClose`.

Пример использования функции `rdsBlockModalWinOpen` приведен в §2.7.5.

См. также:

`rdsBlockModalWinClose` (стр. 146), `rdsModalWindowExists` (стр. 164),
`rdsModalWindowMustClose` (стр. 165).

A.5.2.9. `rdsBringAppToFront` – перемещение РДС на передний план

Функция `rdsBringAppToFront` перемещает приложение РДС на передний план, то есть окна РДС становятся видимыми поверх других окон Windows.

```
void RDSCALL rdsBringAppToFront(void) ;
```

Тип указателя на эту функцию:

```
RDS_VV
```

Примечания:

Эта функция чаще всего применяется в моделях блоков, реагирующих на команды внешних приложений (см. главу 3) для того, чтобы после каких-либо действий управляющего приложения вывести РДС на передний план и показать пользователю результат этих действий.

См. также:

`rdscrtlBringAppToFront` (стр. 619).

A.5.2.10. `rdsCalcProcessIsRunning` – РДС в режиме расчета

Функция `rdsCalcProcessIsRunning` проверяет, находится ли РДС в режиме расчета.

```
BOOL RDSCALL rdsCalcProcessIsRunning(void) ;
```

Тип указателя на эту функцию:

```
RDS_BV
```

Возвращаемое значение:

`TRUE`, если РДС находится в режиме расчета, и `FALSE`, если в режимах редактирования или моделирования.

Примечания:

Несмотря на название функции, в режиме расчета она возвращает `TRUE` независимо от того, в каком потоке выполняется сам расчет: в отдельном потоке расчета, как обычно, или в главном потоке РДС (такой режим можно включить в настройках для совместимости со старыми машинами).

Пример использования функции `rdsCalcProcessIsRunning` приведен в §2.14.1.

См. также:

`rdsSystemInEditMode` (стр. 175).

A.5.2.11. `rdsCalcProcessNeverStarted` – запускался ли расчет

Функция `rdsCalcProcessNeverStarted` проверяет, запускался ли расчет в загруженной схеме.

```
BOOL RDSCALL rdsCalcProcessNeverStarted(void);
```

Тип указателя на эту функцию:

`RDS_BV`

Возвращаемое значение:

`TRUE`, если расчет никогда не запускался или схема сброшена, `FALSE` в противном случае.

Примечания:

Эта функция возвращает `TRUE` в двух случаях:

1. Схема только что загружена в РДС и расчет еще не запускался.
2. Расчет всей системы сброшен пользователем или программно – вызовом сервисной функции `rdsResetSystemState` (стр. 252) для корневой подсистемы.

Следует помнить, что при программном сбросе расчета вся система будет считаться сброшенной только при вызове `rdsResetSystemState` с идентификатором корневой подсистемы или с параметром `NULL`. Если вызвать `rdsResetSystemState` для какой-либо внутренней подсистемы схемы, часть подсистем, включая корневую, не будет затронута, при этом вся схема не будет считаться сброшенной, и функция `rdsCalcProcessNeverStarted` будет возвращать `FALSE`.

РДС запоминает факт запуска расчета после того, как все блоки схемы отреагируют на событие запуска расчета `RDS_BFM_STARTCALC` (стр. 46), поэтому в реакции на это событие можно вызвать `rdsCalcProcessNeverStarted` чтобы узнать, первый это запуск (то есть запуск расчета из исходного состояния схемы, при этом функция вернет `TRUE`) или повторный (продолжение расчета после остановки, при этом функция вернет `FALSE`). На момент первой реакции блоков на событие выполнения такта расчета (`RDS_BFM_MODEL`, стр. 40) факт запуска уже запомнен, и функция будет всегда возвращать `FALSE`.

См. также:

`rdsResetSystemState` (стр. 252), `RDS_BFM_STARTCALC` (стр. 46),
`RDS_BFM_MODEL` (стр. 40).

A.5.2.12. `rdsCancelPaste` – отмена вставки блоков из буфера обмена

Функция `rdsCancelPaste` отменяет вставку одного или нескольких блоков из буфера обмена, если она вызвана из реакции на событие, возникшее в процессе этой вставки.

```
void RDSCALL rdsCancelPaste(void);
```

Тип указателя на эту функцию:

`RDS_VV`

Примечания:

При вставке блоков в схему из буфера обмена последовательно вызывается множество реакций моделей этих блоков. К блокам подключаются модели (событие `RDS_BFM_INIT`,

стр. 38), проверяются статические переменные блоков (событие RDS_BFM_VARCHHECK, стр. 48), загружаются сохраненные параметры блоков (события RDS_BFM_LOADBIN, стр. 52, и RDS_BFM_LOADTXT, стр. 52). Если вставляемые блоки имеют автоматически компилируемые модели, также вызываются реакции связанных с ними модулей автокомпиляции: инициализация модуля, если он еще не подключен (RDS_COMPM_INIT, стр. 103), проверка возможности подключения блока к модели (RDS_COMPM_CANATTACHBLK, стр. 96), инициализация модели, если это ее первое использование в схеме (RDS_COMPM_MODELINIT, стр. 105), подключение модели к блоку (RDS_COMPM_ATTACHBLOCK, стр. 95) и т.д. Если хотя бы в одной из этих реакций будет вызвана функция rdsCancelPaste, вставка будет отменена, а все блоки и связи, которые успели вставиться до вызова функции, будут стерты с вызовом соответствующих реакций их моделей.

Чаще всего эта функция применяется для того, чтобы при вставке блоков, которые связаны с какими-либо глобальными объектами, предупредить об этом пользователя и дать ему возможность, при необходимости, отменить эту вставку. Например, стандартный модуль автоматической компиляции, входящий в состав РДС, при вставке из буфера обмена блоков с автокомпилируемыми моделями спрашивает пользователя, что сделать с моделями: создать их копии для вставляемых блоков, чтобы их можно было независимо изменять, или связать вставляемые блоки с теми же файлами моделей, которые использовались оригинальными, скопированными в буфер, блоками. Если пользователь нажмет в диалоге кнопку “Отмена”, вся вставка блоков будет отменена.

Если функция rdsCancelPaste вызывается не в момент вставки блоков из буфера обмена, вызов будет проигнорирован.

A.5.2.13. rdsChangeRegWinTitle – изменение названия немодального окна

Функция rdsChangeRegWinTitle меняет название (то есть текст в пункте меню “Окна” и на панели окон) немодального окна, зарегистрированного функцией rdsRegisterWindow (стр. 166).

```
void RDCALL rdsChangeRegWinTitle(  
    HWND Handle,           // Дескриптор окна  
    LPSTR Title            // Название окна  
);
```

Тип указателя на эту функцию:

RDS_VHwS

Параметры:

Handle

Дескриптор (HWND, см. стр. 24) немодального окна, ранее зарегистрированного функцией rdsRegisterWindow.

Title

Указатель на строку с новым названием этого окна, которое будет отображаться в соответствующем этому окну пункте меню “Окна” и на соответствующей ему кнопке панели окон.

Примечания:

После вызова этой функции текст на кнопке окна `Handle` и его название в меню “Окна” меняются на строку `Title`. Если окно с дескриптором `Handle` ранее не регистрировалось в РДС функцией `rdsRegisterWindow`, эта функция не выполнит никаких действий. Если в параметре `Title` вместо указателя на строку будет передано значение `NULL`, названием окна станет пустая строка.

См. также:

`rdsRegisterWindow` (стр. 166), `rdsUnregisterWindow` (стр. 175),
`rdsRegWinActivateNotify` (стр. 167).

A.5.2.14. `rdsEnableCommandQueue` – приостановить/продолжить выполнение очереди команд

Функция `rdsEnableCommandQueue` приостанавливает или продолжает выполнение команд, поставленных в очередь на выполнение функцией `rdsExecuteCommand` (стр. 151).

```
void RDSCALL rdsEnableCommandQueue(  
    BOOL On           // Разрешение  
);
```

Тип указателя на эту функцию:

`RDS_VB`

Параметр:

`On`

`FALSE`, если выполнение очереди команд нужно приостановить, или `TRUE`, если продолжить (по умолчанию команды в очереди выполняются).

Примечания:

Эта функция обычно используется для того, чтобы приостановить выполнение команд на время добавления в очередь большой группы таких команд. Если набор последовательных команд добавляется в очередь из потока расчета, они могут начать выполняться до того, как добавление всего набора завершится. Если такое поведение РДС нежелательно, перед добавлением команд следует вызвать функцию с параметром `FALSE`, а после – с параметром `TRUE`:

```
// Остановка очереди  
rdsEnableCommandQueue(FALSE);  
// Постановка команд в очередь  
rdsExecuteCommand(...);  
rdsExecuteCommand(...);  
  
...  
rdsExecuteCommand(...);  
// Продолжение выполнения очереди  
rdsEnableCommandQueue(TRUE);
```

См. также:

`rdsExecuteCommand` (стр. 151).

A.5.2.15. `rdsExecuteCommand` – выполнить общесистемную команду

Функция `rdsExecuteCommand` ставит команду, указанную в ее параметрах, в очередь на исполнение.

```

void RDSCALL rdsExecuteCommand(
    int Command,           // Команда RDS_SYSCMD_*
    DWORD Flags,          // Флаги
    LPSTR Param1,         // Первый параметр команды
    LPSTR Param2         // Второй параметр команды
);

```

Тип указателя на эту функцию:

RDS_VIDwSS

Параметры:

Command

Одна из констант RDS_SYSCMD_* (см. ниже), указывающая на исполняемую РДС команду.

Flags

Битовые флаги команды (зависят от значения Command).

Param1

Указатель на строку с первым параметром команды (зависит от значения Command).

Param2

Указатель на строку со вторым параметром команды (зависит от значения Command).

Примечания:

РДС поддерживает очередь команд, относящихся ко всей загруженной схеме и выполняемых строго последовательно. Если вызов rdsExecuteCommand, поставивший в очередь команду А, предшествует вызову, поставившему в очередь команду Б, команда Б не начнет выполняться до тех пор, пока не завершится выполнение команды А. Таким образом можно, например, скомандовать РДС сохранить текущую загруженную схему, а затем загрузить новую: загрузка новой не начнется, пока не будет сохранена старая.

Выполняемая команда задается в параметре Command одной из следующих констант:

<i>Command</i>	<i>Команда и ее параметры</i>
RDS_SYSCMD_STARTCALC	Запустить расчет. Параметры Flags, Param1 и Param2 не используются.
RDS_SYSCMD_LOADFILE	Загрузить схему из файла. Параметр Flags не используется. В параметре Param1 передается указатель на строку, содержащую полный путь к файлу схемы. В параметре Param2 передается либо указатель на строку, имитирующую дополнительные параметры командной строки РДС, которые могут быть разобраны блоками загруженной схемы, либо NULL.
RDS_SYSCMD_SAVEFILE	Сохранить схему в файл. Параметры Flags и Param2 не используются. В параметре Param1 передается указатель на строку, содержащую полный путь к файлу схемы.
RDS_SYSCMD_LOADTEMPLATE	Загрузить шаблон схемы из файла. Параметры Flags, Param1 и Param2 используются так же, как в команде RDS_SYSCMD_LOADFILE.

<i>Command</i>	<i>Команда и ее параметры</i>
RDS_SYSCMD_MESSAGEBOX	Вывести сообщение пользователю. В параметре Flags передаются флаги сообщения, аналогичные функции Windows API MessageBox и сервисной функции РДС rdsMessageBox (стр. 201). В параметре Param1 передается указатель на строку с текстом сообщения. В параметре Param2 передается указатель на строку с заголовком окна сообщения. Этот вызов отличается от rdsMessageBox только тем, что он не выводит сообщение пользователю немедленно, а ставит его в очередь.
RDS_SYSCMD_EDITMODE	Переводит РДС в режим редактирования. Параметры Flags, Param1 и Param2 не используются.
RDS_SYSCMD_CALCMODE	Переводит РДС в режим моделирования. Параметры Flags, Param1 и Param2 не используются.
RDS_SYSCMD_RESETCALC	Сбрасывает расчет во всей схеме. Параметры Flags, Param1 и Param2 не используются.

Следует помнить, что при загрузке новой схемы очередь команд очищается, поэтому все команды, поставленные в очередь после RDS_SYSCMD_LOADFILE и RDS_SYSCMD_LOADTEMPLATE будут проигнорированы. Например, последовательность

```
rdsExecuteCommand(RDS_SYSCMD_LOADFILE, 0,
                  "c:\\rds\\Scheme.rds", NULL);
rdsExecuteCommand(RDS_SYSCMD_STARTCALC, 0, NULL, NULL);
```

не приведет к запуску расчета после загрузки новой схемы: сразу после загрузки очередь команд будет очищена, и команда RDS_SYSCMD_STARTCALC не будет воспринята РДС.

Технически, вызов rdsExecuteCommand приводит к помещению команды и ее параметров в очередь сообщений Windows, обрабатываемых РДС, поэтому команды не будут выполняться до тех пор, пока РДС в очередной раз не войдет в цикл обработки сообщений. Если, например, модель какого-нибудь блока организует длительный цикл в главном потоке, команды не будут выполняться до завершения этого цикла.

См. также:

rdsEnableCommandQueue (стр. 151), rdsStartCalc (стр. 174),
rdsStopCalc (стр. 175), rdsResetSystemState (стр. 252),
rdsMessageBox (стр. 201).

A.5.2.16. rdsFindCmdParam – найти номер параметра командной строки

Функция rdsFindCmdParam ищет в командной строке запуска РДС параметр с заданным именем и возвращает его порядковый номер.

```
int RDSCALL rdsFindCmdParam(
    LPSTR Param,           // Текст параметра
    BOOL CaseSensitive     // Искать с учетом регистра
);
```

Тип указателя на эту функцию:

RDS_ISB

Параметры:

Param

Слово, которое нужно найти в командной строке.

CaseSensitive

TRUE, если слово Param нужно искать с учетом регистра символов, FALSE в противном случае.

Возвращаемое значение:

Порядковый номер слова в списке параметров, или -1, если такого слова не было в командной строке. Нумерация начинается с нуля.

Примечания:

Функция `rdsFindCmdParam` работает с параметрами командной строки, использовавшейся для запуска РДС с одновременной загрузкой схемы, или со списком параметров, переданных в сервисную функцию `rdsExecuteCommand` (стр. 151) для выполнения команд загрузки схемы `RDS_SYSCMD_LOADFILE` и `RDS_SYSCMD_LOADTEMPLATE`. Если схема загружена из пользовательского интерфейса РДС (например, при помощи пункта меню “Файл | Загрузить”), список параметров будет пустым.

Одним параметром командной строки считается либо слово, отделенное от остальных слов командной строки пробелами, либо текст, заключенный в двойные кавычки. В список, доступный сервисным функциям, попадают только параметры командной строки, не разобранные РДС. Например, слово “/run”, указывающее РДС, что в загруженной схеме необходимо немедленно запустить расчет, в этот список не попадет, так же как и само имя файла загружаемой схемы.

См. также:

`rdsGetCmdParamCount` (стр. 156), `rdsGetCmdParam` (стр. 155), параметры командной строки РДС (стр. 689).

A.5.2.17. `rdsGetAppInstance` – дескриптор приложения РДС

Функция `rdsGetAppInstance` позволяет модели блока или модулю автокомпиляции получить дескриптор загруженного в память приложения РДС.

```
HINSTANCE RDCALL rdsGetAppInstance (void) ;
```

Тип указателя на эту функцию:

`RDS_HiV`

Возвращаемое значение:

Дескриптор (`HINSTANCE`, см. стр. 24) главного приложения, то есть `rds.exe`. Он может использоваться в вызовах функций Windows API.

См. также:

`rdsGetAppWindowHandle` (стр. 154).

A.5.2.18. `rdsGetAppWindowHandle` – дескриптор главного окна РДС

Функция `rdsGetAppWindowHandle` возвращает дескриптор главного окна РДС, с которым должны быть связаны все остальные окна, открываемые в моделях и модулях автокомпиляции.

```
HWND RDSCALL rdsGetAppWindowHandle(void);
```

Тип указателя на эту функцию:

```
RDS_HwV
```

Возвращаемое значение:

Дескриптор (HWND, см. стр. 24) главного окна приложения.

Примечания:

Дескриптор окна, возвращаемый функцией `rdsGetAppWindowHandle`, обычно используется для привязки модальных окон, открываемых моделями блоков и модулями автоматической компиляции, к приложению РДС. Эта привязка выполняется при создании окон функциями Windows API, некоторые проблемы открытия модальных окон моделями блоков рассматривается в §1.8 и §2.2. Пример использования функции `rdsGetAppWindowHandle` приведен в §2.7.5.

Не следует путать эту функцию с функцией `rdsGetMainWindow` (стр. 158), которая возвращает дескриптор главного интерфейсного окна РДС, то есть окна с полосой меню, функциональными кнопками и вкладками панели блоков.

См. также:

`rdsGetAppInstance` (стр. 154), `rdsGetMainWindow` (стр. 158).

A.5.2.19. `rdsGetCmdParam` – параметр командной строки по номеру

Функция `rdsFindCmdParam` возвращает указатель на параметр (слово) с заданным номером в командной строке РДС.

```
LPSTR RDSCALL rdsGetCmdParam(  
    int Num,                // Номер слова  
    BOOL UpperCase         // Вернуть в верхнем регистре  
);
```

Тип указателя на эту функцию:

```
RDS_SIB
```

Параметры:

`Num`

Номер слова в командной строке. Нумерация начинается с нуля. Общее число слов можно получить вызовом `rdsGetCmdParamCount` (стр. 156).

`UpperCase`

TRUE, если необходимо вернуть слово, все символы которого переведены в верхний регистр, и FALSE, если слово нужно в том регистре, в котором оно было введено.

Возвращаемое значение:

Указатель на строку во внутренней памяти РДС, содержащую запрошенное слово в нужном регистре, или NULL, если слово с таким номером отсутствует. Вызвавшая функция не должна изменять эту строку.

Примечания:

Функция `rdsGetCmdParam` работает с параметрами командной строки, использовавшейся для запуска РДС с одновременной загрузкой схемы, или со списком параметров, переданных в сервисную функцию `rdsExecuteCommand` (стр. 151) для выполнения команд загрузки схемы `RDS_SYSCMD_LOADFILE` и

RDS_SYSCMD_LOADTEMPLATE. Если схема загружена из пользовательского интерфейса РДС (например, при помощи пункта меню “Файл | Загрузить”), список параметров будет пустым.

Одним параметром командной строки считается либо слово, отделенное от остальных слов командной строки пробелами, либо текст, заключенный в двойные кавычки. В список, доступный сервисным функциям, попадают только параметры командной строки, не разобранные РДС. Например, слово “/run”, указывающее РДС, что в загруженной схеме необходимо немедленно запустить расчет, в этот список не попадет, так же как и само имя файла загружаемой схемы.

См. также:

rdsGetCmdParamCount (стр. 156), rdsFindCmdParam (стр. 153), параметры командной строки РДС (стр. 689).

A.5.2.20. rdsGetCmdParamCount – число слов в командной строке РДС

Функция rdsGetCmdParamCount возвращает общее число не разобранных РДС параметров командной строки, которые могут анализироваться моделями блоков схемы.

```
int RDSCALL rdsGetCmdParamCount(void);
```

Тип указателя на эту функцию:

```
RDS_IV
```

Возвращаемое значение:

Общее число не разобранных РДС параметров командной строки.

Примечания:

Функция rdsGetCmdParamCount работает с параметрами командной строки, использовавшейся для запуска РДС с одновременной загрузкой схемы, или со списком параметров, переданных в сервисную функцию rdsExecuteCommand (стр. 151) для выполнения команд загрузки схемы RDS_SYSCMD_LOADFILE и RDS_SYSCMD_LOADTEMPLATE. Если схема загружена из пользовательского интерфейса РДС (например, при помощи пункта меню “Файл | Загрузить”), список параметров будет пустым.

Одним параметром командной строки считается либо слово, отделенное от остальных слов командной строки пробелами, либо текст, заключенный в двойные кавычки. В список, доступный сервисным функциям, попадают только параметры командной строки, не разобранные РДС. Например, слово “/run”, указывающее РДС, что в загруженной схеме необходимо немедленно запустить расчет, в этот список не попадет, так же как и само имя файла загружаемой схемы.

См. также:

rdsGetCmdParam (стр. 155), rdsFindCmdParam (стр. 153), параметры командной строки РДС (стр. 689).

A.5.2.21. rdsGetCustomColors – получение массива пользовательских цветов

Функция rdsGetCustomColors возвращает указатель на массив из 16 цветов, которые пользователь самостоятельно определил в диалогах выбора цвета РДС.

```
RDS_PCOLORREF RDSCALL rdsGetCustomColors(void);
```

Тип указателя на эту функцию:

RDS_pCrV

Возвращаемое значение:

Указатель на массив во внутренней памяти РДС, содержащий 16 элементов типа COLORREF (см. стр. 24).

Примечания:

Windows предоставляет пользователю возможность вручную покомпонентно задать 16 цветов, которые могут запоминаться в диалоге выбора цвета для дальнейшего использования. Задача хранения этих цветов между вызовами функции Windows API ChooseColor лежит на приложении. В РДС для этого используется внутренний массив, сохраняемый в файле rds.ini. Если модель блока вызывает функцию ChooseColor, она может передать в нее указатель на этот массив, возвращенный функцией rdsGetCustomColors.

Сервисная функция rdsCallColorDialog (стр. 197), открывающая диалог выбора цвета средствами РДС, сама работает с массивом пользовательских цветов, и в вызове rdsGetCustomColors для нее нет необходимости.

См. также:

rdsCallColorDialog (стр. 197).

A.5.2.22. rdsGetHugeDouble – получение значения-индикатора математической ошибки

Функция rdsGetHugeDouble возвращает специальное значение, используемое моделями блоков для индикации ошибки вещественных вычислений.

```
BOOL RDSCALL rdsGetHugeDouble(  
    double *pVal           // Возвращаемое значение  
);
```

Тип указателя на эту функцию:

RDS_BpD

Параметр:

pVal

Указатель на вещественную переменную двойной точности, в которую функция запишет специальное значение.

Возвращаемое значение:

TRUE, если переданный указатель pVal не равен NULL (в абсолютном большинстве случаев результат возврата этой функции можно не проверять).

Примечания:

Значение, возвращаемое этой функцией через параметр pVal, используется моделями блоков для указания на ошибку математических вычислений (переполнение, деление на ноль и т.п.). В РДС для этого используется стандартная константа HUGE_VAL из математической библиотеки языка C. Модели блоков тоже могут непосредственно использовать эту константу, но надежнее вместо нее пользоваться значением, возвращенным rdsGetHugeDouble: при этом все модели гарантированно получают одно и то же значение, не зависящее от версии РДС и версий библиотек, которые используются в DLL моделей. Пример использования функции rdsGetHugeDouble приведен в §2.5.1.

A.5.2.23. rdsGetMainWindow – дескриптор главного окна пользовательского интерфейса РДС

Функция `rdsGetMainWindow` возвращает дескриптор главного интерфейсного окна РДС, то есть окна с полосой меню, функциональными кнопками и вкладками панели блоков.

```
HWND RDSCALL rdsGetMainWindow(void);
```

Тип указателя на эту функцию:

```
RDS_HwV
```

Возвращаемое значение:

Дескриптор окна (HWND, см. стр. 24).

Примечания:

Дескриптор окна, возвращаемый функцией `rdsGetMainWindow`, может использоваться в функциях Windows API для выполнения каких-либо операций с окном РДС. Например, ему можно послать сообщение WM_CLOSE функцией `PostMessage`, что приведет к закрытию РДС.

Если отображение главного окна РДС запрещено (например, при управлении из внешнего приложения, см. §3.2), это окно, тем не менее, существует, и его дескриптор можно, как обычно, получить вызовом `rdsGetMainWindow`.

См. также:

`rdsGetAppWindowHandle` (стр. 154).

A.5.2.24. rdsGetSystemInt – получить целый системный параметр

Функция `rdsGetSystemInt` возвращает целое значение, являющееся либо общесистемным параметром, либо параметром, характеризующим состояние РДС в данный момент. Эти параметры разнородны, их объединяет только то, что все они целые и относятся к РДС и схеме в целом, а не к отдельному объекту схемы.

```
int RDSCALL rdsGetSystemInt(  
    int ValueId           // Запрашиваемый параметр  
);
```

Тип указателя на эту функцию:

```
RDS_II
```

Параметр:

ValueId

Идентификатор запрашиваемого параметра. Может принимать одно из следующих значений:

RDS_GSIBAKFILESCOUNT	Число файлов резервных копий, создаваемых при сохранении схемы (задается в настройках РДС).
RDS_GSICMDPARAMCOUNT	Число параметров командной строки, не обработанных РДС (аналог функции <code>rdsGetCmdParamCount</code> , стр. 156).
RDS_GSIDEFAULTPORT	Номер порта по умолчанию для сетевых соединений (задается в настройках РДС).
RDS_GSIINSTSTRUCTCOUNT	Общее число структур (структурных типов переменных блока), зарегистрированных в РДС.

RDS_GSIMODIFIED	Признак наличия изменений в схеме (функция возвращает 1, если схема была изменена с момента последнего сохранения, и 0, если изменений не было). Флаг наличия изменений можно установить программно вызовом функции <code>rdsSetModifiedFlag</code> (стр. 171).
RDS_GSISAVELOADACTION	Действие, вызвавшее загрузку или запись параметров блока (одна из констант <code>RDS_LS_*</code> , см. ниже).
RDS_GSISTOPPING	Признак того, что расчет в данный момент останавливается (возвращается 1, если была запрошена остановка расчета, но поток расчета еще не успел остановиться, и 0, если расчет работает в нормальном режиме или уже остановлен).
RDS_GSITICKPARITY	Четность текущего такта расчета (возврат: 0 – четный такт, 1 – нечетный). Этот параметр автоматически инвертируется каждый такт расчета и позволяет выполнять какие-либо действия через один такт.
RDS_GSIUIENABLED	Признак наличия пользовательского интерфейса РДС: возвращается 0, если интерфейс скрыт и вся работа ведется через порты вывода из внешнего приложения (см. §3.6), и 1, если интерфейс доступен пользователю.
RDS_GSIUNDOSIZE	Число шагов отмены действий пользователя при редактировании схемы (0 – отмена отключена). Задается в настройках РДС.

Возвращаемое значение:

Значение запрошенного параметра. Для некоторых параметров это просто целое числовое значение, для некоторых – 1 или 0 в зависимости от состояния РДС. Вызов `rdsGetSystemInt(RDS_GSISAVELOADACTION)` возвращает одну из следующих констант, указывающих на причину загрузки или записи параметров блока, происходящей в данный момент:

RDS_LS_ERROR	В данный момент не производится загрузка или запись параметров блока.
RDS_LS_LOADROOT	Загружаются параметры корневой подсистемы при загрузке схемы.
RDS_LS_SAVEROOT	Записываются параметры корневой подсистемы при записи схемы.
RDS_LS_LOADCONTENT	Загружаются параметры блока в составе схемы.
RDS_LS_SAVECONTENT	Записываются параметры блока в составе схемы.
RDS_LS_LOADCLIPBRD	Параметры блока загружаются из буфера обмена.
RDS_LS_SAVECLIPBRD	Параметры блока записываются в буфер обмена.
RDS_LS_LOADFROMFILE	Одиночный блок загружается из файла или вставляется из библиотеки.
RDS_LS_SAVETOFILE	Одиночный блок записывается в файл или библиотеку.
RDS_LS_LOADUNDO	Параметры блока загружаются из внутреннего буфера РДС для отмены сделанных пользователем изменений.
RDS_LS_SAVEUNDO	Параметры блока записываются во внутренний буфер РДС для обеспечения возможности отмены изменений, сделанных пользователем.

RDS_LS_LOADAUTOCOMP	Параметры блока загружаются после автоматической компиляции его модели.
RDS_LS_SAVEAUTOCOMP	Параметры блока записываются перед автоматической компиляцией его модели.
RDS_LS_LOADTAGGED	Блок схемы загружается в специальном формате (см. §3.5).
RDS_LS_SAVETAGGED	Блок схемы записывается в специальном формате.

Пример использования `rdsGetSystemInt(RDS_GSISAVELOADACTION)` для выяснения причины сохранения параметров блока приведен в §2.10.4.

См. также:

`rdsGetSystemPath` (стр. 160).

A.5.2.25. `rdsGetSystemPath` – получить системную строку

Функция `rdsGetSystemPath` возвращает указатель на одну из строк, описывающих общие параметры или параметры, относящиеся к загруженной схеме в целом. Строки находятся во внутренней памяти РДС.

```
LPSTR RDSCALL rdsGetSystemPath(
    int ValueId          // Запрашиваемая строка
);
```

Тип указателя на эту функцию:

RDS_SI

Параметр:

ValueId

Идентификатор запрашиваемой строки. Может принимать одно из следующих значений:

RDS_GSPAPPEXE	Полный путь к исполняемому файлу РДС <code>rds.exe</code> .
RDS_GSPAPPPATH	Полный путь к папке, в которой находится исполняемый файл РДС <code>rds.exe</code> . Отличается от <code>RDS_GSPAPPEXE</code> отсутствием имени файла в конце пути.
RDS_GSPBAKFILEEXT	Расширение файлов резервных копий схем (по умолчанию – “~rds”, может изменяться пользователем в настройках РДС). Расширение возвращается без символа точки, отделяющей его от имени.
RDS_GSPBLOCKLIBPATH	Полный путь к библиотеке блоков (по умолчанию – папка “Library\” внутри папки РДС).
RDS_GSPBLOCKPANELPATH	Полный путь к панели блоков (по умолчанию – папка “Panel\” внутри папки РДС).
RDS_GSPDEFAULTHOST	Установленное в сетевых настройках РДС имя сервера по умолчанию (см. стр. 389).
RDS_GSPDLLPATH	Полный путь к папке стандартных DLL (по умолчанию – папка “Dll\” внутри папки РДС).
RDS_GSPINCLUDEPATH	Полный путь к папке файлов-заголовков (по умолчанию – папка “Include\” внутри папки РДС).
RDS_GSPINIPATH	Полный путь к папке INI-файлов (по умолчанию совпадает с основной папкой РДС, в которой находится исполняемый файл <code>rds.exe</code>).

RDS_GSPMODELSPATH	Полный путь к папке стандартных автокомпилируемых моделей (по умолчанию – папка “Models\” внутри папки РДС).
RDS_GSPSYSTEMFILE	Полный путь к файлу загруженной в данный момент схемы. Если создана новая схема, вместо полного пути возвращается только имя файла (без пути, обычно “Untitled.rds”). При этом после первого же ее сохранения будет возвращаться полный путь к файлу, в который эта новая схема сохранена.
RDS_GSPSYSTEMFULLPATH	Полный путь к файлу загруженной в данный момент схемы. Если схема только что создана и еще не сохранялась, возвращается пустая строка (этим RDS_GSPSYSTEMFULLPATH отличается от RDS_GSPSYSTEMFILE).
RDS_GSPTEMPLATEPATH	Полный путь к папке шаблонов схем (по умолчанию – папка “Template\” внутри папки РДС).
RDS_GSPTEMPPPATH	Полный путь к собственной временной папке РДС (по умолчанию – папка “Temp\” внутри папки РДС).

Возвращаемое значение:

Указатель на запрошенную строку во внутренней памяти РДС. Вызвавшая функция модели не должна как-либо изменять эту строку. Если параметру ValueId не соответствует никакая из системных строк (он не равен одной из констант RDS_GSP*), возвращается указатель на пустую строку. Таким образом, эта функция *всегда* возвращает указатель на какую-либо существующую строку.

Примечания:

Большинство возвращаемых функцией rdsGetSystemPath строк связаны с путями к различным папкам РДС, отсюда и название функции. Разработчику модели блока или модуля автокомпиляции не следует полагаться на то, что в копии РДС, установленной на машине пользователя, все пути будут стандартными. Например, если в модуле автокомпиляции требуется получить путь к файлам заголовков для включения их в формируемый текст модели блока на языке С, будет ошибкой получить путь к папке установки РДС вызовом rdsGetSystemPath(RDS_GSPAPPPATH) и добавить к нему строку “Include\”. Размещение некоторых папок может быть изменено в настройках или специальными параметрами командной строки РДС, поэтому для получения пути к файлам заголовков следует использовать вызов rdsGetSystemPath(RDS_GSPINCLUDEPATH) – он всегда возвращает путь, указанный в текущих настройках.

Пример использования функции rdsGetSystemPath приведен в §4.4.

См. также:

rdsGetSystemInt (стр. 158).

A.5.2.26. rdsIsValidVarName – проверка синтаксиса имени переменной

Функция rdsIsValidVarName проверяет переданное ей имя переменной на соответствие правилам языка С. Имя, соответствующее этим правилам, заведомо может использоваться в качестве имени переменной или блока в РДС.

```

BOOL RDSCALL rdsIsValidVarName(
    LPSTR VarName,           // Имя переменной
    int MaxLength           // Max длина или 0
);

```

Тип указателя на эту функцию:

RDS_BSI

Параметры:

VarName

Указатель на строку с проверяемым именем переменной.

MaxLength

Максимально допустимое число символов в имени переменной или 0, если проверять длину имени не нужно.

Возвращаемое значение:

TRUE, если переданное в параметре VarName имя прошло проверку, FALSE в противном случае.

Примечания:

Чтобы соответствовать правилам языка C, переданное в параметре VarName имя не должно содержать символов, отличных от латинских букв, цифр и знака подчеркивания, при этом оно не должно начинаться с цифры. Если в параметре MaxLength передано положительное число, имя переменной также не должно быть длиннее MaxLength символов.

Чаще всего функция rdsIsValidVarName используется в модулях автокомпиляции для проверки допустимости вводимых пользователем имен переменных.

A.5.2.27. rdsListVarTypes – список названий типов переменных

Функция rdsListVarTypes возвращает динамически созданную строку, содержащую перечисление названий запрошенных типов переменных РДС, разделенных символом перевода строки “\n” (код 10). Такой список можно показывать пользователю для выбора типа переменной.

```

LPSTR RDSCALL rdsListVarTypes(
    DWORD Flags,           // Битовые флаги RDS_HVAR_F*
    LPSTR ExStruct         // Исключаемая из списка структура
);

```

Тип указателя на эту функцию:

RDS_SDwS

Параметры:

Flags

Битовые флаги, указывающие функции, какие типы переменных (см. стр. 49) нужно включить в список:

RDS_HVAR_FARRAYS	матрицы;
RDS_HVAR_F1INDEX	массивы (только если установлен флаг RDS_HVAR_FARRAYS, разрешить массивы не разрешая матриц нельзя);
RDS_HVAR_FCHAR	однобайтовые целые (char);
RDS_HVAR_FDOUBLE	вещественные двойной точности (double);
RDS_HVAR_FFLOAT	вещественные одинарной точности (float);

RDS_HVAR_FINT	четыребайтовые целые (int);
RDS_HVAR_FLOGICAL	однобайтовые логические;
RDS_HVAR_FRUNTIME	переменные, тип которых может изменяться в процессе работы (произвольный тип);
RDS_HVAR_FSHORT	двухбайтовые целые (short int);
RDS_HVAR_F SIGNAL	сигналы (однобайтовые логические специального вида, см. §1.5 и §2.5.2);
RDS_HVAR_FSTRING	строки;
RDS_HVAR_FSTRUCT	структуры.

Для удобства в “RdsDef.h” описано четыре дополнительных константы, представляющих собой объединение приведенных выше флагов для часто встречающихся случаев. Эти константы можно использовать в параметре функции вместо флагов:

RDS_HVAR_FALL	все типы переменных;
RDS_HVAR_FALLNS	все типы, кроме сигналов;
RDS_HVAR_FALLPLAIN	все простые типы, то есть все, кроме структур, матриц/массивов, строк и произвольных типов;
RDS_HVAR_FALLPLAINNS	все простые типы, кроме сигналов.

Передаваемые в этом параметре флаги частично совпадают с флагами, используемыми вспомогательным объектом редактирования переменных в функции rdsVSExecuteEditor (стр. 438).

ExStruct

Указатель на строку с именем структуры, которая должна быть исключена из формируемого списка. Если в список нужно включить все структуры, в этом параметре передается NULL. При исключении какой-либо структуры вместе с ней из списка исключаются все структуры, в которых эта используется как тип одного из полей.

Возвращаемое значение:

Указатель на сформированную в динамической памяти строку, содержащую названия запрошенных типов, принятые в РДС. Именно такие названия показываются в выпадающих списках стандартных окон редактирования переменных и структур. В случае ошибки (например, если в параметре Flags не установлен ни один флаг типа), функция возвращает NULL.

Примечания:

Списки типов, возвращаемых функцией rdsListVarTypes, чаще всего используются моделями блоков и модулями автокомпиляции, позволяющими пользователю задавать структуру переменных блока. Элементы списка, как и в других сервисных функциях РДС, отделяются друг от друга символом перевода строки (“\n”, код символа 10 в десятичной системе или 0x0A в шестнадцатеричной). Выбранный пользователем элемент списка можно преобразовать в константу типа (стр. 49) при помощи функции rdsProcessText (стр. 192), передав ей код операции RDS_PT_VARTYPECHAR.

В параметре ExStruct можно передать имя структуры которая (вместе со всеми использующими ее структурами) должна быть исключена из формируемого списка. Например, если создается пользовательский интерфейс для редактирования полей какой-либо структуры, сама эта структура должна быть исключена из списка возможных типов полей – структура не может быть полем самой себя. По этой же причине необходимо исключить из списка и другие структуры, в которые входит эта.

Функция создает строку в динамической памяти, поэтому после использования эта строка **обязательно** должна быть освобождена вызовом `rdsFree` (стр. 187).

См. также:

Типы переменных РДС (стр. 49), `rdsProcessText` (стр. 192), `rdsFree` (стр. 187), `rdsVSExecuteEditor` (стр. 438).

A.5.2.28. `rdsMainWindowVisible` – видимость главного окна РДС

Функция `rdsMainWindowVisible` проверяет, отображается ли главное окно РДС (окно с меню, кнопками и панелью блоков) на экране.

```
BOOL RDSCALL rdsMainWindowVisible(void);
```

Тип указателя на эту функцию:

```
RDS_BV
```

Возвращаемое значение:

TRUE, если окно видимо и пользователь может с ним работать, и FALSE, если окно скрыто от пользователя.

Примечания:

Видимость главного окна РДС может задаваться в командной строке (параметр “/hide” скрывает главное окно, см. стр. 690), сервисной функцией `rdsShowMainWindow` (стр. 174) или функцией `rdscrtlShowMainWindow` (стр. 614) при управлении РДС из другого приложения.

См. также:

`rdsShowMainWindow` (стр. 174), `rdscrtlShowMainWindow` (стр. 614), параметры командной строки РДС (стр. 689).

A.5.2.29. `rdsModalWindowExists` – наличие открытых модальных окон

Функция `rdsModalWindowExists` возвращает TRUE, если в данный момент в РДС открыто какое-либо модальное окно.

```
BOOL RDSCALL rdsModalWindowExists(void);
```

Тип указателя на эту функцию:

```
RDS_BV
```

Возвращаемое значение:

TRUE, если в РДС есть открытое модальное окно, и FALSE, если все открытые окна немодальные.

Примечания:

К модальным окнам, наличие которых проверяется этой функцией, относятся:

- собственные модальные окна РДС;
- стандартные сообщения пользователю (выводимые функцией `rdsMessageBox`);
- модальные окна, открываемые сервисными функциями РДС;
- любое другое окно, перед открытием которого вызвана функция `rdsBlockModalWinOpen` (см. стр. 147).

Следует помнить, что для того, чтобы открытие модальных окон функциями Windows API учитывалось РДС и отражалось на результате вызова этой функции, перед открытием

такого окна следует вызывать функцию `rdsBlockModalWinOpen` (см. стр. 147), а после закрытия – `rdsBlockModalWinClose` (стр. 146). Проблемы, которые могут возникнуть при ненадлежащем открытии модальных окон функциями Windows API, описаны в §1.8 и §2.7.5.

Вызов `rdsModalWindowExists` обычно применяется для того, чтобы заблокировать какие-либо действия, пока на экране находится модальное окно. Например, если какой-либо блок сообщает что-то пользователю по таймеру, и нет острой необходимости выводить эти сообщения в точно определенные моменты времени, вывод сообщения лучше не производить поверх открытого модального окна – это может сбить пользователя с толку, поскольку открытое окно не будет иметь ничего общего с появившимся поверх него сообщением.

См. также:

`rdsBlockModalWinOpen` (стр. 147), `rdsBlockModalWinClose` (стр. 146),
`rdsModalWindowMustClose` (стр. 165).

A.5.2.30. `rdsModalWindowMustClose` – проверка принудительного закрытия модальных окон

Функция `rdsModalWindowMustClose` возвращает `TRUE`, если в данный момент РДС принудительно закрывает все модальные окна.

```
BOOL RDSCALL rdsModalWindowMustClose(void) ;
```

Тип указателя на эту функцию:

```
RDS_BV
```

Возвращаемое значение:

`TRUE`, если РДС принудительно закрывает модальные окна, `FALSE` в противном случае.

Примечания:

Принудительное закрытие модальных окон применяется при управлении РДС из внешнего приложения (см. главу 3). При открытых модальных окнах некоторые действия выполнить невозможно (например, нельзя удалить блок, модель которого открыла модальное окно, пока это окно не будет закрыто), поэтому иногда возникает необходимость в принудительном закрытии окон. Для этого РДС посылает модальному окну верхнего уровня сообщение Windows `WM_CLOSE` (команда закрытия окна) до тех пор, пока все модальные окна не будут закрыты. Если процедура окна написана так, что при получении сообщения `WM_CLOSE` оно выводит пользователю запрос о необходимости сохранения данных, это может привести к появлению бесконечного цикла: модальное окно получает сообщение `WM_CLOSE` и выводит запрос, окно запроса тоже получает сообщение `WM_CLOSE` (что равносильно нажатию кнопки “Отмена”) и возвращает управление модальному окну, не закрывая его, модальное окно снова получает сообщение `WM_CLOSE` и т.д. Функция `rdsModalWindowMustClose` позволяет отличить принудительное закрытие окна от нормального: если она вернет `TRUE`, окно следует закрыть без запросов к пользователю, чтобы описанный выше цикл не возник.

Проблемы, связанные с открытием модальных окон в моделях блоков, описаны в §1.8 и §2.7.5.

См. также:

rdsBlockModalWinOpen (стр. 147), rdsBlockModalWinClose (стр. 146),
rdsModalWindowExists (стр. 164).

A.5.2.31. rdsRegisterWindow – регистрация немодального окна

Функция rdsRegisterWindow добавляет указанное немодальное окно в меню “Окна” и на панель с кнопками окон, чтобы пользователь мог активировать это окно, выбрав его название в меню или нажав соответствующую ему кнопку панели.

```
BOOL RDSCALL rdsRegisterWindow(  
    HWND Handle,           // Дескриптор окна  
    LPSTR Title,          // Название окна  
    HBITMAP HBmp,         // Дескриптор картинки окна  
    HINSTANCE HInst,      // Дескриптор модуля с картинкой  
    LPSTR ResName         // Имя ресурса с картинкой  
);
```

Тип указателя на эту функцию:

RDS_BHwSHbHiS

Параметры:

Handle

Дескриптор регистрируемого немодального окна. Именно это окно РДС будет помещать на передний план при выборе пользователем его названия в меню “Окна” или нажатии кнопки на панели окон.

Title

Указатель на строку с названием окна. Это название будет добавлено в меню “Окна” и на панель кнопок окон РДС.

HBmp

Дескриптор (HBITMAP) растрового рисунка, который должен быть помещен на кнопку окна слева от названия. Может иметь значение NULL, тогда растровый рисунок задается параметрами HInst и ResName.

HInst

Дескриптор (HINSTANCE) модуля, из ресурсов которого должен быть загружен растровый рисунок для кнопки окна. Значение этого параметра используется только тогда, когда параметр HBmp равен NULL, в противном случае значение HInst может быть любым (можно передавать NULL). Понятие ресурсов и их использования рассматривается в описании Windows API.

ResName

Имя ресурса, в котором находится растровый рисунок для кнопки окна. Значение этого параметра используется только тогда, когда параметр HBmp равен NULL, а параметр HInst не равен NULL. В противном случае значение ResName может быть любым (можно передавать NULL).

Возвращаемое значение:

TRUE, если окно успешно зарегистрировано, FALSE в противном случае. Как правило, результат возврата этой функции можно не проверять – отказ в регистрации окна обычно связан с недопустимыми параметрами функции (например, если параметр Handle равен NULL), что легко обнаруживается и исправляется на этапе разработки программы.

Примечания:

Если модель блока или модуль автоматической компиляции открывает немодальные окна (то есть окна, разрешающие пользователю переключаться в другие окна), желательно регистрировать их в РДС функцией `rdsRegisterWindow`, чтобы пользователю проще было вызывать их на передний план. При регистрации обязательно указывается дескриптор окна `Handle`, используемый РДС для перемещения окна на передний план, и название окна `Title`, по которому пользователь будет его опознавать в меню и на панели кнопок окон (обычно название окна делают одинаковым с его заголовком, чтобы пользователю было проще сопоставить их). Кроме того, при регистрации может быть задан растровый рисунок, который будет изображаться на кнопке этого окна слева от названия. Рисунок должен иметь размеры 16 x 16 точек, причем цвет его левой нижней точки будет считаться прозрачным, то есть этот цвет при изображении рисунка будет заменен на цвет фона кнопки. Сам рисунок задается тремя параметрами функции: `Hbmp`, `HInst` и `ResName`. Возможны следующие сочетания параметров:

<i>Hbmp</i>	<i>HInst</i>	<i>ResName</i>	<i>Рисунок</i>
NULL	NULL	Не важно (NULL)	Кнопка окна не будет иметь рисунка.
NULL	Дескриптор модуля	Имя ресурса в модуле	Рисунок будет загружен из ресурса с именем <code>ResName</code> , находящегося в модуле <code>HInst</code> .
Дескриптор рисунка	Не важно (NULL)	Не важно (NULL)	Рисунок будет скопирован из рисунка с дескриптором <code>Hbmp</code> .

Как правило, при загрузке рисунка из ресурса этот ресурс размещают в одной DLL с моделью блока или модулем автокомпиляции, и в качестве дескриптора модуля `HInst` указывают дескриптор той библиотеки, из которой вызвана функция `rdsRegisterWindow`. Однако, это не является обязательным требованием: в `HInst` можно передать дескриптор любого модуля, загруженного в память на момент вызова. После регистрации окна этот модуль можно, при желании, немедленно выгрузить – для дальнейшей работы он не нужен.

После регистрации окна функцией `rdsRegisterWindow` процедура этого окна обязательно должна сообщать РДС об активации этого окна вызовом `rdsRegWinActivateNotify` (стр. 167), иначе кнопка этого окна не будет переходить в нажатое состояние, а его название в меню “Окна” не будет помечаться. При закрытии окна его необходимо убрать из меню и с панели вызовом `rdsUnregisterWindow` (стр. 175).

См. также:

`rdsUnregisterWindow` (стр. 175), `rdsChangeRegWinTitle` (стр. 150),
`rdsRegWinActivateNotify` (стр. 167).

A.5.2.32. `rdsRegWinActivateNotify` – уведомление об активации зарегистрированного окна

Функция `rdsRegWinActivateNotify` уведомляет РДС об активации пользователем немодального окна, зарегистрированного ранее функцией `rdsRegisterWindow` (стр. 166). После этого уведомления кнопка окна переходит в нажатое состояние, а его название в меню “Окна” помечается.

```
void RDSCALL rdsRegWinActivateNotify(  
    HWND Handle           // Дескриптор окна  
) ;
```

Тип указателя на эту функцию:

RDS_VHw

Параметр:

Handle

Дескриптор активированного немодального окна, использованный при его регистрации функцией `rdsRegisterWindow`.

См. также:

`rdsRegisterWindow` (стр. 166), `rdsChangeRegWinTitle` (стр. 150),
`rdsUnregisterWindow` (стр. 175).

A.5.2.33. `rdsRunWithoutEvents` – приостановить обработку некоторых некритических событий

Функция `rdsRunWithoutEvents` временно останавливает реакцию РДС на некритические события: обновление окон, реакцию на мышь и т.п. (обработка клавиатуры не прерывается). Эта же функция используется и для возобновления этой реакции.

```
BOOL RDSCALL rdsRunWithoutEvents(  
    BOOL Suspend           // Остановить/продолжить обработку  
);
```

Тип указателя на эту функцию:

RDS_BB

Параметр:

Suspend

TRUE, если необходимо остановить обработку некритических событий, и FALSE, если ее нужно возобновить.

Возвращаемое значение:

TRUE, если после вызова обработка некритических событий разрешена, FALSE – если запрещена.

Примечания:

Остановка реакций на некритические события применяется для кратковременного ускорения работы РДС. Если, например, необходимо рассчитать длительный переходный процесс в какой-либо системе, можно на время этого расчета остановить обновление окон, которое, как правило, занимает много процессорного времени.

При каждом вызове `rdsRunWithoutEvents(TRUE)` внутренний счетчик вызовов этой функции увеличивается на единицу, а при каждом вызове `rdsRunWithoutEvents(FALSE)` – уменьшается (но не может стать отрицательным). Обработка некритических событий разрешена только при нулевом значении этого счетчика. Таким образом, допускаются вложенные вызовы `rdsRunWithoutEvents` – главное, чтобы число вызовов этой функции с параметром TRUE совпадало с числом ее вызовов с параметром FALSE, иначе обработка некритических событий останется запрещенной до загрузки в РДС новой схемы.

Если необходимо разрешить обработку событий независимо от того, сколько раз до этого функция `rdsRunWithoutEvents` была вызвана с параметром TRUE, можно вызывать ее в цикле с параметром FALSE до тех пор, пока она не вернет TRUE (то есть пока обработка событий не будет разрешена).

См. также:

`rdsSetSystemUpdate` (стр. 172).

A.5.2.34. `rdsServiceVersion` – версия РДС

Функция `rdsServiceVersion` возвращает версию РДС в виде одного целого числа.

```
int RDSCALL rdsServiceVersion(void) ;
```

Тип указателя на эту функцию:

`RDS_IV`

Возвращаемое значение:

Целое число, скомпонованное из трех компонентов номера версии сдвигом влево старшего номера на 24 бита, младшего – на 16 бит, и объединения получившихся чисел операцией битового ИЛИ с номером сборки (так же работает макрос `RDS_INTVERSION`, см. стр. 143). Такие числа можно использовать для сравнения версий – чем больше число, тем старше версия.

Примечания:

Номер версии, возвращаемый функцией `rdsServiceVersion`, чаще всего используется в главной функции DLL (см. A.2.2) для вывода сообщения пользователю в том случае, если версия РДС, в которую загружена DLL, слишком старая, и в ней отсутствуют необходимые сервисные функции.

См. также:

`RDS_INTVERSION` (стр. 143).

A.5.2.35. `rdsSetExclusiveCalc` – выделенный расчет подсистемы

Функция `rdsSetExclusiveCalc` включает или выключает выделенный расчет блоков указанной в параметрах подсистемы. При выделенном расчете блоки всех подсистем, внешних по отношению к указанной, исключаются из цикла расчета.

```
BOOL RDSCALL rdsSetExclusiveCalc(  
    RDS_BHANDLE System,    // Подсистема  
    BOOL On                // Включить/выключить выделенный расчет  
);
```

Тип указателя на эту функцию:

`RDS_BBhB`

Параметры:

`System`

Идентификатор подсистемы, для блоков которой нужно включить или выключить выделенный расчет.

`On`

`TRUE`, если выделенный расчет нужно включить, `FALSE` – если выключить.

Возвращаемое значение:

`TRUE`, если выделенный расчет для указанной подсистемы удалось успешно включить или выключить, `FALSE` при невозможности сделать это.

Примечания:

Исходно в расчете, то есть в вызове моделей блоков с параметром `RDS_BFM_MODEL` (стр. 40) и `RDS_BFM_PREMODEL` (стр. 41), который производится циклически в режиме расчета, участвуют все простые блоки загруженной схемы. При включении выделенного расчета какой-либо подсистемы все блоки, не находящиеся непосредственно в этой подсистеме и во вложенных в нее (на любом уровне вложенности) подсистемах исключаются из расчета. Это позволяет бороться с задержками, возникающими в длинных цепочках блоков, если это необходимо для логики работы схемы (пример использования выделенного расчета приведен в §2.14.4).

После включения выделенного расчета подсистемы можно либо выключить его, либо включить выделенный расчет для одной из ее внутренних подсистем – тогда число блоков, участвующих в расчете, будет ограничено еще сильнее. Если для подсистемы А включен выделенный расчет, включить его для подсистемы Б получится только в том случае, если Б находится внутри А, в противном случае `rdsSetExclusiveCalc` вернет `FALSE`. Таким образом, включение и выключение выделенного расчета производится строго иерархически: при выключении в расчете начинают принимать участие те блоки, которые участвовали в нем до включения. Например, если подсистема Б находится внутри подсистемы А, возможна следующая последовательность вызовов:

Параметры <i>rdsSetExclusiveCalc</i>	Блоки, участвующие в расчете после вызова
Исходное состояние	В расчете принимают участие все блоки схемы, включая блоки в подсистемах А и Б.
(А, TRUE)	В расчете принимают участие блоки подсистемы А, включая блоки вложенной в нее подсистемы Б.
(Б, TRUE)	В расчете принимают участие только блоки подсистемы Б.
(Б, FALSE)	В расчете снова принимают участие блоки подсистемы А, включая блоки вложенной в нее подсистемы Б.
(А, FALSE)	В расчете снова принимают участие все блоки схемы.

Выключение выделенного расчета желательно производить в порядке, обратном к включению, как в приведенном выше примере. Если выключить выделенный расчет не для той подсистемы, для которой он был включен в последний раз, выключение будет произведено не немедленно, а в тот момент, когда до нее дойдет очередь:

Параметры <i>rdsSetExclusiveCalc</i>	Блоки, участвующие в расчете после вызова
Исходное состояние	В расчете принимают участие все блоки схемы, включая блоки в подсистемах А и Б
(А, TRUE)	В расчете принимают участие блоки подсистемы А, включая блоки вложенной в нее подсистемы Б.
(Б, TRUE)	В расчете принимают участие только блоки подсистемы Б.
(А, FALSE)	В расчете, как и раньше, принимают участие только блоки подсистемы Б. Факт выключения расчета для подсистемы А запомнен, но он никак не повлиял на включенный выделенный расчет подсистемы Б.

<i>Параметры rdsSetExclusiveCalc</i>	<i>Блоки, участвующие в расчете после вызова</i>
(B, FALSE)	В расчете снова принимают участие все блоки схемы. Выделенный расчет для подсистемы Б выключился, а за ним автоматически выключился и выделенный расчет подсистемы А, поскольку он был выключен непосредственно перед ним.

См. также:

RDS_BFM_MODEL (стр. 40), RDS_BFM_PREMODEL (стр. 41).

A.5.2.36. rdsSetModifiedFlag – установка флага наличия изменений в схеме

Функция rdsSetModifiedFlag программно устанавливает флаг наличия изменений в загруженной в данный момент схеме.

```
void RDSCALL rdsSetModifiedFlag(
    BOOL Modified          // Флаг наличия изменений
);
```

Тип указателя на эту функцию:

RDS_VB

Параметр:

Modified

TRUE – взвести флаг, FALSE – сбросить его.

Примечания:

При вызове rdsSetModifiedFlag с параметром TRUE загруженная система будет считаться измененной, и РДС предупредит об этом пользователя, если последний попытается выйти из программы или загрузить другую схему без сохранения данной. При вызове функции с параметром FALSE схема будет считаться не измененной, независимо от фактического наличия изменений в схеме.

Чаще всего эта функция используется для взведения флага наличия изменений после программного изменения схемы. При изменении схемы пользователем флаг наличия изменений взводится автоматически, и автоматически сбрасывается при сохранении схемы. Пример использования функции rdsSetModifiedFlag приведен в §2.12.8.

См. также:

rdsGetSystemInt (RDS_GSIMODIFIED) (стр. 158).

A.5.2.37. rdsSetSystemInt – установка целого системного параметра

Функция rdsSetSystemInt устанавливает целое значение для различных системных параметров.

```
void RDSCALL rdsSetSystemInt(
    int Param,          // Идентификатор параметра
    int Value           // Значение параметра
);
```

Тип указателя на эту функцию:

RDS_VII

Параметры:

Param

Идентификатор устанавливаемого параметра (одна из констант RDS_SSI*, см. ниже).

Value

Значение устанавливаемого параметра.

Идентификаторы параметров:

RDS_SSIFASTTEXTSAVE

Способ сохранения в текстовом формате. Этот параметр может устанавливаться только в реакции блока на событие RDS_BFM_SAVETXT (стр. 54), во всех остальных реакциях вызов rdsSetSystemInt с этим идентификатором игнорируется. Параметр может принимать одно из следующих значений:

- 0 Обычное сохранение в текстовом формате: ко всем строкам, переданным в РДС для записи, добавляются отступы слева, если это необходимо. Это значение параметр получает автоматически перед каждым вызовом блока для сохранения данных.
- 1 Быстрое сохранение – отступы не добавляются. Это значение имеет смысл устанавливать при сохранении больших объемов текста, чтобы повысить скорость сохранения за счет худшего форматирования получившегося файла схемы. Если не предполагается, что файл схемы будет корректироваться вручную, его форматирование не играет роли – на загрузку файлов в РДС оно не влияет.

RDS_SSIWAITCURSOR

Установить курсор мыши “песочные часы”. Параметр может принимать одно из следующих значений:

- 1 Установить курсор “песочные часы”, его внешний вид не будет зависеть от находящегося под ним элемента.
- 0 Вернуть предыдущий, использовавшийся до установки значения 1, вид курсора мыши.
- 1 Немедленно установить обычный курсор мыши, его внешний вид будет зависеть от находящегося под ним элемента управления.

При каждом вызове функции rdsSetSystemInt с параметрами (RDS_SSIWAITCURSOR, 1) РДС устанавливает курсор “песочные часы” (что обычно указывает на выполнение какой-либо длительной операции, во время которой пользователь не может работать с программой) и увеличивает внутренний счетчик. При вызове функции с параметрами (RDS_SSIWAITCURSOR, 0) внутренний счетчик уменьшается, и, когда он достигнет нуля, РДС вернет курсору стандартную форму. Таким образом, пары вызовов установки/отмены “песочных часов” можно безопасно вкладывать друг в друга. Если необходимо немедленно вернуть курсору его обычную форму, следует вызвать функцию с параметрами (RDS_SSIWAITCURSOR, –1).

См. также:

rdsGetSystemInt (стр. 158).

А.5.2.38. rdsSetSystemUpdate – разрешить/запретить обновление вспомогательных данных

Функция rdsSetSystemUpdate разрешает или запрещает обновление вспомогательных данных, необходимых для правильной работы схемы.

```
void RDSCALL rdsSetSystemUpdate(
    BOOL On          // Разрешить/запретить
);
```

Тип указателя на эту функцию:

RDS_VB

Параметр:

On

TRUE, если обновление нужно разрешить (это режим работы по умолчанию), и FALSE, если его нужно временно запретить.

Примечания:

Для загруженной в память схемы РДС создает и поддерживает в актуальном состоянии достаточно большое количество вспомогательных структур, обеспечивающих правильную передачу данных по связям, работу в режиме расчета и т.п. Если в схему вносятся серьезные изменения – например, добавляется и удаляется большое количество блоков и связей – не имеет смысла обновлять эти структуры после каждого небольшого изменения. Если на время внесения изменений заблокировать обновление вспомогательных структур РДС вызовом `rdsSetSystemUpdate(FALSE)`, а потом снова разрешить их вызовом `rdsSetSystemUpdate(TRUE)`, можно получить весьма существенный выигрыш в скорости работы, поскольку все вспомогательные данные обновятся один раз в момент вызова `rdsSetSystemUpdate(TRUE)`. Следует только помнить, что после внесения изменений **обязательно** снова разрешить обновление структур, иначе схема придет в неработоспособное состояние.

Для пар вызовов `rdsSetSystemUpdate` не поддерживается внутренний счетчик, поэтому их **нельзя** вкладывать друг в друга: два последовательных вызова `rdsSetSystemUpdate(FALSE)` воспринимаются РДС как один, и следующий же вызов `rdsSetSystemUpdate(TRUE)` снова разрешит обновление вспомогательных структур.

Пример использования функции `rdsSetSystemUpdate` приведен в §2.16.2.

См. также:

`rdsRunWithoutEvents` (стр. 168).

A.5.2.39. `rdsShowBlockPanelTab` – управление вкладками панели блоков

Функция `rdsShowBlockPanelTab` позволяет включать и выключать видимость отдельных вкладок панели блоков.

```
void RDSCALL rdsShowBlockPanelTab(
    int Op,                // Операция с вкладкой
    LPSTR TabName          // Имя вкладки
);
```

Тип указателя на эту функцию:

RDS_VIS

Параметры:

Op

Действие, выполняемое с вкладкой, имя которой передано в параметре `TabName`. Может принимать одно из следующих значений:

RDS_BLOCKPANELOP_HIDE	Скрыть вкладку <code>TabName</code> .
RDS_BLOCKPANELOP_SHOW	Показать вкладку <code>TabName</code> .
RDS_BLOCKPANELOP_HIDEEXCEPT	Скрыть все вкладки кроме <code>TabName</code> .

RDS_BLOCKPANELOP_SHOWALL	Показать все вкладки.
RDS_BLOCKPANELOP_SELECT	Сделать вкладку TabName текущей.

TabName

Указатель на строку с именем вкладки (имя вкладки панели блоков – это имя папки, в которой находятся файлы ее блоков, см. §1.1).

Примечания:

Управление вкладками панели блоков обычно используется в тех случаях, когда РДС входит в состав другого приложения и используется как редактор каких-либо схем. При этом пользователю обычно показывают только те вкладки, на которых размещаются блоки, которые можно добавлять в редактируемую схему.

A.5.2.40. rdsShowMainWindow – управление видимостью главного окна РДС

Функция rdsShowMainWindow позволяет скрыть или показать пользователю главное окно РДС (окно с полосой меню, функциональными кнопками и панелью блоков).

```
void RDSCALL rdsShowMainWindow(
    BOOL Show          // Показать/скрыть
);
```

Тип указателя на эту функцию:

RDS_VB

Параметр:

Show

TRUE, если главное окно должно быть видимо пользователю, или FALSE, если оно должно быть скрыто.

Примечания:

Исходное состояние главного окна определяется параметрами командной строки РДС (запуск с параметром “/hide” делает его скрытым) или управляющим приложением, если РДС работает под его управлением. По умолчанию главное окно видимо.

При скрытом главном окне закрытие окна корневой подсистемы завершает РДС.

См. также:

rdsMainWindowVisible (стр. 164).

A.5.2.41. rdsStartCalc – запуск расчета

Функция rdsStartCalc переводит РДС в режим расчета, в котором производится циклический запуск моделей всех простых блоков и передача данных по связям.

```
void RDSCALL rdsStartCalc(void);
```

Тип указателя на эту функцию:

RDS_VV

Примечания:

Эта функция запускает расчет не мгновенно, она ставит команду на запуск расчета в очередь команд РДС и возвращает управление вызвавшей программе. В момент фактического запуска расчета модели всех блоков будут вызваны для реакции на событие RDS_BFM_STARTCALC (стр. 46), а загруженные модули автокомпиляции – на событие RDS_COMPM_MODECHANGE (стр. 104).

Пример использования функции `rdsStartCalc` приведен в §2.14.1.

См. также:

`rdsExecuteCommand` (стр. 151), `rdsStopCalc` (стр. 175).

A.5.2.42. `rdsStopCalc` – остановка расчета

Функция `rdsStopCalc` переводит РДС из режима расчета в режим моделирования.

```
void RDSCALL rdsStopCalc(void) ;
```

Тип указателя на эту функцию:

```
RDS_VV
```

Примечания:

Эта функция останавливает расчет, если он в данный момент работает, переводя РДС в режим моделирования. Если РДС находится в режимах моделирования или редактирования (то есть расчет не работает), вызов `rdsStopCalc` игнорируется. Расчет останавливается не мгновенно, функция просто информирует РДС о необходимости остановить его по окончании очередного такта, и немедленно возвращает управление вызвавшей программе. В момент фактической остановки расчета модели всех блоков будут вызваны для реакции на событие `RDS_BFM_STOPCALC` (стр. 46), а загруженные модули автокомпиляции – на событие `RDS_COMPM_MODECHANGE` (стр. 104).

Пример использования функции `rdsStopCalc` приведен в §2.14.1.

См. также:

`rdsExecuteCommand` (стр. 151), `rdsStartCalc` (стр. 174).

A.5.2.43. `rdsSystemInEditMode` – РДС в режиме редактирования

Функция `rdsSystemInEditMode` проверяет, находится ли РДС в режиме редактирования.

```
BOOL RDSCALL rdsSystemInEditMode(void) ;
```

Тип указателя на эту функцию:

```
RDS_BV
```

Возвращаемое значение:

TRUE, если РДС находится в режиме редактирования, и FALSE, если в режимах моделирования или расчета.

См. также:

`rdsCalcProcessIsRunning` (стр. 148).

A.5.2.44. `rdsUnregisterWindow` – отмена регистрации немодального окна

Функция `rdsUnregisterWindow` убирает ранее зарегистрированное функцией `rdsRegisterWindow` (стр. 166) немодальное окно из меню “Окна” и с панели кнопок окон РДС.

```
void RDSCALL rdsUnregisterWindow(  
    HWND Handle           // Дескриптор окна  
) ;
```

Тип указателя на эту функцию:

RDS_VHw

Параметр:

Handle

Дескриптор немодального окна, использованный при его регистрации функцией rdsRegisterWindow.

Примечания:

Эта функция вызывается перед закрытием ранее зарегистрированного в РДС немодального окна, чтобы убрать его название из интерфейса пользователя.

См. также:

rdsRegisterWindow (стр. 166), rdsChangeRegWinTitle (стр. 150),
rdsRegWinActivateNotify (стр. 167).

A.5.2.45. rdsUpdateExtIdsRange – обновление диапазонов идентификаторов

Функция rdsUpdateExtIdsRange улучшает работу механизма назначения внешних уникальных идентификаторов (см. §3.5) блокам и связям.

```
void RDSCALL rdsUpdateExtIdsRange(void);
```

Тип указателя на эту функцию:

RDS_VV

Примечания:

Эту функцию имеет смысл вызывать после одновременного удаления из схемы большого числа блоков и связей. Она позволит РДС повторно использовать идентификаторы, которые были назначены удаленным объектам. Ее вызов не обязателен – РДС в любом случае будет назначать новым объектам уникальные идентификаторы и сможет использовать освободившиеся повторно, но при активном программном редактировании большой схемы функция rdsUpdateExtIdsRange позволит несколько ускорить процедуру добавления новых объектов.

См. также:

rdsDeleteBlock (стр. 210), rdsDeleteConnection (стр. 211),
RDS_BLOCKDESCRIPTION (стр. 113), RDS_CONNDESCRIPTION (стр. 119).

A.5.3. Синхронизация потоков РДС

Описываются функции, предотвращающие одновременное обращение к данным блоков из двух одновременных работающих потоков РДС: главного потока и потока расчета.

A.5.3.1. rdsBlockDataSyncCall – вызвать функцию с блокировкой данных

Функция rdsBlockDataSyncCall вызывает пользовательскую функцию, указатель на которую передается в ее первом параметре, блокируя доступ ко всем данным РДС на время этого вызова.

```
int RDSCALL rdsBlockDataSyncCall(  
    RDS_IpV Func,           // Функция пользователя  
    LPVOID Param           // Параметр функции пользователя  
);
```


Тип указателя на эту функцию:

RDS_ICb4pV

Параметры:

Func

Указатель на пользовательскую функцию, которую нужно вызвать с блокировкой данных. Пользовательская функция должна иметь следующий вид:

```
int RDSCALL имя_функции(LPVOID param);
```

Param

Параметр типа void*, передаваемый в пользовательскую функцию при вызове.

Возвращаемое значение:

Целое число, возвращенное вызванной функцией пользователя.

Примечания:

Функция `rdsBlockDataSyncCall` обычно используется для выполнения каких-либо действий с данными блоков (например, вызова других сервисных функций РДС или обращения к статическим переменным блока) не из функции модели или функции модуля автокомпиляции, то есть в те моменты, когда данные не заблокированы РДС автоматически. Чаще всего такая ситуация возникает при открытии немодальных окон в модели блока (см. §1.8), процедуры которых вызываются Windows без синхронизации с РДС. Если процедура окна обратится к данным блока одновременно с потоком расчета РДС, могут возникнуть серьезные ошибки, поэтому такие обращения необходимо синхронизировать. При обращении к данным блока из функции модели или модуля автокомпиляции за синхронизацией следит РДС, поэтому в вызове специальных функций нет необходимости.

При вызове `rdsBlockDataSyncCall` РДС выполняет следующие действия:

1. Данные блокируются, как при вызове `rdsLockBlockData` (стр. 178).
2. Вызывается функция, указатель на которую передан в параметре `Func`, с параметром, переданным в параметре `Param`. Возвращенное функцией целое число запоминается.
3. Блокировка данных снимается, как при вызове `rdsUnlockBlockData` (стр. 181).
4. Запомненный результат возврата вызванной функции возвращается вызвавшей программе.

Параметр `Param`, имеющий тип `LPVOID` (произвольный указатель), передается в вызываемую функцию пользователя без изменений. Это единственный способ передать ей какие-либо данные: например, можно передать указатель на какую-либо структуру, а внутри пользовательской функции привести его к нужному типу и обращаться к полям этой структуры.

Фактически, вызов `rdsBlockDataSyncCall` всегда можно заменить парой вызовов `rdsLockBlockData` и `rdsUnlockBlockData`, между которыми вызывается пользовательская функция. Использование `rdsBlockDataSyncCall` с ее внутренней блокировкой данных позволяет уменьшить вероятность ошибки программиста: если при блокировке данных вручную забыть вызвать `rdsUnlockBlockData`, данные останутся заблокированными, что приведет к остановке потока расчета РДС.

Пример:

В этом примере в качестве действий, требующих блокировки данных и выполняемых в пользовательской функции, используется вызов функции `rdsMessageBox` (стр. 201), выводящей сообщение пользователю.

```

// Функция пользователя
int RDSCALL MySyncFunc(LPVOID param)
{ char *text=(char*)param;
  return rdsMessageBox(text, "Сообщение", MB_YESNO);
}

...

// Вызов функции с синхронизацией
int ret=rdsBlockDataSyncCall(MySyncFunc,
                             "Выполнить действие?");

```

Этот пример можно было бы переписать без использования rdsBlockDataSyncCall следующим образом:

```

// Действия с синхронизацией
int ret;
// Блокировка данных
rdsLockBlockData();
// Выполнение действий
ret=rdsMessageBox("Выполнить действие?", "Сообщение", MB_YESNO);
// Снятие блокировки
rdsUnlockBlockData();

```

См. также:

rdsLockBlockData (стр. 178), rdsUnlockBlockData (стр. 181).

A.5.3.2. rdsCallerThreadType – тип вызвавшего потока

Функция rdsCallerThreadType позволяет определить, в каком потоке сейчас выполняется вызвавшая функция – в главном потоке или в потоке расчета.

```
int RDSCALL rdsCallerThreadType(void);
```

Тип указателя на эту функцию:

RDS_IV

Возвращаемое значение:

Одна из двух констант, указывающая тип потока:

RDS_THREADMAIN	Главный поток (обслуживает интерфейс пользователя).
RDS_THREADAUX	Поток расчета (в режиме расчета выполняет циклический запуск моделей блоков).

Примечания:

В режиме расчета в РДС, как правило, работает одновременно два потока: главный поток, обслуживающий окна и интерфейс пользователя, и поток расчета, в котором в цикле вызываются модели блоков схемы и выполняется передача данных по связям. В потоке расчета крайне нежелательно выполнять некоторые действия – например, открытие модальных окон, которое приведет к остановке расчета. Поскольку некоторые события блоков могут возникать как в главном потоке, так и в потоке расчета, функция модели может узнать, в каком именно потоке она сейчас работает, вызовом rdsCallerThreadType.

A.5.3.3. rdsLockBlockData – включение блокировки данных

Функция rdsLockBlockData включает блокировку данных загруженной схемы для всех потоков, кроме вызвавшего эту функцию. Для выключения блокировки используется функция rdsUnlockBlockData (стр. 181).

```
void RDSCALL rdsLockBlockData(void);
```

Тип указателя на эту функцию:

```
RDS_VV
```

Примечания:

В РДС одновременно может работать несколько потоков: в режиме расчета, как правило, вместе с главным потоком приложения работает еще и поток расчета. Кроме того, модели блоков могут создавать свои потоки средствами Windows API. Поэтому в РДС, как и в любом многопоточном приложении, имеются средства блокировки данных: нельзя допускать, чтобы разные потоки одновременно обращались к одним и тем же данным в памяти. Поскольку потоки работают параллельно, если один из них начнет запись в какую-либо область памяти, а другой попытается в то же самое время считать данные из этой области или записать туда свои данные, может возникнуть конфликт, который приведет к считыванию неверных данных и другим серьезным ошибкам.

Блокировка данных включается автоматически при вызове функции модели блока или модуля автоматической компиляции, поэтому внутри этих функций о ней можно не задумываться. Если же нужно обратиться к данным блока, модуля автокомпиляции или вызвать сервисную функцию РДС из процедуры немодального окна (которая вызывается Windows в произвольные моменты времени без синхронизации с РДС) или из потока, созданного средствами Windows API, перед обращением необходимо вызвать функцию `rdsLockBlockData`, а после него – `rdsUnlockBlockData`.

Функция `rdsLockBlockData` работает следующим образом:

- если данные не заблокированы ни одним из потоков, функция блокирует данные, присваивает внутреннему счетчику блокировок единицу и возвращает управление вызвавшему потоку;
- если данные уже заблокированы тем же самым потоком, из которого вызвана функция, внутренний счетчик блокировок увеличивается на единицу и управление возвращается вызвавшему потоку;
- если данные заблокированы другим потоком, функция ждет снятия этой блокировки и, дождавшись ее, блокирует данные (присваивая единицу внутреннему счетчику блокировок), а затем возвращает управление вызвавшему потоку.

Таким образом, вызвав `rdsLockBlockData`, программист может быть уверен, что, после того, как она завершится, данные РДС будут в монопольном распоряжении вызвавшего потока. Поскольку функция имеет внутренний счетчик блокировок, ее можно вызывать из одного потока много раз подряд – если данные заблокированы этим же самым потоком, она будет возвращать ему управление немедленно. Такой счетчик позволяет вкладывать вызовы `rdsLockBlockData/rdsUnlockBlockData` друг в друга: данные будут разблокированы только тогда, когда `rdsUnlockBlockData` будет вызвана столько же раз, сколько раз была вызвана `rdsLockBlockData`.

Крайне важно вызывать `rdsUnlockBlockData` как можно быстрее после вызова `rdsLockBlockData`, поскольку попытки других потоков заблокировать данные будут приводить к их остановке до снятия блокировки этим потоком, что может привести к задержкам в работе РДС. Если необходимо провести над данными какую-либо сложную и длительную операцию в режиме расчета, лучше всего будет заблокировать их, скопировать в какие-либо внутренние структуры, а затем разблокировать и провести необходимые операции уже над данными во внутренних структурах.

Использование функций `rdsLockBlockData` и `rdsUnlockBlockData` рассматривается в §1.8.

См. также:

rdsUnlockBlockData (стр. 181), rdsBlockDataSyncCall (стр. 176),
rdsUnlockAndCall (стр. 180).

A.5.3.4. rdsUnlockAndCall – вызвать функцию, сняв блокировку данных

Функция rdsUnlockAndCall вызывает пользовательскую функцию, указатель на которую передается в ее первом параметре, снимая на время ее работы блокировку данных. Эту функцию можно вызывать *только в главном потоке* РДС.

```
BOOL RDSCALL rdsUnlockAndCall(  
    RDS_IpV Func,           // Функция пользователя  
    LPVOID Param,          // Параметр функции пользователя  
    int *pResult            // Результат возврата функции  
);
```

Тип указателя на эту функцию:

RDS_BCb4pVI

Параметры:

Func

Указатель на пользовательскую функцию, которую нужно вызвать со снятой блокировкой данных. Пользовательская функция должна иметь следующий вид:

```
int RDSCALL имя_функции(LPVOID param);
```

Param

Параметр типа void*, передаваемый в пользовательскую функцию при вызове.

pResult

Указатель на переменную, в которую будет записано целое значение, возвращенное пользовательской функцией. Если это значение не нужно, может быть равен NULL.

Возвращаемое значение:

TRUE, если пользовательская функция выполнена, или FALSE, если вызов не выполнен из-за того, что rdsUnlockAndCall вызвана не в главном потоке, вместо указателя на функцию передано значение NULL или в пользовательской функции возникли исключения.

Примечания:

Вызов rdsUnlockAndCall обычно используется для выполнения каких-либо длительных действий, не затрагивающих данные блоков, в главном потоке РДС. Типичный пример – открытие модального окна в режиме расчета из главного потока (например, в реакции блока на нажатие клавиши или кнопки мыши), этот случай рассматривается в §1.8. Поскольку при вызове модели блока данные всегда блокируются, открытие модального окна приведет к остановке потока расчета: пока окно на экране, функция модели блока не завершится, и РДС не разблокирует данные, поэтому как только поток расчета попытается тоже заблокировать их, он перейдет в состояние ожидания. Чтобы избежать этого, нужно создать функцию специального вида, которая будет открывать модальное окно, после чего передать указатель на эту функцию в параметре Func вызова rdsUnlockAndCall.

При вызове rdsUnlockAndCall РДС выполняет следующие действия:

1. Если вызвавший функцию поток – не главный поток РДС, функция немедленно возвращает FALSE.
2. Блокировка данных полностью снимается – как при вызове rdsUnlockBlockData (стр. 181), но без учета внутреннего счетчика блокировок.

3. Вызывается функция, указатель на которую передан в параметре Func, с параметром, переданным в параметре Param. Возвращенное функцией целое число запоминается.
4. Блокировка данных восстанавливается, причем вместе с ней восстанавливается значение внутреннего счетчика блокировок функции rdsLockBlockData (стр. 178).
5. Запомненный результат возврата вызванной функции записывается по указателю pResult, если этот указатель не равен NULL.

Поскольку на время выполнения пользовательской функции блокировка снимается, *любые* обращения к данным блоков и вызовы сервисных функций РДС следует окружать вызовами rdsLockBlockData/rdsUnlockBlockData.

Пример использования rdsUnlockAndCall рассматривается в §2.7.6.

См. также:

rdsLockBlockData (стр. 178), rdsUnlockBlockData (стр. 181).

A.5.3.5. rdsUnlockBlockData – выключение блокировки данных

Функция rdsUnlockBlockData выключает блокировку данных загруженной схемы, включенную ранее функцией rdsLockBlockData (стр. 178).

```
void RDSCALL rdsUnlockBlockData(void);
```

Тип указателя на эту функцию:

```
RDS_VV
```

Примечания:

Вызов rdsUnlockBlockData выключает блокировку данных, включенную вызовом rdsLockBlockData. Функция работает следующим образом:

- если данные не заблокированы ни одним из потоков или заблокированы не тем потоком, из которого вызвана rdsUnlockBlockData (то есть функция вызвана ошибочно, перед ней не было вызова rdsLockBlockData), функция немедленно возвращает управление вызвавшему потоку;
- если данные заблокированы тем же самым потоком, из которого вызвана функция, и внутренний счетчик блокировок больше единицы, значение счетчика уменьшается и управление возвращается вызвавшему потоку (блокировка при этом не снимается);
- если данные заблокированы тем же самым потоком, из которого вызвана функция, и внутренний счетчик блокировок равен единице, блокировка снимается и счетчику блокировок присваивается нулевое значение.

Используемый в РДС счетчик блокировок позволяет вкладывать вызовы rdsLockBlockData/rdsUnlockBlockData друг в друга: данные будут разблокированы только тогда, когда rdsUnlockBlockData будет вызвана столько же раз, сколько раз была вызвана rdsLockBlockData.

Крайне важно вызывать rdsUnlockBlockData как можно быстрее после вызова rdsLockBlockData, поскольку попытки других потоков заблокировать данные будет приводить к их остановке до снятия блокировки этим потоком, что вызовет задержки в работе РДС. Если необходимо провести над данными какую-либо сложную и длительную операцию в режиме расчета, лучше всего будет заблокировать их, скопировать в какие-либо внутренние структуры, а затем разблокировать и провести необходимые операции уже над данными во внутренних структурах.

Использование функций rdsLockBlockData и rdsUnlockBlockData рассматривается в §1.8.

См. также:

rdsLockBlockData (стр. 178), rdsBlockDataSyncCall (стр. 176),
rdsUnlockAndCall (стр. 180).

А.5.4. Отведение памяти и преобразование строк

Описываются функции управления памятью, которые можно использовать для работы с областями памяти, создаваемыми и уничтожаемыми РДС – в частности, динамически отводимыми строками. Вместе с ними описываются и другие функции, формирующие в результате своей работы динамические строки или преобразующие строки в числа.

А.5.4.1. rdsAddToDynStr – добавление строки к динамически отведенной строке

Функция rdsAddToDynStr добавляет строку в конец динамически созданной другими функциями строки.

```
void RDSCALL rdsAddToDynStr(  
    LPSTR *pString,           // Указатель на указатель на  
                               // динамическую строку  
    LPSTR AddStr,             // Добавляемая строка  
    BOOL NullEmpty           // Можно ли возвращать NULL  
);
```

Тип указателя на эту функцию:

RDS_VpSSB

Параметры:

pString

Указатель на переменную типа LPSTR (или, в терминах языка C, char*), в которой содержится указатель на динамическую строку, в конец которой нужно добавить новую. После выполнения функции старая динамическая строка освобождается, а в эту переменную запишется указатель на новую строку, полученную объединением старой строки с AddStr. pString не может иметь значение NULL (в этом случае функция немедленно завершится), но в переменной, на которую указывает pString, вполне может находиться NULL вместо указателя на динамическую строку. В этом случае старая динамическая строка будет считаться пустой, и новая строка будет динамической копией параметра AddStr.

AddStr

Указатель на строку, которую нужно добавить в конец строки *pString. Это может быть указатель на статическую или динамическую строку – для работы функции это не важно, в отличие от *pString, эта строка никак не изменяется в процессе выполнения функции. Если передать в этом параметре NULL, это будет считаться добавлением к *pString пустой строки.

NullEmpty

TRUE, если вместо пустой строки функция должна записывать в *pString значение NULL, и FALSE, если она должна будет динамически отвести в памяти пустую строку (то есть массив char из единственного символа с нулевым кодом) и записать в *pString указатель на нее.

Примечания:

Функция rdsAddToDynStr введена в набор сервисных функций РДС только для улучшения читаемости программ, работающих с динамическими строками – ее можно

полностью заменить последовательными вызовами `rdsDynStrCat` (стр. 186) и `rdsFree` (стр. 187).

Первый параметр этой функции (`pString`) *всегда* должен содержать указатель на переменную, в которой содержится указатель на динамически отведенную сервисными функциями РДС строку. Если эта строка будет статической, при работе функции возникнут серьезные ошибки, поскольку после объединения строк она попытается освободить старую строку вызовом `rdsFree`, что приведет к непредсказуемым последствиям.

Фактически, внутри функции `rdsAddToDynStr` находится конструкция следующего вида:

```
void rdsAddToDynStr(LPSTR *pString, LPSTR AddStr, BOOL NullEmpty)
{ char *oldstr=*pString;
  *pString=rdsDynStrCat(*pString, AddStr, NullEmpty);
  rdsFree(oldstr);
}
```

Пример использования функции `rdsAddToDynStr` приведен в §2.11.

См. также:

`rdsDynStrCat` (стр. 186), `rdsFree` (стр. 187).

A.5.4.2. `rdsAllocate` – динамическое отведение области памяти

Функция `rdsAllocate` отводит в динамической памяти РДС (т.н. “куча”, “heap”) область заданного размера.

```
LPVOID RDSCALL rdsAllocate(
    DWORD Size          // Размер отводимой области
);
```

Тип указателя на эту функцию:

`RDS_pVDw`

Параметр:

`Size`

Размер отводимой области в байтах.

Возвращаемое значение:

Указатель на отведенную область памяти.

Примечания:

Эта функция используется для отведения памяти для нужд моделей блоков и модулей автоматической компиляции, а также при работе со строковыми статическими переменными блока. Она отводит в динамической памяти РДС область размером в `Size` байтов и возвращает указатель на эту область. Возвращается указатель произвольного вида (`LPVOID`, то есть `void*`), поэтому перед использованием этот указатель обычно приводят к типу, соответствующему данным, которые будут храниться в этой области.

Отведенную таким образом область *обязательно* нужно освободить вызовом функции `rdsFree` (стр. 187) после того, как она станет не нужна.

Использование именно этой функции вместо стандартных для языка C функций `malloc/calloc` или оператора C++ `new` удобно тем, что область памяти, отведенная одним блоком при помощи `rdsAllocate`, может быть освобождена другим блоком при помощи `rdsFree`. Со стандартными функциями это было бы невозможным: нет никакой гарантии, что менеджер памяти, используемый моделью одного блока, совместим с менеджером

памяти модели другого. При использовании функций `rdsAllocate/rdsFree` в роли менеджера памяти выступает РДС, поэтому проблем с совместимостью не возникает.

Пример использования функции `rdsAllocate` для работы со строковыми переменными блока приведен в §2.5.4.

См. также:

`rdsFree` (стр. 187).

A.5.4.3. `rdsAtoD` – преобразование строки в вещественное число

Функция `rdsAtoD` переводит строку с символьным представлением вещественного числа двойной точности в число типа `double`.

```
void RDSCALL rdsAtoD(  
    LPSTR Str,           // Строка с представлением числа  
    double *pVal        // Возвращаемое число  
);
```

Тип указателя на эту функцию:

`RDS_VSpD`

Параметры:

`Str`

Строка с символьным представлением вещественного числа (например, “123.45”).

`pVal`

Указатель на переменную, в которую нужно записать полученное из строки вещественное число.

Примечания:

Эта функция отличается от стандартной функции языка C `atof` способом работы с разделителем целой и дробной части и со значением-индикатором ошибки. Она работает следующим образом:

- если первый символ переданной строки `Str` – вопросительный знак, функция записывает по указателю `pVal` значение-индикатор математической ошибки (см. функцию `rdsGetHugeDouble`, стр. 157);
- если в строке `Str` содержится символ, заданный в настройках Windows в качестве разделителя целой и дробной части, строка преобразуется в вещественное число согласно настройкам Windows;
- во всех остальных случаях функция работает так же, как и `atof` – в качестве разделителя целой и дробной части используется символ точки.

Пример использования функции `rdsAtoD` приведен в §3.3.

См. также:

`rdsGetHugeDouble` (стр. 157), `rdsDtoA` (стр. 185), `rdsAtoI` (стр. 184).

A.5.4.4. `rdsAtoI` – преобразование строки в целое число

Функция `rdsAtoI` переводит строку с символьным представлением целого числа в число типа `int`. В строке может содержаться указание на используемую систему счисления.

```
int RDSCALL rdsAtoI(  
    LPSTR Str,           // Строка с представлением числа  
    int *pRadix          // Возвращаемая система счисления  
);
```


Тип указателя на эту функцию:

RDS_ISpI

Параметры:

Str

Строка с символьным представлением целого числа. В зависимости от префикса, то есть символов, стоящих в строке между необязательным знаком и цифрами числа, для преобразования строки в число будут использоваться следующие системы счисления:

<i>Префикс</i>	<i>Система счисления</i>
0B	Двоичная (строка "0b11" будет преобразована в число 3).
0 или 0O (ноль и латинская буква "O")	Восьмеричная (строка "011" или "0o11" будет преобразована в число 9).
0X или 0H	Шестнадцатеричная (строка "0x1f" будет преобразована в число 31).
0D или без префикса	Десятичная (строка "123" или "0d123" будет преобразована в число 123).

Знак "+" (необязательный) или "-" должен находиться перед префиксом, если он есть. Таким образом, если за необязательным знаком числа в строке находится цифра 1 – 9, то будет использована десятичная система; если цифра 0 и буква, то будет использована система счисления, определяемая этой буквой; если же там будет находиться цифра 0, за которой следуют другие цифры, то будет использована восьмеричная система.

pRadix

Указатель на целую переменную, в которую нужно записать основание системы счисления, использованной в переданной строке для записи целого числа. Если вызывающей программе не нужна информация о системе счисления, в этом параметре можно передать NULL.

Возвращаемое значение:

Полученное из строки целое число или 0, если строка не может быть преобразована в число.

Примечания:

Функция rdsAtOI похожа на стандартную функцию языка C strtol, но поддерживает больше префиксов и двоичную систему счисления.

См. также:

rdsAtOD (стр. 184), rdsItOA (стр. 191).

A.5.4.5. rdsDtoA – преобразование вещественного числа в строку

Функция rdsDtoA преобразует вещественное число двойной точности в динамическую строку с его символьным представлением.

```
LPSTR RDSCALL rdsDtoA(  
    double Val,          // Вещественное число  
    int Dec,             // Число знаков дробной части  
    int *pLength         // Возвращаемая длина строки  
);
```

Тип указателя на эту функцию:

RDS_SDIpI

Параметры:

Val

Вещественное число, преобразуемое в строку.

Dec

Число десятичных знаков в дробной части числа. Можно вместо числа знаков передать значение `-1`, тогда число знаков дробной части будет автоматически подобрано так, чтобы отбросить все незначащие нули справа.

pLength

Указатель на целую переменную, в которую функция должна записать длину получившейся строки. Если вызывающей программе не нужна длина строки, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку с символьным представлением переданного числа. Для отделения дробной части всегда используется десятичная точка. В случае ошибки преобразования возвращается `NULL`. Если в параметре Val передано специальное значение-индикатор ошибки (см. функцию `rdsGetHugeDouble`, стр. 157), созданная строка будет состоять из единственного символа вопросительного знака.

Примечания:

Динамическая строка, созданная функцией `rdsDtoA`, должна быть **обязательно** освобождена функцией `rdsFree` (стр. 187).

Пример использования функции `rdsDtoA` приведен в §2.11.

См. также:

`rdsAtoD` (стр. 184), `rdsGetHugeDouble` (стр. 157), `rdsFree` (стр. 187).

A.5.4.6. `rdsDynStrCat` – сложение двух строк

Функция `rdsDynStrCat` возвращает динамическую строку, содержащую сумму (последовательную запись) двух переданных ей строк.

```
LPSTR RDSCALL rdsDynStrCat(  
    LPSTR String1,    // Первое слагаемое-строка  
    LPSTR String2,    // Второе слагаемое-строка  
    BOOL NullEmpty    // Можно ли возвращать NULL  
);
```

Тип указателя на эту функцию:

`RDS_SSSB`

Параметры:

String1

Указатель на строку, с которой будет начинаться строка-результат. Значение `NULL` в этом параметре эквивалентно передаче пустой строки.

String2

Указатель на строку, которой будет заканчиваться строка-результат. Значение `NULL` в этом параметре эквивалентно передаче пустой строки.

NullOrEmpty

TRUE, если вместо пустой строки функция должна возвращать NULL, и FALSE, если она должна будет динамически отвести в памяти пустую строку (то есть массив char из единственного символа с нулевым кодом) и вернуть указатель на нее.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую сумму двух переданных строк. Например, если в String1 передать “12”, а в String2 – “34”, будет создана строка “1234”.

Примечания:

Динамическая строка, созданная функцией rdsDynStrCat, должна быть **обязательно** освобождена вызовом rdsFree (стр. 187). Пример использования этой функции приведен в §2.5.4.

См. также:

rdsAddToDynStr (стр. 184), rdsFree (стр. 187).

A.5.4.7. rdsDynStrCopy – создание динамической копии строки

Функция rdsDynStrCopy возвращает динамическую строку, являющуюся копией переданной строки.

```
LPSTR RDSCALL rdsDynStrCopy(  
    LPSTR String    // Копируемая строка  
);
```

Тип указателя на эту функцию:

RDS_SS

Параметр:

String

Указатель на строку, для которой создается динамическая копия.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую копию строки String.

Примечания:

Эта функция используется в тех случаях, когда для работы модели блока требуется динамическая строка, совместимая с функцией rdsFree (стр. 187) – например, если требуется изменить строковую статическую переменную блока. Пример использования этой функции приведен в §2.12.7.

Динамическая строка, созданная функцией rdsDynStrCopy, должна быть **обязательно** освобождена функцией rdsFree.

См. также:

rdsFree (стр. 187).

A.5.4.8. rdsFree – освобождение отведенной динамической памяти

Функция rdsFree освобождает область памяти, ранее отведенную функцией rdsAllocate (стр. 183) или уничтожает динамическую строку, созданную одной из сервисных функций.

```
void RDSCALL rdsFree(
    LPVOID Ptr // Указатель на освобождаемую область
);
```

Тип указателя на эту функцию:

RDS_VpV

Параметр:

Ptr

Указатель на область памяти или динамическую строку, которую нужно освободить (при передаче NULL никаких действий не производится).

Примечания:

Эта функция должна быть обязательно вызвана, когда область памяти или динамическая строка больше не нужна. Если этого не сделать, в программе возникнут утечки памяти: в отличие от различных объектов РДС, динамические строки и отводимые области памяти не привязаны к какому-либо блоку и не уничтожаются автоматически при его удалении. Функцию можно безопасно вызывать с параметром NULL – в этом случае она немедленно завершается, не выполняя никаких действий.

Функцию rdsFree необходимо вызывать для освобождения памяти, отведенной следующими функциями:

rdsAddToDynStr (стр. 182)	rdsGetRuntimeTypeData (стр. 330)
rdsAllocate (стр. 183)	rdsGetTextWordDyn (стр. 293)
rdsBlockVarToMem (стр. 320)	rdsInputString (стр. 200)
rdsCallDirDialog (стр. 197)	rdsItoA (стр. 191)
rdsCallFileDialog (стр. 198)	rdsListVarTypes (стр. 162)
rdsCreateFullBlockNameString (стр. 210)	rdsMakeUniqueBlockName (стр. 239)
rdsCreateVarDescriptionString (стр. 324)	rdsProcessText (стр. 192)
rdsCreateVarTypeText (стр. 325)	rdsStringReplace (стр. 194)
rdsDtoA (стр. 185)	rdsStructToFontText (стр. 279)
rdsDynStrCat (стр. 186)	rdsTransformFileName (стр. 196)
rdsDynStrCopy (стр. 187)	rdsVSGetVarDefValueStr (стр. 441)
rdsGetBlockVarDefValueStr (стр. 330)	Команда RDS_CSV_LINE (стр. 561)
rdsGetFullFilePath (стр. 188)	Команда RDS_CSV_TEXT (стр. 568)
rdsGetRelFilePath (стр. 190)	

A.5.4.9. rdsGetFullFilePath – сокращенный путь к файлу в полный

Функция rdsGetFullFilePath преобразует сокращенный (содержащий специальные символические константы РДС) путь к файлу в полный путь, и возвращает динамическую строку, содержащую этот полный путь. Файл, путь к которому обрабатывается, может физически не существовать или быть недоступным в данный момент – функция работает только со строкой пути.

```
LPSTR RDSCALL rdsGetFullFilePath(
    LPSTR FileName, // Сокращенный путь
    LPSTR AltDefPath, // Альтернативный путь по умолчанию
    int *pLength // Возвращаемая длина строки
);
```

Тип указателя на эту функцию:

RDS_SSSpI

Параметры:

FileName

Указатель на строку с сокращенным путем к файлу. Может начинаться с одной из символических констант, которые будут заменены на путь к одной из стандартных папок РДС (см. также функцию `rdsGetSystemPath`, стр. 160):

\$DLL\$	Полный путь к папке стандартных DLL (по умолчанию – папка “Dll\” внутри папки РДС).
\$INI\$	Полный путь к папке INI-файлов (по умолчанию совпадает с основной папкой РДС, в которой находится исполняемый файл <code>rds.exe</code>).
\$LIB\$	Полный путь к библиотеке блоков (по умолчанию – папка “Library\” внутри папки РДС).
\$MODELS\$	Полный путь к папке стандартных автокомпилируемых моделей (по умолчанию – папка “Models\” внутри папки РДС).
\$PANEL\$	Полный путь к панели блоков (по умолчанию – папка “Panel\” внутри папки РДС).
\$RDSINCLUDE\$	Полный путь к папке файлов-заголовков (по умолчанию – папка “Include\” внутри папки РДС).
\$TEMP\$	Полный путь к собственной временной папке РДС (по умолчанию – папка “Temp\” внутри папки РДС).
\$WINTEMP\$	Полный путь к папке временных файлов Windows.

Обнаружив в начале строки `FileName` одну из перечисленных выше констант, функция подставляет вместо нее полный путь к соответствующей папке без завершающей обратной косой черты. Если `FileName` не содержит ни констант, ни полного пути, к этому имени добавляется либо путь из параметра `AltDefPath`, если он не равен `NULL`, либо путь к папке, в которой находится файл загруженной в данный момент схемы, если `AltDefPath` равен `NULL`.

AltDefPath

Указатель на строку, которая добавляется к строке `FileName`, если она не содержит ни приведенных выше констант, ни полного пути. Если этот параметр равен `NULL`, к `FileName` будет добавлен путь к папке, в которой находится файл загруженной в данный момент схемы.

pLength

Указатель на целую переменную, в которую функция запишет длину строки получившегося полного пути к файлу. Если вызвавшей программе не нужно это значение, в `pLength` можно передать `NULL`.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую полный путь к указанному файлу. Если получить полный путь невозможно, возвращается `NULL`.

Примечания:

Динамическая строка, созданная функцией `rdsGetFullFilePath`, должна быть **обязательно** освобождена вызовом `rdsFree` (стр. 187). Пример использования этой функции приведен в §2.13.6.

См. также:

`rdsGetRelFilePath` (стр. 190), `rdsGetSystemPath` (стр. 160),
`rdsFree` (стр. 187).

A.5.4.10. rdsGetRelFilePath – полный путь к файлу в сокращенный

Функция `rdsGetRelFilePath` преобразует полный путь к файлу в сокращенный, содержащий специальные символические константы РДС, и возвращает динамическую строку, содержащую сокращенный путь. Файл, путь к которому обрабатывается, может физически не существовать или быть недоступным в данный момент – функция работает только со строкой пути.

```
LPSTR RDSCALL rdsGetRelFilePath(  
    LPSTR FileName,          // Полный путь  
    LPSTR AltDefPath,        // Альтернативный путь по умолчанию  
    int *pLength             // Возвращаемая длина строки  
);
```

Тип указателя на эту функцию:

RDS_SSSpI

Параметры:

`FileName`

Указатель на строку с полным путем к файлу. Если начало этого пути совпадет с одним из стандартных путей РДС (см. функцию `rdsGetSystemPath`, стр. 160), вместо этого пути будет подставлена одна из символических констант, описанных на стр. 189.

`AltDefPath`

Указатель на путь по умолчанию, который нужно удалить из полного пути. В зависимости от значения этого параметра, сокращенный путь формируется следующим образом:

<i>AltDefPath</i>	<i>Формирование сокращенного пути</i>
NULL	Если начало <code>FileName</code> совпадает с путем к папке, в которой находится загруженная в данный момент схема, эта часть пути будет удалена. Таким образом, в сокращенном пути останутся только подпапки в папке схемы и само имя файла.
путь к папке	Если начало <code>FileName</code> совпадает с <code>AltDefPath</code> , <code>AltDefPath</code> удаляется из сокращенного пути. Таким образом, в сокращенном пути останутся только подпапки в <code>AltDefPath</code> и само имя файла.
"" (пустая строка)	Сравнение начала <code>FileName</code> с какими-либо путями не производится.

`pLength`

Указатель на целую переменную, в которую функция запишет длину строки получившегося сокращенного пути к файлу. Если вызвавшей программе не нужно это значение, в `pLength` можно передать NULL.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую сокращенный путь к указанному файлу. Если получить этот путь невозможно, возвращается NULL.

Примечания:

Динамическая строка, созданная функцией `rdsGetRelFilePath`, должна быть **обязательно** освобождена вызовом `rdsFree` (стр. 187).

Функцию `rdsGetRelFilePath` чаще всего используют для того, чтобы хранить в параметрах блоков сокращенные относительные пути вместо полных. В этом случае перенос папки со схемой на другую машину или в другую папку не приводит к потере работоспособности схемы, поскольку все пути в параметрах блоков будут указаны относительно расположения файла схемы.

См. также:

`rdsGetFullFilePath` (стр. 188), `rdsGetSystemPath` (стр. 160),
`rdsFree` (стр. 187).

A.5.4.11. `rdsItoA` – преобразование целого числа в строку

Функция `rdsItoA` преобразует тридцатидвухбитное целое число со знаком в динамическую строку с его символьным представлением в указанной системе счисления.

```
LPSTR RDSCALL rdsItoA(  
    int Val,           // Целое число  
    int Radix,         // Система счисления  
    int MinDigits      // Минимальное число разрядов  
);
```

Тип указателя на эту функцию:

`RDS_SIII`

Параметры:

`Val`

Целое число, преобразуемое в строку.

`Radix`

Система счисления, в которой нужно представить число. Этот параметр может принимать следующие значения:

- | | |
|----|--|
| 2 | Число будет представлено в двоичной системе счисления с префиксом “0b” (число 3 будет преобразовано в строку “0b11”). |
| 8 | Число будет представлено в восьмеричной системе счисления с префиксом “0o” (число 9 будет преобразовано в строку “0o11”). |
| 10 | Число будет представлено в десятичной системе счисления без какого-либо префикса (число 123 будет преобразовано в строку “123”). |
| 16 | Число будет представлено в шестнадцатеричной системе счисления с префиксом “0x” (число 31 будет преобразовано в строку “0x1f”). |

Любое другое значение параметра `Radix` будет игнорироваться и число будет преобразовываться в строку с использованием десятичной системы.

`MinDigits`

Минимально допустимое число разрядов в символьном представлении числа. Если после преобразования в указанной системе счисления в числе окажется меньше `MinDigits` разрядов, оно будет дополнено нулями слева. Если в числе окажется больше `MinDigits` разрядов, этот параметр игнорируется.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку с символьным представлением переданного числа. В случае ошибки преобразования возвращается `NULL`.

Примечания:

Динамическая строка, созданная функцией `rdsItoA`, должна быть **обязательно** освобождена функцией `rdsFree` (стр. 187).

Пример использования функции приведен в §4.4.

См. также:

rdsAtoi (стр. 184), rdsFree (стр. 187).

A.5.4.12. rdsProcessText – обработка строки

Функция rdsProcessText выполняет над переданной ей строкой указанную операцию и возвращает результат в виде динамически созданной строки.

```
LPSTR RDSCALL rdsProcessText (  
    LPSTR String,      // Исходная строка  
    int Operation,    // Операция (RDS_PT_*)  
    int *pLength      // Возвращаемая длина строки  
);
```

Тип указателя на эту функцию:

RDS_SSIPi

Параметры:

String

Указатель на исходную строку.

Operation

Выполняемое над строкой преобразование:

RDS_PT_TEXTTOSTRING Непечатаемые символы в строке String заменяются на их символические обозначения (возврат каретки заменяется на “\r”, перед знаком кавычки вставляется обратная косая черта, нестандартные коды заменяются на их шестнадцатеричное представление с префиксом “x” и т.д.). После такого преобразования строка гарантированно содержит только печатаемые коды символов и ее можно, например, записать в INI-файл. Такая строка может быть однозначно преобразована обратно вызовом этой же функции с параметром RDS_PT_STRINGTOTEXT.

RDS_PT_STRINGTOTEXT Символические представления непечатаемых символов в строке String заменяются на коды этих символов. Это операция, обратная операции RDS_PT_TEXTTOSTRING.

RDS_PT_VARTYPETEXT Формируется строка, содержащая текстовое описание типа переменной, которое соответствует первому символу строки String, если считать его однобайтовой константой типа RDS_VARTYPE_* (см. стр. 49). Полученные в результате этой операции строки можно показывать пользователю при редактировании переменных блоков.

RDS_PT_VARTYPECHAR Текстовое описание переменной из строки String преобразуется в строку, состоящую из единственного символа типа этой переменной RDS_VARTYPE_*. Это операция, обратная операции RDS_PT_VARTYPETEXT.

Любое другое значение параметра Operation приводит к возврату значения NULL.

pLength

Указатель на целую переменную, в которую функция запишет длину созданной динамической строки. Если вызвавшей программе не нужно это значение, в pLength можно передать NULL.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, являющуюся результатом преобразования Operation, примененного к строке String. В случае недопустимой операции возвращается NULL.

Примечания:

Динамическая строка, созданная функцией rdsProcessText, должна быть **обязательно** освобождена функцией rdsFree (стр. 187).

Операция RDS_PT_TEXTTOSTRING переводит произвольную строку в текст, пригодный для сохранения, записи или вставки в программу путем замены символов на их условные обозначения по правилам языка C согласно следующей таблице:

<i>Символ или его код</i>	<i>Условное обозначение (по правилам языка C)</i>
Пробел, печатаемые спецсимволы, символы латинского и русского алфавитов	Условного обозначения нет, символ не изменяется.
Код 7 (код звукового сигнала)	\a
Код 8 (Backspace)	\b
Код 12 (перевод страницы)	\f
Код 10 (перевод строки)	\n
Код 13 (возврат каретки, Enter)	\r
Код 9 (табуляция)	\t
Код 11 (вертикальная табуляция)	\v
Обратная косая черта	Удвоенная обратная косая черта (“\\”)
' (апостроф, код 39)	\' (обратная косая черта, за которой следует апостроф)
" (кавычка, код 34)	\" (обратная косая черта, за которой следует кавычка)
Прочие непечатаемые символы	\xNN, где NN – шестнадцатеричный код символа. Например, символ с кодом 255 будет преобразован в “\xff”.

Обратная операция, RDS_PT_STRINGTOTOEXT, преобразует строку с условными обозначениями символов в строку с кодами этих символов.

Операция RDS_PT_VARTYPETEXT переводит однобайтовый тип переменной в его название, пригодное для показа пользователю:

<i>Символ</i>	<i>Константа типа в РДС</i>	<i>Название типа согласно RDS_PT_VARTYPETEXT</i>
S	RDS_VARTYPE_SIGNAL	“double”
L	RDS_VARTYPE_LOGICAL	“Логический”
C	RDS_VARTYPE_CHAR	“char”
H	RDS_VARTYPE_SHORT	“short”

<i>Символ</i>	<i>Константа типа в РДС</i>	<i>Название типа согласно RDS_PT_VARTYPETEXT</i>
I	RDS_VARTYPE_INT	“int”
F	RDS_VARTYPE_FLOAT	“float”
D	RDS_VARTYPE_DOUBLE	“double”
A	RDS_VARTYPE_STRING	“Строка”
V	RDS_VARTYPE_RUNTIME	“произвольный” (при обратном преобразовании допускается “Произв.”)
M	RDS_VARTYPE_ARRAY	“Матрица” (при обратном преобразовании допускается “Массив”)

Остальные типы после выполнения операция RDS_PT_VARTYPETEXT дают пустую строку. Обратная операция, RDS_PT_VARTYPECHAR, переводит название типа (включая альтернативные варианты, указанные в скобках) в строку с константой этого типа. Если название типа не опознано, тип считается структурой и возвращается строка из одного символа “{” (RDS_VARTYPE_STRUCT).

См. также:

Типы переменных РДС (стр. 49), rdsListVarTypes (стр. 162), rdsFree (стр. 187), rdsGetFullFilePath (стр. 188), rdsGetRelFilePath (стр. 190), rdsStringReplace (стр. 194).

A.5.4.13. rdsStringReplace – замена фрагментов строки

Функция rdsStringReplace ищет в переданной ей строке заданные фрагменты и формирует новую динамическую строку, в которой эти фрагменты заменены на другие.

```

LPSTR RDSCALL rdsStringReplace(
    LPSTR String,           // Исходная строка
    LPSTR *Search,         // Массив фрагментов для поиска
    LPSTR *Replace,        // Массив фрагментов для замены
    int Count,             // Размер массива Search или -1
    DWORD Flags            // Флаги RDS_SRF_*
);

```

Тип указателя на эту функцию:

RDS_SSpSpSIDw

Параметры:

String

Указатель на исходную строку, в которой будет выполняться поиск фрагментов.

Search

Указатель на первый элемент массива указателей на строки (char*) фрагментов для поиска. Массив либо должен завершаться элементом NULL, либо число его элементов должно быть указано в параметре Count.

Replace

Указатель на первый элемент массива указателей на строки (char*) фрагментов, которыми будут заменяться фрагменты из массива Search с теми же индексами. Значение NULL в элементе этого массива указывает на необходимость заменить найденный фрагмент на пустую строку (то есть просто удалить его из строки). Число

элементов в массиве Replace не должно быть меньше числа элементов в массиве Search.

Count

Число строк в массиве Search или -1, если массив Search завершается элементом со значением NULL.

Flags

Битовые флаги, влияющие на поиск и замену фрагментов:

RDS_SRF_IGNORECASE При поиске не учитывать регистр символов.

RDS_SRF_STDPATHS Заменить символические обозначения стандартных путей (см. стр. 189) на сами пути после того, как выполнен поиск фрагментов из массива Search и замена их на фрагменты из Replace.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, в которой фрагменты из массива Search заменены их на фрагменты из массива Replace. В случае ошибки возвращается NULL.

Примечания:

Динамическая строка, созданная функцией rdsStringReplace, должна быть **обязательно** освобождена функцией rdsFree (стр. 187).

Эта функция последовательно просматривает массив Search, начиная с нулевого индекса, до обнаружения в нем значения NULL или до индекса Count, если Count>0. Каждый элемент массива Search ищется в строке String и, если он найден, все его вхождения заменяются на элемент массива Replace с тем же индексом. Если в параметре Flags взведен флаг RDS_SRF_IGNORECASE, поиск осуществляется без учета регистра символов, в противном случае – с его учетом. Если в параметре Flags взведен флаг RDS_SRF_STDPATHS, то, после того, как перебраны все элементы массива Search, оставшиеся в строке символические обозначения путей (“\$DLL\$”, “\$INI\$” и т.п., см. стр. 189) заменяются на их значения, причем замена производится не только в начале строки, как в функции rdsGetFullFilePath, но и в любом другом ее месте.

Пример использования функции rdsStringReplace приведен в §4.4.

Пример:

Выполнение программы

```
char *names[]={ "_PI_VAL_", "$PI$", NULL};
char *values[]={ "3.1415926", "Пи" };
char *result=rdsStringReplace(
    "Значение $PI$ равно _PI_VAL_",
    names, values, -1, 0);
rdsMessageBox(result, "Результат", MB_OK);
rdsFree(result);
```

приведет к выводу сообщения с текстом “Значение Пи равно 3.1415926”.

См. также:

rdsFree (стр. 187), rdsGetFullFilePath (стр. 188),
rdsTransformFileName (стр. 196).

A.5.4.14. rdsTransformFileName – преобразование имени файла

Функция `rdsTransformFileName` преобразует переданное ей имя файла согласно указанной в параметре операции: выделяет путь, заменяет расширение на другое и т.п.

```
LPSTR RDSCALL rdsTransformFileName(  
    LPSTR FileName,           // Исходное имя файла  
    DWORD Op,                 // Операция (RDS_TFN_*)  
    LPSTR Arg,                // Дополнительный аргумент  
    int *pLength              // Возвращаемая длина строки  
);
```

Тип указателя на эту функцию:

`RDS_SSDwSpI`

Параметры:

`FileName`

Указатель на строку с именем файла, которое нужно преобразовать.

`Op`

Выполняемая над именем файла операция:

`RDS_TFN_CHANGEEXT` Заменить расширение в имени файла на строку из параметра `Arg`. В параметре `Arg` указывается новое расширение (включая символ точки) или `NULL`, если расширение из имени файла нужно удалить.

`RDS_TFN_EXCLUDEPATHBS` Если в конце `FileName` находится символ обратной косой черты, убрать его (в противном случае `FileName` копируется в динамическую строку без изменений).

`RDS_TFN_GETTEXT` Извлечь расширение из имени файла `FileName`, включая символ точки, отделяющий его от собственно имени. При отсутствии расширения в имени файла возвращается `NULL`.

`RDS_TFN_GETNAME` Убрать путь из `FileName` (возвращается собственно имя файла с расширением).

`RDS_TFN_GETPATH` Извлечь путь из имени файла `FileName`, включая завершающий его символ обратной косой черты.

`RDS_TFN_GETPATHNOBS` Извлечь путь из имени файла `FileName`, исключая завершающий его символ обратной косой черты.

`RDS_TFN_INCLUDEPATHBS` Если в конце `FileName` не находится символ обратной косой черты, добавить его (в противном случае `FileName` копируется в динамическую строку без изменений).

`Arg`

Дополнительный аргумент преобразования (новое расширение для операции `RDS_TFN_CHANGEEXT`).

`pLength`

Указатель на целую переменную, в которую функция должна записать длину получившейся строки. Если вызывающей программе не нужна длина строки, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую преобразованное имя файла. Если преобразование дает пустую строку, возвращается NULL.

Примечания:

Динамическая строка, созданная функцией `rdsTransformFileName`, должна быть **обязательно** освобождена функцией `rdsFree` (стр. 187).

Пример использования функции приведен в §4.4.

См. также:

`rdsFree` (стр. 187), `rdsStringReplace` (стр. 194).

A.5.5. Вызов стандартных диалогов

Описываются функции вызова диалогов РДС и стандартных диалогов Windows.

A.5.5.1. `rdsCallColorDialog` – вызов диалога выбора цвета

Функция `rdsCallColorDialog` открывает стандартный диалог, используемый в Windows для выбора цвета.

```
BOOL RDSCALL rdsCallColorDialog(  
    RDS_PCOLORREF pColor // Указатель на переменную цвета  
);
```

Тип указателя на эту функцию:

`RDS_BpCr`

Параметр:

`pColor`

Указатель на переменную типа `COLORREF` (см. стр. 24), в которую запишется выбранный пользователем цвет. Из этой же переменной будет взято значение цвета, которое установится в диалоге сразу после открытия, поэтому параметр `pColor` не может быть равен NULL.

Возвращаемое значение:

TRUE, если пользователь нажал кнопку “OK”, и FALSE, если он нажал кнопку “Отмена” или просто закрыл окно.

Примечания:

Эта функция открывает обычный для Windows диалог выбора цвета, в котором пользователь может выбрать один из стандартных цветов или задать произвольный цвет, введя его компоненты или выбрав точку на цветовом поле. Исходное значение выбранного в диалоге цвета берется из переменной по указателю `pColor`. Если пользователь закроет диалог кнопкой “OK”, по этому указателю запишется новое значение цвета, в противном случае значение `*pColor` не изменится.

A.5.5.2. `rdsCallDirDialog` – вызов диалога выбора папки

Функция `rdsCallDirDialog` открывает стандартный диалог выбора папки.

```
LPSTR RDSCALL rdsCallDirDialog(  
    LPSTR InitialDir, // Исходная папка  
    LPSTR Title,      // Заголовок диалога
```

```

        BOOL AbsPath          // Вернуть абсолютный путь
    );

```

Тип указателя на эту функцию:

```
RDS_SSSB
```

Параметры:

InitialDir

Указатель на строку с путем к исходной папке, которая будет выбрана в диалоге на момент его открытия. Если в этом параметре передано значение NULL или указатель на пустую строку, исходно в диалоге будет выбрана папка с загруженной в данный момент схемой.

Title

Указатель на строку с заголовком окна диалога. Если в этом параметре передано значение NULL, диалог будет иметь заголовок “Папка”.

AbsPath

TRUE, если функция должна вернуть полный путь к выбранной пользователем папке, и FALSE, если в возвращаемой строке необходимо заменить стандартные пути РДС на их символические обозначения (см. стр. 189).

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую путь к выбранной папке, если пользователь нажал кнопку “ОК”. Если пользователь нажал кнопку “Отмена” или просто закрыл окно диалога, возвращается NULL.

Примечания:

Эта функция используется в тех случаях, когда нужно запросить у пользователя имя какой-либо папки на диске (например, в настройках модуля автокомпиляции может потребоваться указание пути к папке стандартных библиотек). Если требуется путь к конкретному файлу, а не к папке, следует использовать функцию `rdsCallFileDialog` (стр. 198).

Динамическая строка, созданная функцией `rdsCallDirDialog`, должна быть **обязательно** освобождена функцией `rdsFree` (стр. 187).

См. также:

`rdsCallFileDialog` (стр. 198), `rdsFree` (стр. 187).

A.5.5.3. `rdsCallFileDialog` – вызов диалога выбора файла

Функция `rdsCallFileDialog` открывает стандартный диалог выбора файла.

```

LPSTR RDSCALL rdsCallFileDialog(
    LPSTR InitialFile,    // Исходное имя файла или папки
    DWORD Flags,          // Флаги диалога (RDS_CFD_*)
    LPSTR Filter,         // Фильтры имен файлов
    LPSTR DefExt,         // Расширение по умолчанию
    LPSTR Title           // Заголовок диалога
);

```

Тип указателя на эту функцию:

```
RDS_SSDwSSS
```

Параметры:

InitialFile

Указатель на строку с путем к исходному файлу, который будет выбран в диалоге на момент его открытия. Если последний символ этой строки – обратная косая черта, она будет трактоваться как путь к исходной папке, которая должна отображаться в диалоге (имя файла при этом будет пустым). Если в этом параметре передано значение NULL или указатель на пустую строку, исходно в диалоге будет отображаться папка с загруженной в данный момент схемой, а имя файла будет пустым.

Flags

Битовые флаги, определяющие поведение диалога:

RDS_CFD_OPEN	Показать диалог открытия файла (не может использоваться вместе с флагом RDS_CFD_SAVE).
RDS_CFD_SAVE	Показать диалог сохранения файла (не может использоваться вместе с флагом RDS_CFD_OPEN).
RDS_CFD_CREATEPROMPT	Для диалога сохранения: пользователь должен подтвердить создание нового файла, если введено имя несуществующего файла.
RDS_CFD_OVERWRITEPROMPT	Для диалога сохранения: пользователь должен подтвердить перезапись файла, если введено имя существующего файла.
RDS_CFD_MUSTEXIST	Для диалога открытия: пользователю запрещено выбирать несуществующие файлы.
RDS_CFD_ABSPATH	В возвращаемом функцией пути к файлу стандартные пути РДС не будут заменены на их символические обозначения (см. стр. 189).

Filter

Указатель на строку с фильтрами имен файлов для диалога, или NULL, если в диалоге всегда нужно отображать все файлы. Каждый фильтр состоит из названия, видимого пользователем в списке фильтров, и шаблона имени файла с использованием обычных метасимволов “*” и “?” (можно указать несколько шаблонов, разделив их точкой с запятой). Название и шаблон разделяются символом вертикальной черты “|”. Фильтры в строке отделяются друг от друга символов перевода строки “\n” (код 10). Например, передача в параметре Filter строки “Текстовые файлы|.txt;*.log\nВсе файлы|*.*” приведет к тому, что в выпадающем списке фильтров диалога будет два варианта: “Текстовые файлы” и “Все файлы”. При выборе первого из них в диалоге будут отображаться только файлы с расширениями “txt” и “log”, при выборе второго – все файлы.

DefExt

Указатель на строку с расширением по умолчанию, которое нужно дать введенному пользователем вручную имени файла, если он не ввел расширение сам, или NULL, если расширение по умолчанию следует взять из самого первого фильтра в строке Filter (если она передана).

Title

Указатель на строку с заголовком диалога, или NULL, если диалог должен иметь заголовок “Сохранить как” (при наличии в параметре Flags флага RDS_CFD_SAVE) или “Открыть” (при наличии флага RDS_CFD_OPEN).

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую путь к выбранному файлу, если пользователь нажал кнопку “ОК”. Если пользователь нажал кнопку “Отмена” или просто закрыл окно диалога, возвращается NULL.

Примечания:

Эта функция используется в тех случаях, когда нужно запросить у пользователя имя какого-либо файла для чтения (тогда указывается флаг RDS_CFD_OPEN) или для записи (тогда указывается флаг RDS_CFD_SAVE) данных. Если не указан ни один из этих флагов, считается, что указан RDS_CFD_OPEN.

Для запроса имен папок следует использовать функцию rdsCallDirDialog (стр. 197).

Динамическая строка, созданная функцией rdsCallFileDialog, должна быть **обязательно** освобождена функцией rdsFree (стр. 187). Пример использования функции приведен в §4.3.

См. также:

rdsCallDirDialog (стр. 197), rdsFree (стр. 187).

А.5.5.4. rdsExecutePrintDialog – открыть окно печати подсистемы

Функция rdsExecutePrintDialog открывает стандартное окно РДС для печати содержимого подсистемы (именно это окно открывается при выборе пункта меню “Файл | Печать...”).

```
void RDSCALL rdsExecutePrintDialog(  
    RDS_BHANDLE System           // Подсистема или NULL  
);
```

Тип указателя на эту функцию:

RDS_VBh

Параметр:

System

Идентификатор подсистемы (RDS_BHANDLE), которая должна быть выбрана в окне печати, или NULL, если какую-либо конкретную подсистему указывать не нужно. Пользователь всегда может выбрать в окне печати любую из открытых в данный момент подсистем.

Примечания:

Эта функция обычно используется для более быстрого доступа к окну печати схемы. Принудительно начать печать с помощью этой функции нельзя.

А.5.5.5. rdsInputString – окно ввода строки

Функция rdsInputString открывает универсальный диалог для ввода одной строки.

```
LPSTR RDSCALL rdsInputString(  
    LPSTR WinCaption,           // Заголовок окна  
    LPSTR StrCaption,          // Заголовок поля ввода  
    LPSTR Default,             // Исходное значение  
    int Width                   // Ширина поля ввода  
);
```


Тип указателя на эту функцию:

RDS_SSSSI

Параметры:

WinCaption

Указатель на строку с заголовком окна или NULL, если заголовок должен быть пустым.

StrCaption

Указатель на строку с заголовком поля ввода строки (текстом, отображаемым слева от поля ввода) или NULL, если этот заголовок должен быть пустым.

Default

Указатель на строку с исходным значением поля ввода диалога или NULL, если на момент открытия окна поле ввода должно быть пустым.

Width

Ширина поля ввода в точках экрана.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую введенный текст, если пользователь нажал кнопку “ОК”. Если пользователь нажал кнопку “Отмена” или просто закрыл окно диалога, возвращается NULL.

Примечания:

Эта функция используется в тех случаях, когда необходимо дать пользователю возможность ввести одну строку, при этом в заголовке окна и поля ввода можно разместить текст, поясняющий назначение вводимой строки. Для ввода нескольких значений или создания более сложных диалогов следует использовать вспомогательные объекты РДС, предназначенные для создания модальных окон (стр. 491).

Динамическая строка, созданная функцией `rdsInputString`, должна быть **обязательно** освобождена функцией `rdsFree` (стр. 187). Пример использования функции приведен в §2.7.1.

См. также:

`rdsFORMCreate` (стр. 491), `rdsFree` (стр. 187).

A.5.5.6. `rdsMessageBox` – вывод окна сообщения

Функция `rdsMessageBox` выводит стандартное окно сообщения Windows, не останавливая при этом поток расчета, даже если она вызвана из него.

```
int RDSCALL rdsMessageBox(  
    LPSTR Text,           // Текст сообщения  
    LPSTR Caption,       // Заголовок окна сообщения  
    int Flags             // Флаги Windows API (MB_*)  
);
```

Тип указателя на эту функцию:

RDS_ISSI

Параметры:

Text

Указатель на строку с текстом выводимого сообщения. Сообщение может состоять из нескольких строк, разделенных символом перевода строки “\n” (код 10).

Caption

Указатель на строку с заголовком окна выводимого сообщения.

Flags

Битовые флаги сообщения, определяющие набор кнопок в окне, иконку сообщения и т.п. Эти флаги совпадают с флагами стандартной функции Windows API `MessageBox`. Для задания иконки сообщения чаще всего используются следующие флаги:

<code>MB_ICONERROR</code>	Знак “стоп” – обычно используется в сообщениях о фатальных ошибках.
<code>MB_ICONINFORMATION</code>	Знак информационного сообщения – обычно используется в различных сообщениях, не связанных с ошибками.
<code>MB_ICONQUESTION</code>	Вопросительный знак – используется в запросах пользователю, когда он должен ответить “Да”/“Нет” или подтвердить/отменить какое-либо действие.
<code>MB_ICONWARNING</code>	Восклицательный знак – обычно используется в предупреждающих сообщениях.

Для задания набора кнопок в окне сообщения используются следующие флаги:

<code>MB_ABORTRETRYIGNORE</code>	В окне три кнопки: “Стоп”, “Повтор” и “Пропустить”.
<code>MB_OK</code>	В окне единственная кнопка “ОК”.
<code>MB_OKCANCEL</code>	В окне две кнопки: “ОК” и “Отмена”.
<code>MB_RETRYCANCEL</code>	В окне две кнопки: “Повтор” и “Отмена”.
<code>MB_YESNO</code>	В окне две кнопки: “Да” и “Нет”.
<code>MB_YESNOCANCEL</code>	В окне три кнопки: “Да”, “Нет” и “Отмена”.

Для задания выбранной по умолчанию кнопки используются следующие флаги:

<code>MB_DEFBUTTON1</code>	По умолчанию выбрана первая кнопка.
<code>MB_DEFBUTTON2</code>	По умолчанию выбрана вторая кнопка.
<code>MB_DEFBUTTON3</code>	По умолчанию выбрана третья кнопка.

В параметре `Flags` могут также указываться прочие флаги, используемые в функции Windows API `MessageBox`.

Возвращаемое значение:

Стандартная константа Windows API, указывающая на нажатую пользователем кнопку окна сообщения:

<code>IDABORT</code>	Нажата кнопка “Стоп”.
<code>IDCANCEL</code>	Нажата кнопка “Отмена”, клавиша Esc, или пользователь просто закрыл окно сообщения.
<code>IDIGNORE</code>	Нажата кнопка “Пропустить”.
<code>IDNO</code>	Нажата кнопка “Нет”.
<code>IDOK</code>	Нажата кнопка “ОК”.
<code>IDRETRY</code>	Нажата кнопка “Повтор”.
<code>IDYES</code>	Нажата кнопка “Да”.

Если сообщение выведено в режиме расчета, всегда возвращается константа `IDCANCEL`.

Примечания:

Работа функции `rdsMessageBox` зависит от того, из какого потока РДС она вызвана. Если функция вызвана из главного потока, она не вернет управление до тех пор, пока пользователь не закроет окно сообщения. В этом случае в параметре `Flags` можно задавать различные флаги кнопок и анализировать возвращенное функцией значение, определяя нажатую пользователем кнопку.

Если же `rdsMessageBox` вызвана из потока расчета, она немедленно вернет константу `IDCANCEL` не открывая окна. Окно с сообщением откроется позже, при первой возможности, в главном потоке РДС, при этом флаги кнопок будут проигнорированы и в окне будет единственная кнопка “ОК”.

Таким образом, функцию `rdsMessageBox` можно безопасно вызывать из потока расчета (например, из реакции блока на такт моделирования `RDS_BFM_MODEL`, см. стр. 40) для вывода сообщений пользователю, не вызывая при этом остановку потока. Разумеется, в этом случае нельзя будет узнать, какую кнопку нажал пользователь, поскольку моменты вызова функции и фактического открытия окна будут разнесены во времени.

См. также:

`rdsBlockMessageBox` (стр. 569).

А.5.6. Операции с блоками и связями

Описываются функции, позволяющие получать и изменять параметры блоков и связей, управлять их работой и программно изменять схему.

А.5.6.1. `rdsActivateOutputConnections` – активация выходных связей блока

Функция `rdsActivateOutputConnections` принудительно запускает передачу выходов указанного блока по связям на входы соединенных с ним блоков.

```
void RDSCALL rdsActivateOutputConnections(  
    RDS_BHANDLE Block,           // Идентификатор блока  
    BOOL Logic                   // Использовать логику выходов  
);
```

Тип указателя на эту функцию:

`RDS_VBhB`

Параметры:

`Block`

Идентификатор блока, выходы которого необходимо немедленно передать по связям. Это обязательно должен быть простой блок (то есть блок типа `RDS_BTSIMPLEBLOCK`, см. стр. 113), вызов функции для подсистем или внешних входов/выходов игнорируется.

`Logic`

`TRUE`, если нужно выполнить нормальную передачу данных, то есть такую же, какая производится в режиме расчета (см. §1.3 и §1.5):

- данные передаются только в том случае, если вторая сигнальная переменная блока `Block` (сигнал готовности “Ready”) не равна нулю;
- не передаются данные выходов, связанная логическая переменная которых равна нулю;
- связанные сигналы входов других блоков, к которым подключены сработавшие связи, взводятся;
- если у входа, к которому подключена сработавшая связь, установлен флаг “пуск”, первая сигнальная переменная (“Start”) его блока взводится;
- после передачи у блока `Block` сбрасывается сигнал готовности.

`FALSE`, если нужно выполнить “инициализационную” передачу, то есть передачу без учета сигналов и логики:

- данные передаются независимо от значения второй сигнальной переменной блока (сигнала готовности “Ready”), после передачи этот сигнал не сбрасывается;
- данные передаются независимо от значения связанных логических переменных выходов;
- значения сигнальных выходов не передаются;
- в блоках-получателях данных не взводятся никакие сигналы.

Примечания:

Эта функция чаще всего используется для передачи данных по связям вне режима расчета. Пример ее использования приведен в §2.13.2.

A.5.6.2. rdsAltConnAppearanceOp – операции с альтернативным внешним видом связи или шины

Функция rdsAltConnAppearanceOp позволяет временно изменять внешний вид связи или шины (визуально выделять ее) или считывать параметры такого измененного внешнего вида.

```
int RDSCALL rdsAltConnAppearanceOp(
    RDS_CHANDLE Conn,           // Идентификатор связи/шины
    int Op,                     // Операция (RDS_CAO*)
    int Num,                    // Номер внешнего вида
    RDS_PCONNAPPEARANCE pData // Описание внешнего вида
);
```

Тип указателя на эту функцию:

RDS_IChIICa

Параметры:

Conn

Идентификатор связи или шины, для которой нужно установить альтернативный внешний вид или считать его параметры.

Op

Выполняемая операция, одна из констант RDS_CAO* (см. ниже).

Num

Номер альтернативного внешнего вида или число таких видов в зависимости от значения Op (см. ниже).

pData

Указатель на структуру описания внешнего вида связи (RDS_CONNAPPEARANCE, см. стр. 118), из которой берутся или в которую записываются, в зависимости от значения Op, параметры альтернативного внешнего вида связи.

Возвращаемое значение:

Зависит от параметра Op, см. ниже.

Примечания:

Эта функция позволяет создавать для связи или шины альтернативные наборы параметров внешнего вида и оперативно переключаться между ними. Чаще всего этот механизм используется для временного визуального выделения какой-либо группы связей: его преимущество перед использованием функции постоянного изменения внешнего вида связи rdsSetConnAppearance (стр. 247) заключается в том, что альтернативные наборы внешнего вида связи не запоминаются при сохранении схемы, поэтому после загрузки схемы

все временные выделения исчезнут автоматически. Кроме того, функция `rdsAltConnAppearanceOp` автоматически запоминает исходный внешний вид связи и позволяет быстро вернуться к нему.

Действия, выполняемые функцией, и возвращаемое ей значение зависят от параметра `Op`, который может принимать одно из следующих значений:

`RDS_CAOCOUNT`

Функция возвращает число запомненных в связи или шине `Conn` альтернативных внешних видов. Значения параметров `Num` и `pData` не используются.

`RDS_CAODELETE`

Удалить из связи или шины `Conn` альтернативный внешний вид с номером `Num`. Все виды с большими номерами сдвигаются вниз на единицу (то есть, если удалить внешний вид с номером 5, вид с номером 6 станет видом 5, вид 7 станет видом 6 и т.д.). Параметр `pData` не используется. Функция возвращает число оставшихся альтернативных внешних видов. При удалении самого последнего внешнего вида автоматически восстанавливается исходный внешний вид связи.

`RDS_CAOGGET`

Считать из `Conn` параметры альтернативного внешнего вида с номером `Num` в структуру, на которую указывает `pData`. Функция возвращает 1, если в `Conn` есть внешний вид с таким номером, и 0 в противном случае.

`RDS_CAOPREALLOCATE`

Подготовиться к созданию в связи или шине `Conn` альтернативных видов общим числом `Num`. Если на данный момент в `Conn` уже больше `Num` видов, “лишние” виды будут удалены. Функция всегда возвращает 0, параметр `pData` не используется. Эта операция позволяет ускорить создание большого количества внешних видов – при желании, ее можно не выполнять, сразу создавая внешние виды вызовами `rdsAltConnAppearanceOp` с параметром `Op`, равным `RDS_CAOSSET`.

`RDS_CAORESTORE`

Восстановить исходный (до установки альтернативных) внешний вид связи или шины `Conn`. Параметры `Num` и `pData` не используются. Функция всегда возвращает 0.

`RDS_CAOSSET`

Запомнить в `Conn` новый альтернативный внешний вид с номером `Num`, взяв его параметры из структуры, на которую указывает `pData`. Если в связи или шине `Conn` уже есть альтернативный внешний вид с таким номером, его параметры будут заменены и функция вернет `Num`, в противном случае будет создан новый внешний вид и функция вернет его номер. Если в параметре `pData` передано значение `NULL`, параметры запоминаемого внешнего вида будут скопированы из исходного внешнего вида связи. Запоминание нового альтернативного внешнего вида не приводит к его установке, для этого нужно явно вызвать `rdsAltConnAppearanceOp` с параметром `Op`, равным `RDS_CAOSSETCURRENT`.

`RDS_CAOSSETCURRENT`

Применить к `Conn` альтернативный внешний вид с номером `Num`. Параметр `pData` не используется. Функция возвращает 1, если внешний вид установлен (то есть в `Conn` есть запомненный внешний вид с таким номером), и 0 в противном случае.

Пример использования функции `rdsAltConnAppearanceOp` проведен в §2.13.4.

См. также:

`rdsSetConnAppearance` (стр. 247), `rdsGetConnAppearance` (стр. 225).

A.5.6.3. rdsBlockByFullName – блок по его полному имени

Функция `rdsBlockByFullName` возвращает идентификатор блока, полное имя которого передано в ее параметрах.

```
RDS_BHANDLE RDCALL rdsBlockByFullName(  
    LPSTR FullName,           // Полное имя блока  
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

`RDS_BhSBd`

Параметры:

`FullName`

Указатель на строку с полным именем блока. Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем `Block1` находится в подсистеме `Sys100`, которая, в свою очередь, находится в подсистеме `Sys1` корневой подсистемы.

`pDescr`

Указатель на структуру описания блока `RDS_BLOCKDESCRIPTION` (стр. 113), которую функция должна заполнить параметрами найденного блока. Если вызывающей программе не нужно описание блока, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Уникальный идентификатор найденного блока (`RDS_BHANDLE`) или `NULL`, если блока с таким именем нет в загруженной в данный момент схеме.

Примечания:

Эта функция позволяет по строке полного имени блока, уникальной для любого блока схемы, получить идентификатор этого блока, который можно использовать в других сервисных функциях для выполнения с этим блоком каких-либо действий. Для поиска блока по имени в конкретной подсистеме следует использовать другую функцию – `rdsGetChildBlockByName` (стр. 224).

См. также:

`RDS_BLOCKDESCRIPTION` (стр. 113), `rdsBlockOrConnByExtId` (стр. 206),
`rdsCreateFullBlockNameString` (стр. 210),
`rdsGetChildBlockByName` (стр. 224).

A.5.6.4. rdsBlockOrConnByExtId – блок или связь по внешнему идентификатору

Функция `rdsBlockOrConnByExtId` ищет блок, связь или шину по уникальному целому идентификатору (такие идентификаторы используются при управлении РДС из другого приложения, см. §3.5).

```
BOOL RDCALL rdsBlockOrConnByExtId(  
    RDS_BHANDLE Parent,           // Родительская подсистема  
    DWORD ExtId,                 // Целый идентификатор
```

```
        RDS_PFINDBYEXTIDDATA pData // Результаты поиска
    );
```

Тип указателя на эту функцию:

RDS_BBhDwFbei

Параметры:

Parent

Идентификатор (RDS_BHANDLE) подсистемы, начиная с которой нужно искать блок, связь или шину – поиск будет осуществляться в этой подсистеме и во всех ее внутренних подсистемах, независимо от уровня вложенности. Если нужно искать блок во всей загруженной схеме, в этом параметре можно передать NULL.

ExtId

Уникальный целый идентификатор блока, связи или шины.

pData

Указатель на структуру RDS_FINDBYEXTIDDATA (стр. 126), в которую функция запишет результат поиска. Если вызывающей программе нужно проверить, существует ли объект с идентификатором ExtId в схеме, но не требуется узнавать его внутренний идентификатор (RDS_BHANDLE или RDS_CHANDLE), в этом параметре можно передать NULL.

Возвращаемое значение:

TRUE, если блок, связь или шина с идентификатором ExtId найдены, FALSE в противном случае.

Примечания:

Эта функция позволяет найти блок, связь или шину, зная его или ее уникальный целый внешний идентификатор ExtId. Параметр Parent позволяет ограничить поиск какой-либо подсистемой и ее внутренними блоками. По результатам поиска заполняется структура RDS_FINDBYEXTIDDATA: в поле Found заносится признак успешности поиска (это же значение возвращается функцией), в поле Block – внутренний идентификатор найденного блока, в поле Conn – внутренний идентификатор найденной шины или связи, в поле Type – тип найденного объекта. Только одно из полей Block и Conn будет заполнено (другое поле получит значение NULL), поскольку идентификаторы блоков и связей не пересекаются, и параметру функции ExtId будет соответствовать либо блок, либо связь или шина.

Целые идентификаторы блоков и связей используются, в основном, при управлении РДС из других приложений – в отличие от внутренних идентификаторов, они не изменяются при сохранении схемы и ее повторной загрузке. В моделях блоков и модулях автоматической компиляции чаще используются внутренние идентификаторы RDS_BHANDLE и RDS_CHANDLE, поскольку обращение к блокам и связям с их помощью выполняется гораздо быстрее. Для однозначной идентификации блока или связи в схеме можно использовать один из трех следующих способов, каждый из которых имеет свои достоинства и недостатки:

<i>Способ идентификации</i>	<i>Достоинства</i>	<i>Недостатки</i>
Полное имя блока	<ul style="list-style-type: none"> ● понятно для пользователя (может вводиться вручную); ● не изменяется после сохранения схемы; ● можно использовать при внешнем управлении РДС. 	<ul style="list-style-type: none"> ● нельзя использовать для связей и шин; ● изменяется при переименовании блока.
Внутренние идентификаторы (RDS_BHANDLE и RDS_CHANDLE)	<ul style="list-style-type: none"> ● очень быстрый доступ к блокам и связям. 	<ul style="list-style-type: none"> ● каждый раз присваиваются заново после загрузки схемы; ● нельзя использовать при внешнем управлении РДС; ● разные типы идентификаторов для блоков и связей/шин.
Внешние идентификаторы (DWORD)	<ul style="list-style-type: none"> ● не изменяются при сохранении схемы и переименовании блоков; ● можно использовать при внешнем управлении РДС. 	<ul style="list-style-type: none"> ● относительно медленный поиск блока, связи или шины по идентификатору.

См. также:

RDS_BLOCKDESCRIPTION (стр. 113), RDS_CONNDESCRIPTION (стр. 119),
rdsUpdateExtIdsRange (стр. 176), rdsctrlBlockNameByExtId (стр. 618),
rdsctrlBlockExtIdByName (стр. 615).

A.5.6.5. rdsCountBlocks – подсчет блоков

Функция rdsCountBlocks возвращает число блоков заданного типа в указанной подсистеме.

```
int RDSCALL rdsCountBlocks(
    RDS_BHANDLE Parent,    // Подсистема
    int Type,              // Маска типов блоков
    BOOL Recurse           // Подсчитывать во вложенных подсистемах
);
```

Тип указателя на эту функцию:

RDS_IBhIB

Параметры:

Parent

Идентификатор (RDS_BHANDLE) подсистемы, в которой нужно подсчитать число блоков. Передача NULL в этом параметре приведет к возврату функцией нуля.

Type

Маска типов блоков, которые нужно подсчитать – стандартные константы типов блоков (RDS_BT*, см. стр. 113), объединенные битовым ИЛИ. Если нужно подсчитать число всех блоков независимо от типа, в этом параметре можно передать 0.

Recurse

TRUE, если нужно подсчитывать число блоков не только в подсистеме Parent, но и во всех ее вложенных подсистемах. FALSE, если нужно подсчитать только блоки, непосредственно находящиеся в подсистеме Parent.

Возвращаемое значение:

Число блоков указанного типа в указанной подсистеме.

См. также:

rdsEnumBlocks (стр. 212).

A.5.6.6. rdsCreateBlockFromFile – загрузить блок из файла

Функция rdsCreateBlockFromFile загружает блок из файла и вставляет его в заданную подсистему.

```
RDS_BHANDLE RDSCALL rdsCreateBlockFromFile(  
    LPSTR FilePath,                // Путь к файлу  
    RDS_BHANDLE Parent,           // Подсистема  
    int x, int y,                 // Координаты  
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

RDS_BhSBhIIBd

Параметры:

FilePath

Указатель на строку с именем файла. Имя файла может содержать как полный путь, так и сокращенный (с символическими обозначениями путей, см. стр. 189). В качестве файла блока можно указать файл схемы, тогда эта схема будет добавлена в загруженную схему как подсистема.

Parent

Идентификатор подсистемы, внутрь которой нужно добавить блок.

x, y

Координаты точки привязки загружаемого блока на рабочем поле подсистемы Parent в масштабе 100%.

pDescr

Указатель на структуру описания блока RDS_BLOCKDESCRIPTION (стр. 113), которую функция должна заполнить параметрами добавленного блока. Если вызывающей программе не нужно описание блока, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор добавленного в подсистему блока или NULL в случае ошибки.

Примечания:

Эта функция загружает блок из файла и вставляет его в подсистему Parent. В параметре FilePath можно указать путь к файлу сохраненного блока (обычно они имеют расширение “.blk”, в таких файлах хранятся блоки в библиотеке и панели блоков РДС, см. §1.1) или файлу сохраненной схемы (с расширением “.rds”) – в последнем случае сохраненная схема станет внутренней подсистемой в Parent. Точкой привязки загруженного блока (для блоков с векторной картинкой – положением начала координат

этой картинке, для программно рисуемых блоков и блоков, изображаемых прямоугольником с текстом – координатами левого верхнего угла изображения) станут значения из параметров *x* и *y*.

Следует помнить, что новый блок может быть добавлен в схему не немедленно: если в данный момент идет расчет, добавление будет отложено до конца очередного такта.

Пример использования функции `rdsCreateBlockFromFile` приведен в §2.16.2.

См. также:

`rdsDuplicateBlock` (стр. 212), `rdsDeleteBlock` (стр. 210).

A.5.6.7. `rdsCreateFullBlockNameString` – полное имя блока

Функция `rdsCreateFullBlockNameString` формирует в памяти динамическую строку с полным именем блока, идентификатор которого передан в ее параметрах.

```
LPSTR RDSCALL rdsCreateFullBlockNameString(  
    RDS_BHANDLE Block,      // Идентификатор блока  
    int *pLength            // Возвращаемая длина строки  
);
```

Тип указателя на эту функцию:

`RDS_SBhpl`

Параметры:

Block

Идентификатор блока, полное имя которого должна вернуть функция.

pLength

Указатель на целую переменную, в которую функция должна записать длину получившейся строки. Если вызывающей программе не нужна длина строки, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, в которой сформировано полное имя блока. Полное имя начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, завершающееся именем самого блока (например, “:Sys1:Sys100:Block1”). В случае ошибки возвращается `NULL`.

Примечания:

Динамическая строка, созданная функцией `rdsCreateFullBlockNameString`, должна быть *обязательно* освобождена функцией `rdsFree` (стр. 187).

См. также:

`rdsFree` (стр. 187), `rdsBlockByFullName` (стр. 206).

A.5.6.8. `rdsDeleteBlock` – удаление блока

Функция `rdsDeleteBlock` удаляет из схемы блок, идентификатор которого передан в ее параметрах.

```
void RDSCALL rdsDeleteBlock(  
    RDS_BHANDLE Block      // Удаляемый блок  
);
```

Тип указателя на эту функцию:

RDS_VBh

Параметр:

Block

Идентификатор блока, который необходимо удалить из схемы.

Примечания:

Эта функция удаляет блок Block из схемы, если это возможно. Блок не может быть удален, если:

- его моделью в данный момент открыто модальное окно;
- это подсистема, моделью одного из внутренних блоков которой открыто модальное окно;
- это корневая подсистема.

Удаление блока, которому принадлежит открытое в данный момент модальное окно, запрещено, так как это может привести к выгрузке DLL с его моделью, в которой, вероятнее всего, находится и функция открытого окна. Поскольку окно останется открытым и будет продолжать вызывать свою функцию в ответ на системные события, это может привести к ошибкам. Корневая подсистема не может быть удалена в принципе, поскольку это приведет к удалению из памяти всей загруженной схемы.

Вызов функции rdsDeleteBlock не всегда удаляет блок немедленно: если в данный момент идет расчет, блок будет удален только по окончании очередного такта. Если модель пытается удалить свой собственный блок, он тоже не будет удален немедленно: РДС дожидается завершения функции его модели, и только тогда удалит этот блок.

Пример использования функции rdsDeleteBlock приведен в §2.16.2.

См. также:

rdsBlockModalWinOpen (стр. 147), rdsDeleteConnection (стр. 211).

A.5.6.9. rdsDeleteConnection – удаление связи или шины

Функция rdsDeleteConnection удаляет из схемы связь или шину, идентификатор которой передан в ее параметрах.

```
void RDSCALL rdsDeleteConnection(  
    RDS_CHANDLE Conn      // Удаляемая связь/шина  
);
```

Тип указателя на эту функцию:

RDS_VCh

Параметр:

Conn

Идентификатор связи или шины, которую необходимо удалить из схемы.

Примечания:

Эта функция удаляет из схемы связь или шину с идентификатором Conn (в РДС шина является частным случаем связи). Ее вызов не всегда удаляет связь немедленно: если в данный момент идет расчет, она будет удалена только по окончании очередного такта.

Пример использования функции rdsDeleteConnection приведен в §2.16.2.

См. также:

rdsDeleteBlock (стр. 210).

A.5.6.10. rdsDuplicateBlock – сделать копию блока

Функция `rdsDuplicateBlock` вставляет копию блока, идентификатор которого передан в ее параметрах, в заданную подсистему.

```
RDS_BHANDLE RDCALL rdsDuplicateBlock(  
    RDS_BHANDLE Block,           // Копируемый блок  
    RDS_BHANDLE Parent,         // Подсистема  
    int x, int y,               // Координаты  
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

`RDS_BhBhBhIIBd`

Параметры:

`Block`

Идентификатор блока, который нужно скопировать.

`Parent`

Идентификатор подсистемы, внутрь которой нужно добавить копию блока.

`x, y`

Координаты точки привязки создаваемого блока на рабочем поле подсистемы `Parent` в масштабе 100%.

`pDescr`

Указатель на структуру описания блока `RDS_BLOCKDESCRIPTION` (стр. 113), которую функция должна заполнить параметрами добавленного блока. Если вызывающей программе не нужно описание блока, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Идентификатор добавленного в подсистему блока или `NULL` в случае ошибки.

Примечания:

Эта функция делает копию блока `Block` и вставляет ее в подсистему `Parent`. Если в этой подсистеме уже есть блок с таким же именем, имя блока-копии будет скорректировано так, чтобы быть уникальным в подсистеме. Точкой привязки нового блока (для блоков с векторной картинкой – положением начала координат этой картинки, для программно рисуемых блоков и блоков, изображаемых прямоугольником с текстом – координатами левого верхнего угла изображения) станут значения из параметров `x` и `y`.

Следует помнить, что новый блок может быть добавлен в схему не немедленно: если в данный момент идет расчет, добавление будет отложено до конца очередного такта.

Пример использования функции `rdsDuplicateBlock` приведен в §2.16.2.

См. также:

`rdsCreateBlockFromFile` (стр. 209), `rdsDeleteBlock` (стр. 210),
`rdsMakeUniqueBlockName` (стр. 239).

A.5.6.11. rdsEnumBlocks – перебрать все блоки подсистемы

Функция `rdsEnumBlocks` вызывает пользовательскую функцию, указатель на которую передается в параметрах, для каждого блока указанной подсистемы.

```

RDS_BHANDLE RDSCALL rdsEnumBlocks(
    RDS_BHANDLE Parent,    // Подсистема
    int Type,              // Маска типов блоков
    BOOL Recurse,          // Вызывать во вложенных подсистемах
    RDS_BBhpV Callback,    // Функция пользователя
    LPVOID Data            // Параметр функции пользователя
);

```

Тип указателя на эту функцию:

RDS_BhBhIBCbpV

Параметры:

Parent

Идентификатор подсистемы, для блоков которой нужно вызвать функцию пользователя Callback.

Type

Маска типов блоков, для которых нужно вызывать функцию – стандартные константы типов блоков (RDS_BT*, см. стр. 113), объединенные битовым ИЛИ. Если нужно вызвать функцию для каждого блока подсистемы независимо от его типа, в этом параметре можно передать 0.

Recurse

TRUE, если нужно вызывать функцию Callback не только для блоков, непосредственно находящихся в подсистеме Parent, но для блоков во всех ее вложенных подсистемах. FALSE, если нужно вызывать функцию только для блоков, непосредственно находящихся в Parent.

Callback

Указатель на пользовательскую функцию, которую нужно вызвать для каждого блока, удовлетворяющего параметрам Type и Recurse. Пользовательская функция должна иметь следующий вид:

```

    BOOL RDSCALL имя_функции(RDS_BHANDLE block, LPVOID param);

```

В параметре block пользовательской функции передается идентификатор блока, для которого вызвана функция, в параметре param – параметр Data (см. ниже) без какой-либо обработки. Функция пользователя должна вернуть TRUE, если перебор блоков необходимо продолжить, и FALSE, если его нужно немедленно остановить.

Data

Параметр типа void*, передаваемый в пользовательскую функцию при каждом ее вызове.

Возвращаемое значение:

Идентификатор блока, для которого функция пользователя вернула FALSE. Если функция пользователя всегда возвращала TRUE (перебраны все блоки), вместо идентификатора блока возвращается NULL.

Примечания:

Функция rdsEnumBlocks обычно используется для выполнения каких-либо однотипных действий над блоками подсистемы. Ее также можно использовать для поиска блока по какому-либо критерию: если написать пользовательскую функцию, которая будет вызываться для каждого блока таким образом, чтобы она возвращала FALSE, если блок, переданный в ее параметре block, удовлетворяет заданному критерию, то результатом возврата rdsEnumBlocks будет искомый блок.

Функция перебирает блоки в порядке, определяемом внутренней логикой РДС, программист никак не может изменить этот порядок.

Параметр Data, имеющий тип LPVOID (произвольный указатель), передается в вызываемую функцию пользователя без изменений. Это единственный способ передать ей какие-либо данные: например, можно передать указатель на какую-либо структуру, а внутри пользовательской функции привести его к нужному типу и обращаться к полям этой структуры.

Для перебора всех блоков заданного типа в какой-либо подсистеме вместо rdsEnumBlocks можно использовать функции rdsGetFirstBlock (стр. 228) и rdsGetNextBlock (стр. 233).

Пример:

В этом примере rdsEnumBlocks используется для поиска первого попавшегося блока схемы, комментарий которого содержит заданный текст.

```
// Функция пользователя
BOOL RDSCALL MyEnumCallback(RDS_BHANDLE block, LPVOID param)
{ char *text=(char*)param; // Приводим тип к char*
  // Получаем описание блока block с комментарием
  RDS_BLOCKDESCRIPTION Descr;
  Descr.servSize=sizeof(Descr);
  rdsGetBlockDescription(block, &Descr);
  // Возвращаем FALSE, если в комментарии есть текст param
  if(strstr(Descr.BlockComment, text))
    return FALSE;
  return TRUE;
}

...

// Поиск во всей схеме простого блока, в комментарии которого
// содержится текст "тест"
RDS_BHANDLE found=rdsEnumBlocks(
  rdsGetRootSystem(NULL), // Корневая подсистема
  RDS_BTSIMPLEBLOCK,     // Только простые блоки
  TRUE,                  // С вложенными подсистемами
  MyEnumCallback,        // Вызываемая функция
  "тест");                // Текст для поиска
// В found находится идентификатор найденного блока или NULL
```

См. также:

rdsEnumConnectedBlocks (стр. 214),
rdsEnumConnectedBlocksByVar (стр. 216),
rdsBroadcastFunctionCallsEx (стр. 308), rdsBCLCreateList (стр. 420),
rdsGetFirstBlock (стр. 228), rdsGetNextBlock (стр. 233).

A.5.6.12. rdsEnumConnectedBlocks – перебрать все соединенные блоки

Функция rdsEnumConnectedBlocks вызывает пользовательскую функцию, указатель на которую передается в параметрах, для блоков, соединенных связями с заданным.

```
RDS_BHANDLE RDSCALL rdsEnumConnectedBlocks(
  RDS_BHANDLE Block,           // Заданный блок
  DWORD Flags,                 // Флаги (RDS_BEN_*)
  RDS_BhPdPdPv CallBack,      // Функция пользователя
```

```

        LPVOID Data // Параметр функции пользователя
    );

```

Тип указателя на эту функцию:

```
RDS_BhBhDwCb1pV
```

Параметры:

Block

Идентификатор блока, связи которого перебираются. Для соединенных с ним блоков будет вызвана функция пользователя `CallBack`.

Flags

Набор битовых флагов, управляющих работой функции, объединенных битовым ИЛИ:

<code>RDS_BEN_INPUTS</code>	Вызывать функцию <code>CallBack</code> для блоков, соединенных связями с входами блока <code>Block</code> .
<code>RDS_BEN_OUTPUTS</code>	Вызывать функцию <code>CallBack</code> для блоков, соединенных связями с выходами блока <code>Block</code> .
<code>RDS_BEN_TRACELINKS</code>	Если этот флаг не установлен, функция <code>CallBack</code> будет вызываться для всех блоков, непосредственно соединенных связями с блоком <code>Block</code> . Если флаг установлен, РДС будет проследивать связи до простых блоков и вызывать функцию только для них (см. ниже).

CallBack

Указатель на пользовательскую функцию, которую нужно вызвать для каждого блока, удовлетворяющего параметрам функции. Пользовательская функция должна иметь следующий вид:

```

BOOL RDSCALL имя_функции(
    RDS_PPOINTDESCRIPTION pNearPoint,
    RDS_PPOINTDESCRIPTION pFarPoint,
    LPVOID param)

```

В параметре `pNearPoint` пользовательской функции передается указатель на структуру `RDS_POINTDESCRIPTION` (стр. 133), описывающую точку связи, соединяющую эту связь с блоком `Block`. В параметре `pFarPoint` передается указатель на структуру описания точки связи, соединяющей ее с найденным блоком на другом конце этой связи (идентификатор этого блока содержится в структуре описания точки). Следует помнить, что при взведенном флаге `RDS_BEN_TRACELINKS` эти точки могут принадлежать разным связям. В параметре `param` передается параметр `Data` функции `rdsEnumConnectedBlocks` (см. ниже) без какой-либо обработки. Функция пользователя должна вернуть `TRUE`, если перебор блоков необходимо продолжить, и `FALSE`, если его нужно немедленно остановить.

Data

Параметр типа `void*`, без изменений передаваемый в пользовательскую функцию при каждом ее вызове.

Возвращаемое значение:

Идентификатор блока, для которого функция пользователя вернула `FALSE`. Если функция пользователя всегда возвращала `TRUE` (перебраны все соединенные блоки), вместо идентификатора блока возвращается `NULL`.

Примечания:

Функция `rdsEnumConnectedBlocks` обычно используется для выполнения каких-либо однотипных действий над блоками, соединенными с заданным. Например, с ее помощью можно информировать блоки, соединенные с выходами данного, об изменении значения выхода (обычно так осуществляется принудительная передача данных по связям вне режима расчета, см. §1.6).

Работой функции управляют битовые флаги, передаваемые в параметре `Flags`. Флаги `RDS_BEN_INPUTS` и `RDS_BEN_OUTPUTS` позволяют вызывать пользовательскую функцию только для блоков, соединенных с выходами данного или только для блоков, соединенных с входами (для вызова функции у всех соединенных блоков независимо от направления связи следует указать оба флага). Флаг `RDS_BEN_TRACELINKS` управляет прослеживанием связей внутри и наружу подсистем. Если он сброшен, функция `CallBack` будет вызываться только у блоков, непосредственно соединенных с блоком `Block` в его родительской подсистеме, при этом вложенные подсистемы и внешние входы и выходы будут считаться конечными точками связи – РДС не будет отслеживать эту связь дальше внутри вложенной подсистемы или наружу родительской. Если же флаг `RDS_BEN_TRACELINKS` взведен, функция будет отслеживать каждую связь внутри и наружу подсистем через внешние входы и выходы и вызывать функцию `CallBack` только для простых блоков, находящихся в конце такой цепочки связей.

Функция перебирает соединенные блоки в порядке, определяемом внутренней логикой РДС, программист никак не может изменить этот порядок.

Параметр `Data`, имеющий тип `LPVOID` (произвольный указатель), передается в вызываемую функцию пользователя без изменений. Это единственный способ передать ей какие-либо данные: например, можно передать указатель на какую-либо структуру, а внутри пользовательской функции привести его к нужному типу и обращаться к полям этой структуры.

Пример использования функции `rdsEnumConnectedBlocks` приведен в §2.13.2 и §2.13.4.

См. также:

`rdsEnumBlocks` (стр. 212), `rdsEnumConnectedBlocksByVar` (стр. 216),
`rdsGetBlockLink` (стр. 223), `RDS_POINTDESCRIPTION` (стр. 133).

A.5.6.13. `rdsEnumConnectedBlocksByVar` – перебрать все блоки, соединенные с заданной переменной

Функция `rdsEnumConnectedBlocksByVar` вызывает пользовательскую функцию, указатель на которую передается в параметрах, для блоков, соединенных связями с заданным входом или выходом заданного блока.

```
RDS_BHANDLE RDSCALL rdsEnumConnectedBlocksByVar (  
    RDS_BHANDLE Block,           // Заданный блок  
    int VarNum,                 // Номер входа или выхода  
    DWORD Flags,                // Флаги (RDS_BEN_*)  
    RDS_BhPdPdV CallBack,      // Функция пользователя  
    LPVOID Data                // Параметр функции пользователя  
);
```

Тип указателя на эту функцию:

```
RDS_BhBhIDwCb1pV
```


Параметры:

Block

Идентификатор блока, связи которого отслеживаются. Для соединенных с ним блоков будет вызвана функция пользователя `CallBack`.

VarNum

Номер статической переменной блока, связи которой нужно отслеживать. Нумерация переменных начинается с нуля.

Flags

Набор битовых флагов, управляющих работой функции, объединенных битовым ИЛИ. На данный момент в этом параметре может использоваться только один флаг (он используется и в функции `rdsEnumConnectedBlocks`, стр. 214):

`RDS_BEN_TRACELINKS` Если этот флаг не установлен, функция `CallBack` будет вызываться для всех блоков, непосредственно соединенных связями с блоком `Block`. Если флаг установлен, РДС будет проследивать связи до простых блоков и вызывать функцию только для них (см. ниже).

CallBack

Указатель на пользовательскую функцию, которую нужно вызвать для каждого блока, удовлетворяющего параметрам функции. Пользовательская функция должна иметь следующий вид:

```
BOOL RDSCALL имя_функции(
    RDS_PPOINTDESCRIPTION pNearPoint,
    RDS_PPOINTDESCRIPTION pFarPoint,
    LPVOID param)
```

В параметре `pNearPoint` пользовательской функции передается указатель на структуру `RDS_POINTDESCRIPTION` (стр. 133), описывающую точку связи, соединяющую связь с переменной `VarNum` блока `Block`. В параметре `pFarPoint` передается указатель на структуру описания точки связи, соединяющей ее с найденным блоком на другом конце этой связи (идентификатор этого блока содержится в структуре описания точки). Следует помнить, что при взведенном флаге `RDS_BEN_TRACELINKS` эти точки могут принадлежать разным связям. В параметре `param` передается параметр `Data` функции `rdsEnumConnectedBlocks` (см. ниже) без какой-либо обработки. Функция пользователя должна вернуть `TRUE`, если перебор блоков необходимо продолжить, и `FALSE`, если его нужно немедленно остановить.

Data

Параметр типа `void*`, без изменений передаваемый в пользовательскую функцию при каждом ее вызове.

Возвращаемое значение:

Идентификатор блока, для которого функция пользователя вернула `FALSE`. Если функция пользователя всегда возвращала `TRUE` (перебраны все соединенные блоки), вместо идентификатора блока возвращается `NULL`.

Примечания:

Функция `rdsEnumConnectedBlocksByVar` представляет собой более специализированную версию функции `rdsEnumConnectedBlocks` (стр. 214) – в отличие от последней, она отслеживает не все связи, соединенные с блоком `Block`, а только связи, подходящие или отходящие от его переменной с номером `VarNum`. В параметре функции указывается именно номер переменной, а не ее имя: имена переменных часто изменяются

пользователями, а их номера обычно жестко фиксированы и их изменение чаще всего приводит к неработоспособности блока (см. описание события RDS_BFM_VARCHHECK на стр. 48 и §2.5).

В параметре `Flags` в данный момент может передаваться только один битовый флаг – `RDS_BEN_TRACELINKS`. Он управляет прослеживанием связей внутри и наружу подсистем. Если он сброшен, функция `CallBack` будет вызываться только у блоков, непосредственно соединенных с блоком `Block` в его родительской подсистеме, при этом вложенные подсистемы и внешние входы и выходы будут считаться конечными точками связи: РДС не будет отслеживать эту связь дальше внутри вложенной подсистемы или наружу родительской. Если же флаг `RDS_BEN_TRACELINKS` взведен, функция будет отслеживать каждую связь внутри и наружу подсистем через внешние входы и выходы и вызывать функцию `CallBack` только для простых блоков, находящихся в конце такой цепочки связей.

Флаги `RDS_BEN_INPUTS` и `RDS_BEN_OUTPUTS`, использующиеся в функции `rdsEnumConnectedBlocks` для указания перебора входов или выходов блока, в этой функции не имеют смысла, так как переменная с номером `VarNum` уже является либо входом, либо выходом (если она внутренняя, к ней, очевидно, не подключено ни одной связи, и функция `CallBack` не будет вызвана ни разу). Их наличие или отсутствие в параметре `Flags` игнорируется.

Функция перебирает соединенные с переменной `VarNum` блоки в порядке, определяемом внутренней логикой РДС, программист никак не может изменить этот порядок.

Параметр `Data`, имеющий тип `LPVOID` (произвольный указатель), передается в вызываемую функцию пользователя без изменений. Это единственный способ передать ей какие-либо данные: например, можно передать указатель на какую-либо структуру, а внутри пользовательской функции привести его к нужному типу и обращаться к полям этой структуры.

См. также:

`rdsEnumBlocks` (стр. 212), `rdsEnumConnectedBlocks` (стр. 214),
`rdsGetBlockLink` (стр. 223), `RDS_POINTDESCRIPTION` (стр. 133).

A.5.6.14. `rdsFindNextConnectedLine` – найти отрезок связи, соединенный с точкой

Функция `rdsFindNextConnectedLine` находит внутри связи или шины очередной отрезок, соединенный с точкой с указанным номером.

```
int RDSCALL rdsFindNextConnectedLine(  
    RDS_CHANDLE Conn,           // Связь  
    int PointNum,               // Номер точки  
    int StartLineNum,           // Начальный номер отрезка  
    RDS_PLINEDescription pLDescr, // Заполняемое описание  
    int *pNextPoint             // Номер точки на другом конце  
);
```

Тип указателя на эту функцию:

`RDS_IChIILdpI`

Параметры:

`Conn`

Идентификатор связи (`RDS_CHANDLE`), внутри которой ищется отрезок.

PointNum

Номер точки, с которой найденный отрезок должен соединяться одним из своих двух концов. Нумерация точек в связи начинается с нуля.

StartLineNum

Номер отрезка, начиная с которого (включительно) производится поиск. Если отрезок StartLineNum соединен с точкой PointNum, функция вернет StartLineNum. Нумерация отрезков в связи начинается с нуля.

pLDescr

Указатель на структуру описания отрезка связи (RDS_LINEDESCRIPTION, см. стр. 129), которую функция должна заполнить параметрами найденного отрезка. Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

pNextPoint

Указатель на целую переменную, в которую функция запишет номер точки, находящейся на противоположном от точки PointNum конце найденного отрезка. Если вызывающей программе не нужен этот номер, в параметре pNextPoint можно передать NULL.

Возвращаемое значение:

Номер найденного отрезка внутри связи Conn. Если отрезок не найден, возвращается значение -1.

Примечания:

Функция rdsFindNextConnectedLine позволяет анализировать геометрию связи Conn. Связь в РДС состоит из точек и соединяющих их отрезков, причем к каждой точке может подходить один отрезок (для конечных точек связи), два отрезка (для точек излома) или больше двух отрезков (для точек ветвления). Отрезок связи может быть прямой линией или кривой Безье. Общее число точек и отрезков в связи можно получить из структуры RDS_CONNDESCRIPTION (стр. 119), заполняемой функцией rdsGetConnDescription (стр. 225).

Зная номер точки в связи, можно перебрать все подходящие к ней отрезки последовательными вызовами rdsFindNextConnectedLine: при первом вызове нужно в параметре StartLineNum передать 0, а при каждом следующем – число, на единицу большее возвращенного функцией, до тех пор, пока она не вернет -1. Возвращаемые функцией числа будут номерами найденных отрезков, причем функция одновременно будет заполнять структуру описания отрезка, указатель на которую передается в параметре pLDescr, и возвращать через параметр pNextPoint номер точки на другом конце найденного отрезка. Таким образом, двигаясь от точки к точке, можно проанализировать граф связи Conn со всеми ее внутренними точками и отрезками.

См. также:

rdsGetConnDescription (стр. 225), RDS_CONNDESCRIPTION (стр. 119), RDS_LINEDESCRIPTION (стр. 129), rdsGetLineDescription (стр. 231).

A.5.6.15. rdsForceBlockRedraw – перерисовать изображение блока

Функция rdsForceBlockRedraw уведомляет РДС о необходимости перерисовать изображение указанного блока при первой возможности.

```
void RDSCALL rdsForceBlockRedraw(
    RDS_BHANDLE Block    // Блок
);
```

Тип указателя на эту функцию:

RDS_VBh

Параметр:

Block

Идентификатор блока, изображение которого должно быть перерисовано.

Примечания:

Эта функция вызывается в тех случаях, когда внешний вид блока изменился неявным для РДС образом, и его изображение в подсистеме должно быть перерисовано при первой возможности. При любом вызове модели блока необходимость его перерисовки запоминается автоматически, поэтому вызов этой функции требуется очень редко: в основном, при работе функций немодальных окон, созданных блоком, которые что-то меняют в параметрах блока.

См. также:

rdsRefreshBlockWindows (стр. 263).

A.5.6.16. rdsGetBlockDescription – получить описание блока

Функция rdsGetBlockDescription заполняет структуру описания указанного блока.

```
int RDSCALL rdsGetBlockDescription(
    RDS_BHANDLE Block,          // Блок
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание
);
```

Тип указателя на эту функцию:

RDS_IBhBd

Параметры:

Block

Идентификатор блока, описание которого нужно получить.

pDescr

Указатель на структуру описания блока (RDS_BLOCKDESCRIPTION, см. стр. 113), которую функция должна заполнить.

Возвращаемое значение:

1 – структура заполнена, 0 – ошибка.

Примечания:

Функция заполняет структуру RDS_BLOCKDESCRIPTION основными параметрами блока. Для получения размеров блока в текущем масштабе следует использовать функцию rdsGetBlockDimensionsEx (стр. 221), для получения параметров окна подсистемы – функцию rdsGetEditorParameters (стр. 258).

Примеры использования функции rdsGetBlockDescription приведены в §§2.6.3, 2.7.1, 2.16.2, 4.3 и некоторых других.

См. также:

RDS_BLOCKDESCRIPTION (стр. 113), rdsGetBlockDimensionsEx (стр. 221),
rdsGetEditorParameters (стр. 258).

A.5.6.17. rdsGetBlockDimensions – получить размеры и положение блока (устаревшая)

Функция rdsGetBlockDimensions возвращает размеры и положение изображения указанного блока в точках экрана. Это устаревшая функция, в настоящее время вместо нее чаще используется rdsGetBlockDimensionsEx (стр. 221).

```
BOOL RDSCALL rdsGetBlockDimensions(  
    RDS_BHANDLE Block,           // Блок  
    RDS_PBLOCKDIMENSIONS pDim,   // Заполняемая структура размеров  
    BOOL UseZoom                 // Учитывать масштаб подсистемы  
);
```

Тип указателя на эту функцию:

RDS_BBhBrB

Параметры:

Block

Идентификатор блока, размеры которого нужно получить.

pDim

Указатель на структуру размеров и положения блока (RDS_BLOCKDIMENSIONS, см. стр. 117), которую функция должна заполнить.

UseZoom

TRUE, если нужно вернуть размеры и положение в текущем масштабе родительской подсистемы блока, FALSE, если нужно вернуть размеры и положение в масштабе 100%.

Возвращаемое значение:

TRUE – структура заполнена, FALSE – ошибка.

Примечания:

Функция заполняет структуру RDS_BLOCKDIMENSIONS размерами изображения блока и его координатами на рабочем поле подсистемы. Размеры и координаты указываются в точках экрана в текущем масштабе родительской подсистемы (при UseZoom, равном TRUE) или в масштабе 100% (при UseZoom, равном FALSE). Возможные связи размеров и положения с переменными блока в режимах расчета и моделирования в этой функции не учитываются.

Пример использования функции rdsGetBlockDimensions приведен в §2.16.2.

См. также:

RDS_BLOCKDIMENSIONS (стр. 117), rdsGetBlockDimensionsEx (стр. 221).

A.5.6.18. rdsGetBlockDimensionsEx – получить размеры и положение блока

Функция rdsGetBlockDimensionsEx возвращает размеры и положение изображения указанного блока в точках экрана.

```

BOOL RDSCALL rdsGetBlockDimensionsEx(
    RDS_BHANDLE Block,           // Блок
    RDS_PBLOCKDIMENSIONS pDim, // Заполняемая структура размеров
    DWORD Flags                 // Флаги (RDS_GBD_*)
);

```

Тип указателя на эту функцию:

RDS_BBhBrDw

Параметры:

Block

Идентификатор блока, размеры которого нужно получить.

pDim

Указатель на структуру размеров и положения блока (RDS_BLOCKDIMENSIONS, см. стр. 117), которую функция должна заполнить.

Flags

Набор битовых флагов, объединенных битовым ИЛИ:

RDS_GBD_NONE	Специальная константа, обозначающая отсутствие флагов (имеет значение 0, введена для лучшей читаемости программ).
RDS_GBD_USEVARS	В режимах моделирования и расчета возвращать размеры и положение блока с учетом их возможной связи с переменными этого блока.
RDS_GBD_USEZOOM	Возвращать размер не для масштаба 100%, а для текущего масштаба, установленного в родительской подсистеме блока.

Возвращаемое значение:

TRUE – структура заполнена, FALSE – ошибка.

Примечания:

Функция заполняет структуру RDS_BLOCKDIMENSIONS размерами изображения блока и его координатами на рабочем поле подсистемы. При установленном флаге RDS_GBD_USEZOOM размеры и координаты указываются в точках экрана в текущем масштабе родительской подсистемы, при сброшенном – в масштабе 100%. Связи размеров и положения с переменными блока в режимах расчета и моделирования учитываются только при установленном флаге RDS_GBD_USEVARS.

Пример использования функции rdsGetBlockDimensionsEx приведен в §2.13.4.

См. также:

RDS_BLOCKDIMENSIONS (стр. 117), rdsGetBlockDimensions (стр. 221).

A.5.6.19. rdsGetBlockFlags – получить флаги параметров блока

Функция rdsGetBlockFlags возвращает битовые флаги, описывающие различные параметры блока.

```

DWORD RDSCALL rdsGetBlockFlags(
    RDS_BHANDLE Block           // Блок
);

```

Тип указателя на эту функцию:

RDS_DwBh

Параметр:

Block

Идентификатор блока, флаги которого нужно получить.

Возвращаемое значение:

Набор битовых флагов параметров блока (RDS_BDF_*, см. стр. 115).

Примечания:

Функция возвращает набор битовых флагов, идентичный содержимому поля Flags структуры RDS_BLOCKDESCRIPTION (стр. 113). Если нужно получить только флаги описания параметров, работать с функцией rdsGetBlockFlags удобнее, чем с rdsGetBlockDescription.

См. также:

RDS_BLOCKDESCRIPTION (стр. 113), rdsGetBlockDescription (стр. 220),
rdsSetBlockFlags (стр. 243).

А.5.6.20. rdsGetBlockLink – найти очередную связь, соединенную с блоком

Функция rdsGetBlockLink возвращает идентификатор очередной (первой или следующей, в зависимости от параметров) связи, соединенной с заданным блоком.

```
RDS_CHANDLE RDSCALL rdsGetBlockLink(  
    RDS_BHANDLE Block,      // Блок  
    RDS_CHANDLE Conn,      // Предыдущая найденная связь  
    BOOL Inputs,           // Перебирать связи ко входам  
    BOOL Outputs,          // Перебирать связи от выходов  
    RDS_PPOINTDESCRIPTION pPoint // Заполняемое описание точки  
);
```

Тип указателя на эту функцию:

RDS_ChBhChBBPd

Параметры:

Block

Идентификатор блока, с которым должна быть соединена связь.

Conn

Идентификатор предыдущей найденной связи. В этом параметре передают NULL, если нужно найти самую первую связь, присоединенную к блоку, или результат предыдущего вызова этой функции, если нужно найти следующую связь.

Inputs

TRUE, если нужно искать среди связей, соединенных со входами блока Block, и
FALSE, если такие связи нужно пропускать.

Outputs

TRUE, если нужно искать среди связей, соединенных с выходами блока Block, и
FALSE, если такие связи нужно пропускать.

pPoint

Указатель на заполняемую функцией структуру описания точки связи (RDS_POINTDESCRIPTION, см. стр. 133), соединяющей блок Block с найденной связью. Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденной связи (RDS_CHANDLE) или NULL, если больше связей не найдено (то есть Conn была последней связью, соединенной с блоком Block).

Примечания:

Эта функция позволяет последовательно перебрать все связи, соединенные с указанным блоком. Для этого нужно сначала вызвать ее с параметром NULL для поиска самой первой связи, а затем вызывать в цикле, каждый раз передавая в параметре Conn идентификатор предыдущей найденной связи, то есть результат ее прошлого вызова. Цикл нужно завершить тогда, когда функция вернет NULL, указывая на то, что больше соединенных связей нет:

```
RDS_CHANDLE c=NULL;
for (;;)
{ // Очередная связь, соединенная с блоком Block
  c=rdsGetBlockLink(Block,c,TRUE,TRUE,NULL);
  if(c==NULL) // Больше нет связей
    break;
  // Какие-то действия со связью c
  ...
}
```

Такой перебор может использоваться, например, для визуального выделения связей, или при анализе графа схемы.

Параметры Inputs и Outputs определяют допустимые направления связей – включать ли в поиск связи, соединенные с входами и выходами блока соответственно. Хотя бы один из них должен иметь значение TRUE, иначе функция не найдет ни одной связи. Функция перебирает связи в порядке, определяемом внутренней логикой РДС, программист никак не может изменить этот порядок.

Примеры использования функции rdsGetBlockLink приведены в §2.7.4 и §2.13.4.

См. также:

RDS_POINTDESCRIPTION (стр. 133), rdsEnumConnectedBlocks (стр. 214),
rdsEnumConnectedBlocksByVar (стр. 216).

A.5.6.21. rdsGetChildBlockByName – блок подсистемы по имени

Функция rdsGetChildBlockByName возвращает идентификатор блока с заданным именем в заданной подсистеме.

```
RDS_BHANDLE RDSCALL rdsGetChildBlockByName(
    RDS_BHANDLE Parent,           // Подсистема
    LPSTR Name,                   // Имя блока
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание блока
);
```

Тип указателя на эту функцию:

RDS_BhBhSBd

Параметры:

Parent

Идентификатор подсистемы, в которой нужно найти блок.

Name

Указатель на строку с именем блока.

pDescr

Указатель на заполняемую функцией структуру описания найденного блока (RDS_BLOCKDESCRIPTION, см. стр. 113). Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденного блока или NULL, если в подсистеме нет блока с указанным именем.

Примечания:

Эта функция ищет в подсистеме Parent блок с именем Name. В отличие от функции rdsBlockByFullName (стр. 206), которая ищет блок в схеме по полному имени, она ищет блок только в конкретной подсистеме по его собственному имени, в которое, в отличие от полного, не входит вся цепочка подсистем от корневой до родительской данного блока.

Пример использования функции rdsGetChildBlockByName приведен в §2.16.2.

См. также:

rdsBlockByFullName (стр. 206).

A.5.6.22. rdsGetConnAppearance – получить внешний вид связи

Функция rdsGetConnAppearance заполняет структуру, указатель на которую передан в ее параметрах, описанием внешнего вида указанной связи или шины.

```
void RDSCALL rdsGetConnAppearance(  
    RDS_CHANDLE Conn,           // Связь или шина  
    RDS_PCONNAPPEARANCE pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

RDS_VChCa

Параметры:

Conn

Идентификатор связи или шины, описание внешнего вида которой нужно получить.

pDescr

Указатель на заполняемую функцией структуру описания внешнего вида связи/шины (RDS_CONNAPPEARANCE, см. стр. 118).

Примечания:

Эта функция записывает в структуру RDS_CONNAPPEARANCE толщину, цвет, размер стрелки и другие параметры внешнего вида связи/шины. Пример ее использования приведен в §2.13.4.

См. также:

RDS_CONNAPPEARANCE (стр. 118), rdsSetConnAppearance (стр. 247),
rdsAltConnAppearanceOp (стр. 204),
rdsGetConnStyleAppearance (стр. 227).

A.5.6.23. rdsGetConnDescription – получить описание связи

Функция rdsGetConnDescription заполняет структуру описания указанной связи или шины.

```

int RDSCALL rdsGetConnDescription(
    RDS_CHANDLE Conn,           // Связь или шина
    RDS_PCONNDESCRIPTION pDescr // Заполняемое описание
);

```

Тип указателя на эту функцию:

RDS_IChCd

Параметры:

Conn

Идентификатор связи или шины, описание которой нужно получить.

pDescr

Указатель на структуру описания связи/шины (RDS_CONNDESCRIPTION, см. стр. 119), которую функция должна заполнить.

Возвращаемое значение:

1 – структура заполнена, 0 – ошибка.

Примечания:

Функция заполняет структуру RDS_CONNDESCRIPTION основными параметрами связи или шины. Для получения параметров ее внешнего вида (толщины линии, цвета и т.п.) следует использовать функцию rdsGetConnAppearance (стр. 225), для получения размеров и положения занимаемой ей области рабочего поля подсистемы – функцию rdsGetConnDimensions (стр. 226).

Пример использования функции rdsGetConnDescription приведен в §2.13.4.

См. также:

RDS_CONNDESCRIPTION (стр. 119), rdsGetConnDimensions (стр. 226), rdsGetConnAppearance (стр. 225).

A.5.6.24. rdsGetConnDimensions – получить размеры и положение связи

Функция rdsGetConnDimensions возвращает размеры и положение описывающего прямоугольника указанной связи или шины.

```

BOOL RDSCALL rdsGetConnDimensions(
    RDS_CHANDLE Conn,           // Связь или шина
    RDS_PBLOCKDIMENSIONS pDim, // Заполняемая структура размеров
    BOOL UseZoom               // Учитывать масштаб подсистемы
);

```

Тип указателя на эту функцию:

RDS_BChBrB

Параметры:

Conn

Идентификатор связи или шины, размеры которой нужно получить.

pDim

Указатель на структуру размеров и положения (RDS_BLOCKDIMENSIONS, см. стр. 117), которую функция должна заполнить.

UseZoom

TRUE, если нужно вернуть размеры и положение в текущем масштабе подсистемы, в которой находится связь или шина, FALSE, если нужно вернуть размеры и положение в масштабе 100%.

Возвращаемое значение:

TRUE – структура заполнена, FALSE – ошибка.

Примечания:

Функция заполняет структуру RDS_BLOCKDIMENSIONS размерами и координатами описывающего прямоугольника связи или шины, то есть прямоугольника минимального размера, которым можно полностью покрыть изображение этой связи/шины. Размеры и координаты указываются в точках экрана в текущем масштабе подсистемы (при UseZoom, равном TRUE) или в масштабе 100% (при UseZoom, равном FALSE).

См. также:

RDS_BLOCKDIMENSIONS (стр. 117).

A.5.6.25. rdsGetConnStyleAppearance – получить внешний вид стиля связи/шины

Функция rdsGetConnStyleAppearance заполняет структуру, указатель на которую передан в ее параметрах, описанием внешнего вида стиля связей или шин с указанным именем.

```
BOOL RDSCALL rdsGetConnStyleAppearance(  
    LPSTR StyleName,           // Имя стиля  
    RDS_PCONNAPPEARANCE pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

RDS_BSCa

Параметры:

StyleName

Указатель на строку, содержащую имя стиля, внешний вид которого запрашивается.

pDescr

Указатель на заполняемую функцией структуру описания внешнего вида связи/шины (RDS_CONNAPPEARANCE, см. стр. 118).

Возвращаемое значение:

TRUE – структура заполнена, FALSE – ошибка.

Примечания:

Стили связей и шин хранятся вместе со схемой, в них указываются наборы параметров внешнего вида (толщина, цвет, размер стрелки и т.п.), которые пользователь может быстро назначать связям и шинам. РДС также может автоматически назначать новой созданной связи какой-либо стиль в зависимости от типа значения, передаваемого по этой связи. Функция rdsGetConnStyleAppearance заполняет структуру RDS_CONNAPPEARANCE параметрами стиля с указанным в StyleName именем. Если стиль с таким именем отсутствует в схеме, функция возвращает FALSE.

См. также:

RDS_CONNAPPEARANCE (стр. 118), rdsGetConnAppearance (стр. 225).

A.5.6.26. rdsGetFirstBlock – первый блок в подсистеме

Функция rdsGetFirstBlock возвращает идентификатор самого первого блока в указанной подсистеме.

```
RDS_BHANDLE RDSCALL rdsGetFirstBlock(  
    RDS_BHANDLE Parent,           // Подсистема  
    int Type,                     // Маска типов (RDS_BT*)  
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание блока  
);
```

Тип указателя на эту функцию:

RDS_BhBhIBd

Параметры:

Parent

Идентификатор подсистемы, в которой нужно найти самый первый блок.

Type

Маска типов блоков, среди которых ищется блок – стандартные константы типов блоков (RDS_BT*, см. стр. 113), объединенные битовым ИЛИ. Если нужно найти первый блок любого типа, в этом параметре можно передать 0.

pDescr

Указатель на заполняемую функцией структуру описания найденного блока (RDS_BLOCKDESCRIPTION, см. стр. 113). Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденного блока или NULL, если в подсистеме Parent нет ни одного блока с указанными типами.

Примечания:

Эта функция ищет в подсистеме Parent самый первый блок, принадлежащий одному из типов, указанных флагами в параметре Type. Порядок следования блоков в подсистеме (и понятие “самый первый блок” для этой функции) определяется внутренней логикой РДС и не может быть изменен программистом. Чаще всего эта функция используется вместе с функцией rdsGetNextBlock (стр. 233) для перебора всех блоков заданной подсистемы.

См. также:

rdsGetNextBlock (стр. 233), rdsEnumBlocks (стр. 212),
RDS_BLOCKDESCRIPTION (стр. 113).

A.5.6.27. rdsGetFirstConn – первая связь в подсистеме

Функция rdsGetFirstConn возвращает идентификатор самой первой связи или шины в указанной подсистеме.

```
RDS_CHANDLE RDSCALL rdsGetFirstConn(  
    RDS_BHANDLE Parent,           // Подсистема  
    int Type,                     // Маска типов (RDS_CT*)  
    RDS_PCONNDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

RDS_ChBhICd

Параметры:

Parent

Идентификатор подсистемы, в которой нужно найти самую первую связь или шину.

Type

Маска типов, среди которых ищется связь или шина – стандартные константы типов блоков (RDS_CT*, см. стр. 120), объединенные битовым ИЛИ. Если нужно найти первую связь, указывается RDS_CTCONNECTION, если шину – RDS_CTBUS, если любой из этих двух видов связей, можно указать 0.

pDescr

Указатель на заполняемую функцией структуру описания найденной связи/шины (RDS_CONNDESCRIPTION, см. стр. 119). Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденной связи/шины или NULL, если в подсистеме Parent нет таких объектов.

Примечания:

Эта функция ищет в подсистеме Parent самую первую связь или шину (в зависимости от флагов в параметре Type). Порядок следования связей и шин в подсистеме (и понятие “самая первая связь” для этой функции) определяется внутренней логикой РДС и не может быть изменен программистом. Чаще всего эта функция используется вместе с функцией rdsGetNextConn (стр. 234) для перебора всех связей или шин в заданной подсистеме.

См. также:

rdsGetNextConn (стр. 234), RDS_CONNDESCRIPTION (стр. 119).

A.5.6.28. rdsGetIOBlockByVarName – внешний вход/выход по имени переменной подсистемы

Функция rdsGetIOBlockByVarName возвращает идентификатор внутреннего блока-входа или выхода в подсистеме, соответствующего ее переменной с заданным именем.

```
RDS_BHANDLE RDSCALL rdsGetIOBlockByVarName (  
    RDS_BHANDLE Parent,           // Подсистема  
    LPSTR VarName,               // Имя переменной подсистемы  
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание блока  
);
```

Тип указателя на эту функцию:

RDS_BhBhSBd

Параметры:

Parent

Идентификатор подсистемы, внутри которой нужно найти внешний вход или выход.

VarName

Указатель на строку с именем переменной подсистемы (переменной-входу подсистемы будет соответствовать блок-вход внутри нее, переменной-выходу – блок-выход).

pDescr

Указатель на заполняемую функцией структуру описания найденного внутри подсистемы блока (RDS_BLOCKDESCRIPTION, см. стр. 113). Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденного блока или NULL, если блок не найден (например, если в подсистеме нет переменной с указанным именем).

Примечания:

Эта функция ищет в подсистеме Parent внешний вход или внешний выход, связанный с ее переменной с именем VarName. Чаще всего функция используется для прослеживания связей внутри подсистемы в процессе программного анализа структуры схемы: обнаружив соединение связи с подсистемой, из описания соединительной точки этой связи (RDS_POINTDESCRIPTION, см. стр. 133) можно узнать имя переменной, к которой подключена связь, а затем, определив с помощью rdsGetIOBlockByVarName внутренний блок подсистемы, соответствующий этой переменной, анализировать связи, идущие от него или к нему внутри подсистемы.

См. также:

rdsGetIOBlockLink (стр. 230), rdsGetBlockLink (стр. 223),
RDS_BLOCKDESCRIPTION (стр. 113).

A.5.6.29. rdsGetIOBlockLink – очередная связь снаружи подсистемы по идентификатору внутреннего блока-входа или выхода

Функция rdsGetIOBlockLink возвращает идентификатор очередной (первой или следующей) связи, присоединенной к переменной подсистемы, соответствующей переданному в параметрах функции идентификатору блока-входа или выхода.

```
RDS_CHANDLE RDSCALL rdsGetIOBlockLink(  
    RDS_BHANDLE Block,           // Блок-вход или выход  
    RDS_CHANDLE Conn,           // Предыдущая связь  
    RDS_PPOINTDESCRIPTION pPoint // Заполняемое описание точки  
);
```

Тип указателя на эту функцию:

RDS_ChBhChPd

Параметры:

Block

Идентификатор внешнего входа или выхода, для которого нужно найти внешнюю связь.

Conn

Идентификатор предыдущей найденной связи, если нужно найти следующую, или NULL, если нужно найти самую первую связь.

pPoint

Указатель на заполняемую функцией структуру описания точки соединения найденной связи с родительской подсистемой блока Block (RDS_POINTDESCRIPTION, см. стр. 133). Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденной связи или NULL, если связь не найдена.

Примечания:

Эта функция позволяет перебрать все связи, идущие к переменной подсистемы, соответствующей ее внутреннему блоку-внешнему входу или выходу Block. Чаще всего она используется для прослеживания связей наружу подсистемы в процессе программного анализа структуры схемы: обнаружив соединение какой-либо связи с блоком-входом или выходом, с помощью rdsGetIOBlockLink можно найти все связи, которые являются логическим продолжением этой связи снаружи подсистемы.

Для перебора всех наружных связей нужно сначала вызвать функцию с параметром Conn, равным NULL, для поиска самой первой связи, а затем вызывать ее в цикле, каждый раз передавая в параметре Conn идентификатор предыдущей найденной связи, то есть результат ее прошлого вызова. Цикл нужно завершить тогда, когда функция вернет NULL, указывая на то, что больше связей нет:

```
RDS_CHANDLE conn=NULL;
for (;;)
{ // Очередная связь, соединенная с переменной,
  // соответствующей блоку IOBlock
  conn=rdsGetIOBlockLink(IOBlock,conn,NULL);
  if (c==NULL) // Больше нет связей
    break;
  // Какие-то действия со связью conn (снаружи подсистемы)
  ...
}
```

См. также:

rdsGetIOBlockByVarName (стр. 229), rdsGetBlockLink (стр. 223),
RDS_POINTDESCRIPTION (стр. 133).

A.5.6.30. rdsGetLineDescription – получить описание отрезка связи

Функция rdsGetLineDescription возвращает описания отрезка связи или шины с заданным номером и точек, которые он соединяет.

```
BOOL RDSCALL rdsGetLineDescription(
    RDS_CHANDLE Conn,           // Связь
    int LineNum,                // Номер отрезка
    RDS_PLINEDescription pLine, // Описание отрезка
    RDS_PPOINTDESCRIPTION pPoint1, // Начальная точка
    RDS_PPOINTDESCRIPTION pPoint2 // Конечная точка
);
```

Тип указателя на эту функцию:

RDS_BChILdPdPd

Параметры:

Conn

Идентификатор связи или шины, которой принадлежит отрезок.

LineNum

Номер отрезка (нумерация отрезков в связях начинается с нуля).

pLine

Указатель на структуру описания отрезка связи (RDS_LINEDESCRIPTION, см. стр. 129), которую функция должна заполнить параметрами отрезка с номером LineNum. Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

pPoint1

Указатель на структуру описания точки связи (RDS_POINTDESCRIPTION, см. стр. 133), которую функция должна заполнить параметрами начальной точки отрезка с номером LineNum. Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

pPoint2

Указатель на структуру описания точки связи, которую функция должна заполнить параметрами конечной точки отрезка с номером LineNum. Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

TRUE, если отрезок с номером LineNum есть в связи Conn и размеры всех переданных структур правильные, FALSE в противном случае.

Примечания:

Эта функция позволяет заполнить структуры, указатели на которые переданы в ее параметрах, описаниями отрезка связи с заданным номером и соединяемых им точек. В данном случае, начальная и конечная точка отрезка – условные понятия, они никак не связаны с фактическим направлением связи. Начальной точкой считается точка, номер которой заносится в поле nPoint1 структуры RDS_LINEDESCRIPTION, параметры этой точки записываются в структуру по указателю pPoint1. Конечной считается точка, номер которой занесен в поле nPoint2, и ее параметры записываются по указателю pPoint1.

Общее число отрезков связи можно узнать из поля NumLines структуры RDS_CONNDESCRIPTION (стр. 119), заполняемой функцией rdsGetConnDescription (стр. 225). Пример использования функции rdsGetLineDescription приведен в §2.13.4.

См. также:

RDS_LINEDESCRIPTION (стр. 129), RDS_POINTDESCRIPTION (стр. 133),
rdsGetConnDescription (стр. 225), RDS_CONNDESCRIPTION (стр. 119).

A.5.6.31. rdsGetMouseEventId – элемент векторной картинки блока под курсором мыши

Функция rdsGetMouseEventId возвращает целый идентификатор элемента векторной картинки блока, находящегося под курсором мыши.

```
int RDSCALL rdsGetMouseEventId(  
    RDS_PMOUSEDATA pMouseEventData // Описание события  
);
```


Тип указателя на эту функцию:

RDS_IMd

Параметр:

pMouseData

Указатель на структуру RDS_MOUSEDATA (стр. 65), используемую в реакциях блоков на события, связанные с действиями пользователя мышью.

Возвращаемое значение:

Целый идентификатор элемента, находящегося под курсором мыши (координаты курсора считываются из параметра pMouseData) или 0, если под курсором нет элементов картинки.

Примечания:

Эту функция обращается к картинке блока, из модели которого она вызвана, поэтому ее можно вызывать *только из модели блока*. Если вызвать ее из модуля автоматической компиляции или, например, из функции немодального окна, РДС не сможет определить, к какому блоку относится вызов, и функция немедленно вернет нулевое значение.

Чаще всего rdsGetMouseObjectId вызывают из реакций функции модели на нажатие/отпускание кнопок мыши и перемещение ее курсора, поскольку в этих случаях в модель передается указатель на структуру RDS_MOUSEDATA, который можно без изменения передать в rdsGetMouseObjectId (впрочем, можно самостоятельно описать такую структуру, заполнить ее параметрами и вызывать функцию из других реакций модели блока). Функция возвращает целый идентификатор, присвоенный пользователем одному из элементов картинки в графическом редакторе. По умолчанию всем элементам присваиваются нулевые идентификаторы, с точки зрения rdsGetMouseObjectId попадание курсора в такие элементы не отличается от попадания в свободное место – в обоих случаях функция вернет нулевое значение. Если пользователю нужно сделать какие-то элементы картинки активными и отслеживать попадание в них курсора мыши, ему следует дать им ненулевые идентификаторы.

Пример использования функции rdsGetMouseObjectId приведен в §2.12.1.

См. также:

RDS_MOUSEDATA (стр. 65), rdsGetPictureObjectId (стр. 236).

A.5.6.32. rdsGetNextBlock – следующий блок в подсистеме

Функция rdsGetNextBlock возвращает идентификатор блока, следующего за указанным в той же подсистеме.

```
RDS_BHANDLE RDSCALL rdsGetNextBlock(  
    RDS_BHANDLE PrevBlock,          // Предыдущий блок  
    int Type,                       // Маска типов (RDS_BT*)  
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание блока  
);
```

Тип указателя на эту функцию:

RDS_BhBhIBd

Параметры:

PrevBlock

Идентификатор предыдущего блока (функция вернет следующий за ним).

Type

Маска типов блоков, среди которых ищется блок – стандартные константы типов блоков (RDS_BT*, см. стр. 113), объединенные битовым ИЛИ. Если нужно найти следующий блок любого типа, в этом параметре можно передать 0. Тип блока, переданного в параметре PrevBlock, не обязательно должен попадать в маску Type – функция в любом случае вернет блок указанного типа, следующий за PrevBlock.

pDescr

Указатель на заполняемую функцией структуру описания найденного блока (RDS_BLOCKDESCRIPTION, см. стр. 113). Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденного блока или NULL, если в подсистеме, в которой находится блок PrevBlock, больше нет блоков с указанными в Type типами.

Примечания:

Эта функция ищет в родительской подсистеме блока PrevBlock следующий блок, принадлежащий одному из типов, указанных флагами в параметре Type. Порядок следования блоков в подсистеме определяется внутренней логикой РДС и не может быть изменен программистом. Чаще всего эта функция используется вместе с функцией rdsGetFirstBlock (стр. 228) для перебора всех блоков заданной подсистемы. Например, пару этих функций можно вызвать в цикле for:

```
// Перебор всех блоков подсистемы sys
for (RDS_BHANDLE blk=rdsGetFirstBlock(sys,0,NULL);
     blk!=NULL;
     blk=rdsGetNextBlock(blk,0,NULL))
{
    // Какие-то действия с блоком blk
}
```

См. также:

rdsGetFirstBlock (стр. 228), rdsEnumBlocks (стр. 212),
RDS_BLOCKDESCRIPTION (стр. 113).

A.5.6.33. rdsGetNextConn – следующая связь в подсистеме

Функция rdsGetNextConn возвращает идентификатор связи или шины, следующей за указанной в той же подсистеме.

```
RDS_CHANDLE RDSCALL rdsGetNextConn(
    RDS_CHANDLE PrevConn,          // Предыдущая связь/шина
    int Type,                      // Маска типов (RDS_CT*)
    RDS_PCONNDESCRIPTION pDescr // Заполняемое описание
);
```

Тип указателя на эту функцию:

RDS_ChChICd

Параметры:

PrevConn

Идентификатор шины или связи (функция вернет следующую за ней).

Type

Маска типов, среди которых ищется связь или шина – стандартные константы типов связей (RDS_CT*, см. стр. 120), объединенные битовым ИЛИ. Если нужно найти связь, указывается RDS_CTCONNECTION, если шину – RDS_CTBUS, если любой из этих двух видов связей, можно указать 0. Связь или шина, переданная в параметре PrevConn, не обязательно должна сама попадать в маску Type – функция в любом случае вернет связь указанного типа, следующий за PrevConn.

pDescr

Указатель на заполняемую функцией структуру описания найденной связи/шины (RDS_CONNDESCRIPTION, см. стр. 119). Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденной связи/шины или NULL, если в подсистеме, в которой находится PrevConn, больше нет связей указанных в Type типов.

Примечания:

Эта функция ищет в родительской подсистеме связи/шины PrevConn следующую связь или шину (в зависимости от флагов в параметре Type). Порядок следования связей и шин в подсистеме определяется внутренней логикой РДС и не может быть изменен программистом. Чаще всего эта функция используется вместе с функцией rdsGetFirstConn (стр. 228) для перебора всех связей или шин в заданной подсистеме. Например, пару этих функций можно вызвать в цикле for:

```
// Перебор всех связей (без шин) подсистемы sys
for (RDS_CHANDLE conn=rdsGetFirstConn(sys,RDS_CTCONNECTION,NULL) ;
     conn!=NULL;
     conn=rdsGetNextConn(conn,RDS_CTCONNECTION,NULL) )
{
    // Какие-то действия со связью conn
}
```

См. также:

rdsGetFirstConn (стр. 228), RDS_CONNDESCRIPTION (стр. 119).

A.5.6.34. rdsGetParentBlock – родительская подсистема блока

Функция rdsGetParentBlock возвращает идентификатор родительской подсистемы указанного блока.

```
RDS_BHANDLE RDSCALL rdsGetParentBlock(
    RDS_BHANDLE Block,    // Блок
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание
);
```

Тип указателя на эту функцию:

RDS_BhBhBd

Параметры:

Block

Идентификатор блока, для которого нужно вернуть идентификатор родительской подсистемы.

pDescr

Указатель на заполняемую функцией структуру описания родительской подсистемы (RDS_BLOCKDESCRIPTION, см. стр. 113). Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор родительской подсистемы блока Block или NULL, если Block – корневая подсистема (у нее нет родительской).

Примечания:

Эта функция возвращает идентификатор родительской подсистемы, то есть подсистемы, в которой непосредственно находится блок Block. Идентификатор родительской подсистемы находится в поле Parent структуры данных блока RDS_BLOCKDATA (стр. 28) и структуры описания блока RDS_BLOCKDESCRIPTION (стр. 113), поэтому, если у вызывающей программы есть доступ к этим структурам, в функции rdsGetParentBlock нет необходимости.

См. также:

RDS_BLOCKDATA (стр. 28), RDS_BLOCKDESCRIPTION (стр. 113),
rdsGetRootSystem (стр. 238).

A.5.6.35. rdsGetPictureObjectId – элемент векторной картинки блока

Функция rdsGetPictureObjectId возвращает целый идентификатор элемента векторной картинки блока, находящегося по указанным координатам.

```
int RDSCALL rdsGetPictureObjectId(  
    int x, int y,      // Координаты точки  
    BOOL UseVars      // Учитывать значения переменных  
);
```

Тип указателя на эту функцию:

RDS_IIIB

Параметры:

x, y

Координаты (в точках экрана) проверяемой точки относительно точки привязки блока (начала координат векторной картинки) в масштабе 100%.

UseVars

TRUE – в режимах моделирования и расчета учитывать связи положения, размера, поворота и видимости элементов картинки с переменными блока. FALSE – не учитывать связи с переменными.

Возвращаемое значение:

Целый идентификатор элемента, на который в векторной картинке приходятся координаты (x,y), или 0, если по этим координатам нет элементов картинки.

Примечания:

Эту функция обращается к картинке блока, из модели которого она вызвана, поэтому ее можно вызывать *только из модели блока*. Если вызвать ее из модуля автоматической компиляции или, например, из функции немодального окна, РДС не сможет определить, к какому блоку относится вызов, и функция немедленно вернет нулевое значение.

Чаще всего `rdsGetPictureObjectId` вызывают из реакций функции модели на нажатие/отпускание кнопок мыши и перемещение ее курсора, хотя для этого есть более удобная функция `rdsGetMouseObjectId` (стр. 232). Функция возвращает целый идентификатор, присвоенный пользователем одному из элементов картинки в графическом редакторе. По умолчанию всем элементам присваиваются нулевые идентификаторы, с точки зрения `rdsGetPictureObjectId` попадание курсора в такие элементы не отличается от попадания в свободное место – в обоих случаях функция вернет нулевое значение. Если пользователю нужно сделать какие-то элементы картинки активными и отслеживать попадание в них курсора мыши, ему следует дать им ненулевые идентификаторы.

См. также:

`RDS_MOUSEDATA` (стр. 65), `rdsGetMouseObjectId` (стр. 232).

A.5.6.36. `rdsGetPointDescription` – получить описание точки связи

Функция `rdsGetPointDescription` возвращает описание точки связи или шины с заданным номером.

```
int RDSCALL rdsGetPointDescription(  
    RDS_CHANDLE Conn,           // СВЯЗЬ  
    int PointNum,               // Номер точки  
    RDS_PPOINTDESCRIPTION pPoint // Описание точки  
);
```

Тип указателя на эту функцию:

`RDS_IChIPd`

Параметры:

`Conn`

Идентификатор связи или шины, которой принадлежит точка.

`PointNum`

Номер точки (нумерация точек в связях начинается с нуля).

`pPoint`

Указатель на структуру описания точки связи (`RDS_POINTDESCRIPTION`, см. стр. 133), которую функция должна заполнить параметрами указанной точки. Если вызывающей программе не нужно это описание, в этом параметре можно передать `NULL`.

Возвращаемое значение:

1 – точка с номером `PointNum` есть в связи `Conn` и размеры переданной в `pPoint` структуры правильные, 0 – точки с таким номером нет или размер структуры (поле `servSize`) не соответствует ожиданиям функции.

Примечания:

Эта функция позволяет получить описание точки связи с заданным номером. Общее число точек связи можно узнать из поля `NumPoints` структуры `RDS_CONNDESCRIPTION` (стр. 119), заполняемой функцией `rdsGetConnDescription` (стр. 225). Пример использования функции `rdsGetPointDescription` приведен в §2.13.4.

См. также:

`RDS_POINTDESCRIPTION` (стр. 133), `rdsGetConnDescription` (стр. 225),
`RDS_CONNDESCRIPTION` (стр. 119).

A.5.6.37. rdsGetRootSystem – корневая подсистема

Функция `rdsGetRootSystem` возвращает идентификатор корневой подсистемы загруженной схемы.

```
RDS_BHANDLE RDSCALL rdsGetRootSystem(  
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

`RDS_BhBd`

Параметр:

`pDescr`

Указатель на заполняемую функцией структуру описания корневой подсистемы (`RDS_BLOCKDESCRIPTION`, см. стр. 113). Если вызывающей программе не нужно это описание, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Идентификатор корневой подсистемы схемы или `NULL`, если в данный момент никакая схема не загружена.

Примечания:

Эта функция возвращает идентификатор корневой подсистемы, то есть подсистемы, содержащей в себе на разных уровнях вложенности все остальные блоки и связи схемы. У корневой подсистемы нет родительской, это самая верхняя подсистема в иерархии.

См. также:

`RDS_BLOCKDESCRIPTION` (стр. 113), `rdsGetParentBlock` (стр. 235).

A.5.6.38. rdsIsRoot – является ли подсистема корневой

Функция `rdsIsRoot` проверяет, является ли блок, идентификатор которого передан в ее параметре, корневой подсистемой схемы.

```
BOOL RDSCALL rdsIsRoot(  
    RDS_BHANDLE Block // Проверяемый блок  
);
```

Тип указателя на эту функцию:

`RDS_BBh`

Параметр:

`Block`

Идентификатор проверяемого блока.

Возвращаемое значение:

`TRUE`, если `Block` является корневой подсистемой схемы, `FALSE` в противном случае (не корневая подсистема или вообще не подсистема).

Примечания:

Главным признаком корневой подсистемы является отсутствие у нее родительской, поэтому вместо этой функции можно вызвать `rdsGetParentBlock(Block)` (стр. 235): если она вернет `NULL`, значит, `Block` является корневой подсистемой схемы.

Если у вызывающей программы есть доступ к структуре данных блока RDS_BLOCKDATA (стр. 28) или структуре описания блока RDS_BLOCKDESCRIPTION (стр. 113), можно также проверить поле Parent в этих структурах: значение NULL в нем будет указывать на то, что структуры описывают корневую подсистему.

См. также:

rdsGetParentBlock (стр. 235), RDS_BLOCKDESCRIPTION (стр. 113),
RDS_BLOCKDATA (стр. 28).

A.5.6.39. rdsMakeUniqueBlockName – создать уникальное имя блока

Функция rdsMakeUniqueBlockName возвращает динамическую строку, содержащую имя блока, которое будет уникальным в указанной подсистеме. Имя формируется на основе переданной в параметрах строки.

```
LPSTR RDSCALL rdsMakeUniqueBlockName(  
    RDS_BHANDLE System,          // Подсистема  
    LPSTR Name                   // Желаемое имя блока  
);
```

Тип указателя на эту функцию:

RDS_SBhS

Параметры:

System

Идентификатор подсистемы, для которой нужно создать уникальное имя блока.

Name

Указатель на строку с желаемым именем блока. Если передать в этом параметре NULL, будет сформировано имя вида “Block*N*”, где *N* – некоторый номер.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую созданное уникальное (то есть не встречающееся в подсистеме System) имя блока, или NULL, если System – не подсистема, а блок какого-либо другого типа.

Примечания:

Уникальное имя блока формируется из строки Name следующим образом: если в подсистеме System нет блока с именем Name, возвращается динамическая копия строки Name. Если блок с таким именем есть, то цифры в конце строки Name изменяются так, чтобы получившееся имя стало уникальным в подсистеме (например, если в подсистеме есть блоки с именами “Mul1” – “Mul64”, а в параметре Name передана строка “Mul1”, функция вернет “Mul65”). Если строка Name не оканчивается на цифры, к ней будет добавлен номер, делающий ее уникальной (номера начинаются с единицы). Точно таким же образом РДС корректирует имена вставляемых из буфера обмена или из библиотеки блоков, если их имена совпадают с уже имеющимися в подсистеме.

Пример использования функции rdsMakeUniqueBlockName приведен в §2.16.2. Динамическая строка, созданная этой функцией, должна быть **обязательно** освобождена вызовом rdsFree (стр. 187).

См. также:

rdsFree (стр. 187), rdsGetChildBlockByName (стр. 224),
rdsDuplicateBlock (стр. 212).

A.5.6.40. rdsMoveBlock – переместить блок

Функция `rdsMoveBlock` перемещает блок в указанную точку рабочего поля подсистемы.

```
void RDSCALL rdsMoveBlock(  
    RDS_BHANDLE Block,      // Блок  
    int x,int y             // Координаты  
);
```

Тип указателя на эту функцию:

`RDS_VBhIII`

Параметры:

`Block`

Идентификатор перемещаемого блока.

`x, y`

Новые координаты точки привязки блока (для блоков с векторной картинкой – положение начала координат этой картинки, для программно рисуемых блоков и блоков, изображаемых прямоугольником с текстом – координаты левого верхнего угла изображения) на рабочем поле подсистемы в масштабе 100%. Горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля.

Примечания:

После перемещения блока в указанную точку его модель будет вызвана для реакции на событие `RDS_BFM_MOVED` (стр. 81), при этом в параметре `MoveReason` структуры описания события будет записана константа `RDS_MR_SET`.

См. также:

`rdsSetBlockRect` (стр. 246).

A.5.6.41. rdsParentIsRoot – является ли родительская подсистема корневой

Функция `rdsParentIsRoot` проверяет, располагается ли блок, идентификатор которого передан в ее параметре, в корневой подсистеме схемы.

```
BOOL RDSCALL rdsParentIsRoot(  
    RDS_BHANDLE Block // Проверяемый блок  
);
```

Тип указателя на эту функцию:

`RDS_BBh`

Параметр:

`Block`

Идентификатор проверяемого блока.

Возвращаемое значение:

`TRUE`, если `Block` расположен в корневой подсистеме схемы (то есть если его родительская подсистема является корневой), `FALSE` в противном случае.

См. также:

`rdsIsRoot` (стр. 238).

A.5.6.42. rdsRenameBlock – переименовать блок

Функция rdsRenameBlock дает указанному блоку имя, переданное в ее параметрах.

```
BOOL RDSCALL rdsRenameBlock(  
    RDS_BHANDLE Block,           // Блок  
    LPSTR NewName,              // Новое имя  
    RDS_PBLOCKDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

RDS_BBhSBd

Параметры:

Block

Идентификатор переименовываемого блока.

NewName

Указатель на строку с новым именем блока.

pDescr

Указатель на структуру описания блока RDS_BLOCKDESCRIPTION (стр. 113), которую функция должна заполнить параметрами блока после переименования. Если вызывающей программе не нужно описание блока, в этом параметре можно передать NULL.

Возвращаемое значение:

TRUE, если блок переименован, FALSE в противном случае (например, в имени NewName содержатся недопустимые символы или блок с таким именем уже есть в подсистеме).

Примечания:

Блок будет переименован только в том случае, если в его родительской подсистеме нет блока с именем NewName и в этом имени не содержатся недопустимые для имен блоков символы (см. §1.4). После переименования блока его модель будет вызвана для реакции на событие RDS_BFM_RENAME (стр. 82) в том же потоке, который вызвал rdsRenameBlock.

Пример использования функции rdsRenameBlock приведен в §2.16.2.

См. также:

RDS_BLOCKDESCRIPTION (стр. 113), RDS_BFM_RENAME (стр. 82).

A.5.6.43. rdsSelectBlock – выделить блок в редакторе

Функция rdsSelectBlock добавляет указанный блок к выделенным в подсистеме в режиме редактирования или убирает его из выделения.

```
void RDSCALL rdsSelectBlock(  
    RDS_BHANDLE Block,           // Блок  
    BOOL Select,                // Выделить/снять выделение  
    BOOL UpdateWindow           // Обновить окно  
);
```

Тип указателя на эту функцию:

RDS_VBhBB

Параметры:

Block

Идентификатор блока.

Select

Добавить блок к выделению (TRUE) или убрать оттуда (FALSE).

UpdateWindow

TRUE, если после включения или выключения выделения блока следует обновить окно подсистемы, и FALSE, если окно обновлять не нужно.

Примечания:

Включение и выключение выделения блока возможно только в открытой подсистеме в режиме редактирования. В других режимах и для блоков закрытых подсистем этот вызов игнорируется.

Эта функция чаще всего применяется для облегчения редактирования схемы: например, с ее помощью можно программно выделить какие-либо блоки, чтобы пользователь мог скопировать, переместить или удалить всю выделенную группу. Выделение блока этой функцией полностью эквивалентно выделению блока пользователем щелчком мыши с нажатой клавишей Shift.

A.5.6.44. rdsSetBlockAltNameText – вывод текста вместо имени блока

Функция rdsSetBlockAltNameText устанавливает текст, который должен выводиться вместо имени указанного блока.

```
void RDSCALL rdsSetBlockAltNameText(  
    RDS_BHANDLE Block,    // Блок  
    LPSTR Text,           // Текст  
    int Alignment          // Выравнивание (RDS_ALTBLKNAME_*)  
);
```

Тип указателя на эту функцию:

RDS_VBhSI

Параметры:

Block

Идентификатор блока.

Text

Указатель на строку, содержащую текст, который должен выводиться на рабочем поле вместо имени блока. Текст может состоять из нескольких строк – в этом случае строки отделяются друг от друга символом перевода строки “\n” (код 10). Для того, чтобы восстановить вывод имени блока, в этом параметре нужно передать NULL или указатель на пустую строку.

Alignment

Выравнивание текста по горизонтали, если он состоит из нескольких строк (см. также стр. 116):

RDS_ALTBLKNAME_LEFT	строки выровнены по левому краю
RDS_ALTBLKNAME_CENTER	строки выровнены по центру
RDS_ALTBLKNAME_RIGHT	строки выровнены по правому краю

Примечания:

Обычно рядом с изображением блока на рабочем поле подсистемы выводится его имя, если вывод имени всех блоков или этого конкретного блока явно не отключен

пользователем. Функция `rdsSetBlockAltNameText` позволяет вывести вместо имени блока произвольный текст, который, как и имя блока, будет отображаться под изображением, над изображением или в произвольном месте рабочего поля по желанию пользователя. В этом тексте может содержаться какая-либо информация о блоке, его условное название для пользователя и т.д. Этот текст никак не связан с именем блока и его изменение не приводит к переименованию блока. Для отмены вывода текста и возврата к выводу настоящего имени блока нужно вызвать эту же функцию, передав в параметре `Text` значение `NULL` или указатель на пустую строку.

См. также:

`RDS_BLOCKDESCRIPTION` (стр. 113).

A.5.6.45. `rdsSetBlockComment` – задать комментарий блока

Функция `rdsSetBlockComment` устанавливает текст комментария указанного блока.

```
void RDSCALL rdsSetBlockComment (  
    RDS_BHANDLE Block,      // Блок  
    LPSTR Comment          // Текст комментария  
);
```

Тип указателя на эту функцию:

`RDS_VBhS`

Параметры:

`Block`

Идентификатор блока.

`Text`

Указатель на строку, содержащую новый текст комментарий блока. Текст может состоять из нескольких строк – в этом случае строки отделяются друг от друга символом перевода строки “\n” (код 10).

Примечания:

Комментарий блока редко используется в моделях. Он может быть как угодно изменен пользователем в окне параметров блока (см. §1.4), и, чаще всего, модель блока не вмешивается в процесс его редактирования. Тем не менее, комментарий можно использовать для добавления к блоку каких-либо пометок, которые пользователь сможет легко найти и прочесть. Иногда, для самых простых блоков, комментарий используется для хранения текстовых параметров – в этом случае можно не делать специальный интерфейс для ввода этих параметров, но нужно обязать пользователя заполнять комментарий определенным, заранее оговоренным способом.

Пример использования функции `rdsSetBlockComment` приведен в §2.7.1.

См. также:

`RDS_BLOCKDESCRIPTION` (стр. 113), `rdsGetBlockDescription` (стр.220).

A.5.6.46. `rdsSetBlockFlags` – установить флаги параметров блока

Функция `rdsSetBlockFlags` устанавливает битовые флаги, описывающие различные параметры блока.

```
void RDSCALL rdsSetBlockFlags (  
    RDS_BHANDLE Block,      // Блок
```

```

        DWORD Flags,           // Флаги
        DWORD Mask             // Маска
    );

```

Тип указателя на эту функцию:

RDS_VBhDwDw

Параметры:

Block

Идентификатор блока.

Flags

Набор битовых флагов параметров блока (RDS_BDF_*, см. стр. 115), *кроме* флагов RDS_BDF_TEXTRECT и RDS_BDF_HASPICTURE.

Mask

Маска изменяемых битовых флагов (единичные биты в позиции тех флагов, которые нужно изменить в блоке Block согласно Flags).

Примечания:

Эта функция позволяет установить различные флаги параметров блока. Она не может использоваться для установки флагов RDS_BDF_TEXTRECT (блок отображается прямоугольником с текстом) и RDS_BDF_HASPICTURE (у блока есть векторная картинка), поскольку эти флаги – информационные, они сигнализируют о настройках, сделанных пользователем, которые не могут быть изменены программно простым включением или выключением какого-либо параметра.

Для того, чтобы изменить флаги блока, нужно в параметре Flags передать целое число, у которого в позициях, соответствующих взводимым флагам, будут единичные биты, а в позициях, соответствующих сбрасываемым – нулевые. При этом в параметре Mask должно быть передано целое число, у которого единичные биты соответствуют изменяемым (взводимым или сбрасываемым) флагам, а нулевые – флагам, остающимся неизменными. Например, для взведения флага RDS_BDF_ALLOWRESIZE в блоке Block нужно вызвать функцию следующим образом:

```

rdsSetBlockFlags(Block, RDS_BDF_ALLOWRESIZE, RDS_BDF_ALLOWRESIZE);

```

Для сброса этого же флага нужно вызвать функцию так:

```

rdsSetBlockFlags(Block, 0, RDS_BDF_ALLOWRESIZE);

```

См. также:

RDS_BLOCKDESCRIPTION (стр. 113), rdsGetBlockDescription (стр. 220), rdsGetBlockFlags (стр. 222).

A.5.6.47. rdsSetBlockLayer – задать слой блока

Функция rdsSetBlockLayer перемещает блок на слой с указанным идентификатором.

```

BOOL RDSCALL rdsSetBlockLayer(
    RDS_BHANDLE Block,    // Блок
    int LayerId           // Идентификатор слоя
);

```

Тип указателя на эту функцию:

RDS_BBhI

Параметры:

Block

Идентификатор блока.

LayerId

Идентификатор слоя, на который нужно переместить блок.

Возвращаемое значение:

TRUE, если блок перемещен на слой LayerId, FALSE в противном случае (блок уже на этом слое или такого слоя нет в подсистеме).

Примечания:

Эта функция перемещает блок Block на слой LayerId. От того, на каком слое находится блок, зависит его перекрытие другими блоками – чем дальше слой блока находится от переднего плана в текущей выбранной конфигурации слоев, тем большее число блоков сможет перекрыть данный блок.

Следует учитывать, что вызов rdsSetBlockLayer не приводит к немедленной перерисовке окна подсистемы. Обычно окно автоматически обновляется после завершения функции модели вызванного блока, поэтому если rdsSetBlockLayer вызвана не из функции модели, окно подсистемы следует обновить вручную вызовом rdsRefreshBlockWindows (стр. 263).

См. также:

RDS_BLOCKDESCRIPTION (стр. 113), rdsRefreshBlockWindows (стр. 263),
rdsSetLayerPosition (стр. 276).

A.5.6.48. rdsSetBlockModel – подключить к блоку модель

Функция rdsSetBlockModel подключает к указанному блоку новую функцию модели.

```
BOOL RDSCALL rdsSetBlockModel(  
    RDS_BHANDLE Block,      // Блок  
    LPSTR DllFile,          // Имя файла DLL  
    LPSTR FunctionName      // Имя экспортированной функции  
);
```

Тип указателя на эту функцию:

RDS_BBhSS

Параметры:

Block

Идентификатор блока.

DllFile

Указатель на строку, содержащую путь к DLL с функцией модели. В строке пути могут использоваться стандартные символические обозначения РДС (см. стр. 189). Если нужно отключить от блока текущую модель не подключая новой, в этом параметре можно передать NULL.

FunctionName

Указатель на строку, содержащую имя экспортированной из DLL функции модели (в том виде, в котором ее можно обработать функцией Windows API GetProcAddress).

Возвращаемое значение:

TRUE, если операция выполнена успешно, FALSE в противном случае (например, если нет файла DLL или указанная функция не экспортирована).

Примечания:

Эта функция сначала отключает от блока старую модель, а затем подключает новую, если она указана. Если функция вызвана из режима расчета или для смены модели вызвавшего блока, ее выполнение будет отложено до завершения модели блока или, в случае режима расчета, до завершения очередного такта. При таком отложенном выполнении функция вернет FALSE, если смена модели невозможна: например, если на момент вызова файл, указанный в параметре `DllFile`, отсутствует, функция немедленно вернет FALSE и смена модели блока запланирована не будет.

См. также:

`RDS_BLOCKDESCRIPTION` (стр. 113).

A.5.6.49. `rdsSetBlockRect` – задать прямоугольник блока

Функция `rdsSetBlockRect` задает для блока новый описывающий прямоугольник (это приводит к изменению его положения и размеров).

```
void RDSCALL rdsSetBlockRect(  
    RDS_BHANDLE Block,    // Блок  
    int left, int top,     // Левый верхний угол  
    int width, int height // Размеры  
);
```

Тип указателя на эту функцию:

`RDS_VBhIIII`

Параметры:

`Block`

Идентификатор блока, описывающий прямоугольник которого необходимо изменить.

`left, top`

Новая горизонтальная (`left`) и вертикальная (`top`) координаты левого верхнего угла изображения блока на рабочем поле в масштабе 100% (горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля).

`width, height`

Новые ширина (`width`) и высота (`height`) изображения блока в точках экрана в масштабе 100%.

Примечания:

В результате вызова этой функции изображение блока будет занимать прямоугольник с указанными в параметрах координатами, независимо от того, задается оно векторной картинкой, прямоугольником с текстом, или модель блока рисует его программно. Для этого РДС изменит размеры блока и переместит его, если это потребуется.

После перемещения блока его модель будет вызвана для реакции на событие `RDS_BFM_MOVED` (стр. 81), при этом в параметре `MoveReason` структуры описания события будет записана константа `RDS_MR_SET`. Если внешний вид блока рисуется программно, перед перемещением его модель будет вызвана для реакции на событие `RDS_BFM_RESIZE` (стр. 82).

См. также:

rdsMoveBlock (стр. 240), RDS_BFM_MOVED (стр. 81), RDS_BFM_RESIZE (стр. 82),
rdsGetBlockDimensionsEx (стр. 221).

A.5.6.50. rdsSetBlockSetupFuncName – задать имя функции настройки блока

Функция rdsSetBlockSetupFuncName устанавливает текст пункта меню, вызывающего функцию настройки блока, а также позволяет выключить функцию настройки.

```
void RDSCALL rdsSetBlockSetupFuncName (  
    RDS_BHANDLE Block,      // Блок  
    LPSTR MenuName          // Текст пункта меню  
);
```

Тип указателя на эту функцию:

RDS_VBhS

Параметры:

Block

Идентификатор блока.

Text

Указатель на строку, содержащую текст пункта меню, который будет вызывать функцию настройки блока. Передача NULL в этом параметре запретит вызов функции настройки. Передача указателя на пустую строку даст пункту меню название по умолчанию (“Настройка”).

Примечания:

При вызове пользователем функции настройки модель блока обычно открывает окно для ввода параметров этого блока (см. §2.7). Выбор пользователем пункта контекстного меню, соответствующего функции настройки (или двойной щелчок, если в параметрах блока задан вызов функции настройки по нему) генерирует событие RDS_BFM_SETUP (см. стр. 71). Функция rdsSetBlockSetupFuncName позволяет программно задать имя этого пункта контекстного меню или вообще отключить его.

Наличием функции настройки также управляет флаг параметров блока RDS_BDF_SETUPFUNC, который можно получить и установить функциями rdsGetBlockFlags (стр. 222) и rdsSetBlockFlags (стр. 243) соответственно.

См. также:

RDS_BLOCKDESCRIPTION (стр. 113), rdsGetBlockDescription (стр.220),
rdsGetBlockFlags (стр. 222), rdsSetBlockFlags (стр. 243),
RDS_BFM_SETUP (стр. 71).

A.5.6.51. rdsSetConnAppearance – задать внешний вид связи

Функция rdsSetConnAppearance устанавливает внешний вид указанной связи или шины согласно структуре описания, переданной в ее параметрах.

```
void RDSCALL rdsSetConnAppearance (  
    RDS_CHANDLE Conn,          // СВЯЗЬ ИЛИ ШИНА  
    RDS_PCONNAPPEARANCE pDescr // Описание  
);
```

Тип указателя на эту функцию:

RDS_VChCa

Параметры:

Conn

Идентификатор связи или шины, описание внешнего вида которой нужно установить.

pDescr

Указатель на структуру описания внешнего вида связи или шины (RDS_CONNAPPEARANCE, см. стр. 118), из которой будут считаны устанавливаемые параметры.

Примечания:

Эта функция устанавливает толщину, цвет, размер стрелки и другие параметры внешнего вида связи/шины.

См. также:

RDS_CONNAPPEARANCE (стр. 118), rdsGetConnAppearance (стр. 225), rdsAltConnAppearanceOp (стр. 204).

A.5.6.52. rdsSetConnLayer – задать слой связи

Функция rdsSetConnLayer перемещает связь или шину на слой с указанным идентификатором.

```
BOOL RDSCALL rdsSetConnLayer(  
    RDS_CHANDLE Conn,      // Связь  
    int LayerId             // Идентификатор слоя  
);
```

Тип указателя на эту функцию:

RDS_BChI

Параметры:

Conn

Идентификатор связи или шины.

LayerId

Идентификатор слоя, на который нужно переместить связь или шину.

Возвращаемое значение:

TRUE, если связь или шина перемещена на слой LayerId, FALSE в противном случае (связь уже на этом слое или такого слоя нет в подсистеме).

Примечания:

Эта функция перемещает связь или шину с идентификатором Conn на слой LayerId. От того, на каком слое находится связь, зависит ее перекрытие другими связями и блоками – чем дальше слой находится от переднего плана в текущей выбранной конфигурации слоев, тем большее число объектов сможет перекрыть данную связь. В пределах одного слоя связи всегда рисуются поверх блоков.

Следует учитывать, что вызов rdsSetConnLayer не приводит к немедленной перерисовке окна подсистемы. Обычно окно автоматически обновляется после завершения функции модели вызванного блока, поэтому если rdsSetConnLayer вызвана не из функции модели, окно подсистемы следует обновить вручную, вызвав rdsRefreshBlockWindows (стр. 263).

См. также:

RDS_CONNDESCRIPTION (стр. 119), rdsRefreshBlockWindows (стр. 263),
rdsSetLayerPosition (стр. 276).

A.5.6.53. rdsSetHintText – текст всплывающей подсказки

Функция rdsSetHintText передает в РДС текст всплывающей подсказки при вызове модели блока для запроса параметров этой подсказки.

```
void RDSCALL rdsSetHintText(  
    LPSTR HintString      // Текст подсказки  
);
```

Тип указателя на эту функцию:

RDS_VS

Параметр:

HintString

Указатель на строку с текстом выводимой подсказки. Текст может состоять из нескольких строк, разделенных символом перевода строки “\n” (код 10). Передача в этом параметре значения NULL подавит вывод всплывающей подсказки.

Примечания:

Эту функцию следует вызывать из функции модели блока в момент реакции на событие RDS_BFM_POPUPHINT (см. стр. 70). Вызов ее из любых других реакций будет игнорироваться. С помощью этой функции модель должна сообщить, какой именно текст нужно вывести во всплывающей подсказке в данный момент. В РДС не предусмотрены средства задания постоянной всплывающей подсказки, перед каждым ее выводом модель блока будет вызываться для реакции на событие RDS_BFM_POPUPHINT.

Пример использования функции rdsSetHintText приведен в §2.11.

См. также:

RDS_BFM_POPUPHINT (стр. 70).

A.5.6.54. rdsSetPointPosition – задать координаты точки связи

Функция rdsSetPointPosition перемещает указанную точку связи или шины в указанную позицию рабочего поля подсистемы.

```
BOOL RDSCALL rdsSetPointPosition(  
    RDS_CHANDLE Conn,      // СВЯЗЬ  
    int PointNum,          // Номер точки  
    int x, int y,          // Новые координаты  
    DWORD Flags            // Флаги (RDS_SPP_*)  
);
```

Тип указателя на эту функцию:

RDS_BChIIIDw

Параметры:

Conn

Идентификатор связи или шины, точку которой нужно переместить в новое положение.

PointNum

Порядковый номер перемещаемой точки в связи или шине. Обращение к точкам всегда идет по их номеру, общее число точек можно получить через поле NumPoints структуры RDS_CONNDESCRIPTION (стр. 119).

x, y

Новые координаты точки на рабочем поле подсистемы в масштабе 100%.

Flags

Набор битовых флагов, управляющих работой функции, объединенных битовым ИЛИ:

RDS_SPP_RELATIVE Только для точек связи с блоком (RDS_PTBLOCK, см. стр. 119): в x и y переданы не абсолютные координаты на рабочем поле, а смещения относительно точки привязки блока. Для всех остальных типов точек этот флаг игнорируется, координаты всегда абсолютные.

RDS_SPP_REFRESH Обновить окно подсистемы после перемещения точки.

Возвращаемое значение:

TRUE, если точка перемещена, FALSE в противном случае (нет точки с таким номером).

Примечания:

Эта функция перемещает отдельную точку связи или шины. С ее помощью нельзя добавлять или удалять точки, она может только перемещать уже существующие. Для более сложного редактирования связей и шин используется вспомогательный объект РДС (стр. 408).

См. также:

RDS_CONNDESCRIPTION (стр. 119), RDS_POINTDESCRIPTION (стр. 133), вспомогательный объект для редактирования связей (стр. 408).

А.5.7. Сохранение и загрузка состояния схемы

Описываются функции управления сохраненными состояниями блоков схемы (см. §2.14.3), а также функция сброса состояния отдельной подсистемы.

А.5.7.1. rdsDeleteSystemState – удалить сохраненное состояние

Функция rdsDeleteSystemState удаляет ранее сохраненное состояние схемы.

```
void RDSCALL rdsDeleteSystemState(  
    int StateNum           // Номер состояния  
);
```

Тип указателя на эту функцию:

RDS_VI

Параметр:

StateNum

Номер удаляемого состояния схемы, присвоенный ему при сохранении.

Примечания:

Эта функция удаляет из памяти состояние схемы, ранее сохраненное функцией rdsSaveSystemState (стр. 252). Сохраненные состояния – глобальные объекты, они не привязываются к вызвавшей функцию rdsSaveSystemState блоку и не удаляются

автоматически при его удалении (тем не менее, они автоматически удаляются при загрузке новой схемы). Когда сохраненное состояние станет не нужным, его следует удалить вручную вызовом `rdsDeleteSystemState`, чтобы не тратить память зря.

Пример использования функции приведен в §2.14.3.

См. также:

`rdsSaveSystemState` (стр. 252).

A.5.7.2. `rdsLoadSystemState` – загрузить сохраненное состояние

Функция `rdsLoadSystemState` загружает ранее сохраненное состояние схемы.

```
BOOL RDSCALL rdsLoadSystemState(  
    int StateNum,           // Номер состояния  
    RDS_BBhpB Callback     // Функция проверки  
);
```

Тип указателя на эту функцию:

`RDS_BICb2`

Параметры:

`StateNum`

Номер загружаемого состояния схемы, присвоенный ему при сохранении.

`Callback`

Указатель на пользовательскую функцию, которую нужно вызвать для каждого блока, состояние которого будет загружаться, или `NULL`, если такой функции нет. С помощью этой пользовательской функции можно, при необходимости, запретить загрузку состояния отдельных блоков. Она должна иметь следующий вид:

```
BOOL RDSCALL имя_функции(RDS_BHANDLE block, BOOL *pgo);
```

В параметре `block` пользовательской функции передается идентификатор блока, состояние которого должно сейчас загрузиться. Если состояние этого блока нужно загрузить, функция должна вернуть `TRUE`, если же его нужно пропустить и перейти к загрузке состояния следующего блока, функция должна вернуть `FALSE`. В параметре `pgo` передается указатель на логическую переменную, в которую функция может записать `FALSE` (исходно в этой переменной находится `TRUE`), чтобы после обработки блока `block` прекратить загрузку состояний оставшихся блоков.

Возвращаемое значение:

`TRUE`, если состояние системы загружено, `FALSE` в противном случае (нет состояния с номером `StateNum`).

Примечания:

Эта функция загружает состояние схемы, ранее сохраненное функцией `rdsSaveSystemState` (стр. 252). При необходимости, в параметре `Callback` можно передать указатель на функцию обратного вызова, с помощью которой можно разрешать или запрещать загрузку состояния каждого из блоков, а также прервать загрузку состояния в нужный момент. Если в `Callback` передано значение `NULL`, будут загружены все состояния сохраненных блоков.

При загрузке состояния всем статическим переменным блока присваиваются значения, которые они имели на момент сохранения, после чего модель блока вызывается для реакции на событие `RDS_BFM_LOADSTATE` (стр. 39), в которой она должна загрузить из памяти все данные, записанные туда при реакции на событие `RDS_BFM_SAVESTATE` (стр. 45).

Если после сохранения состояния один из блоков схемы был удален или изменил структуру статических переменных (что делает невозможным загрузку состояния этого блока, поскольку новая структура не соответствует сохраненной), вызов `rdsLoadSystemState` не приведет к ошибкам – состояния проблемных блоков просто не будут загружены.

Пример использования функции `rdsLoadSystemState` приведен в §2.14.3.

См. также:

`rdsSaveSystemState` (стр. 252), `RDS_BFM_LOADSTATE` (стр. 39),
`RDS_BFM_SAVESTATE` (стр. 45), `rdsReadBlockData` (стр. 277).

A.5.7.3. `rdsResetSystemState` – сбросить состояние блока или подсистемы

Функция `rdsResetSystemState` возвращает указанный блок или все блоки указанной подсистемы в начальное (до первого запуска расчета) состояние.

```
void RDSCALL rdsResetSystemState(  
    RDS_BHANDLE Block      // Сбрасываемый блок  
);
```

Тип указателя на эту функцию:

`RDS_VBh`

Параметр:

`Block`

Идентификатор сбрасываемого блока или подсистемы.

Примечания:

Эта функция сбрасывает блок `Block` в исходное состояние: всем статическим переменным блока присваиваются значения по умолчанию, после чего модель блока вызывается для реакции на событие `RDS_BFM_RESETCALC` (стр. 44). Если `Block` – не простой блок, а подсистема, будут также сброшены и все внутренние блоки этой подсистемы на всех уровнях иерархии.

В режиме расчета сброс состояния блоков происходит не немедленно, а только по окончании очередного такта расчета.

Пример использования функции `rdsResetSystemState` приведен в §2.14.2.

См. также:

`RDS_BFM_RESETCALC` (стр. 44).

A.5.7.4. `rdsSaveSystemState` – сохранить состояние блока/подсистемы

Функция `rdsSaveSystemState` сохраняет в памяти состояние указанного блока или подсистемы. Позже это состояние может быть загружено функцией `rdsLoadSystemState` (стр. 251).

```
int RDSCALL rdsSaveSystemState(  
    RDS_BHANDLE Block,      // Блок или подсистема  
    int StateNum,          // Номер состояния или -1  
    BOOL Recurse,          // Включать вложенные блоки  
    RDS_BBhpB CallBack     // Функция проверки  
);
```

Тип указателя на эту функцию:

`RDS_IBhIBCb2`

Параметры:

Block

Идентификатор блока (подсистемы), состояние которого нужно сохранить.

StateNum

Номер ранее сохраненного состояния схемы, которое нужно заменить на сохраняемое, или -1, если нужно создать в памяти новое сохраненное состояние.

Recurse

Если Block – подсистема, в этом параметре передается TRUE, если нужно сохранять состояние не только этой подсистемы и всех блоков, непосредственно находящихся в ней, но всех блоков во всех ее вложенных подсистемах. FALSE, если нужно вызывать функцию только для блоков, непосредственно находящихся в подсистеме Block. Если Block – не подсистема, этот параметр игнорируется.

CallBack

Указатель на пользовательскую функцию, которую нужно вызвать для каждого блока, состояние которого будет сохраняться, или NULL, если такой функции нет. С помощью этой пользовательской функции можно, при необходимости, запретить запись состояния отдельных блоков. Она должна иметь следующий вид:

```
BOOL RDSCALL имя_функции(RDS_BHANDLE block, BOOL *pgo);
```

В параметре block пользовательской функции передается идентификатор блока, состояние которого должно сейчас сохраниться. Если состояние этого блока нужно сохранять, функция должна вернуть TRUE, если же его нужно пропустить и перейти к записи состояния следующего блока, функция должна вернуть FALSE. В параметре pgo передается указатель на логическую переменную, в которую функция может записать FALSE (исходно в этой переменной находится TRUE), чтобы после обработки блока block прекратить запись состояний оставшихся блоков.

Возвращаемое значение:

Номер сохраненного состояния, который может использоваться для его загрузки функцией `rdsLoadSystemState` и удаления функцией `rdsDeleteSystemState` (стр. 250).

Примечания:

Эта функция сохраняет в памяти текущее состояние отдельного блока или подсистемы. Если в параметре Block передан идентификатор простого блока, внешнего входа/выхода или ввода шины, сохраняется состояние только этого блока. Если в параметре Block передан идентификатор подсистемы, сохраняется состояние этой подсистемы и всех непосредственно находящихся в ней блоков. Если параметр Recurse имеет значение TRUE, будут также сохранены состояния всех блоков, находящихся в подсистеме Block на более низких уровнях иерархии (внутренние блоки внутренних подсистем). При необходимости, в параметре CallBack можно передать указатель на функцию обратного вызова, с помощью которой можно разрешать или запрещать сохранение состояния каждого из блоков, а также прервать сохранение состояния в нужный момент. Если в CallBack передано значение NULL, будут записаны все состояния блоков, удовлетворяющих параметрам функции.

При сохранении состояния блока в память записываются значения всех его статических переменных, после чего модель блока вызывается для реакции на событие `RDS_BFM_SAVESTATE` (стр. 45), в которой она должна записать в память все остальные данные, относящиеся к текущему состоянию блока. Сохраненному состоянию присваивается уникальный целый номер, который можно использовать для его последующей загрузки или удаления. Можно не создавать новое сохраненное состояние, а заменить им уже

существующее, передав в параметре StateNum номер этого существующего состояния (функция в этом случае вернет число, равное StateNum).

Пример использования функции rdsSaveSystemState приведен в §2.14.3.

См. также:

rdsLoadSystemState (стр. 251), rdsDeleteSystemState (стр. 250),
RDS_BFM_LOADSTATE (стр. 39), RDS_BFM_SAVESTATE (стр. 45),
rdsWriteBlockData (стр. 280).

A.5.8. Работа с окнами подсистем

Описываются функции, открывающие, закрывающие и перемещающие окна подсистем, а также задающие и считывающие их параметры.

A.5.8.1. rdsCheckRectVisibility – проверить видимость прямоугольника

Функция rdsCheckRectVisibility проверяет, видна ли полностью указанная прямоугольная область в окне подсистемы при текущем положении полос прокрутки.

```
BOOL RDCALL rdsCheckRectVisibility(  
    RDS_BHANDLE System,           // Подсистема  
    int Left,int Top,             // Левый верхний угол  
    int Width,int Height          // Размеры прямоугольника  
);
```

Тип указателя на эту функцию:

RDS_BBhIIII

Параметры:

System

Идентификатор подсистемы, окно которой проверяется.

Left, Top

Горизонтальная (Left) и вертикальная (Top) координаты верхнего левого угла прямоугольника на рабочем поле подсистемы в масштабе 100% (горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля).

Width, Height

Ширина (Width) и высота (Height) прямоугольника точках экрана в масштабе 100%.

Возвращаемое значение:

TRUE – указанный прямоугольник полностью видим в окне подсистемы, FALSE – прямоугольник не виден или виден частично.

Примечания:

Эта функция возвращает TRUE, если окно подсистемы System открыто и прямоугольник с указанными в параметрах координатами целиком находится на видимой в данный момент части рабочего поля подсистемы. Если окно подсистемы перекрыто другими окнами, прямоугольник все равно будет считаться видимым, если он попадет в отображаемую в окне часть рабочей области.

См. также:

rdsScrollWindowToRect (стр. 264).

A.5.8.2. rdsCheckSystemWindow – открыто ли окно подсистемы

Функция rdsCheckSystemWindow проверяет, открыто ли в данный момент окно указанной подсистемы.

```
BOOL RDSCALL rdsCheckSystemWindow(  
    RDS_BHANDLE System          // Подсистема  
);
```

Тип указателя на эту функцию:

RDS_BBh

Параметр:

System

Идентификатор подсистемы, окно которой проверяется.

Возвращаемое значение:

TRUE – окно подсистемы System открыто, FALSE – закрыто.

См. также:

rdsOpenSystemWindow (стр. 261), rdsCloseSystemWindow (стр. 255).

A.5.8.3. rdsCloseSystemWindow – закрыть окно подсистемы

Функция rdsCloseSystemWindow закрывает окно указанной подсистемы.

```
void RDSCALL rdsCloseSystemWindow(  
    RDS_BHANDLE System          // Подсистема  
);
```

Тип указателя на эту функцию:

RDS_VBh

Параметр:

System

Идентификатор подсистемы, окно которой нужно закрыть. Если System окажется не подсистемой, а блоком другого типа, будет закрыто окно родительской подсистемы этого блока.

См. также:

rdsOpenSystemWindow (стр. 261), rdsCheckSystemWindow (стр. 255).

A.5.8.4. rdsEnableWindowRefresh – разрешение/запрет обновления окон

Функция rdsEnableWindowRefresh разрешает или запрещает обновление окна указанной подсистемы или немодальных окон указанного блока в режимах моделирования и расчета.

```
void RDSCALL rdsEnableWindowRefresh(  
    RDS_BHANDLE Block,          // Блок или подсистема  
    BOOL Enable,                // Разрешить/запретить обновление  
    BOOL Recurse                // Включать вложенные блоки  
);
```

Тип указателя на эту функцию:

RDS_VBhBB

Параметры:

Block

Идентификатор блока, обновление немодальных окон которого нужно разрешить или запретить. Если Block – подсистема, в ее немодальные окна входит и само окно подсистемы (за редким исключением, это – единственное окно, которым владеет подсистема).

Enable

TRUE, если обновление окон нужно разрешить, или FALSE, если его необходимо временно запретить.

Recurse

Если Block – подсистема, значение TRUE в этом параметре также разрешит или запретит (в зависимости от значения Enable) обновление немодальных окон всех блоков этой подсистемы на всех уровнях иерархии (в частности, всех окон вложенных подсистем).

Примечания:

Эта функция обычно используется для временного запрета обновления окон в режиме расчета, если какие-либо вычисления занимают несколько тактов и в середине этих вычислений, пока их результаты не готовы, отображать что-либо в окнах нежелательно. В режиме редактирования обновление окон всегда разрешено, и вызов этой функции игнорируется.

При запрещении обновления окон в параметрах блока Block взводится флаг RDS_NOWINREFRESH (см. описание структуры RDS_BLOCKDATA на стр. 28). При этом обновление окна подсистемы (если Block – подсистема) запрещается автоматически, а для всех остальных немодальных окон, принадлежащих блоку Block, если таковые имеются, анализировать состояние этого флага необходимо вручную. Например, функция немодального окна, открытого средствами Windows API, не должно перерисовывать содержимое окна при получении сообщения WM_PAINT или при вызове сервисной функции rdsRefreshBlockWindows (стр. 263), если флаг RDS_NOWINREFRESH взведен.

Если при запрещенном обновлении окон (взведенном флаге RDS_NOWINREFRESH) будет затребовано обновление окна, в флагах структуры RDS_BLOCKDATA блока, которому принадлежит окно, должен быть взведен флаг RDS_WINREFRESHWAITING (для окон подсистем и в результате вызова rdsRefreshBlockWindows это делается автоматически, во всех остальных случаях этим должна заниматься модель блока). Когда обновление окон данного блока будет снова разрешено (флаг RDS_NOWINREFRESH сброшен), если флаг RDS_WINREFRESHWAITING окажется взведенным, для блока будет автоматически вызвана функция rdsRefreshBlockWindows.

Таким образом, состояние флагов RDS_NOWINREFRESH и RDS_WINREFRESHWAITING описывает ситуацию с обновлением окон блока следующим образом:

<i>RDS_NOWINREFRESH</i>	<i>RDS_WINREFRESHWAITING</i>	<i>Состояние блока</i>
0	0	Обновление окон разрешено.
0	1	Обновление окон разрешено, но, пока оно было запрещено, произошли какие-то изменения, и их нужно отобразить.
1	0	Обновление окон запрещено.

<i>RDS_NOWINREFRESH</i>	<i>RDS_WINREFRESHWAITING</i>	<i>Состояние блока</i>
1	1	Обновление окон запрещено, требуется отобразить изменения при первой возможности.

Если вызовом `rdsEnableWindowRefresh` разрешается или запрещается обновление окон подсистемы, модели блока не нужно следить за флагами `RDS_NOWINREFRESH` и `RDS_WINREFRESHWAITING` – все действия по прекращению и возобновлению обновления этих окон будут выполнены автоматически.

Следует учитывать, что у функции `rdsEnableWindowRefresh` нет внутреннего счетчика вызовов, поэтому если вызвать ее для запрещения обновления (с параметром `Enable`, равным `FALSE`) несколько раз, все вызовы кроме первого будут проигнорированы, и обновление будет снова разрешено при первом же вызове функции с параметром `Enable=TRUE`.

См. также:

`rdsRefreshBlockWindows` (стр. 263), `RDS_BLOCKDATA` (стр. 28),
`rdscrtlEnableWinRefresh` (стр. 625).

A.5.8.5. `rdsGetEditorFont` – получить параметры шрифта окна подсистемы

Функция `rdsGetEditorFont` возвращает параметры шрифтов, используемых в окне подсистемы для вывода имен блоков и переменных.

```

BOOL RDSCALL rdsGetEditorFont (
    RDS_BHANDLE System,      // Подсистема
    int FontType,           // Тип шрифта (RDS_GEF_*)
    LOGFONT *pFont,        // Заполняемая структура
    DWORD lfSize,          // Размер структуры *pFont
    int *pPixHeight         // Высота шрифта в точках
);

```

Тип указателя на эту функцию:

`RDS_BBhIpLfDwpI`

Параметры:

`System`

Идентификатор подсистемы, из которой нужно получить шрифты (ее окно не обязательно должно быть открытым).

`FontType`

Вид шрифта, параметры которого нужно получить. В этом параметре передается одна из следующих констант:

`RDS_GEF_BLOCKNAME` Шрифт имен блоков.
`RDS_GEF_VARNAME` Шрифт имен переменных.

`pFont`

Указатель на стандартную структуру описания шрифта Windows `LOGFONT`, которую функция должна заполнить. Если вызывающей программе не нужно описание шрифта, в `pFont` можно передать `NULL`.

`lfSize`

Размер структуры, на которую указывает `pFont` (функция использует этот параметр для проверки правильности параметров, поскольку в структуре `LOGFONT` не

предусмотрено специального поля для размера). Если в `pFont` передано значение `NULL`, этот параметр игнорируется.

`pPixHeight`

Указатель на целую переменную, в которую функция должна записать высоту запрашиваемого шрифта в точках экрана в масштабе 100%. Если вызывающей программе не нужна эта высота, в `pPixHeight` можно передать `NULL`.

Возвращаемое значение:

`TRUE` – параметры шрифта получены, `FALSE` – в параметрах `FontType` или `lfSize` переданы недопустимые значения.

Примечания:

Эта функция чаще всего используется в тех случаях, когда желательно вывести какую-либо надпись тем же шрифтом, который используется в окне подсистемы для отображения имен блоков или переменных. Ее также можно использовать для определения высоты шрифта, чтобы, например, программно перемещая блок по рабочему полю, оставить над ним или под ним достаточно места для вывода его имени.

См. также:

`rdsGetEditorParameters` (стр. 258).

A.5.8.6. `rdsGetEditorParameters` – получить описание окна подсистемы

Функция `rdsGetEditorParameters` заполняет структуру описания окна указанной подсистемы.

```
BOOL RDCALL rdsGetEditorParameters(  
    RDS_BHANDLE System,           // Подсистема  
    RDS_EDITORPARAMETERS pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

`RDS_BBhEd`

Параметры:

`System`

Идентификатор подсистемы, описание окна которой нужно получить. Само окно при этом может быть закрыто, его описание хранится в параметрах подсистемы независимо от его наличия на экране.

`pDescr`

Указатель на структуру описания окна (`RDS_EDITORPARAMETERS`, см. стр. 122), которую функция должна заполнить.

Возвращаемое значение:

`TRUE` – структура заполнена, `FALSE` – ошибка (`System` – не подсистема, или размер заполняемой структуры не соответствует ожиданиям функции).

Примечания:

Примеры использования этой функции описаны в §2.10.4 и §2.12.5.

См. также:

`RDS_EDITORPARAMETERS` (стр. 122), `rdsGetEditorFont` (стр. 257),
`rdsGetEditorWindowFlags` (стр. 259).

A.5.8.7. rdsGetEditorToolBars – состояние панелей окна подсистемы (устаревшая)

Функция `rdsGetEditorToolBars` записывает видимость отдельных панелей окна подсистемы в структуру `RDS_EDITORTOOLBARS` (стр. 126). Это устаревшая функция, сейчас вместо нее используется `rdsGetEditorWindowFlags` (стр. 259).

```
void RDSCALL rdsGetEditorToolBars(  
    RDS_BHANDLE System,          // Подсистема  
    RDS_PEDITORTOOLBARS pBars   // Структура описания  
);
```

Тип указателя на эту функцию:

`RDS_VBhEt`

Параметры:

`System`

Идентификатор подсистемы, состояние панелей окна которой нужно получить (окно подсистемы не обязательно должно быть открыто).

`pBars`

Указатель на структуру описания панелей, которую функция должна заполнить.

Примечания:

Эта функция устарела и позволяет получить состояние только некоторых панелей окна.

См. также:

`RDS_EDITORTOOLBARS` (стр. 126), `rdsGetEditorWindowFlags` (стр. 259).

A.5.8.8. rdsGetEditorWindowFlags – получить флаги панелей окна подсистемы

Функция `rdsGetEditorWindowFlags` возвращает флаги, описывающие состояние панелей окна подсистемы.

```
DWORD RDSCALL rdsGetEditorWindowFlags(  
    RDS_BHANDLE System          // Подсистема  
);
```

Тип указателя на эту функцию:

`RDS_DwBh`

Параметр:

`System`

Идентификатор подсистемы, состояние панелей окна которой нужно получить (окно подсистемы не обязательно должно быть открыто).

Возвращаемое значение:

Набор битовых флагов, описывающих состояние панелей (флаг взведен – панель видима, флаг сброшен – панель скрыта):

<code>RDS_EWF_CALCSTOOLBAR</code>	Панель расчета (с кнопками переключения режимов).
<code>RDS_EWF_DISPLAYTOOLBAR</code>	Панель элементов (с кнопками управления сеткой и отображением имен блоков и переменных).
<code>RDS_EWF_LAYERSTOOLBAR</code>	Панель слоев (со списками слоев и их конфигураций).

RDS_EWF_PRINTTOOLBAR	Панель печати (с кнопками вызова окна печати и управления зоной печати).
RDS_EWF_STATUSBAR	Строка состояния.
RDS_EWF_ZOOMTOOLBAR	Панель масштаба (со списком масштабов и кнопками увеличения и перетаскивания).

См. также:

rdsSetEditorWindowFlags (стр. 265), rdsGetEditorParameters (стр. 258).

A.5.8.9. rdsGetScreenCoords – вычислить координаты на экране по координатам на рабочем поле

Функция rdsGetScreenCoords переводит координаты рабочего поля окна подсистемы в экранные координаты.

```

BOOL RDSCALL rdsGetScreenCoords(
    RDS_BHANDLE System,    // Подсистема
    int wx,int wy,        // Координаты на рабочем поле
    int *psx,int *psy     // Экранные координаты
);

```

Тип указателя на эту функцию:

RDS_BBhIIPpI

Параметры:

System

Идентификатор подсистемы, для окна которой нужно провести преобразование координат (окно должно быть открыто). Функция также может работать при открытом для данной подсистемы порте вывода (см. ниже).

wx, wy

Горизонтальная (wx) и вертикальная (wy) координаты точки на рабочем поле подсистемы в текущем масштабе. Горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля.

psx, psy

Указатели на целые переменные, в которые функция должна записать горизонтальную (psx) и вертикальную (psy) координаты точки экрана, соответствующей точке рабочего поля (wx, wy).

Возвращаемое значение:

TRUE – преобразование выполнено, FALSE – ошибка (System – не подсистема, или окно подсистемы закрыто и нет ни одного активного порта вывода).

Примечания:

Эта функция чаще всего используется для позиционирования каких-либо объектов Windows, (например, окон или контекстных меню), относительно блоков в окне подсистемы. Если окно подсистемы System открыто, функция выполнит преобразование координат для этого окна. Если окно закрыто, но при этом РДС управляется из другого приложения, для данной подсистемы создан порт вывода (см. §3.6), и в данный момент выполняется обновление порта вывода или его реакция на мышь, функция выполнит преобразование координат для данного порта вывода. В противном случае никакого преобразования координат выполнено не будет, и функция вернет FALSE.

См. также:

Функции работы с портами вывода (стр. 670).

A.5.8.10. `rdsGetTopWindowBlock` – блок-владелец активного окна

Функция `rdsGetTopWindowBlock` возвращает идентификатор блока, которому принадлежит текущее активное окно РДС.

```
RDS_BHANDLE RDSCALL rdsGetTopWindowBlock(void);
```

Тип указателя на эту функцию:

```
RDS_BhV
```

Возвращаемое значение:

Идентификатор блока, которому принадлежит самое верхнее немодальное окно из зарегистрированных в РДС. Сюда входят и окна подсистем: если активное окно – это окно подсистемы, функция вернет идентификатор этой подсистемы.

См. также:

`rdsRegisterWindow` (стр. 166).

A.5.8.11. `rdsHideAllEditorToolBars` – скрыть все панели окна подсистемы

Функция `rdsHideAllEditorToolBars` скрывает все панели окна указанной подсистемы. Вместо нее можно также использовать вызов `rdsSetEditorWindowFlags` с параметрами 0 и `RDS_EWF_ALLBARS` (стр. 265).

```
void RDSCALL rdsHideAllEditorToolBars(  
    RDS_BHANDLE System    // Подсистема  
);
```

Тип указателя на эту функцию:

```
RDS_VBh
```

Параметр:

`System`

Идентификатор подсистемы, панели окна которой нужно скрыть (окно подсистемы не обязательно должно быть открыто).

См. также:

`rdsSetEditorWindowFlags` (стр. 265).

A.5.8.12. `rdsOpenSystemWindow` – открыть окно подсистемы

Функция `rdsOpenSystemWindow` открывает окно указанной подсистемы, если оно закрыто.

```
void RDSCALL rdsOpenSystemWindow(  
    RDS_BHANDLE System    // Подсистема  
);
```

Тип указателя на эту функцию:

```
RDS_VBh
```

Параметр:

System

Идентификатор подсистемы, окно которой нужно открыть. В этом параметре вместо идентификатора подсистемы можно передать идентификатор блока другого типа, в этом случае будет открыто окно родительской подсистемы этого блока.

Примечания:

Если функция вызвана не из главного потока (например, в реакции блока на такт расчета), окно может открыться с некоторой задержкой. Пример использования этой функции описан в §2.12.7.

См. также:

rdsOpenSystemWindowEx (стр. 262), rdsCloseSystemWindow (стр. 255).

А.5.8.13. rdsOpenSystemWindowEx – открыть окно подсистемы с указанием его координат

Функция rdsOpenSystemWindowEx открывает окно указанной подсистемы, одновременно задавая его размер и положение.

```
void RDSCALL rdsOpenSystemWindowEx(  
    RDS_BHANDLE System,    // Подсистема  
    BOOL Maximized,        // На весь экран  
    int Left, int Top,      // Левый верхний угол окна  
    int Width, int Height   // Размеры окна  
);
```

Тип указателя на эту функцию:

RDS_VBhBIIII

Параметры:

System

Идентификатор подсистемы, окно которой нужно открыть. В этом параметре вместо идентификатора подсистемы можно передать идентификатор блока другого типа, в этом случае будет открыто окно родительской подсистемы этого блока.

Maximized

TRUE – окно нужно развернуть на весь экран, FALSE – размер окна указан в параметрах функции.

Left, Top

Координаты точки экрана (Left – горизонтальная координата, Top – вертикальная), в которой должен оказаться левый верхний угол открываемого окна.

Width, Height

Ширина (Width) и высота (Height) окна в точках экрана.

Примечания:

Эта функция открывает окно подсистемы, помещает его верхний левый угол в точку (Left, Top) и задает ему размер Width x Height точек. Если окно подсистемы уже открыто, функция просто меняет его размер и положение. Если функция вызвана не из главного потока (например, в реакции блока на такт моделирования), окно может открыться с некоторой задержкой.

См. также:

`rdsOpenSystemWindow` (стр. 261), `rdsCloseSystemWindow` (стр. 255),
`rdsSetSystemWindowBounds` (стр. 267), `rdsSetSystemWindowRect` (стр. 268).

A.5.8.14. `rdsRefreshBlockWindows` – обновить немодальные окна

Функция `rdsRefreshBlockWindows` обновляет все немодальные окна, принадлежащие указанному блоку (включая окна подсистем).

```
void RDSCALL rdsRefreshBlockWindows(  
    RDS_BHANDLE Block,      // Блок  
    BOOL Recurse           // Включая вложенные блоки  
);
```

Тип указателя на эту функцию:

`RDS_VBhB`

Параметры:

`Block`

Идентификатор блока, немодальные окна которого нужно обновить. Если `Block` – подсистема, обновляется также окно этой подсистемы.

`Recurse`

`TRUE`: если `Block` – подсистема, обновить также и все окна ее внутренних блоков и подсистем. `FALSE`: обновить только окна, принадлежащие `Block`.

Примечания:

Чаще всего эта функция вызывается для обновления окна подсистемы (и, возможно, окон всех вложенных в нее подсистем) в тех случаях, когда оно не обновляется автоматически. При вызове любой модели блока РДС взводит флаг обновления его родительской подсистемы, что приведет к обновлению окна этой подсистемы при первой возможности, поэтому здесь функцию `rdsRefreshBlockWindows` вызывать не нужно. Если же изменения во внешний вид одного из блоков внесены в результате каких-либо других действий – например, из функции какого-либо окна, открытого моделью средствами Windows API, или изменением динамической переменной без вызова специальной функции `rdsNotifyDynVarSubscribers` (стр. 352) – обновить окно подсистемы нужно вручную.

Эта же функция может использоваться для обновления немодальных окон, открытых моделями блоков для своих целей – ее вызов приводит к возникновению для этих блоков события `RDS_BFM_WINREFRESH` (стр. 77), в реакции на которое и нужно обновить принадлежащие блоку окна.

Пример использования функции `rdsRefreshBlockWindows` описан в §2.7.6.

См. также:

`RDS_BFM_WINREFRESH` (стр. 77), `rdsEnableWindowRefresh` (стр. 255).

A.5.8.15. `rdsScrollWindowToBlock` – показать блок в окне подсистемы

Функция `rdsScrollWindowToBlock` показывает указанный блок в окне его родительской подсистемы, прокручивая ее рабочее поле.

```
void RDSCALL rdsScrollWindowToBlock(  
    RDS_BHANDLE Block      // Блок  
);
```

Тип указателя на эту функцию:

RDS_VBh

Параметр:

Block

Идентификатор блока, который должен стать видимым пользователю в окне родительской подсистемы.

Примечания:

Эта функция прокручивает окно родительской подсистемы блока Block таким образом, чтобы изображение этого блока оказалось в видимой пользователю части рабочего поля. Окно должно быть открыто, при закрытом окне подсистемы вызов функции игнорируется. Если изображение блока больше видимой части рабочего поля, масштаб подсистемы уменьшается, чтобы изображение уместилось в видимую часть целиком.

См. также:

rdsScrollWindowToRect (стр. 264), rdsSetZoomPercent (стр. 269),
rdsSelectBlock (стр. 241).

A.5.8.16. rdsScrollWindowToRect – показать область в окне подсистемы

Функция rdsScrollWindowToRect показывает указанную прямоугольную область в окне подсистемы, прокручивая ее рабочее поле.

```
void RDSCALL rdsScrollWindowToRect(  
    RDS_BHANDLE System,    // Подсистема  
    int Left, int Top,      // Левый верхний угол области  
    int Width, int Height, // Размеры области  
    BOOL ZoomOut           // Разрешить уменьшать масштаб  
);
```

Тип указателя на эту функцию:

RDS_VBhIIIB

Параметры:

System

Идентификатор подсистемы, в окне которой нужно показать прямоугольную область.

Left, Top

Горизонтальная (Left) и вертикальная (Top) координаты левого верхнего угла прямоугольной области на рабочем поле в масштабе 100%. Горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля.

Width, Height

Ширина (Width) и высота (Height) прямоугольной области в масштабе 100%.

ZoomOut

TRUE – разрешить уменьшать масштаб подсистемы, если указанная прямоугольная область не умещается в видимую часть рабочего поля. FALSE – не менять масштаб, прокрутить рабочее поле так, чтобы центр прямоугольной области оказался в центре видимой части поля.

Примечания:

Эта функция прокручивает окно подсистемы System таким образом, чтобы прямоугольная область с указанными в параметрах функции координатами оказалась в видимой пользователю части рабочего поля. Окно должно быть открыто, при закрытом окне подсистемы вызов функции игнорируется. Если область больше видимой части рабочего поля и в параметре ZoomOut передано TRUE, масштаб подсистемы уменьшается, чтобы область уместилась в видимую часть целиком.

См. также:

rdsScrollWindowToBlock (стр. 263), rdsSetZoomPercent (стр. 269).

A.5.8.17. rdsSetEditorToolBars – задать состояние панелей окна подсистемы (устаревшая)

Функция rdsSetEditorToolBars устанавливает видимость отдельных панелей окна подсистемы согласно структуре RDS_EDITORTOOLBARS (стр. 126). Это устаревшая функция, сейчас вместо нее используется rdsSetEditorWindowFlags (стр. 265).

```
void RDSCALL rdsSetEditorToolBars(  
    RDS_BHANDLE System,          // Подсистема  
    RDS_PEDITORTOOLBARS pBars   // Структура описания  
);
```

Тип указателя на эту функцию:

RDS_VBhEt

Параметры:

System

Идентификатор подсистемы, состояние панелей окна которой нужно установить (окно подсистемы не обязательно должно быть открыто).

pBars

Указатель на структуру описания, согласно которой функция установит видимость панелей.

Примечания:

Эта функция устарела и позволяет установить видимость только некоторых панелей окна.

См. также:

RDS_EDITORTOOLBARS (стр. 126), rdsSetEditorWindowFlags (стр. 265).

A.5.8.18. rdsSetEditorWindowFlags – задать флаги панелей окна подсистемы

Функция rdsSetEditorWindowFlags устанавливает флаги, описывающие видимость панелей окна подсистемы.

```
void RDSCALL rdsSetEditorWindowFlags(  
    RDS_BHANDLE System,          // Подсистема  
    DWORD Flags,                 // Флаги  
    DWORD Mask                   // Маска  
);
```

Тип указателя на эту функцию:

RDS_VBhDwDw

Параметры:

System

Идентификатор подсистемы, состояние панелей окна которой нужно установить (окно подсистемы не обязательно должно быть открыто).

Flags

Набор битовых флагов, описывающих состояние панелей (эти же флаги возвращаются функцией `rdsGetEditorWindowFlags`, см. стр. 259). Введенный флаг указывает на то, что соответствующая панель должна быть видима, сброшенный – скрыта.

Mask

Маска панелей, видимость которых должна быть изменена. Это набор тех же флагов, что используются в параметре `Flags`, но, в данном случае, введенный флаг указывает на то, что видимость соответствующей панели должна быть изменена, сброшенный – видимость не меняется. Можно также использовать дополнительно описанную константу `RDS_EWF_ALLBARS`, являющуюся объединением всех флагов (ее передача в параметре `Mask` указывает на то, что видимость всех панелей должна быть изменена).

Примечание:

Эта функция устанавливает видимость панелей, указанных в параметре `Mask`, согласно флагам в параметре `Flags`. Единица в каком-либо разряде `Mask` указывает на то, что видимость соответствующей панели меняется в соответствии со значением того же самого бита в параметре `Flags`:

<i>Бит N в Mask</i>	<i>Бит N в Flags</i>	<i>Действие</i>
0	не важно	Видимость панели N не меняется.
1	0	Панель N делается скрытой.
1	1	Панель N делается видимой.

Пример:

Показать панель расчета в подсистеме с идентификатором `Sys`:

```
rdsSetEditorWindowFlags(  
    Sys,  
    RDS_EWF_CALCTOOLBAR,  
    RDS_EWF_CALCTOOLBAR);
```

или

```
rdsSetEditorWindowFlags(  
    Sys,  
    RDS_EWF_ALLBARS,          // Все флаги установлены  
    RDS_EWF_CALCTOOLBAR);
```

Скрыть строку состояния:

```
rdsSetEditorWindowFlags(Sys, 0, RDS_EWF_STATUSBAR);
```

Скрыть все панели:

```
rdsSetEditorWindowFlags(Sys, 0, RDS_EWF_ALLBARS);
```

См. также:

`rdsGetEditorWindowFlags` (стр. 259).

A.5.8.19. rdsSetSystemWindowBounds – задать границы окна подсистемы

Функция `rdsSetSystemWindowBounds` задает положение и размер окна указанной подсистемы.

```
void RDSCALL rdsSetSystemWindowBounds(  
    RDS_BHANDLE System,    // Подсистема  
    BOOL Maximized,        // На весь экран  
    int Left,int Top,      // Левый верхний угол окна  
    int Width,int Height  // Размеры окна  
);
```

Тип указателя на эту функцию:

```
RDS_VBhBIIII
```

Параметры:

`System`

Идентификатор подсистемы, границы окна которой задаются. В этом параметре вместо идентификатора подсистемы можно передать идентификатор блока другого типа, в этом случае будут заданы границы окна родительской подсистемы этого блока.

`Maximized`

`TRUE` – окно нужно развернуть на весь экран, `FALSE` – размер окна указан в параметрах функции.

`Left, Top`

Координаты точки экрана (`Left` – горизонтальная координата, `Top` – вертикальная), в которой будет располагаться левый верхний угол открываемого окна.

`Width, Height`

Ширина (`Width`) и высота (`Height`) окна в точках экрана.

Примечания:

Эта функция задает положение и размер окна подсистемы, либо разворачивая его на весь экран, либо помещая его верхний левый угол в точку (`Left, Top`) и присваивая ему размер `Width x Height` точек. В отличие от функции `rdsOpenSystemWindowEx` (стр. 262), эта функция не открывает окно подсистемы, если оно закрыто: новые размеры будут запомнены в параметрах самой подсистемы и использованы при следующем открытии окна.

Пример использования функции `rdsSetSystemWindowBounds` приведен в §2.12.5.

См. также:

`rdsSetSystemWindowRect` (стр. 268), `rdsOpenSystemWindowEx` (стр. 262).

A.5.8.20. rdsSetSystemWindowCaption – задать заголовок окна подсистемы

Функция `rdsSetSystemWindowCaption` задает временный или постоянный текст заголовка окна подсистемы.

```
void RDSCALL rdsSetSystemWindowCaption(  
    RDS_BHANDLE System,    // Подсистема  
    LPSTR Caption,        // Заголовок или NULL  
    BOOL Temp              // Временный заголовок  
);
```

Тип указателя на эту функцию:

RDS_VBhSB

Параметры:

System

Идентификатор подсистемы, заголовок окна которой задается (ее окно не обязательно должно быть открыто).

Caption

Указатель на строку с заголовком окна. NULL в этом параметре указывает на необходимость вернуться к стандартному заголовку, то есть к имени подсистемы.

Temp

TRUE – заголовок задается временно, если закрыть окно и открыть его снова он будет потерян. FALSE – заголовок задается постоянно, он будет запомнен в параметрах подсистемы и будет использоваться при каждом открытии окна (в том числе и после сохранения схемы).

Примечания:

Обычно в заголовке окна подсистемы, на соответствующей ему кнопке панели окон и в соответствующем пункте меню “Окна” отображается имя подсистемы или текст, заданный пользователем в ее параметрах. Функция `rdsSetSystemWindowCaption` позволяет программно изменить этот текст – постоянно (при `Temp==FALSE`) или временно (при `Temp==TRUE`).

A.5.8.21. `rdsSetSystemWindowRect` – задать границы окна подсистемы

Функция `rdsSetSystemWindowRect` задает положение и размер окна указанной подсистемы и открывает его, если это необходимо.

```
void RDSCALL rdsSetSystemWindowRect(  
    RDS_BHANDLE System,    // Подсистема  
    int Left,int Top,      // Левый верхний угол окна  
    int Width,int Height  // Размеры окна  
    BOOL OpenWindow       // Открыть окно  
);
```

Тип указателя на эту функцию:

RDS_VBhIIIB

Параметры:

System

Идентификатор подсистемы, границы окна которой задаются. Окно подсистемы не обязательно должно быть открыто.

Left,Top

Координаты точки экрана (Left – горизонтальная координата, Top – вертикальная), в которой будет располагаться левый верхний угол окна.

Width,Height

Ширина (Width) и высота (Height) окна в точках экрана. -1 в любом из этих параметров будет означать, что окно нужно развернуть на весь экран.

OpenWindow

TRUE – если окно закрыто, его нужно открыть. FALSE – если окно закрыто, нужно просто запомнить новый размер.

Примечания:

Эта функция задает положение и размер окна подсистемы, либо разворачивая его на весь экран, либо помещая его верхний левый угол в точку (Left,Top) и присваивая ему размер Width x Height точек. Если в параметре OpenWindow передано значение TRUE, функция откроет окно, если оно в данный момент закрыто.

См. также:

rdsSetSystemWindowBounds (стр. 267), rdsOpenSystemWindowEx (стр. 262).

A.5.8.22. rdsSetZoomPercent – задать масштаб окна подсистемы

Функция rdsSetZoomPercent задает масштаб окна подсистемы в процентах и, при необходимости, выводит в центр окна заданную точку рабочего поля.

```
void RDSCALL rdsSetZoomPercent(  
    RDS_BHANDLE System,           // Подсистема  
    int ZoomPercent,              // Масштаб в %  
    int CenterX, int CenterY      // Центральная точка или -1  
);
```

Тип указателя на эту функцию:

RDS_VBhIIIB

Параметры:

System

Идентификатор подсистемы, масштаб которой задается. Окно подсистемы должно быть открыто.

ZoomPercent

Новый масштаб в процентах.

CenterX, CenterY

Координаты точки рабочего поля в масштабе 100% (CenterX – горизонтальная координата, CenterY – вертикальная), которая должна оказаться в центре видимой в окне части рабочего поля после изменения масштаба. В этих параметрах можно передать -1, в этом случае РДС изменит масштаб так, чтобы в центре окна осталась та же точка рабочего поля, что и до изменения масштаба.

Примечания:

Эта функция изменяет масштаб подсистемы System, если ее окно открыто. В параметрах (CenterX,CenterY) можно указать координаты точки рабочего поля, которая должна оказаться в центре окна после изменения масштаба, или (-1,-1), чтобы сохранить текущую центральную точку. Если вывести указанную точку в центр окна невозможно (например, если она находится слишком близко к краю рабочего поля), в центр окна будет выставлена ближайшая точка из возможных.

Пример использования этой функции приведен в §2.12.5.

См. также:

rdsScrollWindowToRect (стр. 264).

A.5.9. Работа со слоями

Описываются функции управления слоями и конфигурациями слоев в подсистеме.

A.5.9.1. rdsAddLayer – добавить слой

Функция rdsAddLayer добавляет в подсистему слой с указанным именем.

```
int RDSCALL rdsAddLayer(  
    RDS_BHANDLE System,    // Подсистема  
    LPSTR LayerName        // Имя слоя  
);
```

Тип указателя на эту функцию:

RDS_IBhS

Параметры:

System

Идентификатор подсистемы.

LayerName

Указатель на строку с именем добавляемого слоя.

Возвращаемое значение:

Уникальный идентификатор добавленного слоя или -1, если слой с именем LayerName уже есть в подсистеме System.

Примечания:

Эта функция добавляет слой с именем LayerName в подсистему System, если в ней еще нет слоя с таким именем. Новый слой добавляется во все конфигурации на самый верх (то есть объекты, размещенные на нем, будут перекрывать все остальные).

См. также:

rdsSetLayerParams (стр. 275), rdsSetLayerPosition (стр. 276).

A.5.9.2. rdsGetLayerConfigName – имя конфигурации слоев по номеру

Функция rdsGetLayerConfigName возвращает имя конфигурации слоев с указанным номером в указанной подсистеме.

```
LPSTR RDSCALL rdsGetLayerConfigName(  
    RDS_BHANDLE System,    // Подсистема  
    int Num                // Номер конфигурации  
);
```

Тип указателя на эту функцию:

RDS_SBhI

Параметры:

System

Идентификатор подсистемы.

Num

Номер конфигурации слоев в подсистеме.

Возвращаемое значение:

Указатель на строку с именем конфигурации или NULL, если конфигурации с номером Num нет в подсистеме System. Строка располагается во внутренней памяти РДС, вызывающая программа не должна ее изменять.

Примечания:

Эта функция возвращает имя конфигурации слоев подсистемы с заданным номером. Конфигурации нумеруются с нуля, общее их число можно узнать из поля NumConfigs структуры RDS_EDITORPARAMETERS (стр. 122).

См. также:

rdsSetCurLayerConfig (стр. 274), RDS_EDITORPARAMETERS (стр. 122).

A.5.9.3. rdsGetLayerId – идентификатор слоя по имени

Функция rdsGetLayerId возвращает идентификатор слоя в указанной подсистеме по его имени.

```
int RDSCALL rdsGetLayerId(  
    RDS_BHANDLE System,    // Подсистема  
    LPSTR LayerName        // Имя слоя  
);
```

Тип указателя на эту функцию:

RDS_IBhS

Параметры:

System

Идентификатор подсистемы.

LayerName

Указатель на строку с именем слоя.

Возвращаемое значение:

Уникальный идентификатор слоя с именем LayerName, или -1, если слоя с таким именем нет в подсистеме System.

Примечания:

Имена слоев чувствительны к регистру: “Слой 1” и “слой 1” будут считаться разными слоями.

См. также:

rdsGetLayerName (стр. 272), rdsGetLayerParams (стр. 273).

A.5.9.4. rdsGetLayerIdInConfig – идентификатор слоя по номеру в конфигурации

Функция rdsGetLayerIdInConfig возвращает идентификатор слоя в указанной конфигурации слоев указанной подсистемы по его порядковому номеру.

```
int RDSCALL rdsGetLayerIdInConfig(  
    RDS_BHANDLE System,    // Подсистема  
    int ConfigNum,         // Номер конфигурации  
    int LayerNum           // Порядковый номер слоя  
);
```

Тип указателя на эту функцию:

RDS_IBhII

Параметры:

System

Идентификатор подсистемы.

ConfigNum

Номер конфигурации слоев в подсистеме System или -1 для текущей конфигурации.

LayerNum

Порядковый номер слоя в указанной конфигурации (0 – ближайший к переднему плану) или -1 для текущего слоя.

Возвращаемое значение:

Уникальный идентификатор слоя с номером LayerNum в конфигурации ConfigNum подсистемы System, или -1, если нет такого слоя или такой конфигурации слоев.

Примечания:

Конфигурации слоев в подсистеме нумеруются с нуля, общее их число можно узнать из поля NumConfigs структуры RDS_EDITORPARAMETERS (стр. 122). Общее число слоев (оно одинаковое во всех конфигурациях подсистемы) можно узнать из поля NumLayers той же структуры RDS_EDITORPARAMETERS. Слои в конфигурации нумеруются с нуля начиная с переднего плана. Невидимые слои тоже входят в нумерацию.

См. также:

rdsGetLayerName (стр. 272), rdsGetLayerParams (стр. 273),
RDS_EDITORPARAMETERS (стр. 122).

A.5.9.5. rdsGetLayerName – имя слоя по идентификатору

Функция rdsGetLayerName возвращает имя слоя с указанным идентификатором в указанной подсистеме.

```
LPSTR RDSCALL rdsGetLayerName(  
    RDS_BHANDLE System,    // Подсистема  
    int LayerId            // Идентификатор слоя  
);
```

Тип указателя на эту функцию:

RDS_SbHI

Параметры:

System

Идентификатор подсистемы.

LayerId

Идентификатор слоя.

Возвращаемое значение:

Указатель на строку с именем слоя или NULL, если слоя с идентификатором LayerId нет в подсистеме System. Строка располагается во внутренней памяти РДС, вызывающая программа не должна ее изменять.

См. также:

rdsGetLayerId (стр. 271), rdsGetLayerParams (стр. 273).

A.5.9.6. rdsGetLayerParams – параметры слоя в заданной конфигурации

Функция `rdsGetLayerParams` возвращает флаги видимости, разрешения редактирования и установки в качестве текущего для слоя с указанным идентификатором в указанной конфигурации слоев.

```
BOOL RDSCALL rdsGetLayerParams(  
    RDS_BHANDLE System,    // Подсистема  
    int ConfigNum,         // Номер конфигурации  
    int LayerId,           // Идентификатор слоя  
    BOOL *pVisible,        // Видимость  
    BOOL *pEditable,       // Разрешенность  
    BOOL *pCurrent         // Признак текущего  
);
```

Тип указателя на эту функцию:

`RDS_BBhIIPBpBpB`

Параметры:

`System`

Идентификатор подсистемы.

`ConfigNum`

Номер конфигурации слоев в подсистеме `System` или `-1` для текущей конфигурации.

`LayerId`

Идентификатор слоя в конфигурации `ConfigNum`.

`pVisible`

Указатель на логическую (BOOL) переменную, в которую функция запишет TRUE, если слой `LayerId` видим в указанной конфигурации, и FALSE, если он скрыт.

`pEditable`

Указатель на логическую (BOOL) переменную, в которую функция запишет TRUE, если в указанной конфигурации для слоя `LayerId` разрешено редактирование и реакции блоков на мышшь, и FALSE в противном случае.

`pCurrent`

Указатель на логическую (BOOL) переменную, в которую функция запишет TRUE, если в указанной конфигурации слой `LayerId` установлен в качестве текущего, и FALSE в противном случае.

Возвращаемое значение:

TRUE, если функция записала параметры слоя по переданным указателям, и FALSE, если конфигурации `ConfigNum` или слоя `LayerId` нет в подсистеме `System`.

Примечания:

Конфигурации слоев в подсистеме (параметр `ConfigNum`) нумеруются с нуля, общее их число можно узнать из поля `NumConfigs` структуры `RDS_EDITORPARAMETERS` (стр. 122). Любой из параметров `pVisible`, `pEditable` и `pCurrent` может иметь значение NULL, если соответствующий параметр слоя не нужен вызывающей программе.

См. также:

`rdsGetLayerName` (стр. 272), `rdsSetLayerParams` (стр. 275),
`RDS_EDITORPARAMETERS` (стр. 122).

A.5.9.7. rdsSetCurLayerConfig – установить текущую конфигурацию

Функция rdsSetCurLayerConfig устанавливает конфигурацию слоев с заданным номером в указанной подсистеме в качестве текущей.

```
BOOL RDSCALL rdsSetCurLayerConfig(  
    RDS_BHANDLE System,    // Подсистема  
    int ConfigNum          // Номер конфигурации  
);
```

Тип указателя на эту функцию:

RDS_BBhI

Параметры:

System

Идентификатор подсистемы.

ConfigNum

Номер конфигурации слоев в подсистеме System.

Возвращаемое значение:

TRUE, если конфигурации ConfigNum успешно установлена в качестве текущей в подсистеме System, FALSE в противном случае (например, если в подсистеме отсутствует конфигурация с указанным номером).

Примечания:

Конфигурации слоев в подсистеме нумеруются с нуля, общее их число можно узнать из поля NumConfigs структуры RDS_EDITORPARAMETERS (стр. 122).

См. также:

rdsSetCurLayerConfigByName (стр. 274),
rdsGetLayerConfigName (стр. 270), RDS_EDITORPARAMETERS (стр. 122).

A.5.9.8. rdsSetCurLayerConfigByName – установить текущую конфигурацию по имени

Функция rdsSetCurLayerConfigByName устанавливает конфигурацию слоев с заданным именем в указанной подсистеме в качестве текущей.

```
BOOL RDSCALL rdsSetCurLayerConfigByName(  
    RDS_BHANDLE System,    // Подсистема  
    LPSTR ConfigName      // Имя конфигурации  
);
```

Тип указателя на эту функцию:

RDS_BBhS

Параметры:

System

Идентификатор подсистемы.

ConfigName

Имя конфигурации слоев в подсистеме System.

Возвращаемое значение:

TRUE, если конфигурации ConfigName успешно установлена в качестве текущей в подсистеме System, и FALSE в противном случае (например, если в подсистеме отсутствует конфигурация с указанным именем).

Примечания:

Имена конфигураций слоев чувствительны к регистру символов: “Основная” и “основная” будут считаться разными конфигурациями.

См. также:

rdsSetCurLayerConfig (стр. 274), rdsGetLayerConfigName (стр. 270).

A.5.9.9. rdsSetLayerParams – задать параметры слоя в конфигурации

Функция rdsSetLayerParams устанавливает флаги видимости, разрешения редактирования и текущего слоя для слоя с указанным идентификатором в указанной конфигурации слоев.

```
BOOL RDSCALL rdsSetLayerParams(  
    RDS_BHANDLE System,    // Подсистема  
    int ConfigNum,         // Номер конфигурации  
    int LayerId,           // Идентификатор слоя  
    BOOL Visible,          // Видимость  
    BOOL Editable,         // Разрешенность  
    BOOL Current           // Признак текущего  
);
```

Тип указателя на эту функцию:

RDS_BBhIIBBB

Параметры:

System

Идентификатор подсистемы.

ConfigNum

Номер конфигурации слоев в подсистеме System или –1 для текущей конфигурации.

LayerId

Идентификатор слоя.

Visible

Слой должен быть видимым (TRUE) или скрытым (FALSE).

Editable

Разрешить (TRUE) или запретить (FALSE) редактирование и реакции блоков на мышь для указанного слоя.

Current

TRUE – указанный слой нужно сделать текущим слоем конфигурации. FALSE – текущий слой конфигурации не изменится.

Возвращаемое значение:

TRUE, если функция установила параметры указанного слоя, и FALSE, если конфигурации ConfigNum или слоя LayerId нет в подсистеме System.

Примечания:

Конфигурации слоев в подсистеме (параметр `ConfigNum`) нумеруются с нуля, общее их число можно узнать из поля `NumConfigs` структуры `RDS_EDITORPARAMETERS` (стр. 122). Действие параметров функции `Visible` и `Editable` несколько отличается от параметра `Current`: если `Visible` и `Editable` устанавливают и сбрасывают соответствующие флаги слоя, то `Current` влияет на слой, только если в нем передано значение `TRUE`: в этом случае слой `LayerId` станет новым текущим слоем в конфигурации `ConfigNum`. Значение `FALSE` в параметре `Current` не заставит слой `LayerId` перестать быть текущим, если он им был – оно просто будет проигнорировано. Для того, чтобы `LayerId` перестал быть текущим слоем конфигурации, нужно сделать в ней текущим какой-либо другой слой.

См. также:

`rdsGetLayerParams` (стр. 273), `rdsSetLayerPosition` (стр. 276),
`RDS_EDITORPARAMETERS` (стр. 122).

А.5.9.10. `rdsSetLayerPosition` – задать положение слоя в заданной конфигурации

Функция `rdsSetLayerPosition` устанавливает положение слоя в указанной конфигурации относительно других ее слоев.

```
BOOL RDSCALL rdsSetLayerPosition(  
    RDS_BHANDLE System,      // Подсистема  
    int ConfigNum,          // Номер конфигурации  
    int LayerId,             // Идентификатор слоя  
    int PostType,            // Операция (RDS_SLP_*)  
    int RefLayer             // Базовый слой  
);
```

Тип указателя на эту функцию:

`RDS_BBhIII`

Параметры:

`System`

Идентификатор подсистемы.

`ConfigNum`

Номер конфигурации слоев в подсистеме `System` или `-1` для текущей конфигурации.

`LayerId`

Идентификатор слоя, перемещаемого в указанной конфигурации.

`PostType`

Как задается положение слоя (одна из констант `RDS_SLP_*`):

`RDS_SLP_TOP` Переместить слой `LayerId` на самый верх (ближе всего к переднему плану).

`RDS_SLP_BOTTOM` Переместить слой `LayerId` в самый низ (дальше всего от переднего плана).

`RDS_SLP_BEFORE` Разместить слой `LayerId` непосредственно перед слоем `RefLayer` (`LayerId` будет ближе к переднему плану, чем `RefLayer`).

RDS_SLP_AFTER Разместить слой LayerId непосредственно после слоя RefLayer (LayerId будет дальше от переднего плана, чем RefLayer).

RefLayer

Идентификатор слоя, относительно которого позиционируется LayerId при PostType, равном RDS_SLP_BEFORE или RDS_SLP_AFTER (в остальных случаях значение этого параметра игнорируется).

Возвращаемое значение:

TRUE, если функция успешно переместила указанный слой, и FALSE, если конфигурации ConfigNum или слоя LayerId (и RefLayer, если он нужен) нет в подсистеме System.

Примечания:

Конфигурации слоев в подсистеме (параметр ConfigNum) нумеруются с нуля, общее их число можно узнать из поля NumConfigs структуры RDS_EDITORPARAMETERS (стр. 122).

См. также:

rdsSetLayerParams (стр. 275), RDS_EDITORPARAMETERS (стр. 122).

A.5.10. Загрузка и сохранение данных блока

Описываются функции, используемые при загрузке и сохранении параметров и состояния блока (см. §2.8).

A.5.10.1. rdsReadBlockData – считать данные блока в двоичном формате

Функция rdsReadBlockData считывает двоичные данные блока, ранее записанные функцией rdsWriteBlockData (стр. 280).

```
BOOL RDSCALL rdsReadBlockData(  
    LPVOID Buffer,          // Указатель на начало данных  
    int Size                // Размер данных  
);
```

Тип указателя на эту функцию:

RDS_BpVI

Параметры:

Buffer

Указатель на начало области памяти, в которую нужно загрузить считанные данные.

Size

Размер считываемых данных в байтах.

Возвращаемое значение:

TRUE – данные считаны успешно, FALSE – ошибка чтения (осталось меньше Size еще не считанных данных).

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на события загрузки данных RDS_BFM_LOADBIN (стр. 52) и загрузки состояния RDS_BFM_LOADSTATE (стр. 39) в двоичном формате. Во всех остальных случаях функция немедленно возвращает FALSE.

При загрузке данных или состояния каждого блока в двоичном формате РДС известен источник этих данных (это может быть файл на диске или область памяти) и общий размер сохраненных данных. Каждый вызов функции `rdsReadBlockData` считывает из этого источника `Size` байтов начиная с текущей позиции, записывает их в память по адресу `Buffer` и перемещает внутренний указатель этого источника данных вперед на `Size`. Эта функция очень похожа на большинство функций стандартных библиотек, читающих данные из файла, только в ней не нужно указывать дескриптор файла или другого источника данных – он определяется РДС и инициализируется перед вызовом реакций на события `RDS_BFM_LOADBIN` и `RDS_BFM_LOADSTATE`.

Функция не может считать данных больше, чем было записано функцией `rdsWriteBlockData` при их сохранении. Если, например, при сохранении данных (событие `RDS_BFM_SAVEBIN`, стр. 53) или состояния (событие `RDS_BFM_SAVESTATE`, стр. 45) было всего записано 20 байтов одним или несколькими последовательными вызовами `rdsWriteBlockData`, вызовами `rdsReadBlockData` тоже можно считать не более двадцати байтов, не важно, будет это 20 вызовов для чтения по одному байту, два последовательных вызова для чтения пяти и пятнадцати байтов, один вызов для чтения двадцати байтов и т.п. Если при очередном вызове в параметре `Size` будет указано больше байтов, чем осталось в источнике, функция вернет `FALSE`.

Примеры использования функции `rdsReadBlockData` приведены в §§2.8.2, 2.12.4 и 2.14.3.

См. также:

`rdsWriteBlockData` (стр. 280), `RDS_BFM_LOADBIN` (стр. 52),
`RDS_BFM_LOADSTATE` (стр. 39).

A.5.10.2. `rdsReportTextLoadError` – сообщение об ошибке текстового формата

Функция `rdsReportTextLoadError` добавляет указанное сообщение в общий список сообщений об ошибках при загрузке данных блока в текстовом формате.

```
void RDSCALL rdsReportTextLoadError(  
    LPSTR ErrorMessage           // Сообщение  
) ;
```

Тип указателя на эту функцию:

`RDS_VS`

Параметры:

`ErrorMessage`

Указатель на строку с сообщением об ошибке.

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на событие загрузки данных блока в текстовом формате `RDS_BFM_LOADTXT` (стр. 52), во всех остальных случаях вызов игнорируется.

При загрузке данных блока в текстовом формате обнаруженные в данных ошибки (неизвестные ключевые слова, ошибки формата и т.п.) обычно не показываются пользователю немедленно – при большом количестве ошибок ему пришлось бы постоянно нажимать кнопку “ОК” в окне сообщения. Вместо этого из сообщений о возникших ошибках формируется список, который показывается пользователю по окончании загрузки схемы (или блока, если загружался только один блок). Функция `rdsReportTextLoadError`

позволяет добавить в этот список произвольное сообщение, при этом имя блока, из реакции которого вызвана функция, добавится к сообщению автоматически.

См. также:

RDS_BFM_LOADTXT (стр. 52).

A.5.10.3. rdsStructToFontText – формирование описания шрифта

Функция `rdsStructToFontText` возвращает динамическую строку, представляющую собой описание указанного в параметрах шрифта, пригодное для разбора функциями `rdsFontTextToStruct` (стр. 290) и `rdsReadFontText` (стр. 295).

```
LPSTR RDSCALL rdsStructToFontText(  
    RDS_PSERVFONTPARAMS pStr, // Описание шрифта  
    int *pLength              // Длина строки  
);
```

Тип указателя на эту функцию:

RDS_SpFspI

Параметры:

`pStr`

Указатель на структуру описания шрифта `RDS_SERVFONTPARAMS` (стр. 135), по данным которой формируется строка.

`pLength`

Указатель на целую переменную, в которую функция должна записать длину получившейся строки. Если вызывающей программе не нужна длина строки, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую описание шрифта из структуры `pStr`, или `NULL`, если значение поля структуры `servSize` не соответствует ее размеру.

Примечания:

Строка, формируемая функцией, состоит из тех же ключевых слов, которые используются при сохранении схемы в текстовом формате для описания шрифтов окна подсистемы. Эти же слова используются при записи параметров шрифта функцией `rdsWriteFontText` (стр. 282). Чаще всего эта функция используется при сохранении данных блока в текстовом формате или для перевода описания шрифта в строку, которую можно где-нибудь хранить с тем, чтобы потом расшифровать ее с помощью функции `rdsFontTextToStruct`.

Пример использования функции `rdsStructToFontText` приведен в §2.10.1. Динамическая строка, созданная этой функцией, должна быть **обязательно** освобождена вызовом `rdsFree` (стр. 187).

См. также:

`RDS_SERVFONTPARAMS` (стр. 135), `rdsWriteFontText` (стр. 282),
`rdsFree` (стр. 187), `rdsFontTextToStruct` (стр. 290),
`rdsReadFontText` (стр. 295).

A.5.10.4. rdsWriteBlockData – записать данные блока в двоичном формате

Функция rdsWriteBlockData записывает двоичные данные блока – при загрузке они могут быть считаны функцией rdsReadBlockData (стр. 277).

```
BOOL RDSCALL rdsWriteBlockData(  
    LPVOID Buffer,    // Указатель на начало области памяти  
    int Size          // Размер данных  
);
```

Тип указателя на эту функцию:

RDS_BpVI

Параметры:

Buffer

Указатель на начало области памяти, в которой находятся записываемые данные.

Size

Размер записываемых данных в байтах.

Возвращаемое значение:

TRUE – данные записаны успешно, FALSE – ошибка записи.

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на события записи данных RDS_BFM_SAVEBIN (стр. 53) и записи состояния RDS_BFM_SAVESTATE (стр. 45) в двоичном формате. Во всех остальных случаях функция немедленно возвращает FALSE.

При сохранении данных или состояния блока в двоичном формате данные, переданные в функцию rdsWriteBlockData, записываются либо во внутренний буфер, либо непосредственно в файл. Данные пишутся последовательно, каждый вызов функции добавляет к записанным данным Size байтов начиная с указателя Buffer. Эта функция очень похожа на большинство функций стандартных библиотек, записывающих данные в файл, только в ней не нужно указывать дескриптор файла или другого получателя данных – он определяется РДС и инициализируется перед вызовом реакций на события RDS_BFM_SAVEBIN и RDS_BFM_SAVESTATE.

Примеры использования функции rdsWriteBlockData приведены в §§2.8.2, 2.12.4 и 2.14.3.

См. также:

rdsReadBlockData (стр. 277), RDS_BFM_SAVEBIN (стр. 53),
RDS_BFM_SAVESTATE (стр. 45).

A.5.10.5. rdsWriteBlockDataText – добавление текста к сохраняемым в текстовом формате данным блока

Функция rdsWriteBlockDataText добавляет указанный в параметрах текст к данным, сохраняемым блоком в текстовом формате.

```
void RDSCALL rdsWriteBlockDataText(  
    LPSTR String,      // Текст  
    BOOL NewLine      // Перевод строки перед текстом  
);
```


Тип указателя на эту функцию:

RDS_VSB

Параметры:

String

Указатель на строку с текстом, добавляемым к записанным данным.

NewLine

TRUE – перевести строку перед добавлением текста String, FALSE – добавить перед ним пробел.

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на событие записи данных блока в текстовом формате RDS_BFM_SAVETXT (стр. 54), во всех остальных случаях вызов игнорируется. Она добавляет указанный в параметрах произвольный текст к общему набору текстовых данных блока, после завершения реакции на событие RDS_BFM_SAVETXT этот набор будет записан в файл или буфер обмена (в зависимости от того, для чего сохраняются данные блока).

Перед текстом из строки String всегда добавляется либо пробел (при NewLine==FALSE), либо перевод строки (при NewLine==TRUE), таким образом, при записи различных ключевых слов этой функцией не нужно отдельно добавлять между ними пробелы. Например, программа

```
int w=100;
char *string=rdsItoA(w,10,0);
rdsWriteBlockDataText("width",FALSE);
rdsWriteBlockDataText(string,FALSE);
rdsFree(string);
```

добавит к текстовым данным блока строку “ width 100”.

Функцию rdsWriteBlockDataText можно использовать для перевода строки в наборе данных блока, для этого в параметре String нужно передать NULL, а в параметре NewLine – TRUE:

```
rdsWriteBlockDataText(NULL,TRUE);
```

Пример использования функции rdsWriteBlockDataText приведен в §2.8.3.

См. также:

RDS_BFM_SAVETXT (стр. 54), RDS_BFM_LOADTXT (стр. 52).

A.5.10.6. rdsWriteColorText – запись цвета в текстовом формате

Функция rdsWriteColorText добавляет к данным, сохраняемым блоком в текстовом формате, значение указанного в параметрах цвета с предваряющим его, если это необходимо, ключевым словом.

```
void RDSCALL rdsWriteColorText(
    LPSTR Keyword,           // Ключевое слово
    COLORREF Color,          // Цвет
    BOOL Rgb                 // Запись по компонентам
);
```

Тип указателя на эту функцию:

RDS_VSCrB

Параметры:

Keyword

Указатель на строку с ключевым словом, или NULL, если ключевое слово добавлять не нужно.

Color

Значение цвета (COLORREF, см. стр. 24).

Rgb

Выводить цвет покомпонентно (TRUE) или одним числом (FALSE).

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на события записи данных блока в текстовом формате RDS_BFM_SAVETXT (стр. 54), во всех остальных случаях вызов игнорируется. Она добавляет описание цвета к общему набору текстовых данных блока, после завершения реакции на событие RDS_BFM_SAVETXT этот набор будет записан в файл или буфер обмена (в зависимости от того, для чего сохраняются данные блока). Перед описанием цвета всегда добавляется пробел.

Функция может описывать цвет двумя способами: покомпонентно с дополнительным словом “rgb”, за которым следуют три числа интенсивности цвета по каналам (красный, затем зеленый, затем синий канал), или одним десятичным числом. Перед собственно описанием цвета можно добавить произвольное ключевое слово. Например:

<i>Вызов функции</i>	<i>Добавленный текст</i>
<code>rdsWriteColorText(NULL, 0xAAFF, FALSE);</code>	“ 43775”
<code>rdsWriteColorText(NULL, 0xAAFF, TRUE);</code>	“ rgb 255 170 0”
<code>rdsWriteColorText("fill", 0xAAFF, FALSE);</code>	“ fill 43775”
<code>rdsWriteColorText("fill", 0xAAFF, TRUE);</code>	“ fill rgb 255 170 0”

Формат записи цвета, используемый в функции `rdsWriteColorText`, совместим с функцией `rdsReadColorText` (стр. 294) и текстовым форматом схем РДС.

См. также:

RDS_BFM_SAVETXT (стр. 54), RDS_BFM_LOADTXT (стр. 52),
`rdsReadColorText` (стр. 294).

A.5.10.7. `rdsWriteFontText` – запись параметров шрифта в текстовом формате

Функция `rdsWriteFontText` добавляет к данным, сохраняемым блоком в текстовом формате, указанные параметры шрифта.

```
void RDSCALL rdsWriteFontText(  
    int Mask,           // Флаги записываемых параметров  
    LPSTR Name,         // Имя шрифта  
    int SizeHeight,     // Высота или размер  
    COLORREF Color,     // Цвет  
    int Charset,        // Набор символов  
    int Escapement,     // Угол поворота  
    BOOL Bold,          // Жирность  
    BOOL Italic,        // Курсив  
    BOOL Underline,     // Подчеркивание  
    BOOL StrikeOut      // Перечеркивание  
);
```

Тип указателя на эту функцию:

RDS_VISICrIIBBBB

Параметры:

Mask

Набор битовых флагов, указывающих, какие параметры нужно записать и как интерпретировать переданные значения:

RDS_GFNAME	Записать имя шрифта, переданное в параметре Name, заключив его в двойные кавычки и предварив его ключевым словом “font”.
RDS_GFSIZE	Записать размер шрифта в типографских точках (points), переданный в параметре SizeHeight, предварив его ключевым словом “size”. Не может использоваться одновременно с флагом RDS_GFHEIGHT.
RDS_GFHEIGHT	Записать высоту шрифта в точках экрана, переданную в параметре SizeHeight, предварив ее ключевым словом “height”. Не может использоваться одновременно с флагом RDS_GFSIZE.
RDS_GFCHARSET	Записать набор символов шрифта, переданный в параметре Charset, предварив его ключевым словом “charset”. Русский, ANSI, OEM и символьный наборы записываются ключевыми словами “rus”, “ansi”, “oem” и “symbol” соответственно, все остальные – в виде целого числа.
RDS_GFESCAPEMENT	Записать угол поворота шрифта в градусах, переданный в параметре Escapement, предварив его ключевым словом “esc”.
RDS_GFCOLOR	Записать цвет шрифта, переданный в параметре Color, предварив его ключевым словом “color”. Цвет записывается в виде одного целого числа.
RDS_GFBOLD	Если в параметре Bold передано TRUE, записать ключевое слово “bold”.
RDS_GFITALIC	Если в параметре Italic передано TRUE, записать ключевое слово “italic”.
RDS_GFUNDERLINE	Если в параметре Underline передано TRUE, записать ключевое слово “underline”.
RDS_GFSTRIKEOUT	Если в параметре StrikeOut передано TRUE, записать ключевое слово “strikeout”.

Все эти флаги совпадают с флагами, используемыми графической функцией РДС rdsXGSetFont (стр. 379). Кроме них в параметре Mask можно указывать следующие константы, объединяющие несколько флагов вместе:

RDS_GFFONTALLHEIGHT	Все указанные выше флаги, кроме RDS_GFSIZE (размер шрифта задается высотой в точках экрана).
RDS_GFFONTBASIC	Все указанные выше флаги, кроме RDS_GFSIZE (размер шрифта задается высотой в точках экрана) и RDS_GFESCAPEMENT (угол поворота не записывается).
RDS_GFFONTSTYLES	Жирность, курсив, зачеркивание и подчеркивание (флаги RDS_GFBOLD, RDS_GFITALIC, RDS_GFUNDERLINE и RDS_GFSTRIKEOUT).

Если в параметре `Mask` передать 0, будут записаны все параметры шрифта, высота его при этом будет считаться заданной в точках экрана, как при указании флага `RDS_GFHEIGHT`.

`Name`

Указатель на строку с именем шрифта.

`SizeHeight`

Размер шрифта в типографских точках (при указанном в `Mask` флаге `RDS_GFSIZE`) или в точках экрана (при указанном флаге `RDS_GFHEIGHT`).

`Color`

Цвет шрифта (`COLORREF`, см. стр. 24).

`Charset`

Набор символов шрифта.

`Escapement`

Угол поворота шрифта в градусах относительно горизонтали.

`Bold`

Шрифт жирный (`TRUE`) или обычный (`FALSE`).

`Italic`

Курсив (`TRUE`) или обычный шрифт (`FALSE`).

`Underline`

Шрифт подчеркнут (`TRUE`) или нет (`FALSE`).

`StrikeOut`

Шрифт перечеркнут (`TRUE`) или нет (`FALSE`).

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на событие записи данных блока в текстовом формате `RDS_BFM_SAVETXT` (стр. 54), во всех остальных случаях вызов игнорируется. Она добавляет описание шрифта к общему набору текстовых данных блока. После завершения реакции на событие `RDS_BFM_SAVETXT` этот набор будет записан в файл или буфер обмена (в зависимости от того, для чего сохраняются данные блока). Перед описанием, как и между всеми словами внутри описания, добавляется пробел.

Функция добавляет к тексту, сохраняемому блоком, ключевые слова и значения параметров, описывающих шрифт, согласно флагам в параметре `Mask`. Значения параметров функции, для которых не установлены соответствующие флаги в `Mask`, могут быть любыми: например, если не установлен флаг `RDS_GFNAME`, в параметре `Name` можно передать `NULL`.

Формат записи цвета, используемый в функции `rdsWriteFontText`, совместим с функциями `rdsStructToFontText` (стр. 279), `rdsReadFontText` (стр. 295), `rdsFontTextToStruct` (стр. 290) и текстовым форматом схем РДС.

Пример:

Вызов функции

```
rdsWriteFontText(RDS_GFFONTBASIC,  
                "Times New Roman", 20, 0xffffffff,  
                RUSSIAN_CHARSET, 0, TRUE, TRUE, FALSE, FALSE);
```

добавит к текстовым данным блока пробел и следующий текст (порядок пар “ключевое слово – значение” может отличаться, переводов строки внутри текста на самом деле не будет):

```
font "Times New Roman" height 20 color 16777215
charset rus bold italic
```

См. также:

RDS_BFM_SAVETXT (стр. 54), RDS_BFM_LOADTXT (стр. 52),
rdsReadFontText (стр. 295), rdsFontTextToStruct (стр. 290),
rdsStructToFontText (стр. 279).

A.5.10.8. rdsWriteHexText – запись блока двоичных данных в текстовом формате

Функция `rdsWriteHexText` добавляет к данным, сохраняемым блоком в текстовом формате, строку в двойных кавычках, представляющую собой записанные последовательно шестнадцатеричные значения байтов указанной области памяти.

```
void RDSCALL rdsWriteHexText (
    LPVOID Data,           // Указатель на начало области памяти
    int Size,              // Размер области памяти
    int MaxWidth            // Число символов в строке
);
```

Тип указателя на эту функцию:

RDS_VpVII

Параметры:

Data

Указатель на начало области памяти, в которой находятся записываемые данные.

Size

Размер записываемых данных в байтах.

MaxWidth

Число символов в шестнадцатеричной записи, после которого нужно перевести строку, добавив символ продолжения “+” и продублировав двойные кавычки, или 0, если строки переводить не нужно.

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на события записи данных блока в текстовом формате RDS_BFM_SAVETXT (стр. 54), во всех остальных случаях вызов игнорируется. Она добавляет строку с шестнадцатеричными кодами к общему набору текстовых данных блока. После завершения реакции на событие RDS_BFM_SAVETXT этот набор будет записан в файл или буфер обмена (в зависимости от того, для чего сохраняются данные блока). Перед строкой всегда добавляется пробел.

Чаще всего функция `rdsWriteHexText` используется для того, чтобы в текстовом формате записать массив каких-либо двоичных данных, не разбирая их структуру. Каждый байт области памяти Data функция переводит в два шестнадцатеричных символа, записывает эти пары символов друг за другом и окружает строку двойными кавычками. Если значение параметра MaxWidth не нулевое (в текущей версии РДС значение MaxWidth, меньшее пяти, будет эквивалентно нулевому), то функция, сформировав строку из MaxWidth символов, закрывает кавычку, добавит символ “+”, переведет строку, добавит еще один символ “+”, потом снова откроет кавычку и продолжит запись (в текстовом формате РДС допускается таким образом переносить длинный текст на следующую строку файла).

Для чтения такой строки шестнадцатеричных кодов в заданную область памяти при загрузке данных блока следует использовать функцию `rdsReadHexText` (стр. 297).

Пример:

Следующий фрагмент программы запишет в данные блока строку из последовательно возрастающих шестнадцатеричных кодов:

```
int maxchars=0;
BYTE array[20];
for(int i=0;i<20;i++) // Массив возрастающих байтов
    array[i]=i;
// Запись массива
rdsWriteHexText(array,20,maxchars);
```

В результате ее работы в данные блока добавится пробел и следующий текст:

```
"000102030405060708090A0B0C0D0E0F10111213"
```

Если заменить в первой строке программы “maxchars=0” на “maxchars=15”, в сохраненные данные блока добавится текст

```
"0001020304050"+
+"60708090A0B0C"+
+"0D0E0F1011121"+
+"3"
```

Каждая строка в этом тексте, исключая символы продолжения “+”, не длиннее 15 символов. При чтении данных блока такая разбитая строка будет автоматически собрана.

См. также:

RDS_BFM_SAVETXT (стр. 54), RDS_BFM_LOADTXT (стр. 52),
rdsReadHexText (стр. 297).

A.5.10.9. rdsWriteLineStyleText – запись стиля линии в текстовом формате

Функция rdsWriteLineStyleText добавляет к данным, сохраняемым блоком в текстовом формате, стандартное ключевое слово, обозначающее стиль линии.

```
void RDSCALL rdsWriteLineStyleText(
    int Style,           // Стиль линии (PS_*)
    BOOL WriteNull       // Писать ли PS_NULL
);
```

Тип указателя на эту функцию:

RDS_VIB

Параметры:

Style

Одна из стандартных констант Windows API, обозначающая стиль линии:

PS_DASH	Пунктирная линия (ключевое слово “dash”).
PS_DASHDOT	Линия из чередующихся отрезков и точек (ключевое слово “dashdot”).
PS_DASHDOTDOT	Линия из повторяющихся групп “отрезок-точка-точка” (ключевое слово “dashdotdot”).
PS_DOT	Линия, состоящая из точек (ключевое слово “dot”).
PS_NULL	Невидимая линия (ключевое слово “empty”).
PS_SOLID	Сплошная линия (ключевое слово “solid”).

PS_INSIDEFRAME Специальный стиль сплошной линии, разрешающий Windows скорректировать размеры геометрической фигуры, ограниченной этой линией, так, чтобы она уместилась в заданный прямоугольник (ключевое слово “inside”). В РДС используется редко.

Все эти флаги совпадают с флагами, используемыми графической функцией РДС `rdsXGSetPenStyle` (стр. 382).

WriteNull

TRUE – если параметр `Style` равен `PS_NULL` (отсутствие линии), записывать ключевое слово “empty”. **FALSE** – для `PS_NULL` не записывать ничего.

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на события записи данных блока в текстовом формате `RDS_BFM_SAVETXT` (стр. 54), во всех остальных случаях вызов игнорируется. Она добавляет ключевое слово стиля линии к общему набору текстовых данных блока. После завершения реакции на событие `RDS_BFM_SAVETXT` этот набор будет записан в файл или буфер обмена (в зависимости от того, для чего сохраняются данные блока). Перед ключевым словом добавляется пробел.

Ключевые слова, используемые в функции `rdsWriteLineStyleText`, совместимы с функцией `rdsReadLineStyleText` (стр. 298) и текстовым форматом схем РДС.

Пример:

Вызов функции

```
rdsWriteLineStyleText(PS_DASHDOT, FALSE);
```

добавит к текстовым данным блока пробел и слово “dashdot”.

См. также:

`RDS_BFM_SAVETXT` (стр. 54), `RDS_BFM_LOADTXT` (стр. 52),
`rdsReadLineStyleText` (стр. 298), `rdsXGSetPenStyle` (стр. 382).

A.5.10.10. `rdsWriteWordDoubleText` – запись вещественного числа в текстовом формате

Функция `rdsWriteWordDoubleText` добавляет к данным, сохраняемым блоком в текстовом формате, значение указанного в параметрах вещественного числа с предваряющим его, если это необходимо, ключевым словом.

```
void RDSCALL rdsWriteWordDoubleText(  
    LPSTR Keyword,          // Ключевое слово  
    double Value            // Число  
);
```

Тип указателя на эту функцию:

`RDS_VSD`

Параметры:

Keyword

Указатель на строку с ключевым словом, или `NULL`, если ключевое слово добавлять не нужно.

Value

Число, значение которого записывается (при записи отбрасываются незначащие нули).

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на события записи данных блока в текстовом формате RDS_BFM_SAVETXT (стр. 54), во всех остальных случаях вызов игнорируется. Она добавляет преобразованное в текст вещественное число (с ключевым словом или без него) к общему набору текстовых данных блока. После завершения реакции на событие RDS_BFM_SAVETXT это набор будет записан в файл или буфер обмена (в зависимости от того, для чего сохраняются данные блока). Перед текстом всегда добавляется пробел.

Если в параметре Value передано специальное значение-индикатор ошибки (см. функцию rdsGetHugeDouble, стр. 157), вместо значения числа будет записан вопросительный знак.

Пример использования функции rdsWriteWordDoubleText приведен в §2.8.4.

Пример:

Вызов функции

```
rdsWriteWordDoubleText("scale",100.10);
```

добавит к текстовым данным блока пробел и следующий текст:

```
scale 100.1
```

См. также:

RDS_BFM_SAVETXT (стр. 54), RDS_BFM_LOADTXT (стр. 52), rdsDtoA (стр. 185), rdsAtoD (стр. 184), rdsGetHugeDouble (стр. 157).

A.5.10.11. rdsWriteWordStringText – запись строки в текстовом формате

Функция rdsWriteWordStringText добавляет к данным, сохраняемым блоком в текстовом формате, указанную строку текста, заменяя в ней непечатаемые символы на их обозначения и заключая ее в кавычки. Перед строкой, если необходимо, добавляется ключевое слово.

```
void RDSCALL rdsWriteWordStringText(  
    LPSTR Keyword,          // Ключевое слово  
    LPSTR String            // Строка  
);
```

Тип указателя на эту функцию:

RDS_VSS

Параметры:

Keyword

Указатель на строку с ключевым словом, или NULL, если ключевое слово добавлять не нужно.

String

Указатель на записываемую строку.

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на события записи данных блока в текстовом формате RDS_BFM_SAVETXT (стр. 54), во всех остальных случаях вызов игнорируется. Она добавляет преобразованную для записи строку (с ключевым словом или без него) к общему набору текстовых данных блока. После завершения реакции на событие RDS_BFM_SAVETXT этот набор будет записан в файл или

буфер обмена (в зависимости от того, для чего сохраняются данные блока). Перед строкой и ключевым словом всегда добавляется пробел.

Преобразование строки, переданной в параметре `String`, в вид, пригодный для записи в текстовые данные блока, осуществляется точно так же, как и в функции `rdsProcessText` с параметром `RDS_PT_TEXTTOSTRING` (стр. 193). При чтении текстовых данных блока функцией `rdsGetTextWord` (стр. 291) строки в кавычках автоматически преобразуются обратно в исходный вид (все обозначения непечатаемых символов заменяются на их коды).

Пример:

Вызов функции

```
rdsWriteWordStringText("remark", "Строка1\nСтрока2");
```

добавит к текстовым данным блока пробел и следующий текст:

```
remark "Строка1\nСтрока2"
```

См. также:

`RDS_BFM_SAVETXT` (стр. 54), `RDS_BFM_LOADTXT` (стр. 52),
`rdsProcessText` (стр. 192), `rdsGetTextWord` (стр. 291).

A.5.10.12. `rdsWriteWordValueText` – запись целого числа в текстовом формате

Функция `rdsWriteWordValueText` добавляет к данным, сохраняемым блоком в текстовом формате, значение указанного в параметрах целого числа с предваряющим его, если это необходимо, ключевым словом.

```
void RDSCALL rdsWriteWordValueText(  
    LPSTR Keyword,          // Ключевое слово  
    int Value               // Число  
);
```

Тип указателя на эту функцию:

`RDS_VSI`

Параметры:

`Keyword`

Указатель на строку с ключевым словом, или `NULL`, если ключевое слово добавлять не нужно.

`Value`

Число, значение которого записывается (всегда используется десятичная система счисления).

Примечания:

Эта функция может вызываться только из функции модели блока в момент реакции на события записи данных блока в текстовом формате `RDS_BFM_SAVETXT` (стр. 54), во всех остальных случаях вызов игнорируется. Она добавляет преобразованное в текст целое число (с ключевым словом или без него) к общему набору текстовых данных блока. После завершения реакции на событие `RDS_BFM_SAVETXT` этот набор будет записан в файл или буфер обмена (в зависимости от того, для чего сохраняются данные блока). Перед числом и ключевым словом всегда добавляется пробел.

Пример использования функции `rdsWriteWordValueText` приведен в §2.8.4.

Пример:

Вызов функции

```
rdsWriteWordValueText("count", 34);
```

добавит к текстовым данным блока пробел и следующий текст:

```
count 34
```

См. также:

RDS_BFM_SAVETXT (стр. 54), RDS_BFM_LOADTXT (стр. 52), rdsItoA (стр. 191), rdsAtoI (стр. 184).

A.5.11. Разбор текста

Описываются функции, служащие для разбора текста и поиска в нем ключевых слов. Для этой цели могут также использоваться вспомогательные объекты, описанные в A.5.26, A.5.27 и A.5.32.

A.5.11.1. rdsFontTextToStruct – разбор описания шрифта

Функция `rdsFontTextToStruct` считывает из переданной в параметрах строки описание шрифта и записывает его в указанную структуру.

```
BOOL RDSCALL rdsFontTextToStruct(  
    LPSTR Start,                // Начало текста  
    LPSTR *pNextWord,           // Возврат – конец описания  
    RDS_PSERVFONTPARAMS pStr    // Заполняемая структура  
);
```

Тип указателя на эту функцию:

RDS_BSpSpFs

Параметры:

Start

Указатель на начало текста описания шрифта. Пробелы и табуляции в начале текста будут пропущены.

pNextWord

Указатель на переменную типа `char*`, в которую будет записан указатель на начало первого слова строки `Start`, которое не относится к описанию шрифта. Если вызывающей программе не нужно знать, где кончается описание шрифта, в этом параметре можно передать `NULL`.

pStr

Указатель на структуру описания шрифта `RDS_SERVFONTPARAMS` (стр. 135) в которую функция запишет считанные из строки значения.

Возвращаемое значение:

`TRUE`, если в описании шрифта не обнаружено ошибок, `FALSE` в противном случае.

Примечания:

Эта функция двигается по строке, начиная с указателя `Start`, считывает из нее ключевые слова описания шрифта (те же, что используются в функции `rdsWriteFontText`, стр. 282) и следующие за ними значения параметров, записывая при этом эти значения в структуру по указателю `pStr`. Как только функция встретит в строке ключевое слово, не относящееся к словам описания шрифта, она запишет указатель на начало этого слова в переменную по указателю `pNextWord` и завершит работу.

Чаще всего `rdsFontTextToStruct` используется как обратная функция к `rdsStructToFontText` (стр. 279) для преобразования строки обратно в структуру описания шрифта. Функция автоматически обрабатывает знак продолжения “+”, используемый в текстовом формате схем РДС – обнаружив его, она пропускает следующий за ним перевод строки и повторение знака продолжения, считая следующую строку текста продолжением текущей строки.

Пример использования функции приведен в §2.10.1.

См. также:

`RDS_SERVFONTPARAMS` (стр. 135), `rdsStructToFontText` (стр. 279),
`rdsWriteFontText` (стр. 282), `rdsReadFontText` (стр. 295).

A.5.11.2. `rdsGetTextWord` – извлечение слова из текста

Функция `rdsGetTextWord` считывает из переданной в параметрах строки очередное слово и возвращает указатель на это слово во внутреннем буфере РДС.

```
LPSTR RDSCALL rdsGetTextWord(
    LPSTR Start,           // Начало текста
    LPSTR *pNextWord,      // Возврат – следующее слово
    char *pSym,           // Вид слова
    BOOL LowerCase         // Перевести в нижний регистр
);
```

Тип указателя на эту функцию:

`RDS_SSpSpCB`

Параметры:

`Start`

Указатель на начало текста, из которого нужно извлечь слово. Все пробелы и табуляции перед словом будут пропущены.

`pNextWord`

Указатель на переменную типа `char*`, в которую будет записан указатель на начало следующего слова. Если вызывающей программе оно не нужно, в этом параметре можно передать `NULL`.

`pSym`

Указатель на переменную типа `char`, в которую будет записан вид считанного слова, или `NULL`, если вызывающей программе не нужен вид слова. От вида слова зависит его предварительная обработка внутри функции, от него также может зависеть реакция на это слово вызывавшей функцию программы. По указателю `pSym` может быть записан один из следующих символов:

<i>*pSym (код)</i>	<i>Значение</i>
0	Считан конец текста, в нем больше нет слов. Функция при этом вернет пустую строку.
10 (“\n”)	Считан конец строки, следующее слово находится на следующей строке. Функция при этом вернет слово, состоящее из единственного символа “\n”.
Кавычка (“\ ”)	Считанное слово на самом деле является строкой в двойных кавычках (возможно, с пробелами), в которой непечатаемые символы заменены на их символические обозначения (см. стр. 193). Функция при этом вернет строку без кавычек, в которой

<i>*pSym (код)</i>	<i>Значение</i>
	символические обозначения заменены на сами символы, в т.ч. и непечатаемые.
Другие коды	Считано слово, состоящее из слитно набранных печатаемых символов. Функция вернет это слово, а в *pSym будет записан первый символ этого слова.

LowerCase

TRUE – считанное слово нужно перевести в нижний регистр (за исключением считанных строк в кавычках – их регистр никогда не изменяется). FALSE – считанное слово возвращается как есть.

Возвращаемое значение:

Указатель на внутренний буфер РДС, в котором размещено считанное слово. Функция никогда не возвращает NULL – даже если слова нет (текст закончился), она вернет указатель на буфер с пустой строкой.

Примечания:

Обычно эта функция используется для разбиения текста на слова (например, в процессе загрузки данных блока в текстовом формате) с целью последующего анализа этих слов. Она извлекает из текста, указатель на который передан в параметре Start, первое слово, записывает его во внутренний буфер, записывает в *pNextWord указатель на начало следующего слова (этот указатель можно передать в параметре Start при следующем вызове этой функции для чтения очередного слова), в *pSym – тип считанного слова как указано выше, и возвращает указатель на внутренний буфер со считанным словом. Словом считается либо последовательность любых печатаемых символов, либо строка в двойных кавычках, разделителями слов – пробелы и табуляции. Код перевода строки “\n” считается отдельным словом из одного символа. Нулевой байт, завершающий текст, тоже считается отдельным словом (пустой строкой). Если в параметре LowerCase передано TRUE, считанное слово, если оно не является строкой в кавычках, будет переведено в нижний регистр.

Функция отдельно обрабатывает знак продолжения “+”, используемый в текстовом формате схем РДС. Обнаружив его, она пропускает следующий за ним перевод строки и повторение знака продолжения, считая следующую строку текста продолжением текущей строки и не возвращая отдельно слово конца строки “\n”. Таким образом, например, текст

```
word1 word2 +
+ word3 "word 4"
```

будет считаться одной строкой, состоящей из четырех слов: “word1”, “word2”, “word3” и “word 4”.

Строка в кавычках считается одним словом, какой бы длинной она ни была и сколько бы пробелов внутри ни содержала. В приведенном выше примере “word 4” будет считано функцией как одно слово, а не как два. Длинные строки тоже могут быть разбиты с помощью символов продолжения “+”, при этом функция автоматически соберет их вместе. Например, текст

```
word5 "abcd"+
+"efgh" word6
```

будет разобран на три слова: “word5”, “abcdefgh” и “word6”.

Следует помнить, что извлеченное из текста слово хранится во внутреннем буфере РДС только до тех пор, пока функция не будет вызвана в следующий раз, поэтому анализировать это слово нужно до следующего вызова rdsGetTextWord. Если из текста

нужно извлечь сразу несколько слов, можно воспользоваться похожей функцией `rdsGetTextWordDyn` (стр. 293), которая работает с текстом точно так же, как `rdsGetTextWord`, и имеет те же самые параметры, но вместо того, чтобы возвращать указатель на слово во внутреннем буфере, создает и возвращает динамическую строку с этим словом.

Использование функции подробно рассматривается в §2.8.3.

См. также:

`rdsGetTextWordDyn` (стр. 293), `RDS_BFM_LOADTXT` (стр. 52).

A.5.11.3. `rdsGetTextWordDyn` – извлечение слова из текста

Функция `rdsGetTextWordDyn` считывает из переданной в параметрах строки очередное слово и возвращает его в виде динамически созданной строки.

```
LPSTR RDCALL rdsGetTextWordDyn(  
    LPSTR Start,           // Начало текста  
    LPSTR *pNextWord,      // Возврат – следующее слово  
    char *pSym,           // Вид слова  
    BOOL LowerCase        // Перевести в нижний регистр  
);
```

Тип указателя на эту функцию:

`RDS_SSpSpCB`

Параметры:

`Start`

Указатель на начало текста, из которого нужно извлечь слово. Все пробелы и табуляции перед словом будут пропущены.

`pNextWord`

Указатель на переменную типа `char*`, в которую будет записан указатель на начало следующего слова. Если вызывающей программе оно не нужно, в этом параметре можно передать `NULL`.

`pSym`

Указатель на переменную типа `char`, в которую будет записан вид считанного слова (см. стр. 291). `NULL`, если вызывающей программе не нужен вид слова.

`LowerCase`

`TRUE` – считанное слово нужно перевести в нижний регистр (за исключением считанных строк в кавычках – их регистр никогда не изменяется). `FALSE` – считанное слово возвращается как есть.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, содержащую считанное слово, или `NULL`, если эта строка пустая (текст закончился).

Примечания:

Функция `rdsGetTextWordDyn` полностью аналогична функции `rdsGetTextWord` (стр. 291), за исключением того, что последняя возвращает указатель на считанное слово во внутреннем буфере РДС, а эта функция формирует из считанного слова динамическую строку, которая **обязательно** должна быть освобождена вызовом `rdsFree` (стр. 187). Принципы разбора текста и смысл параметров у этих двух функций полностью совпадают.

`rdsGetTextWordDyn` обычно используется в тех случаях, когда необходимо сначала выделить из текста несколько слов, а потом уже их анализировать. `rdsGetTextWord` для этого не годится, поскольку каждый ее вызов приводит к изменению строки во внутреннем буфере и, следовательно, к потере предыдущего выделенного из текста слова.

См. также:

`rdsGetTextWord` (стр. 291), `rdsFree` (стр. 187), `RDS_BFM_LOADTXT` (стр. 52).

A.5.11.4. `rdsReadColorText` – разбор описания цвета

Функция `rdsReadColorText` считывает из переданной в параметрах строки описание цвета и возвращает считанное значение типа `COLORREF`.

```
COLORREF RDSCALL rdsReadColorText(  
    LPSTR Start,           // Начало текста  
    LPSTR *pNextWord      // Возврат – следующее слово  
);
```

Тип указателя на эту функцию:

`RDS_CrSpS`

Параметры:

`Start`

Указатель на начало текста, из которого нужно извлечь описание цвета. Все пробелы и табуляции перед описанием будут пропущены.

`pNextWord`

Указатель на переменную типа `char*`, в которую будет записан указатель на начало следующего после описания цвета слова. Если вызывающей программе оно не нужно, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Считанное из текста значение цвета в виде целого числа (тип `COLORREF`, см. стр. 24). Если текст не соответствует формату описания цвета, возвращается нулевое значение (оно соответствует черному цвету).

Примечания:

Эта функция разбирает текстовое описание цвета в том же формате, который используется в функции `rdsWriteColorText` (стр. 281). Сначала она извлекает из текста, указатель на который передан в параметре `Start`, первое слово. Если это слово “rgb”, она считывает следующие за ним три слова, преобразует их в целые числа тем же способом, который используется в функции `rdsAtOI` (стр. 184), и собирает из этих трех чисел значение цвета, считая их интенсивностями его красного, зеленого и синего компонентов (каждое из трех целых чисел может принимать значения в интервале 0...255). В противном случае, то есть если первое слово в тексте по указателю `Start` – не “rgb”, функция преобразует это слово в целое число и считает его значением цвета, которое нужно вернуть. По указателю `pNextWord` функция записывает указатель на следующее после описания цвета слово, то есть указатель на следующее слово после считанного числа, если цвет был представлен числом, или указатель на слово, следующее за значением синего компонента, если цвет описывается словом “rgb” и тремя интенсивностями компонентов.

Функция `rdsReadColorText` автоматически обрабатывает знак продолжения “+”, используемый в текстовом формате схем РДС – обнаружив его, она пропускает следующий

за ним перевод строки и повторение знака продолжения, считая следующую строку текста продолжением текущей строки.

См. также:

`rdsWriteColorText` (стр. 281), `rdsAtoI` (стр. 184), `RDS_BFM_LOADTXT` (стр. 52).

A.5.11.5. `rdsReadFontText` – разбор описания шрифта

Функция `rdsReadFontText` считывает из переданной в параметрах строки описание шрифта и записывает его в переменные, указатели на которые переданы в других параметрах функции.

```
BOOL RDSCALL rdsReadFontText(  
    LPSTR Start,           // Начало текста  
    LPSTR *pNextWord,     // Возврат – конец описания  
    LPSTR NameBuf,        // Указатель на массив для возврата  
                           // имени шрифта  
    int NameBufLen,       // Макс длина имени  
    int *pHeight,         // Возврат – высота шрифта  
    COLORREF *pColor,    // Возврат – цвет шрифта  
    int *pCharSet,       // Возврат – набор символов  
    int *pEscapement,     // Возврат – угол  
    BOOL *pBold,         // Возврат – жирность  
    BOOL *pItalic,       // Возврат – курсив  
    BOOL *pUnderline,    // Возврат – подчеркивание  
    BOOL *pStrikeOut     // Возврат – зачеркивание  
);
```

Тип указателя на эту функцию:

`RDS_BSpSSIpIpCpIpIpBpBpBpB`

Параметры:

`Start`

Указатель на начало текста описания шрифта. Пробелы и табуляции в начале текста будут пропущены.

`pNextWord`

Указатель на переменную типа `char*`, в которую будет записан указатель на начало первого слова строки `Start`, которое не относится к описанию шрифта. Если вызывающей программе не нужно знать, где кончается описание шрифта, в этом параметре можно передать `NULL`.

`NameBuf`

Указатель на массив символов, в который функция запишет имя шрифта. Может равняться `NULL`, если имя шрифта не нужно вызвавшей программе.

`NameBufLen`

Максимально допустимая длина имени шрифта (размер массива `NameBuf` минус один). При `NameBuf==NULL` игнорируется.

`pHeight`

Указатель на целую переменную, в которую функция запишет высоту шрифта в точках экрана, или `NULL`, если это значение не нужно вызвавшей программе.

`pColor`

Указатель на переменную типа `COLORREF` (см. стр. 24), в которую функция запишет цвет шрифта, или `NULL`, если это значение не нужно вызвавшей программе.

pCharset

Указатель на целую переменную, в которую функция запишет набор символов шрифта, или NULL, если это значение не нужно вызвавшей программе.

pEscapement

Указатель на целую переменную, в которую функция запишет угол поворота шрифта в градусах относительно горизонтали, или NULL, если это значение не нужно вызвавшей программе.

pBold

Указатель на логическую переменную, в которую функция запишет TRUE, если шрифт жирный, или FALSE в противном случае. Может равняться NULL, если этот параметр не нужен вызвавшей программе.

pItalic

Указатель на логическую переменную, в которую функция запишет TRUE, если шрифт курсивный, или FALSE в противном случае. Может равняться NULL, если этот параметр не нужен вызвавшей программе.

pUnderline

Указатель на логическую переменную, в которую функция запишет TRUE, если шрифт подчеркнутый, или FALSE в противном случае. Может равняться NULL, если этот параметр не нужен вызвавшей программе.

pStrikeOut

Указатель на логическую переменную, в которую функция запишет TRUE, если шрифт перечеркнутый, или FALSE в противном случае. Может равняться NULL, если этот параметр не нужен вызвавшей программе.

Возвращаемое значение:

TRUE, если в описании шрифта не обнаружено ошибок, FALSE в противном случае.

Примечания:

Эта функция двигается по строке, начиная с указателя Start, считывает из нее ключевые слова описания шрифта (те же, что используются в функции rdsWriteFontText, стр. 282) и следующие за ними значения, записывая при этом эти значения по указателям, переданным в ее параметрах. Как только функция встретит в строке ключевое слово, не относящееся к словам описания шрифта, она запишет указатель на начало этого слова в переменную по указателю pNextWord и завершит работу. Функция работает так же, как rdsFontTextToStruct (стр. 290), но записывает считанные значения не в структуру, а в переменные вызвавшей программы. В отличие от rdsFontTextToStruct она всегда возвращает высоту шрифта в точках экрана, даже если в разбираемом тексте высота указана в типографских точках (преобразование в точки экрана в этом случае выполняется автоматически).

Функция автоматически обрабатывает знак продолжения “+”, используемый в текстовом формате схем РДС – обнаружив его, она пропускает следующий за ним перевод строки и повторение знака продолжения, считая следующую строку текста продолжением текущей строки.

См. также:

rdsFontTextToStruct (стр. 290), rdsStructToFontText (стр. 279),
rdsWriteFontText (стр. 282).

A.5.11.6. rdsReadHexText – разбор шестнадцатеричного блока текста

Функция `rdsReadHexText` считывает шестнадцатеричный блок из переданного в параметрах текста и, преобразовав каждую пару шестнадцатеричных цифр в этом блоке в один байт, записывает эти байты последовательно в указанный буфер.

```
int RDSCALL rdsReadHexText(  
    LPSTR Start,           // Начало текста  
    LPSTR *pNextWord,      // Возврат – конец описания  
    LPVOID Buffer,         // Буфер для записи  
    int MaxSize            // Размер буфера  
);
```

Тип указателя на эту функцию:

`RDS_ISpSpVI`

Параметры:

`Start`

Указатель на начало текста. Пробелы и табуляции в начале текста будут пропущены.

`pNextWord`

Указатель на переменную типа `char*`, в которую будет записан указатель на начало следующего за шестнадцатеричным блоком слова строки `Start`. Если вызывающей программе не нужно знать начало следующего слова, в этом параметре можно передать `NULL`.

`Buffer`

Указатель общего вида (`void*`) на заполняемый функцией буфер.

`MaxSize`

Размер буфера, переданного в параметре `Buffer` (функция не станет писать в него больше `MaxSize` байтов).

Возвращаемое значение:

Число байтов, фактически записанное в буфер.

Примечания:

Эта функция извлекает из строки `Start` блок шестнадцатеричных данных и записывает его побайтно в буфер `Buffer`. Блок данных может быть заключен в кавычки, в этом случае функция будет автоматически обрабатывать знак продолжения “+”, используемый в текстовом формате схем РДС – обнаружив его, она пропустит следующий за ним перевод строки и повторение знака продолжения, считая следующую строку текста продолжением текущей строки. Обычно функцией `rdsReadHexText` обрабатывают текст, записанный функцией `rdsWriteHexText` (стр. 285) при сохранении данных блока – форматы данных этих двух функций полностью совместимы.

Функция запишет в буфер `Buffer` не более `MaxSize` байтов. Если шестнадцатеричный блок окажется длиннее (то есть будет содержать больше `2xMaxSize` шестнадцатеричных цифр), лишние байты в конце блока будут потеряны. Функция пишет данные в буфер побайтно, то есть в результате выполнения программы

```
char *text=" \"ff2e1f\"";  
BYTE Buffer[3];  
rdsReadHexText(text, NULL, Buffer, 3);
```

в `Buffer[0]` будет записано значение 255 (0xff), в `Buffer[1]` – 46 (0x2e), а в `Buffer[2]` – 31 (0x1f).

См. также:

`rdsWriteHexText` (стр. 285).

A.5.11.7. `rdsReadLineStyleText` – разбор стиля линии

Функция `rdsReadLineStyleText` считывает слово из переданного в параметрах текста и возвращает соответствующую ему константу стиля линии.

```
int RDSCALL rdsReadLineStyleText(  
    LPSTR Start,           // Начало текста  
    LPSTR *pNextWord,     // Возврат – конец описания  
);
```

Тип указателя на эту функцию:

`RDS_ISpS`

Параметры:

`Start`

Указатель на начало текста. Пробелы и табуляции в начале текста будут пропущены.

`pNextWord`

Указатель на переменную типа `char*`, в которую будет записан указатель на начало следующего слова строки `Start`. Если вызывающей программе не нужно знать начало следующего слова, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Константа стиля линии Windows API, соответствующая считанному ключевому слову.

Примечания:

Эта функция извлекает слово из строки `Start` и преобразует его в стандартную константу стиля линии, используемую в Windows API. Она использует те же ключевые слова, что и функция `rdsWriteLineStyleText` (стр. 286). Если слово, считанное функцией, не совпадает ни с одним из ключевых слов стиля линий, функция возвращает `PS_SOLID` (сплошная линия). Если считано слово конца строки (“\n”, см. описание функции `rdsGetTextWord` на стр. 291) или в тексте нет ни одного слова, функция возвращает `PS_NULL` (нет линии).

См. также:

`rdsWriteLineStyleText` (стр. 286), `rdsGetTextWord` (стр. 291).

A.5.12. Таймеры блоков

Описываются функции создания, программирования и удаления таймеров блоков (см. §2.9).

A.5.12.1. `rdsDeleteBlockTimer` – удалить таймер

Функция `rdsDeleteBlockTimer` удаляет таймер вызвавшего блока с указанным идентификатором.

```
void RDSCALL rdsDeleteBlockTimer(  
    RDS_TIMERID Timer      // Идентификатор таймера  
);
```

Тип указателя на эту функцию:

`RDS_VTh`

Параметр:

Timer

Идентификатор удаляемого таймера.

Примечания:

Эта функция удаляет таймер с идентификатором `Timer` в блоке, из модели которого она вызвана. Если этот таймер принадлежит другому блоку, он не будет удален (то есть нельзя удалить таймер, не принадлежащий блоку, модель которого в данный момент выполняется). После удаления таймера модель блока не будет больше вызываться для реакции на события, связанные с этим таймером.

При отключении модели от блока (например, из-за удаления блока, или подключения к нему другой модели) все таймеры этого блока удаляются автоматически.

Пример использования функции `rdsDeleteBlockTimer` приведен в §2.9.2.

См. также:

`rdsSetBlockTimer` (стр. 300), `rdsStopBlockTimer` (стр. 302),
`RDS_BFM_TIMER` (стр. 47).

A.5.12.2. `rdsGetBlockTimerDescr` – получить описание таймера

Функция `rdsGetBlockTimerDescr` заполняет структуру описания указанного таймера.

```
BOOL RDSCALL rdsGetBlockTimerDescr(  
    RDS_TIMERID Timer           // Идентификатор таймера  
    RDS_PTIMERDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

`RDS_BThTd`

Параметры:

Timer

Идентификатор таймера, описание которого нужно получить.

pDescr

Указатель на структуру описания таймера `RDS_TIMERDESCRIPTION` (стр. 136), которую функция должна заполнить.

Возвращаемое значение:

`TRUE` – структура заполнена, `FALSE` – ошибка.

Примечания:

Функция заполняет структуру `RDS_TIMERDESCRIPTION` описанием таймера `Timer`. Пример использования функции `rdsGetBlockTimerDescr` приведен в §2.9.2.

См. также:

`RDS_TIMERDESCRIPTION` (стр. 136), `rdsSetBlockTimer` (стр. 300).

A.5.12.3. `rdsRestartBlockTimer` – перезапустить таймер

Функция `rdsRestartBlockTimer` перезапускает таймер с указанным идентификатором.

```

void RDSCALL rdsRestartBlockTimer(
    RDS_TIMERID Timer      // Идентификатор таймера
    DWORD Delay           // Новый интервал
);

```

Тип указателя на эту функцию:

RDS_VThDw

Параметры:

Timer

Идентификатор перезапускаемого таймера.

Delay

Нулевое значение в этом параметре продолжает работу остановленного таймера, ненулевое – задает ему новый интервал срабатывания в Delay миллисекунд и начинает отсчет заново.

Примечания:

При Delay, равном нулю, эта функция продолжает работу ранее остановленного таймера Timer. При ненулевом значении Delay функция устанавливает таймеру Timer новый интервал срабатывания и запускает его отсчет с самого начала, независимо от того, работал ли он на момент вызова функции или был остановлен.

При отключении модели от блока (например, из-за удаления блока, или подключения к нему другой модели) все таймеры этого блока удаляются автоматически.

Пример использования функции rdsRestartBlockTimer приведен в §2.9.2.

См. также:

rdsSetBlockTimer (стр. 300), rdsStopBlockTimer (стр. 302),
RDS_BFM_TIMER (стр. 47).

A.5.12.4. rdsSetBlockTimer – создать таймер

Функция rdsSetBlockTimer создает новый таймер с указанными параметрами или изменяет параметры существующего.

```

RDS_TIMERID RDSCALL rdsSetBlockTimer(
    RDS_TIMERID Timer,      // Идентификатор существующего таймера
    DWORD Delay,           // Интервал срабатывания, мс
    DWORD Mode,            // Режим работы
    BOOL Start             // Запустить немедленно
);

```

Тип указателя на эту функцию:

RDS_ThThDwDwB

Параметры:

Timer

NULL, если нужно создать новый таймер, или идентификатор таймера, параметры которого нужно изменить.

Delay

Интервал срабатывания таймера в миллисекундах.

Mode

Режим работы таймера (аналогичен одноименному полю структуры RDS_TIMERDESCRIPTION, см. стр. 136). Формируется как объединение битовым

ИЛИ константы режима (RDS_TIMERM_*), константы способа срабатывания (RDS_TIMERS_*) и флагов режима (RDS_TIMERF_*). Константа режима может быть одной из следующих:

RDS_TIMERM_LOOP	Таймер циклический, после срабатывания он автоматически перезапускается с той же задержкой.
RDS_TIMERM_STOP	Таймер однократный, после срабатывания таймера он будет остановлен. Его можно перезапустить повторным вызовом <code>rdsSetBlockTimer</code> или функцией <code>rdsRestartBlockTimer</code> (стр. 299).
RDS_TIMERM_DELETE	Таймер однократный, после срабатывания таймера он будет автоматически удален.

Константа способа срабатывания может быть одной из следующих:

RDS_TIMERS_SIGNAL	При срабатывании таймера у блока-владельца взводится сигнал запуска, то есть его первой сигнальной переменной присваивается значение 1. Таймер работает только в режиме расчета.
RDS_TIMERS_TIMER	При срабатывании таймера модель блока-владельца вызывается в потоке расчета для реакции на событие RDS_BFM_TIMER (см. стр. 47). Таймер работает только в режиме расчета.
RDS_TIMERS_WINREF	При срабатывании таймера модель блока-владельца вызывается в главном потоке для реакции на событие RDS_BFM_WINREFRESH (см. стр. 77). Таймер работает только в режиме расчета.
RDS_TIMERS_SYSTIMER	При срабатывании таймера модель блока-владельца вызывается в главном потоке для реакции на событие RDS_BFM_TIMER (см. стр. 47). Таймер работает во всех режимах.

На данный момент поддерживается единственный флаг режима:

RDS_TIMERF_FIXFREQ	При способе срабатывания RDS_TIMERS_WINREF к этому таймеру не будет применяться автоматическое снижение частоты при чрезмерной загрузке процессора процедурами обновления окон. При остальных способах срабатывания флаг игнорируется.
--------------------	--

Start

TRUE – запустить таймер немедленно после создания или изменения параметров,
FALSE – не запускать таймер (он будет находиться в остановленном состоянии).

Возвращаемое значение:

Идентификатор созданного таймера. Если функция была вызвана не для создания таймера, а для изменения параметров существующего (параметр `Timer` не равен NULL), возвращаемое значение будет совпадать с `Timer`.

Примечания:

Эта функция создает новый таймер в блоке, модель которого в данный момент выполняется, или изменяет параметры ранее созданного таймера в этом блоке. Параметры функции определяют интервал срабатывания таймера, реакцию блока на его срабатывание, автоматический перезапуск таймера и т.п.

Пример использования функции `rdsSetBlockTimer` приведен в §2.9.2.

См. также:

RDS_TIMERDESCRIPTION (стр. 136), rdsDeleteBlockTimer (стр. 298),
RDS_BFM_TIMER (стр. 47).

A.5.12.5. rdsStopBlockTimer – остановить таймер

Функция rdsStopBlockTimer останавливает указанный таймер, не удаляя его.

```
void RDSCALL rdsStopBlockTimer(  
    RDS_TIMERID Timer    // Идентификатор таймера  
);
```

Тип указателя на эту функцию:

RDS_VTh

Параметр:

Timer

Идентификатор останавливаемого таймера.

Примечания:

Эта функция останавливает таймер с идентификатором Timer. После его остановки модель блока не будет больше вызываться для реакции на события, связанные с этим таймером, до тех пор, пока таймер не будет запущен заново. Функция rdsStopBlockTimer только останавливает таймер, но не удаляет его – если таймер больше не нужен, его можно удалить вызовом rdsDeleteBlockTimer (стр. 298).

Пример использования функции rdsStopBlockTimer приведен в §2.9.2.

См. также:

rdsSetBlockTimer (стр. 300), rdsRestartBlockTimer (стр. 299),
rdsDeleteBlockTimer (стр. 298).

A.5.13. Вызов функций блоков

Описываются функции и макросы, предназначенные для вызова моделей блоков из моделей других блоков (см. §2.13).

A.5.13.1. Макрос RDS_FUNCPARAMCAST – приведение параметра функции к нужному типу

Макрос RDS_FUNCPARAMCAST предназначен для приведения поля Data структуры параметров функций RDS_FUNCTIONCALldata (см. стр. 35), которое является указателем общего вида, к указанному в параметре типу.

```
RDS_FUNCPARAMCAST(  
    pFuncData,    // Указатель на RDS_FUNCTIONCALldata  
    Type          // Тип, к указателю на который  
                  // приводится поле Data  
)
```

Определение:

```
#define RDS_FUNCPARAMCAST(pFuncData, Type) \  
    ((Type*) ((RDS_PFUNCTIONCALldata) (pFuncData)) ->Data))
```

Параметры:

pFuncData

Указатель на структуру данных вызываемой функции RDS_FUNCTIONCALldata. Этот параметр не обязательно должен иметь тип "RDS_FUNCTIONCALldata*", он может быть и указателем произвольного типа (void*), поскольку внутри макроса он явно приводится к нужному типу. В нем, например, можно передать значение третьего параметра (ExtParam) функции модели блока при реакции на событие RDS_BFM_FUNCTIONCALL (стр. 35) – несмотря на то, что он имеет тип void*, в нем всегда передается указатель на структуру данных функции.

Type

Имя типа, к указателю на который будет приведено поле Data указанной в первом параметре структуры.

Возвращаемое значение:

Значение поля Data структуры данных функции, приведенное к указанному типу.

Примечания:

В структуре данных функции RDS_FUNCTIONCALldata поле Data, являющееся указателем на параметры функции, для универсальности имеет тип "указатель общего вида", то есть LPVOID. Макрос RDS_FUNCPARAMCAST позволяет в некоторых случаях упростить приведение этого указателя к нужному типу. На практике, в большинстве случаев, в его применении нет острой необходимости – все приведения типов всегда можно записать вручную.

Пример:

Допустим, в глобальной переменной FuncId находится идентификатор какой-либо функции (см. rdsRegisterFunction, стр. 316), параметром которой является указатель на некую структуру TMyStruct:

```
typedef struct
{
    DWORD servSize;      // Размер структуры для проверки
    int IntField;
    double DoubleField;
} TMyStruct;
```

Вызов этой функции может выглядеть, например, так:

```
TMyStruct str;
str.servSize=sizeof(str);
str.IntField=1;
RDS_BHANDLE Block=... // Идентификатор вызываемого блока
rdsCallBlockFunction(Block,FuncId,&str);
```

Реакция на вызов этой функции в модели блока может выглядеть так:

```
// Модель блока
extern "C" __declspec(dllexport) int RDSCALL Model(
    int CallMode,           // Событие
    RDS_PBLOCKDATA BlockData, // Данные блока
    LPVOID ExtParam)        // Дополнительные параметры
{
    switch(CallMode)
    {
        ...
        case RDS_BFM_FUNCTIONCALL: // Вызов функции
```

```

        if ((RDS_PFUNCTIONCALldata) ExtParam) ->Function == FuncId)
        { if (RDS_FUNCPARAMCAST (ExtParam, TMyStruct) ->servSize ==
                                sizeof (TMyStruct))
            { // Выполнение функции
              RDS_FUNCPARAMCAST (ExtParam, TMyStruct) ->DoubleField =
              RDS_FUNCPARAMCAST (ExtParam, TMyStruct) ->IntField*2;
            }
        }
        break;
        ...
    }
}

```

См. также:

RDS_FUNCTIONCALldata (стр. 35).

A.5.13.2. Макрос RDS_FUNCPARAMPVOID – поле параметра функции

Макрос RDS_FUNCPARAMPVOID предназначен для доступа к полю Data структуры параметров функции RDS_FUNCTIONCALldata (см. стр. 35), если указатель на эту структуру имеет неправильный тип.

```

RDS_FUNCPARAMPVOID (
    pFuncData          // Указатель на RDS_FUNCTIONCALldata
)

```

Определение:

```

#define RDS_FUNCPARAMPVOID (pFuncData) \
    ((RDS_PFUNCTIONCALldata) (pFuncData)) ->Data

```

Параметр:

pFuncData

Указатель на структуру данных функции RDS_FUNCTIONCALldata, имеющий неправильный тип.

Возвращаемое значение:

Значение поля Data структуры данных функции, находящейся по переданному указателю.

Примечания:

Этот макрос используется достаточно редко. Единственное его назначение – несколько сократить запись приведения третьего параметра (ExtParam) функции модели блока при реакции на событие RDS_BFM_FUNCTIONCALL (стр. 35), в котором передается указатель на структуру данных функции RDS_FUNCTIONCALldata, к нужному типу. В отличие от макроса RDS_FUNCPARAMCAST (стр. 302), значение поля Data, возвращаемое этим макросом, не приводится ни к какому типу и остается указателем общего вида (void*). На практике, в большинстве случаев, в применении этого макроса нет острой необходимости – все приведения типов всегда можно записать вручную.

См. также:

RDS_FUNCTIONCALldata (стр. 35), RDS_FUNCPARAMCAST (стр. 302).

A.5.13.3. Макрос RDS_FUNCPROVIDERLINK_SUCCESS – проверка успешности подписки на исполнителя функции

Макрос RDS_FUNCPROVIDERLINK_SUCCESS предназначен для проверки успешности подписки на идентификатор блока-исполнителя функции (регистрация блоков-исполнителей и подписка на них подробно рассмотрены в §2.13.6).

```
RDS_FUNCPROVIDERLINK_SUCCESS(  
    pLink           // Указатель на RDS_FUNCPROVIDERLINK  
)
```

Определение:

```
#define RDS_FUNCPROVIDERLINK_SUCCESS(pLink) \  
    ( ((pLink)!=NULL) && \  
      ((RDS_PFUNCPROVIDERLINK)(pLink))->Block!=NULL) )
```

Параметр:

pLink

Указатель на структуру подписки на идентификатор блока-исполнителя RDS_FUNCPROVIDERLINK (стр. 129).

Возвращаемое значение:

TRUE – блок-исполнитель найден, FALSE – подписка невозможна или в схеме нет блока-исполнителя данной функции.

Примечания:

В параметре макроса передается указатель на структуру RDS_FUNCPROVIDERLINK, созданную функцией rdsSubscribeToFuncProvider (стр. 317). Значение, возвращаемое макросом, указывает на возможность вызова блока-исполнителя. На практике, в большинстве случаев, в применении этого макроса нет острой необходимости – эту проверку можно выполнить вручную.

См. также:

RDS_FUNCPROVIDERLINK (стр. 129), rdsSubscribeToFuncProvider (стр. 317).

A.5.13.4. rdsBroadcastFuncCallsDelayed – отложенный вызов функции всех блоков подсистемы

Функция rdsBroadcastFuncCallsDelayed ставит указанную функцию в очередь на выполнение для всех блоков указанной подсистемы. После этого управление немедленно возвращается модели вызвавшего блока, а модели блоков указанной подсистемы будут вызваны только после завершения этой модели.

```
void RDSCALL rdsBroadcastFuncCallsDelayed(  
    RDS_BHANDLE Parent,    // Подсистема с вызываемыми блоками  
    int FuncId,            // Идентификатор функции  
    LPVOID pParamBuf,      // Указатель на область  
                          // параметров функции  
    DWORD ParamBufSize,    // Размер области параметров  
    DWORD Flags            // Флаги (RDS_BCALL_*)  
) ;
```

Тип указателя на эту функцию:

RDS_VBhIpVDwDw

Параметры:

Parent

Идентификатор подсистемы, у блоков которой будет вызываться функция (модели этих блоков будут реагировать на событие RDS_BFM_FUNCTIONCALL, см. стр. 35).

FuncId

Целый идентификатор вызываемой функции, полученный при ее регистрации вызовом rdsRegisterFunction (стр. 316).

pParamBuf

Указатель на область параметров функции, которая будет скопирована во внутреннюю память РДС. Область параметров не должна содержать внутри себя какие-либо указатели на другие области памяти и объекты. Может равняться NULL.

ParamBufSize

Размер (в байтах) области параметров, указатель на которую передан в pParams.

Flags

Один или несколько объединенных битовым ИЛИ флагов, управляющих вызовом функции:

RDS_BCALL_ALLOWSTOP	Разрешить вызываемым моделям блоков прекращать дальнейшие вызовы присвоением значения TRUE полю Stop структуры RDS_FUNCTIONCALLDATA (стр. 35).
RDS_BCALL_CHECKSUPPORT	Перед вызовом проверить поддержку этой функции блоком, вызвав его модель для реакции на событие RDS_BFM_CHECKFUNCSUPPORT (стр. 32). В настоящее время этот режим практически не используется.
RDS_BCALL_SUBSYSTEMS	Вызывать функции у всех блоков внутри подсистемы Parent на всех уровнях иерархии (то есть не только у блоков и подсистем, непосредственно находящихся в Parent, но и у всех внутренних блоков этих подсистем).
RDS_BCALL_FIRST	Поставить вызов функции блоков в начало очереди (не может использоваться одновременно с RDS_BCALL_LAST).
RDS_BCALL_LAST	Поставить вызов функции блоков в конец очереди (не может использоваться одновременно с RDS_BCALL_FIRST).

Примечания:

Вызов этой функции приводит к тому, что после завершения модели вызвавшего блока все модели блоков в подсистеме Parent будут последовательно (в произвольном порядке) вызываться для реакции на событие RDS_BFM_FUNCTIONCALL. Поля структуры данных функции RDS_FUNCTIONCALLDATA, указатель на которую передается в модель блока в параметре ExtParam, при этом будут заполнены следующим образом:

Поле	Значение
int Function	значение параметра FuncId
LPVOID Data	указатель на внутреннюю область памяти РДС размером в ParamBufSize байтов, в которую скопирована область pParamBuf

<i>Поле</i>	<i>Значение</i>
RDS_BHANDLE Caller	идентификатор блока, из модели которого вызвана rdsBroadcastFuncCallsDelayed
BOOL Broadcast	TRUE
int BroadcastCnt	исходно – 0, увеличивается на 1 при вызове каждого следующего блока
BOOL Stop	исходно – FALSE, вызванная модель может присвоить TRUE, что прекратит вызов других моделей (при указании флага RDS_BCALL_ALLOWSTOP)
BOOL Delayed	TRUE
DWORD DataBufSize	значение параметра ParamBufSize

Модели блоков, находящихся в Parent, будут вызываться до тех пор, пока не будут вызваны все, или пока какая-либо модель не присвоит полю Stop структуры RDS_FUNCTIONCALldata значение TRUE.

Отложенный вызов функций блоков рассматривается в §2.13.5.

См. также:

RDS_BFM_FUNCTIONCALL, RDS_FUNCTIONCALldata (стр. 35),
rdsBroadcastFunctionCallsEx (стр. 308),
rdsQueueCallBlockFunction (стр. 313), rdsRegisterFunction (стр. 316).

A.5.13.5. rdsBroadcastFunctionCalls – прямой вызов функции всех блоков подсистемы (устаревшая)

Функция rdsBroadcastFunctionCalls вызывает одну и ту же функцию у всех блоков указанной подсистемы (с блоками вложенных подсистем или без них). Управление возвращается модели вызвавшего блока только после вызова всех функций. Сейчас вместо rdsBroadcastFunctionCalls чаще используется rdsBroadcastFunctionCallsEx (стр. 308), обладающая более широкими возможностями.

```
void RDSCALL rdsBroadcastFunctionCalls(
    RDS_BHANDLE Parent,    // Подсистема с вызываемыми блоками
    int FuncId,            // Идентификатор функции
    LPVOID pParams,        // Указатель на область
                           // параметров функции
    BOOL CallSubsystems    // Вызывать блоки вложенных подсистем
);
```

Тип указателя на эту функцию:

RDS_VBhIpVB

Параметры:

Parent

Идентификатор подсистемы, у блоков которой будет вызываться функция (модели этих блоков будут реагировать на событие RDS_BFM_FUNCTIONCALL, см. стр. 35).

FuncId

Целый идентификатор вызываемой функции, полученный при ее регистрации вызовом rdsRegisterFunction (стр. 316).

pParams

Указатель на область параметров функции (этот указатель передается модели вызванного блока). Может равняться NULL.

CallSubsystems

TRUE, если функция FuncId должна быть вызвана у всех блоков всех внутренних подсистем подсистемы Parent на всех уровнях иерархии. FALSE, если функцию FuncId нужно вызвать только у блоков, непосредственно находящихся в подсистеме Parent (разумеется, включая подсистемы непосредственно вложенные в нее, но не включая содержимое этих подсистем).

Примечания:

Вызов этой функции приводит к последовательному (в произвольном порядке) вызову всех моделей блоков в подсистеме Parent для реакции на событие RDS_BFM_FUNCTIONCALL. Поля структуры данных функции RDS_FUNCTIONCALLDATA (стр. 35), указатель на которую передается в модель блока в параметре ExtParam, при этом будут заполнены следующим образом:

<i>Поле</i>	<i>Значение</i>
int Function	значение параметра FuncId
LPVOID Data	значение параметра pParams
RDS_BHANDLE Caller	идентификатор блока, из модели которого вызвана rdsBroadcastFunctionCalls
BOOL Broadcast	TRUE
int BroadcastCnt	исходно – 0, увеличивается на 1 при вызове каждого следующего блока
BOOL Stop	FALSE
BOOL Delayed	FALSE
DWORD DataBufSize	0

Вызов функций всех блоков подсистемы рассматривается в §2.13.3.

См. также:

RDS_BFM_FUNCTIONCALL, RDS_FUNCTIONCALLDATA (стр. 35),
rdsBroadcastFunctionCallsEx (стр. 308), rdsRegisterFunction (стр. 316).

A.5.13.6. rdsBroadcastFunctionCallsEx – прямой вызов функции всех блоков подсистемы

Функция rdsBroadcastFunctionCallsEx вызывает одну и ту же функцию у всех блоков указанной подсистемы (с блоками вложенных подсистем или без них). Управление возвращается модели вызвавшего блока только после вызова всех функций.

```
int RDSCALL rdsBroadcastFunctionCallsEx(  
    RDS_BHANDLE Parent,    // Подсистема с вызываемыми блоками  
    int FuncId,            // Идентификатор функции  
    LPVOID pParams,       // Указатель на область  
                           // параметров функции  
    DWORD Flags           // Флаги (RDS_BCALL_*)  
);
```

Тип указателя на эту функцию:

RDS_IBhIpVDw

Параметры:

Parent

Идентификатор подсистемы, у блоков которой будет вызываться функция (модели этих блоков будут реагировать на событие RDS_BFM_FUNCTIONCALL, см. стр. 35).

FuncId

Целый идентификатор вызываемой функции, полученный при ее регистрации вызовом rdsRegisterFunction (стр. 316).

pParams

Указатель на область параметров функции (этот указатель передается модели вызванного блока). Может равняться NULL.

Flags

Один или несколько объединенных битовым ИЛИ флагов, управляющих вызовом функции:

RDS_BCALL_ALLOWSTOP Разрешить вызываемым моделям блоков прекращать дальнейшие вызовы присвоением значения TRUE полю Stop структуры RDS_FUNCTIONCALldata (стр. 35).

RDS_BCALL_CHECKSUPPORT Перед вызовом проверить поддержку этой функции блоком, вызвав его модель для реакции на событие RDS_BFM_CHECKFUNCSUPPORT (стр. 32). В настоящее время этот режим практически не используется.

RDS_BCALL_SUBSYSTEMS Вызывать функции у всех блоков внутри подсистемы Parent на всех уровнях иерархии (то есть не только у блоков и подсистем, непосредственно находящихся в Parent, но и у всех внутренних блоков этих подсистем).

Возвращаемое значение:

Общее число блоков, модели которых были вызваны в процессе выполнения функции.

Примечания:

Вызов этой функции приводит к последовательному (в произвольном порядке) вызову всех моделей блоков в подсистеме Parent для реакции на событие RDS_BFM_FUNCTIONCALL. Поля структуры данных функции RDS_FUNCTIONCALldata, указатель на которую передается в модель блока в параметре ExtParam, при этом будут заполнены следующим образом:

<i>Поле</i>	<i>Значение</i>
int Function	значение параметра FuncId
LPVOID Data	значение параметра pParams
RDS_BHANDLE Caller	идентификатор блока, из модели которого вызвана rdsBroadcastFunctionCallsEx
BOOL Broadcast	TRUE

<i>Поле</i>	<i>Значение</i>
int BroadcastCnt	исходно – 0, увеличивается на 1 при вызове каждого следующего блока
BOOL Stop	исходно – FALSE, вызванная модель может присвоить TRUE, что прекратит вызов других моделей (при указании флага RDS_BCALL_ALLOWSTOP)
BOOL Delayed	FALSE
DWORD DataBufSize	0

Модели блоков, находящихся в Parent, будут вызываться до тех пор, пока не будут вызваны все, или пока какая-либо модель не присвоит полю Stop структуры RDS_FUNCTIONCALldata значение TRUE.

Пример использования функции rdsBroadcastFunctionCallsEx приведен в §2.13.3.

См. также:

RDS_BFM_FUNCTIONCALL, RDS_FUNCTIONCALldata (стр. 35),
rdsBroadcastFuncCallsDelayed (стр. 305),
rdsRegisterFunction (стр. 316).

A.5.13.7. rdsCallBlockFunction – прямой вызов функции блока

Функция rdsCallBlockFunction вызывает указанную функцию указанного блока и не возвращает управление вызвавшей программе, пока модель вызванного блока не завершится.

```
int RDSCALL rdsCallBlockFunction(
    RDS_BHANDLE Block,      // Вызываемый блок
    int FuncId,             // Идентификатор функции
    LPVOID pParams          // Указатель на параметры функции
);
```

Тип указателя на эту функцию:

RDS_IBhIPv

Параметры:

Block

Идентификатор блока, функция которого вызывается (модель этого блока будет реагировать на событие RDS_BFM_FUNCTIONCALL, см. стр. 35).

FuncId

Целый идентификатор вызываемой функции, полученный при ее регистрации вызовом rdsRegisterFunction (стр. 316).

pParams

Указатель на параметры функции (этот указатель передается модели вызванного блока). Может равняться NULL.

Возвращаемое значение:

Целое число, возвращенное функцией модели вызванного блока.

Примечания:

Вызов этой функции приводит к немедленному вызову модели блока Block для реакции на событие RDS_BFM_FUNCTIONCALL, при этом поля структуры данных функции RDS_FUNCTIONCALLDATA (стр. 35), указатель на которую передается в модель блока в параметре ExtParam, будут заполнены следующим образом:

<i>Поле</i>	<i>Значение</i>
int Function	значение параметра FuncId
LPVOID Data	значение параметра pParams
RDS_BHANDLE Caller	идентификатор блока, из модели которого вызвана rdsCallBlockFunction
BOOL Broadcast	FALSE
int BroadcastCnt	0
BOOL Stop	FALSE
BOOL Delayed	FALSE
DWORD DataBufSize	0

Пример использования функции rdsCallBlockFunction приведен в §2.13.2.

См. также:

RDS_BFM_FUNCTIONCALL, RDS_FUNCTIONCALLDATA (стр. 35),
rdsQueueCallBlockFunction (стр. 313),
rdsBroadcastFunctionCallsEx (стр. 308), rdsRegisterFunction (стр. 316).

A.5.13.8. rdsCallBlockFunctionDelayed – отложенный вызов функции блока (устаревшая)

Функция rdsCallBlockFunctionDelayed ставит указанную функцию указанного блока в конец очереди на выполнение. После этого управление немедленно возвращается модели вызвавшего функцию блока, а поставленная в очередь функция будет вызвана только после завершения этой модели. Сейчас вместо rdsCallBlockFunctionDelayed чаще используется rdsQueueCallBlockFunction (стр. 313), обладающая более широкими возможностями.

```
void RDSCALL rdsCallBlockFunctionDelayed(  
    RDS_BHANDLE Block,      // Вызываемый блок  
    int FuncId,             // Идентификатор функции  
    LPVOID pParamBuf,       // Указатель на область  
                                // параметров функции  
    DWORD ParamBufSize      // Размер области параметров  
);
```

Тип указателя на эту функцию:

RDS_VBhIpVDw

Параметры:

Block

Идентификатор блока, функция которого будет вызвана (модель этого блока будет реагировать на событие RDS_BFM_FUNCTIONCALL, см. стр. 35).

FuncId

Целый идентификатор вызываемой функции, полученный при ее регистрации вызовом `rdsRegisterFunction` (стр. 316).

pParamBuf

Указатель на область параметров функции, которая будет скопирована во внутреннюю память РДС. Область параметров не должна содержать внутри себя какие-либо указатели на другие области памяти и объекты. Может равняться NULL.

ParamBufSize

Размер (в байтах) области параметров, указатель на которую передан в `pParams`.

Примечания:

Вызов этой функции приводит к постановке вызова модели блока `Block` для реакции на событие `RDS_BFM_FUNCTIONCALL` в конец специальной очереди. Вызовы из этой очереди будут выполнены после завершения модели вызвавшего функцию блока. Когда модель блока `Block` будет вызвана в режиме `RDS_BFM_FUNCTIONCALL`, поля структуры данных функции `RDS_FUNCTIONCALLDATA` (стр. 35), указатель на которую передается в модель блока в параметре `ExtParam`, будут заполнены следующим образом:

<i>Поле</i>	<i>Значение</i>
int Function	значение параметра FuncId
LPVOID Data	указатель на внутреннюю область памяти РДС размером в ParamBufSize байтов, в которую скопирована область pParamBuf
RDS_BHANDLE Caller	идентификатор блока, из модели которого вызвана <code>rdsCallBlockFunctionDelayed</code>
BOOL Broadcast	FALSE
int BroadcastCnt	0
BOOL Stop	FALSE
BOOL Delayed	TRUE
DWORD DataBufSize	значение параметра ParamBufSize

В отличие от прямого вызова функции блока, при отложенном вызове модели вызванного блока передается не указатель на область параметров функции, а указатель на копию этой области, сделанную РДС (именно поэтому при отложенном вызове всегда указывается размер области параметров и требуется, чтобы эта область не содержала указателей). Поскольку вызов функции будет выполнен уже после завершения модели вызвавшего блока, оригинальная область параметров на момент вызова может быть уже уничтожена, но сделанная РДС копия останется в памяти до фактического вызова функции блока.

Отложенный вызов функций блоков рассматривается в §2.13.5.

См. также:

`RDS_BFM_FUNCTIONCALL`, `RDS_FUNCTIONCALLDATA` (стр. 35),
`rdsQueueCallBlockFunction` (стр. 313), `rdsCallBlockFunction` (стр. 310),
`rdsBroadcastFunctionCallsEx` (стр. 308), `rdsRegisterFunction` (стр. 316).

A.5.13.9. rdsCheckBlockFunctionSupport – проверка поддержки функции блока

Функция `rdsCheckBlockFunctionSupport` проверяет, поддерживает ли указанный блок указанную функцию, вызывая его модель для реакции на событие `RDS_BFM_CHECKFUNCSUPPORT` (стр. 32). Сейчас эта функция практически не используется, поскольку все модели блоков пишутся таким образом, чтобы их можно было безопасно вызывать для функций, которые не поддерживаются этими блоками.

```
BOOL RDSCALL rdsCheckBlockFunctionSupport(  
    RDS_BHANDLE Block,      // Вызываемый блок  
    int FuncId              // Идентификатор функции  
);
```

Тип указателя на эту функцию:

`RDS_BBhI`

Параметры:

`Block`

Идентификатор блока, для которого проверяется поддержка функции (модель этого блока будет реагировать на событие `RDS_BFM_CHECKFUNCSUPPORT`).

`FuncId`

Целый идентификатор проверяемой функции, полученный при ее регистрации вызовом `rdsRegisterFunction` (стр. 316).

Возвращаемое значение:

`TRUE`, если функция `FuncId` поддерживается блоком `Block` (его модель вернула ненулевое целое число), или `FALSE` в противном случае.

См. также:

`RDS_BFM_CHECKFUNCSUPPORT` (стр. 32), `rdsRegisterFunction` (стр. 316).

A.5.13.10. rdsQueueCallBlockFunction – отложенный вызов функции блока

Функция `rdsQueueCallBlockFunction` ставит указанную функцию указанного блока в очередь на выполнение. После этого управление немедленно возвращается модели вызвавшего функцию блока, а поставленная в очередь функция будет вызвана только после завершения этой модели.

```
void RDSCALL rdsQueueCallBlockFunction(  
    RDS_BHANDLE Block,      // Вызываемый блок  
    int FuncId,             // Идентификатор функции  
    LPVOID pParamBuf,      // Указатель на область  
                           // параметров функции  
    DWORD ParamBufSize,    // Размер области параметров  
    DWORD Flags             // Флаги (RDS_BCALL_*)  
);
```

Тип указателя на эту функцию:

`RDS_VBhIpVDwDw`

Параметры:

Block

Идентификатор блока, функция которого будет вызвана (модель этого блока будет реагировать на событие RDS_BFM_FUNCTIONCALL, см. стр. 35).

FuncId

Целый идентификатор вызываемой функции, полученный при ее регистрации вызовом rdsRegisterFunction (стр. 316).

pParamBuf

Указатель на область параметров функции, которая будет скопирована во внутреннюю память РДС. Область параметров не должна содержать внутри себя какие-либо указатели на другие области памяти и объекты. Может равняться NULL.

ParamBufSize

Размер (в байтах) области параметров, указатель на которую передан в pParams.

Flags

Один или несколько объединенных битовым ИЛИ флагов, управляющих вызовом функции:

RDS_BCALL_CHECKSUPPORT Перед вызовом проверить поддержку этой функции блоком, вызвав его модель для реакции на событие RDS_BFM_CHECKFUNCSUPPORT (стр. 32). В настоящее время этот режим практически не используется.

RDS_BCALL_FIRST Поставить вызов функции блока в начало очереди (не может использоваться одновременно с RDS_BCALL_LAST).

RDS_BCALL_LAST Поставить вызов функции блока в конец очереди (не может использоваться одновременно с RDS_BCALL_FIRST).

Примечания:

Вызов этой функции приводит к постановке вызова модели блока Block для реакции на событие RDS_BFM_FUNCTIONCALL в начало или конец (в зависимости от флагов) специальной очереди. Вызовы функций из этой очереди будут выполнены последовательно после завершения модели вызвавшего функцию блока. Когда модель блока Block будет вызвана в режиме RDS_BFM_FUNCTIONCALL, поля структуры данных функции RDS_FUNCTIONCALLDATA (стр. 35), указатель на которую передается в модель блока в параметре ExtParam, будут заполнены следующим образом:

<i>Поле</i>	<i>Значение</i>
int Function	значение параметра FuncId
LPVOID Data	указатель на внутреннюю область памяти РДС размером в ParamBufSize байт, в которую скопирована область pParamBuf
RDS_BHANDLE Caller	идентификатор блока, из модели которого вызвана rdsQueueCallBlockFunction
BOOL Broadcast	FALSE
int BroadcastCnt	0
BOOL Stop	FALSE

<i>Поле</i>	<i>Значение</i>
BOOL Delayed	TRUE
DWORD DataBufSize	значение параметра ParamBufSize

В отличие от прямого вызова функции блока, при отложенном вызове модели вызванного блока передается не указатель на область параметров функции, а указатель на копию этой области, сделанную РДС (именно поэтому при отложенном вызове всегда указывается размер области параметров и требуется, чтобы эта область не содержала указателей). Поскольку вызов функции будет выполнен уже после завершения модели вызвавшего блока, оригинальная область параметров на момент вызова может быть уже уничтожена, но сделанная РДС копия останется в памяти до фактического вызова функции блока.

Пример использования функции `rdsQueueCallBlockFunction` приведен в §2.13.5.

См. также:

`RDS_BFM_FUNCTIONCALL`, `RDS_FUNCTIONCALLDATA` (стр. 35),
`rdsCallBlockFunction` (стр. 310),
`rdsCallBlockFunctionDelayed` (стр. 311),
`rdsBroadcastFunctionCallsEx` (стр. 308), `rdsRegisterFunction` (стр. 316).

A.5.13.11. `rdsRegisterFuncProvider` – регистрация блока как исполнителя функции

Функция `rdsRegisterFuncProvider` регистрирует вызвавший ее блок в качестве исполнителя функции с указанным идентификатором. После этого другие блоки схемы смогут легко найти этот блок.

```
int RDSCALL rdsRegisterFuncProvider(
    int FuncId,           // Идентификатор функции
    BOOL ChildrenOnly     // Только для вложенных блоков
);
```

Тип указателя на эту функцию:

`RDS_IIB`

Параметры:

`FuncId`

Целый идентификатор функции, полученный при ее регистрации вызовом `rdsRegisterFunction` (стр. 316).

`ChildrenOnly`

Этот параметр используется только тогда, когда функция `rdsRegisterFuncProvider` вызывается из модели подсистемы, в противном случае он игнорируется. Если в этом параметре передано значение `TRUE`, подсистема будет зарегистрирована как исполнитель функции только для блоков, находящихся внутри нее (на всех уровнях вложенности). Если в параметре передано `FALSE`, подсистема будет зарегистрирована как исполнитель функции не только для своих внутренних блоков, но и для всех блоков и подсистем, находящихся в одной с ней подсистеме (“соседей” по подсистеме) и для всех их внутренних блоков.

Возвращаемое значение:

Ненулевое значение при успешной регистрации, ноль при ошибке.

Примечания:

Вызов этой функции регистрирует вызвавший блок в качестве исполнителя указанной функции для всех блоков, находящихся в одной с ним подсистеме и во всех вложенных подсистемах на всех уровнях иерархии (за исключением случая, когда `rdsRegisterFuncProvider` вызвана из модели подсистемы и в параметре `ChildrenOnly` передано `TRUE` – в этом случае доступ к исполнителю получают только внутренние блоки этой подсистемы). Любой блок может подписаться на идентификатор зарегистрированного исполнителя функции вызовом `rdsSubscribeToFuncProvider` (стр. 317).

Пример использования функции `rdsRegisterFuncProvider` приведен в §2.13.6.

См. также:

`rdsUnregisterFuncProvider` (стр. 317),
`rdsSubscribeToFuncProvider` (стр. 317).

А.5.13.12. `rdsRegisterFunction` – регистрация функции блока

Функция `rdsRegisterFunction` регистрирует в РДС функцию с указанным именем, присваивая ей уникальный целый идентификатор.

```
int RDSCALL rdsRegisterFunction(  
    LPSTR FuncName           // Имя функции  
);
```

Тип указателя на эту функцию:

`RDS_IS`

Параметр:

`FuncName`

Указатель на строку с именем регистрируемой функции.

Возвращаемое значение:

Уникальный целый идентификатор, присвоенный функции.

Примечания:

Вызов `rdsRegisterFunction` регистрирует функцию с указанным именем в РДС: если эта функция регистрируется в первый раз с момента загрузки схемы в память или создания новой схемы, имя функции заносится во внутреннюю таблицу и ей присваивается уникальный целый идентификатор, который будет использоваться для вызова этой функции и для опознания ее в модели блока при реакции на событие `RDS_BFM_FUNCTIONCALL` (стр. 35). Если функция с этим именем уже регистрировалась, `rdsRegisterFunction` возвратит идентификатор, присвоенный этой функции при первой регистрации. Таким образом, регистрацию функции можно безопасно проводить несколько раз подряд – ее идентификатор от этого не изменится. Следует помнить, что при выгрузке схемы из памяти внутренняя таблица функций РДС очищается, поэтому при следующей загрузке той же самой схемы функции блоков могут получить другие идентификаторы.

Поскольку функцию блока достаточно зарегистрировать один раз за все время нахождения схемы в памяти, все вызовы `rdsRegisterFunction` часто выносят в главную функцию DLL (стр. 27), но можно регистрировать функции и при инициализации моделей блоков. РДС никогда не дает регистрируемой функции нулевой идентификатор, поэтому значение 0 в переменной, в которой будет храниться идентификатор функции, можно использовать как признак того, что эта функция еще не регистрировалась.

Пример использования функции `rdsRegisterFunction` приведен в §2.13.1.

См. также:

`RDS_BFM_FUNCTIONCALL` (стр. 35), главная функция DLL (стр. 27),
`RDS_BFM_INIT` (стр. 38).

A.5.13.13. `rdsSubscribeToFuncProvider` – подписка на блок-исполнитель функции

Функция `rdsSubscribeToFuncProvider` создает во внутренних данных блока, из модели которого она вызвана, специальную структуру, в которую РДС будет записывать идентификатор блока, зарегистрировавшегося как исполнитель указанной функции.

```
RDS_FUNCPROVIDERLINK RDCALL rdsSubscribeToFuncProvider(  
    int FuncId           // Идентификатор функции  
);
```

Тип указателя на эту функцию:

`RDS_FLI`

Параметр:

`FuncId`

Целый идентификатор функции, полученный при ее регистрации вызовом `rdsRegisterFunction` (стр. 316).

Возвращаемое значение:

Указатель на созданную во внутренней памяти РДС структуру `RDS_FUNCPROVIDERLINK` (стр. 129).

Примечания:

Вызов этой функции запрашивает у РДС подписку блока, из модели которого вызвана `rdsSubscribeToFuncProvider`, на исполнителя указанной функции. В результате вызова РДС создает структуру `RDS_FUNCPROVIDERLINK` и начинает отслеживать наличие в схеме блока-исполнителя функции `FuncId`, записывая его идентификатор в поле `Block` этой структуры. Если в схеме нет такого блока, в поле `Block` будет записано значение `NULL`. При любых изменениях в регистрации исполнителя этой функции (появление исполнителя ближе по иерархии подсистем, отмена регистрации и т.п.) РДС будет автоматически обновлять данные в структуре `RDS_FUNCPROVIDERLINK`. Эта структура будет уничтожена при отмене подписки вызовом `rdsUnsubscribeFromFuncProvider` (стр. 318) или при удалении подписавшегося блока.

Пример использования функции `rdsSubscribeToFuncProvider` приведен в §2.13.6.

См. также:

`RDS_FUNCPROVIDERLINK` (стр. 129), `rdsRegisterFuncProvider` (стр. 315),
`rdsUnsubscribeFromFuncProvider` (стр. 318).

A.5.13.14. `rdsUnregisterFuncProvider` – отмена регистрации блока как исполнителя функции

Функция `rdsUnregisterFuncProvider` отменяет регистрацию блока, из модели которого она вызвана, в качестве исполнителя функции с указанным идентификатором.

```

BOOL RDSCALL rdsUnregisterFuncProvider(
    int FuncId           // Идентификатор функции
);

```

Тип указателя на эту функцию:

RDS_BI

Параметр:

FuncId

Целый идентификатор функции, полученный при ее регистрации вызовом `rdsRegisterFunction` (стр. 316).

Возвращаемое значение:

TRUE – регистрация отменена, FALSE – ошибка (блок не был исполнителем функции).

Примечания:

Вызов этой функции отменяет ранее выполненную вызовом `rdsRegisterFuncProvider` (стр. 315) регистрацию вызвавшего блока в качестве исполнителя указанной функции. После этого РДС автоматически обновит поле `Block` во всех структурах `RDS_FUNCPROVIDERLINK` (стр. 129), созданных для подписавшихся блоков, записав туда идентификатор другого блока-исполнителя (или NULL, если исполнителей не осталось).

Пример использования функции `rdsUnregisterFuncProvider` приведен в §2.13.6.

См. также:

`rdsRegisterFuncProvider` (стр. 315),
`rdsSubscribeToFuncProvider` (стр. 317).

A.5.13.15. `rdsUnsubscribeFromFuncProvider` – отмена подписки на блок-исполнитель функции

Функция `rdsUnsubscribeFromFuncProvider` отменяет подписку блока, из модели которого она вызвана, на идентификатор блока-исполнителя указанной функции.

```

void RDSCALL rdsUnsubscribeFromFuncProvider(
    int FuncId           // Идентификатор функции
);

```

Тип указателя на эту функцию:

RDS_VI

Параметр:

FuncId

Целый идентификатор функции, полученный при ее регистрации вызовом `rdsRegisterFunction` (стр. 316).

Примечания:

Вызов этой функции отменяет ранее выполненную вызовом `rdsSubscribeToFuncProvider` (стр. 317) подписку вызвавшего блока на идентификатор блока-исполнителя функции `FuncId`. Структура `RDS_FUNCPROVIDERLINK` (стр. 129), созданная во внутренней памяти РДС при вызове `rdsSubscribeToFuncProvider`, при этом уничтожается.

Пример использования функции `rdsUnsubscribeFromFuncProvider` приведен в §2.13.6.

См. также:

`rdsRegisterFuncProvider` (стр. 315),
`rdsSubscribeToFuncProvider` (стр. 317), `RDS_FUNCPROVIDERLINK` (стр. 129).

A.5.14. Общие функции работы с переменными блока

Описываются функции, позволяющие работать со значениями и параметрами переменных блока. Специализированные функции для работы с матрицами и массивами описаны в A.5.15, функции работы с динамическими переменными – в A.5.16.

A.5.14.1. `rdsBlockVarFromMem` – считать значение переменной блока из буфера в памяти

Функция `rdsBlockVarFromMem` считывает значения какой-либо статической переменной указанного блока или всей его структуры переменных из области памяти, в которую эти значения были записаны вызовом `rdsBlockVarToMem` (стр. 320).

```
BOOL RDSCALL rdsBlockVarFromMem(  
    RDS_BHANDLE Block,      // Идентификатор блока  
    int VarNum,             // Номер переменной или -1  
    LPVOID Buffer,          // Указатель на начало данных  
    int Size                // Размер данных  
);
```

Тип указателя на эту функцию:

`RDS_BBhIpVI`

Параметры:

`Block`

Идентификатор блока, значение переменной которого нужно загрузить из области памяти `Buffer`. Если в этом параметре передано значение `NULL`, функция будет работать с блоком, из модели которого она вызвана.

`VarNum`

Порядковый номер переменной (начиная с нуля), значение которой нужно загрузить из памяти, или `-1`, если из памяти нужно загрузить значения всех переменных блока.

`Buffer`

Указатель на начало области памяти, в которой находится значение переменной.

`Size`

Размер области памяти `Buffer` в байтах.

Возвращаемое значение:

`TRUE` – значение переменной (переменных) считаны успешно, `FALSE` – в области памяти `Buffer` находятся данные, не соответствующие типу указанной переменной или формату записи функции `rdsBlockVarToMem`.

Примечания:

Эта функция, вместе с `rdsBlockVarToMem`, позволяет сохранять значение переменной блока в последовательную область памяти и восстанавливать его оттуда. Такую область памяти можно скопировать, сохранить на диск или передать по сети с тем, чтобы позже (или в другой схеме) считать эту область и восстановить из нее значение переменной.

Тип переменной с номером VarNum, в которую считывается значение из области Buffer длиной в Size байтов, должен в точности совпадать с типом переменной, значение которой было записано в эту область функцией rdsBlockVarToMem. Если восстанавливается вся структура переменных блока (в VarNum передано -1), вся структура должна соответствовать структуре, записанной в области Buffer.

Пример использования функции rdsBlockVarFromMem приведен в §2.16.1.

См. также:

rdsBlockVarToMem (стр. 320).

A.5.14.2. rdsBlockVarToMem – записать значение переменной блока в буфер в памяти

Функция rdsBlockVarToMem записывает значение какой-либо статической переменной указанного блока или всей его структуры переменных в динамически отводимую область памяти. Это значение может быть загружено обратно в переменную вызовом rdsBlockVarFromMem (стр. 319).

```
LPVOID RDSCALL rdsBlockVarToMem(  
    RDS_BHANDLE Block,      // Идентификатор блока  
    int VarNum,             // Номер переменной или -1  
    BOOL Check,             // Записать данные типа переменной  
    int *pSize              // Возвращаемый размер данных  
);
```

Тип указателя на эту функцию:

RDS_pVBhIBpI

Параметры:

Block

Идентификатор блока, значение переменной которого нужно записать. Если в этом параметре передано значение NULL, функция будет работать с блоком, из модели которого она вызвана.

VarNum

Порядковый номер переменной (начиная с нуля), значение которой нужно записать, или -1, если нужно записать значения всех переменных блока.

Check

TRUE – записывать вместе со значением переменной информацию о ее типе для проверки при загрузке функцией rdsBlockVarFromMem. FALSE – не записывать информацию о типе (данные займут меньше места, но надежность программы снизится).

pSize

Указатель на целую переменную, в которую функция запишет размер (в байтах) динамически отведенной области памяти.

Возвращаемое значение:

Указатель на динамически отведенную область памяти, содержащую записанное значение переменной (или переменных, если в VarNum передано -1).

Примечания:

Эта функция, вместе с rdsBlockVarFromMem, позволяет сохранять значение переменной блока в память и восстанавливать его оттуда. Такую область памяти можно

скопировать, сохранить на диск или передать по сети с тем, чтобы позже (или в другой схеме) считать эту область и восстановить из нее значение переменной. Динамическая область памяти, созданная этой функцией, должна быть **обязательно** освобождена вызовом `rdsFree` (стр. 187).

Пример использования функции `rdsBlockVarToMem` приведен в §2.16.1.

См. также:

`rdsBlockVarFromMem` (стр. 319), `rdsFree` (стр. 187).

A.5.14.3. `rdsClearRuntimeType` – очистка переменной произвольного типа

Функция `rdsClearRuntimeType` очищает переменную произвольного типа с указанным базовым адресом.

```
void RDSCALL rdsClearRuntimeType(  
    LPVOID BaseAddr           // Базовый адрес переменной  
);
```

Тип указателя на эту функцию:

`RDS_VpV`

Параметр:

`BaseAddr`

Базовый адрес переменной в дереве, то есть указатель на блок из восьми байтов, которые эта переменная занимает в дереве переменных блока (см. §2.5.6).

Примечания:

Вызов этой функции присваивает указанной переменной пустой фактический тип.

Пример:

Модель блока-переключателя (мультиплексора), который подает на выход `y` произвольного типа элемент входного массива переменных произвольного типа `X` с номером, определяемым целым входом `N`. Если `N` меньше нуля или больше числа элементов в массиве `X`, выход `y` очищается, то есть ему присваивается пустой фактический тип.

Блок должен иметь следующую структуру переменных:

<i>Смещение</i>	<i>Имя</i>	<i>Тип</i>	<i>Размер</i>	<i>Вход/выход</i>
0	Start	Сигнал	1	Вход
1	Ready	Сигнал	1	Выход
2	X	Массив произвольных	8	Вход
10	N	int	4	Вход
14	y	Произвольный	8	Выход

Модель блока:

```
extern "C" __declspec(dllexport) int RDSCALL MultiplexMVariant(  
    int CallMode, RDS_PBLOCKDATA BlockData, LPVOID ExtParam)  
// Макросы для переменных  
#define pStart ((char *) (BlockData->VarData))  
#define pReady ((char *) (pStart+1))  
#define pX ((void **) (pStart+2))  
#define pN ((int *) (pStart+10))
```

```

#define py ((void **)(pStart+14))
{
    switch(CallMode)
    { case RDS_BFM_VARCHHECK:
        if(strcmp((char*)ExtParam,"{SSMVIV}")==0)
            return RDS_BFR_DONE;
        return RDS_BFR_BADVARSMSG;

        case RDS_BFM_MODEL:
            if(RDS_ARRAYEXISTS(pX)) // Массив X не пустой
            { int n=RDS_ARRAYROWS(pX)*RDS_ARRAYCOLS(pX),N=*pN;
              BYTE *data=(BYTE*)RDS_ARRAYDATA(pX); // 1-й элемент
              if(N>=0 && N<n) // Копируем в выход
                  rdsCopyRuntimeType(py,data+N*2*sizeof(void*));
              else
                  rdsClearRuntimeType(py); // Очищаем выход
            }
            else // Нет массива X
                rdsClearRuntimeType(py); // Очищаем выход
            break;
        }
    }
    return RDS_BFR_DONE;
}
#undef py
#undef pN
#undef pX
#undef pReady
#undef pStart
//=====

```

См. также:

rdsCopyRuntimeType (стр. 322), rdsSetRuntimeType (стр. 335),
rdsGetBlockVarBase (стр. 329).

А.5.14.4. rdsCopyRuntimeType – копировать переменную произвольного типа

Функция rdsCopyRuntimeType копирует значение одной переменной произвольного типа в другую. Переменная-получатель данных после этого будет иметь тот же фактический тип, что и переменная-источник.

```

BOOL RDSCALL rdsCopyRuntimeType(
    LPVOID DestBaseAddr, // Базовый адрес получателя
    LPVOID SrcBaseAddr   // Базовый адрес источника
);

```

Тип указателя на эту функцию:

RDS_BpVpV

Параметры:

DestBaseAddr

Базовый адрес переменной-получателя данных в дереве, то есть указатель на область из восьми байтов, которые эта переменная занимает в дереве переменных блока (см. §2.5.6).

SrcBaseAddr

Базовый адрес переменной-источника данных в дереве переменных блока.

Возвращаемое значение:

TRUE – значение переменной скопировано успешно, FALSE – произошла ошибка (недопустимые параметры функции).

Примечания:

Эта функция используется в тех случаях, когда требуется скопировать одну переменную произвольного типа в другую, не разбираясь в ее внутренней структуре и фактическом типе (например, при создании моделей блоков-переключателей, которые, при выполнении каких-либо условий, передают значение своего входа на выход без изменений).

Пример использования функции rdsCopyRuntimeType приведен в §2.5.6.

См. также:

rdsClearRuntimeType (стр. 321), rdsSetRuntimeType (стр. 335),
rdsGetBlockVarBase (стр. 329).

A.5.14.5. rdsCopyVarGeneral – копировать значение переменной в другую переменную

Функция rdsCopyVarGeneral копирует значение одной переменной в другую.

```
BOOL RDSCALL rdsCopyVarGeneral (  
    RDS_VHANDLE DestVar, // Переменная-получатель  
    LPVOID DestBaseAddr, // Базовый адрес получателя  
    RDS_VHANDLE SrcVar, // Переменная-источник  
    LPVOID SrcBaseAddr // Базовый адрес источника  
);
```

Тип указателя на эту функцию:

RDS_BVhpVVhpV

Параметры:

DestVar

Идентификатор переменной-получателя данных (тип RDS_VHANDLE, см. стр. 23).

DestBaseAddr

Базовый адрес переменной-получателя данных в дереве, то есть указатель на начало области, которую эта переменная занимает в дереве переменных блока (см. §2.5.1).

SrcVar

Идентификатор переменной-источника данных.

SrcBaseAddr

Базовый адрес переменной-источника данных в дереве переменных блока.

Возвращаемое значение:

TRUE – значение переменной скопировано успешно, FALSE – произошла ошибка (недопустимые параметры функции).

Примечания:

Эта функция позволяет скопировать значение одной переменной в другую, если для каждой из переменных известен уникальный идентификатор типа RDS_VHANDLE и базовый адрес этой переменной. Идентификатор статической переменной блока можно получить вызовом rdsGetBlockVar (стр. 328) или rdsFindBlockVar (стр. 326), идентификатор

динамической хранится в структуре RDS_DYNVARLINK (стр. 121), создаваемой РДС при подписке на эту динамическую переменную.

Чаще всего модели блоков работают с переменными непосредственно по адресам (см. §2.5.1), и в использовании функции rdsCopyVarGeneral нет необходимости. Она может пригодиться в тех случаях, когда блок имеет изменяемую структуру переменных (например, настраиваемую пользователем) и при этом выполняет над своими переменными однотипные действия (например, копирует значение входа на один из выходов в зависимости от какого-либо условия).

См. также:

rdsCopyVarArray (стр. 342), rdsCopyRuntimeType (стр. 322),
rdsGetBlockVar (стр. 328), rdsFindBlockVar (стр. 326),
RDS_DYNVARLINK (стр. 121), rdsSubscribeToDynamicVar (стр. 353),
rdsCreateAndSubscribeDV (стр. 347), rdsGetBlockVarBase (стр. 329).

A.5.14.6. rdsCreateVarDescriptionString – текстовое описание переменной

Функция rdsCreateVarDescriptionString формирует в памяти динамическую строку с описанием переменной, идентификатор которой передан в ее параметрах.

```
LPSTR RDSCALL rdsCreateVarDescriptionString(  
    RDS_VHANDLE Var,          // Идентификатор переменной  
    BOOL StructFields,        // Описывать поля структуры  
    int Indent,               // Отступ в пробелах  
    int *pLength              // Возвращаемая длина текста  
);
```

Тип указателя на эту функцию:

RDS_SVhBIpI

Параметры:

Var

Идентификатор переменной (тип RDS_VHANDLE, см. стр. 23).

StructFields

Если переменная Var – структура, значение TRUE в этом параметре приведет к тому, что каждое из полей этой структуры в формируемом тексте будет описано с указанием имени, типа и т.д. на отдельной строке (строки разделяются символом перевода строки “\n” с кодом 10). При передаче FALSE в этом параметре для Var будет записано только имя типа структуры, а описания полей будут опущены. Если Var – не структура, этот параметр игнорируется.

Indent

Число пробелов, автоматически добавляемое в формируемый текст описания в начале каждой строки.

pLength

Указатель на целую переменную, в которую функция должна записать длину сформированного текста строки. Если вызывающей программе не нужна длина текста, в этом параметре можно передать NULL.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, в которой сформировано текстовое описание переменной Var. Это описание совместимо с текстовым форматом файлов схем РДС. В случае ошибки возвращается NULL.

Примечания:

Эта функция часто используется для временного хранения описания какой-либо переменной блока или всей структуры его переменных (структура переменных блока тоже является переменной с точки зрения РДС). По этому описанию может быть потом создана реальная переменная при помощи функции rdsVSCreateByDescr (стр. 437). Пример использования функции rdsCreateVarDescriptionString приведен в §2.16.1 и §4.3.

Динамическая строка, созданная функцией rdsCreateVarDescriptionString, должна быть **обязательно** освобождена функцией rdsFree (стр. 187).

См. также:

rdsVSCreateByDescr (стр. 437), rdsFree (стр. 187).

A.5.14.7. rdsCreateVarTypeText – название типа переменной

Функция rdsCreateVarTypeText формирует в памяти динамическую строку с названием типа переменной, которое можно показывать пользователю.

```
LPSTR RDSCALL rdsCreateVarTypeText(  
    RDS_VHANDLE Var          // Идентификатор переменной  
);
```

Тип указателя на эту функцию:

RDS_SVh

Параметр:

Var

Идентификатор переменной (тип RDS_VHANDLE, см. стр. 23).

Возвращаемое значение:

Указатель на созданную в динамической памяти строку, в которой сформировано название типа переменной Var. Для описания используются те же слова, что и в функции rdsProcessText с параметром RDS_PT_VARTYPETEXT (стр. 193). В случае ошибки возвращается NULL.

Если переменная Var – матрица, название типа будет состоять из нескольких строк: сначала будет несколько раз (по числу вложенности матриц) повторено слово “Матрица”, а затем – тип элемента. Например:

<i>Тип</i>	<i>Текст описания</i>
Матрица целых чисел (int)	“Матрица\nint”
Матрица вещественных чисел (double)	“Матрица\ndouble”
Матрица структур “Complex”	“Матрица\nComplex”
Матрица матриц целых (int)	“Матрица\nМатрица\nint”

Примечания:

Эта функция чаще всего используется для индикации типа переменной в понятном пользователю виде.

Динамическая строка, созданная функцией `rdsCreateVarTypeText`, должна быть **обязательно** освобождена функцией `rdsFree` (стр. 187).

См. также:

`RDS_PT_VARTYPETEXT` (стр.193), `rdsFree` (стр. 187).

A.5.14.8. `rdsFindBlockVar` – найти переменную блока по имени

Функция `rdsFindBlockVar` ищет статическую переменную с указанным именем в указанном блоке и возвращает ее уникальный идентификатор.

```
RDS_VHANDLE RDSCALL rdsFindBlockVar(  
    RDS_BHANDLE Block,           // Идентификатор блока  
    LPSTR VarName,              // Имя переменной  
    BOOL FullName,              // В имени могут быть поля  
    RDS_PVARDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

`RDS_VhBhSBVd`

Параметры:

`Block`

Идентификатор блока, в котором нужно найти переменную.

`VarName`

Указатель на строку с именем переменной.

`FullName`

`TRUE` – в параметре `VarName` передано имя переменной с возможным указанием полей структур и элементов массивов. `FALSE` – в `VarName` передано имя переменной, непосредственно находящейся в блоке (функция в этом случае выполняется несколько быстрее).

`pDescr`

Указатель на структуру описания переменной `RDS_VARDESCRIPTION` (стр. 138), которую функция должна заполнить параметрами найденной переменной. Если вызывающей программе не нужно описание переменной, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Идентификатор найденной переменной (тип `RDS_VHANDLE`, см. стр. 23) или `NULL`, если переменной с именем `VarName` нет в блоке `Block`.

Примечания:

Эта функция позволяет найти идентификатор статической переменной блока `Block` по имени этой переменной `VarName`. К имени переменной может быть добавлено имя поля, если эта переменная – структура, или индекс элемента, если переменная – массив. В этом случае в параметре `FullName` необходимо передать значение `TRUE`, и функция вернет не идентификатор самой переменной блока, а идентификатор поля структуры или элемента массива, которые, с точки зрения РДС, являются вложенными переменными. Допустим, например, что в блоке `Block` есть следующие переменные:

- вещественная (`double`) переменная `x`;
- структура `in` типа “Complex”, содержащая вещественные поля `Re` и `Im`;
- матрица целых чисел `Y`;

- массив `out` структур “Complex”.

В этом случае вызовы функции `rdsFindBlockVar` будут возвращать следующие идентификаторы:

<i>VarName</i>	<i>Возврат при FullName=TRUE</i>	<i>Возврат при FullName=FALSE</i>
“x”	переменная x	переменная x
“in”	переменная in	переменная in
“out”	переменная out	переменная out
“in.Re”	поле Re структуры in	NULL
“Y[0][1]”	элемент матрицы Y	NULL
“out[3]”	элемент массива out (структура типа “Complex”)	NULL
“out[1].Im”	поле Im структуры, являющейся элементом массива out	NULL

Следует помнить, что все элементы конкретной матрицы (массива) в РДС обслуживаются единственной вложенной переменной с типом элемента этой матрицы – при работе этой вложенной переменной автоматически передаются базовые адреса конкретных элементов. Таким образом, вызов `rdsFindBlockVar` для “out[0]”, “out[1]”, “out[2]” и т.д. будет возвращать один и тот же идентификатор переменной-элемента массива “out”.

См. также:

`rdsGetBlockVar` (стр. 328), `rdsGetVarField` (стр. 332),
`RDS_VARDESCRIPTION` (стр. 138).

A.5.14.9. `rdsFindStructVar` – найти структуру по имени типа

Функция `rdsFindStructVar` ищет структуру с указанным именем типа в общем списке зарегистрированных в схеме структур и возвращает ее уникальный идентификатор.

```
RDS_VHANDLE RDSCALL rdsFindStructVar(
    LPSTR TypeName,           // Имя типа структуры
    RDS_PVARDESCRIPTION pDescr // Заполняемое описание
);
```

Тип указателя на эту функцию:

`RDS_VhSVd`

Параметры:

`TypeName`

Указатель на строку с именем типа структуры.

`pDescr`

Указатель на структуру описания переменной `RDS_VARDESCRIPTION` (стр. 138), которую функция должна заполнить параметрами найденной структуры. Если вызывающей программе не нужно описание, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Идентификатор найденной структуры (тип `RDS_VHANDLE`, см. стр. 23) или `NULL`, если структура с именем типа `TypeName` не зарегистрирована в схеме.

Примечания:

Эта функция позволяет найти идентификатор одной из структур в общем списке структур РДС. Каждая из этих структур представляет собой обычную переменную и, как и переменные блоков, имеет идентификатор RDS_VHANDLE.

См. также:

rdsGetVarField (стр. 332), RDS_VARDESCRIPTION (стр. 138),
rdsGetStructVar (стр. 331).

A.5.14.10. rdsGetBlockVar – переменная блока по номеру

Функция rdsGetBlockVar возвращает идентификатор переменной с заданным номером в указанном блоке или идентификатор всей структуры переменных блока.

```
RDS_VHANDLE RDSCALL rdsGetBlockVar(  
    RDS_BHANDLE Block,           // Идентификатор блока  
    int VarNum,                  // Номер переменной или -1  
    RDS_PVARDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

RDS_VhBhVd

Параметры:

Block

Идентификатор блока, в котором нужно найти переменную. Если в этом параметре передано значение NULL, функция будет работать с блоком, из модели которого она вызвана.

VarNum

Порядковый номер переменной (начиная с нуля) или -1, если нужно получить идентификатор всей структуры переменных блока как единого целого.

pDescr

Указатель на структуру описания переменной RDS_VARDESCRIPTION (стр. 138), которую функция должна заполнить параметрами переменной с указанным номером. Если вызывающей программе не нужно описание переменной, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденной переменной (тип RDS_VHANDLE, см. стр. 23) или NULL, если переменная с номером VarNum отсутствует в блоке Block.

Примечания:

Эта функция позволяет найти идентификатор переменной с заданным номером и заполнить структуру ее описания. Для того, чтобы определить общее число переменных в блоке, следует вызвать эту функцию с параметром VarNum=-1, передав в параметре pDescr указатель на структуру RDS_VARDESCRIPTION, при этом в этой структуре окажется описание всей структуры переменных блока как одной большой переменной, и в ее поле StructFields будет содержаться общее число переменных в этом блоке.

Пример использования функции rdsGetBlockVar приведен в §2.7.4 и §2.16.1.

См. также:

rdsFindBlockVar (стр. 326), rdsGetVarField (стр. 332),
RDS_VARDESCRIPTION (стр. 138).

A.5.14.11. rdsGetBlockVarBase – базовый адрес переменной блока по ее номеру

Функция rdsGetBlockVarBase возвращает указатель на начало данных переменной с указанным номером в дереве переменных блока и размер области данных этой переменной.

```
LPVOID RDSCALL rdsGetBlockVarBase(  
    RDS_BHANDLE Block,      // Идентификатор блока  
    int VarNum,             // Номер переменной  
    int *pSize              // Возвращаемый размер  
);
```

Тип указателя на эту функцию:

RDS_pVBhIpI

Параметры:

Block

Идентификатор блока, в котором нужно найти переменную.

VarNum

Порядковый номер переменной (начиная с нуля).

pSize

Указатель на целую переменную, в которую функция запишет число байтов, которое данные этой переменной занимают в дереве блока. Добавив к возвращенному функцией адресу это число, можно получить начало области данных следующей переменной. Если вызывающей программе не нужен размер области данных переменной, в этом параметре можно передать NULL.

Возвращаемое значение:

Указатель на начало области данных переменной с указанным номером, или NULL, если такой переменной нет.

Примечания:

Чаще всего базовые адреса переменных вычисляются непосредственно в модели блока по значению поля VarData структуры данных блока RDS_BLOCKDATA (стр. 28) – диаграммы размещения переменных в памяти и способ доступа к их данным подробно рассматриваются в §2.5. Функция rdsGetBlockVarBase позволяет получить базовый адрес переменной указанного блока в тех случаях, когда структура данных этого блока по какой-либо причине недоступна. Пример ее использования приведен в §2.7.4.

См. также:

rdsGetBlockVar (стр. 328), rdsGetRuntimeTypeData (стр. 330).

A.5.14.12. rdsGetBlockVarDefValueStr – получить значение переменной блока по умолчанию

Функция `rdsGetBlockVarDefValueStr` возвращает динамическую строку, содержащую текстовое представление значения по умолчанию указанной переменной в указанном блоке.

```
LPSTR RDSCALL rdsGetBlockVarDefValueStr(  
    RDS_BHANDLE Block,    // Идентификатор блока  
    int VarNum,           // Номер переменной  
    int *pLength          // Возвращаемая длина строки  
);
```

Тип указателя на эту функцию:

`RDS_SBhIpI`

Параметры:

`Block`

Идентификатор блока, в котором нужно найти переменную.

`VarNum`

Порядковый номер переменной (начиная с нуля).

`pLength`

Указатель на целую переменную, в которую функция должна записать длину получившейся строки. Если вызывающей программе не нужна длина строки, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку со значением по умолчанию указанной переменной, или `NULL`, если такой переменной нет.

Примечания:

Эта функция чаще всего используется в тех случаях, когда создатель модели блока решает хранить настроечные параметры этого блока в значениях по умолчанию его статических переменных – так ему не приходится следить за их загрузкой и сохранением. Пример использования этой функции приведен в §2.7.4 и §2.16.2.

Динамическая строка, созданная функцией `rdsGetBlockVarDefValueStr`, должна быть **обязательно** освобождена функцией `rdsFree` (стр. 187).

См. также:

```
rdsSetBlockVarDefValueStr (стр. 334),  
rdsSetBlockVarDefValueByCur (стр. 333),  
rdsVSGetVarDefValueStr (стр. 441), rdsFree (стр. 187).
```

A.5.14.13. rdsGetRuntimeTypeData – получить фактические данные переменной произвольного типа

Функция `rdsGetRuntimeTypeData` служит для доступа к фактическим данным переменной произвольного типа, если у нее в данный момент есть фактический тип. Работа с переменными произвольного типа подробно рассматривается в §2.5.6.

```
LPVOID RDSCALL rdsGetRuntimeTypeData(  
    LPVOID BaseAddr,    // Базовый адрес переменной  
    LPSTR *pRealTypeStr // Возвращаемая строка типа  
);
```

Тип указателя на эту функцию:

RDS_pVpVpS

Параметры:

BaseAddr

Базовый адрес переменной произвольного типа в дереве, то есть указатель на блок из восьми байтов, которые эта переменная занимает в дереве переменных блока.

pRealTypeStr

Указатель на переменную типа LPSTR (char*), в которую функция запишет указатель на созданную ей динамическую строку, содержащую описание фактического типа, который в данный момент имеет переменная. Эта строка состоит из одного (для простых переменных) или нескольких (для сложных переменных) стандартных символов, используемых в РДС для указания типа переменной (см. стр. 49). Если строка типа не нужна вызывающей программе, в этом параметре можно передать NULL.

Возвращаемое значение:

Указатель на начало области данных переменной какого-либо фактического типа, созданной внутри переменной произвольного типа. Если переменная в данный момент не имеет фактического типа, функция вернет NULL.

Примечания:

Эта функция, в основном, используется для получения строки фактического типа переменной произвольного типа – базовый адрес переменной фактического типа обычно можно получить и без вызова функций. Пример использования функции rdsGetRuntimeTypeData приведен в §2.5.6, там же описывается принцип размещения переменных произвольного типа в памяти.

Динамическая строка, записанная функцией rdsGetRuntimeTypeData по указателю из параметра pRealTypeStr (если он не равен NULL), должна быть **обязательно** освобождена функцией rdsFree (стр. 187).

См. также:

rdsClearRuntimeType (стр. 321), rdsSetRuntimeType (стр. 335),
rdsFree (стр. 187).

A.5.14.14. rdsGetStructVar – структура по номеру

Функция rdsGetStructVar возвращает идентификатор структуры с заданным номером в общем списке структур РДС.

```
RDS_VHANDLE RDSCALL rdsGetStructVar(  
    int StructNum,           // Номер структуры  
    RDS_PVARDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

RDS_VhIVd

Параметры:

StructNum

Порядковый номер структуры в общем списке (начиная с нуля).

pDescr

Указатель на структуру описания переменной RDS_VARDESCRIPTION (стр. 138), которую функция должна заполнить параметрами структуры с указанным номером. Если вызывающей программе не нужно это описание, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор структуры (тип RDS_VHANDLE, см. стр. 23) или NULL, если структура с номером StructNum отсутствует в общем списке структур, зарегистрированных в схеме.

Примечания:

Эта функция позволяет найти идентификатор структуры с заданным номером в общем списке структур РДС. Каждая из этих структур представляет собой обычную переменную и, как и переменные блоков, имеет идентификатор RDS_VHANDLE. Общее число зарегистрированных структур можно получить вызовом функции rdsGetSystemInt с параметром RDS_GSIINSTSTRUCTCOUNT (стр. 158).

См. также:

rdsFindStructVar (стр. 327), RDS_VARDESCRIPTION (стр. 138),
rdsGetSystemInt (стр. 158).

A.5.14.15. rdsGetVarField – поле структуры или элемент массива по номеру

Функция rdsGetVarField возвращает идентификатор поля структуры с указанным номером или элемента указанного массива/матрицы.

```
RDS_VHANDLE RDSCALL rdsGetVarField(  
    RDS_VHANDLE ParentVar,      // Переменная-владелец  
    int VarNum,                 // Номер переменной  
    RDS_PVARDESCRIPTION pDescr // Заполняемое описание  
);
```

Тип указателя на эту функцию:

RDS_VhVhIvD

Параметры:

ParentVar

Идентификатор структуры или матрицы (массива), идентификатор элемента которой требуется получить.

VarNum

Порядковый номер поля структуры, начиная с нуля. Если ParentVar – массив или матрица, этот параметр игнорируется, поскольку все элементы массива или матрицы всегда обслуживаются одной и той же переменной-элементом (при изменении индекса меняется только базовый адрес данных, с которыми работает эта переменная).

pDescr

Указатель на структуру описания переменной RDS_VARDESCRIPTION (стр. 138), которую функция должна заполнить параметрами поля структуры с указанным номером или элемента массива. Если вызывающей программе не нужно описание переменной, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденной переменной (тип `RDS_VHANDLE`, см. стр. 23) или `NULL`, если такой переменной нет.

Примечания:

С помощью этой функции можно получить доступ к идентификаторам и описаниям внутренних полей какой-либо структуры или элементу массива. Это бывает нужно при анализе сложных переменных блока – например, в модуле автокомпиляции (см. главу 4) при формировании для переменных каких-либо вспомогательных классов доступа.

См. также:

`RDS_VARDESCRIPTION` (стр. 138), `rdsGetBlockVar` (стр. 328).

A.5.14.16. `rdsSetBlockVarDefValueByCur` – сделать текущее значение переменной блока значением по умолчанию

Функция `rdsSetBlockVarDefValueByCur` копирует текущее значение статической переменной указанного блока с заданным номером в ее значение по умолчанию.

```
BOOL RDCALL rdsSetBlockVarDefValueByCur(  
    RDS_BHANDLE Block,      // Идентификатор блока  
    int VarNum              // Номер переменной  
);
```

Тип указателя на эту функцию:

`RDS_BBhI`

Параметры:

`Block`

Идентификатор блока, которому принадлежит переменная.

`VarNum`

Порядковый номер переменной в блоке (начиная с нуля).

Возвращаемое значение:

`TRUE` – значение переменной по умолчанию установлено, `FALSE` – в блоке нет переменной с таким номером.

Примечания:

Эта функция позволяет сделать текущее значение какой-либо статической переменной блока значением по умолчанию, то есть значением, автоматически присваиваемым переменной после загрузки схемы или сброса расчета. Пример использования функции `rdsSetBlockVarDefValueByCur` приведен в §2.7.5.

Следует помнить, что для элемента массива и матрицы может быть задано только одно значение по умолчанию, поэтому, если переменная с номером `VarNum` – массив или матрица, после вызова `rdsSetBlockVarDefValueByCur` в качестве значения по умолчанию для этой переменной будет запомнен текущий размер этой матрицы (массива) и значение ее нулевого элемента.

См. также:

`rdsSetBlockVarDefValueStr` (стр. 334),
`rdsGetBlockVarDefValueStr` (стр. 330).

A.5.14.17. rdsSetBlockVarDefValueStr – установить значение переменной по умолчанию

Функция rdsSetBlockVarDefValueStr устанавливает значение по умолчанию и текущее значение для указанной переменной указанного блока согласно переданной строке.

```
BOOL RDSCALL rdsSetBlockVarDefValueStr(  
    RDS_BHANDLE Block,      // Идентификатор блока  
    int VarNum,             // Номер переменной  
    LPSTR DefValue          // Строка значения  
);
```

Тип указателя на эту функцию:

RDS_BBhIS

Параметры:

Block

Идентификатор блока, которому принадлежит переменная.

VarNum

Порядковый номер переменной в блоке (начиная с нуля).

DefValue

Указатель на строку, содержащую символьное представление значения переменной (в том виде, в котором его обычно вводит пользователь).

Возвращаемое значение:

TRUE – значение переменной установлено, FALSE – в блоке нет переменной с таким номером.

Примечания:

Эта функция позволяет задать текущее значение переменной и ее значение по умолчанию. Устанавливаемое значение указывается в символьном виде, соответствующем правилам РДС:

- для вещественных переменных – строка, соответствующая правилам функции rdsAtoD (стр. 184);
- для целых, логических и сигнальных переменных – строка, соответствующая правилам функции rdsAtoI (стр. 184);
- для массивов – строка вида “[N]val”, где N – число элементов, val – значение элемента (одно на все элементы);
- для матриц – строка вида “[Nr,Nc]val”, где Nr – число строк, Nc – число столбцов, val – значение элемента (одно на все элементы);
- для структур – строка вида “{val1,val2, ..., valN}”, где val1...valN – значения полей.

Чаще всего эта функция используется для записи введенного пользователем значения одновременно в текущее значение и в значение переменной по умолчанию. Пример использования функции rdsSetBlockVarDefValueStr приведен в §2.7.4.

См. также:

rdsSetBlockVarDefValueByCur (стр. 333),
rdsGetBlockVarDefValueStr (стр. 330), rdsAtoD (стр. 184),
rdsAtoI (стр. 184).

A.5.14.18. rdsSetBlockVarFlags – установить флаги переменной

Функция `rdsSetBlockVarFlags` устанавливает различные битовые флаги указанной переменной указанного блока.

```
BOOL RDCALL rdsSetBlockVarFlags(  
    RDS_BHANDLE Block,      // Идентификатор блока  
    int VarNum,             // Номер переменной  
    DWORD Flags,           // Флаги  
    DWORD Mask             // Маска  
);
```

Тип указателя на эту функцию:

`RDS_BBhIDwDw`

Параметры:

Block

Идентификатор блока, которому принадлежит переменная. Если в этом параметре передано значение `NULL`, функция будет работать с блоком, из модели которого она вызвана.

VarNum

Порядковый номер переменной в блоке (начиная с нуля).

Flags

Набор битовых флагов переменной `RDS_VARFLAG_*` (см. поле `Flags` структуры `RDS_VARDESCRIPTION`, стр. 138).

Mask

Маска изменяемых битовых флагов (единичные биты в позиции тех флагов, которые нужно изменить в переменной `VarNum` согласно `Flags`).

Возвращаемое значение:

`TRUE` – флаги переменной установлены, `FALSE` – в блоке нет переменной с таким номером.

Примечания:

Эта функция устанавливает и сбрасывает битовые флаги, определяющие поведение переменной с номером `VarNum` в блоке `Block`. В параметре `Flags` передается целое число, у которого в позициях, соответствующих взводимым флагам, будут находиться единичные биты, а в позициях, соответствующих сбрасываемым – нулевые. При этом в параметре `Mask` должно быть передано целое число, у которого единичные биты соответствуют изменяемым (взводимым или сбрасываемым) флагам, а нулевые – флагам, остающимся неизменными.

См. также:

`RDS_VARDESCRIPTION` (стр. 138), `rdsVSSetVarFlags` (стр. 443).

A.5.14.19. rdsSetRuntimeType – установить фактический тип переменной произвольного типа

Функция `rdsSetRuntimeType` служит для присвоения переменной произвольного типа какого-либо конкретного фактического типа. Работа с переменными произвольного типа подробно рассматривается в §2.5.6.

```
LPCVOID RDCALL rdsSetRuntimeType(  
    LPCVOID BaseAddr,      // Базовый адрес переменной
```

```

        LPSTR TypeStr          // Строка типа
    );

```

Тип указателя на эту функцию:

```
RDS_pVpVS
```

Параметры:

BaseAddr

Базовый адрес переменной произвольного типа в дереве, то есть указатель на блок из восьми байтов, которые эта переменная занимает в дереве переменных блока.

TypeStr

Указатель на строку, описывающую новый фактический тип переменной. Эта строка состоит из одного (для простых переменных) или нескольких (для сложных переменных) стандартных символов, используемых в РДС для указания типов переменных (см. стр. 49).

Возвращаемое значение:

Указатель на начало области данных новой переменной указанного фактического типа, созданной внутри переменной произвольного типа. Если строка типа пустая, функция вернет NULL.

Примечания:

Эта функция позволяет изменять фактический тип переменной произвольного типа. Пример ее использования приведен в §2.5.6, там же описывается принцип размещения переменных произвольного типа в памяти.

См. также:

rdsClearRuntimeType (стр. 321), rdsGetRuntimeTypeData (стр. 330).

A.5.14.20. rdsVarUsesStructType – проверить использование структуры внутри переменной

Функция rdsVarUsesStructType проверяет, используется ли где-нибудь внутри указанной переменной (в элементах матриц или в полях структур) структура указанного типа.

```

BOOL RDSCALL rdsVarUsesStructType(
    RDS_VHANDLE Var,          // Переменная
    LPSTR StructName         // Имя типа структуры
);

```

Тип указателя на эту функцию:

```
RDS_BVhS
```

Параметры:

Var

Идентификатор переменной (тип RDS_VHANDLE, см. стр. 23), внутри которой ищется структура с указанным именем типа. Очевидно, эта переменная должна быть массивом, матрицей или структурой – внутри простых переменных не может быть других переменных, а, значит, никакая структура в них заведомо не используется.

StructName

Указатель на строку с именем типа, под которым структура зарегистрирована в РДС.

Возвращаемое значение:

TRUE – структура StructName используется где-то внутри переменной Var. FALSE – структура не используется.

Примечания:

С помощью этой функции можно проверить, требуется ли описание структуры с типом StructName для работы с переменной Var. Она может применяться в модулях автокомпиляции (см. главу 4) для поиска структур РДС, описания которых нужны для работы блока.

См. также:

rdsFindStructVar (стр. 327), rdsGetStructVar (стр. 331),
rdsVSUsesStructType (стр. 444).

А.5.15. Работа с матрицами и массивами

Описываются функции, выполняющие различные действия с матрицами и массивами в составе переменных блока.

А.5.15.1. Макрос RDS_ARRAYCOLS – число столбцов матрицы/массива

Макрос RDS_ARRAYCOLS возвращает число столбцов (второй индекс) в статической или динамической переменной блока, являющейся матрицей или массивом. В массиве число строк всегда равно единице, поэтому число столбцов равно числу элементов.

```
RDS_ARRAYCOLS (  
    baseaddr          // Указатель на переменную  
)
```

Определение:

```
#define RDS_ARRAYCOLS(baseaddr) \  
    ( *( *( (int**) (baseaddr)) + 1 ) )
```

Параметр:

baseaddr

Указатель (может быть любого типа, в т.ч. и void*) на данные переменной в дереве переменных блока.

Возвращаемое значение:

Число столбцов в матрице/массиве.

Примечания:

По указателю baseaddr в дереве переменных блока находятся восемь байтов, описывающих матрицу или массив. Четыре первых байта занимает указатель на данные матрицы. Если матрица пуста, этот указатель будет равен NULL, если нет, в области памяти, на которую он указывает, будут размещаться два тридцатидвухбитных целых числа, в первом из которых хранится число строк матрицы, во втором – число столбцов, а за ними будут последовательно размещаться значения элементов матрицы. Макрос RDS_ARRAYCOLS возвращает второе из этих двух чисел. Массивы в РДС хранятся в памяти так же, как и матрицы, поэтому этот макрос можно применять как для матриц, так и для массивов. Пример его использования приведен в §2.5.3.

Поскольку в макросе RDS_ARRAYCOLS отсутствует проверка указателя на данные матрицы на значение NULL, и он сразу обращается к этим данным, использовать его можно только после проверки на существование матрицы, например, макросом

RDS_ARRAYEXISTS (стр. 339). При желании, вместо макросов, работающих непосредственно с данными переменной, для получения описания матрицы или массива можно использовать структуру RDS_ARRAYACCESSDATA (стр. 112) и функции, работающие с ней.

См. также:

RDS_ARRAYROWS (стр. 341), RDS_ARRAYEXISTS (стр. 339),
RDS_ARRAYDATA (стр. 338), RDS_ARRAYACCESSDATA (стр. 112).

A.5.15.2. Макрос RDS_ARRAYDATA – указатель на первый элемент матрицы/массива

Макрос RDS_ARRAYDATA возвращает указатель на первый элемент статической или динамической переменной блока, являющейся матрицей или массивом.

```
RDS_ARRAYDATA(  
    baseaddr          // Указатель на переменную  
)
```

Определение:

```
#define RDS_ARRAYDATA(baseaddr) \  
    ( *((int**) (baseaddr)) + 2 )
```

Параметр:

baseaddr

Указатель (может быть любого типа, в т.ч. и void*) на данные переменной в дереве переменных блока.

Возвращаемое значение:

Указатель на первый элемент матрицы. Хотя технически он возвращается как указатель на целое (int*), его следует считать указателем произвольного типа и явно приводить к типу указателя на тип элемента матрицы (для матрицы вещественных чисел двойной точности – double* и т.п.).

Примечания:

По указателю baseaddr в дереве переменных блока находятся восемь байтов, описывающих матрицу или массив. Четыре первых байта занимает указатель на данные матрицы. Если матрица пуста, этот указатель будет равен NULL, если нет, в области памяти, на которую он указывает, будут размещаться два тридцатидвухбитных целых числа, в первом из которых хранится число строк матрицы, во втором – число столбцов, а за ними последовательно располагаются значения элементов матрицы. Макрос RDS_ARRAYDATA возвращает указатель на первый после числа столбцов байт области данных матрицы, то есть на первый ее элемент. Массивы в РДС хранятся в памяти так же, как и матрицы, поэтому этот макрос можно применять как для матриц, так и для массивов. Пример его использования приведен в §2.5.3.

Поскольку в макросе RDS_ARRAYDATA отсутствует проверка указателя на данные матрицы на значение NULL, и он сразу обращается к этим данным, использовать его можно только после проверки на существование матрицы, например, макросом RDS_ARRAYEXISTS (стр. 339). При желании, вместо макросов, работающих непосредственно с данными переменной, для получения описания матрицы или массива можно использовать структуру RDS_ARRAYACCESSDATA (стр. 112) и функции, работающие с ней.

См. также:

RDS_ARRAYCOLS (стр. 337), RDS_ARRAYROWS (стр. 341),
RDS_ARRAYEXISTS (стр. 339), RDS_ARRAYACCESSDATA (стр. 112).

A.5.15.3. Макрос RDS_ARRAYEXISTS – проверка наличия элементов в матрице/массиве

Макрос RDS_ARRAYEXISTS проверяет, есть ли элементы в статической или динамической переменной блока, являющейся матрицей или массивом.

```
RDS_ARRAYEXISTS(  
    baseaddr          // Указатель на переменную  
)
```

Определение:

```
#define RDS_ARRAYEXISTS(baseaddr) \  
    (*(void**) (baseaddr)) != NULL
```

Параметр:

baseaddr

Указатель (может быть любого типа, в т.ч. и void*) на данные переменной в дереве переменных блока.

Возвращаемое значение:

TRUE – в массиве или матрице есть хотя бы один элемент. FALSE – матрица или массив пусты, то есть не содержат ни одного элемента, и область данных для них не отведена.

Примечания:

По указателю baseaddr в дереве переменных блока находятся восемь байтов, описывающих матрицу. Четыре первых байта занимает указатель на данные матрицы. Если матрица пуста, этот указатель будет равен NULL, и макрос RDS_ARRAYEXISTS вернет значение FALSE. Если этот указатель не равен NULL, значит, для матрицы создана область данных со значениями элементов, и макрос вернет TRUE. Пример использования RDS_ARRAYEXISTS приведен в §2.5.3. Массивы в РДС хранятся в памяти так же, как и матрицы, поэтому этот макрос можно применять как для матриц, так и для массивов. При желании, вместо макросов, работающих непосредственно с данными переменной, для получения описания матрицы или массива можно использовать структуру RDS_ARRAYACCESSDATA (стр. 112) и функции, работающие с ней.

См. также:

RDS_ARRAYCOLS (стр. 337), RDS_ARRAYROWS (стр. 341),
RDS_ARRAYDATA (стр. 338), RDS_ARRAYACCESSDATA (стр. 112).

A.5.15.4. Макрос RDS_ARRAYITEM – элемент матрицы с заданными индексами

Макрос RDS_ARRAYITEM возвращает элемент статической или динамической переменной блока, являющейся матрицей или массивом, с указанными в параметрах индексами. Он может использоваться только для матриц и массивов простых переменных и только вместе со структурой RDS_ARRAYACCESSDATA (стр. 112).

```
RDS_ARRAYITEM(  
    type,          // Тип элемента  
    acc,           // Указатель на структуру описания
```

```

        r,          // Строка
        c          // Столбец
    )

```

Определение:

```

#define RDS_ARRAYITEM(type,acc,r,c) \
    ( *((type *)RDS_ARRAYITEMADDR(acc,r,c)) )

```

Параметры:

type

Простой тип элемента матрицы/массива (double, int и т.д.).

acc

Указатель на структуру описания матрицы RDS_ARRAYACCESSDATA.

r, c

Строка (r) и столбец (c) элемента, нумерация начинается с нуля.

Возвращаемое значение:

Значение элемента матрицы, описываемой структурой acc. Это значение может использоваться как в правой, так и в левой (lvalue) части выражений.

Примечания:

Для использования этого макроса необходимо иметь заполненную структуру описания матрицы RDS_ARRAYACCESSDATA (указатель на нее передается во втором параметре макроса). Сам макрос определен через другой макрос – RDS_ARRAYITEMADDR (стр. 340) – который возвращает указатель на данные элемента матрицы с указанным номером. Этот адрес затем приводится к типу “указатель на type”, после чего оператором * берется его содержимое (именно поэтому этот макрос можно использовать только для матриц, элементы которых имеют простой тип, совпадающий с одним из типов языка C). Массивы в РДС хранятся в памяти так же, как и матрицы, поэтому этот макрос можно применять как для матриц, так и для массивов – в массиве в качестве номера строки всегда следует передавать 0.

Пример использования RDS_ARRAYITEM приведен в §2.5.3.

См. также:

RDS_ARRAYACCESSDATA (стр. 112), RDS_ARRAYITEMADDR (стр. 340).

A.5.15.5. Макрос RDS_ARRAYITEMADDR – указатель на элемент матрицы с заданными индексами

Макрос RDS_ARRAYITEMADDR возвращает указатель на элемент статической или динамической переменной блока, являющейся матрицей или массивом, с указанными в параметрах индексами. Он может использоваться только вместе со структурой RDS_ARRAYACCESSDATA (стр. 112).

```

RDS_ARRAYITEMADDR (
    acc,          // Указатель на структуру описания
    r,           // Строка
    c           // Столбец
)

```

Определение:

```
#define RDS_ARRAYITEMADDR(acc,r,c) \
    ( ((char*)((acc)->Data)) + \
      ((r)*((acc)->Cols)+(c))*((acc)->ItemSize) )
```

Параметры:

acc

Указатель на структуру описания матрицы RDS_ARRAYACCESSDATA.

r, c

Строка (r) и столбец (c) элемента, нумерация начинается с нуля..

Возвращаемое значение:

Указатель на элемент матрицы, описываемой структурой acc. Тип этого указателя не соответствует фактическому типу элемента, поэтому приведение типов необходимо выполнить явно.

Примечания:

Для использования этого макроса необходимо иметь заполненную структуру описания матрицы RDS_ARRAYACCESSDATA (указатель на нее передается во втором параметре макроса). Используя поля этой структуры, макрос вычисляет указатель на элемент с индексами r и c. В отличие от макроса RDS_ARRAYITEM (стр. 339), он может применяться и для матриц сложных переменных, поскольку возвращает только указатель на элемент, не выполняя взятие содержимого этого указателя.

Массивы в РДС хранятся в памяти так же, как и матрицы, поэтому этот макрос можно применять как для матриц, так и для массивов – в массиве в качестве номера строки всегда следует передавать 0.

См. также:

RDS_ARRAYACCESSDATA (стр. 112), RDS_ARRAYITEM (стр. 339).

A.5.15.6. Макрос RDS_ARRAYROWS – число строк матрицы/массива

Макрос RDS_ARRAYROWS возвращает число строк (первый индекс) в статической или динамической переменной блока, являющейся матрицей или массивом. В массиве число строк всегда равно единице.

```
RDS_ARRAYROWS (
    baseaddr          // Указатель на переменную
)
```

Определение:

```
#define RDS_ARRAYROWS(baseaddr) \
    ( *((int**) (baseaddr)) )
```

Параметр:

baseaddr

Указатель (может быть любого типа, в т.ч. и void*) на данные переменной в дереве переменных блока.

Возвращаемое значение:

Число строк в матрице/массиве.

Примечания:

По указателю `baseaddr` в дереве переменных блока находятся восемь байтов, описывающих матрицу. Четыре первых байта занимает указатель на данные матрицы. Если матрица пуста, этот указатель будет равен `NULL`, если нет, в области памяти, на которую он указывает, будут размещаться два тридцатидвухбитных целых числа, в первом из которых хранится число строк матрицы, во втором – число столбцов, а за ними последовательно размещаются значения элементов матрицы. Макрос `RDS_ARRAYROWS` возвращает первое из этих двух чисел. Массивы в РДС хранятся в памяти так же, как и матрицы, поэтому этот макрос можно применять как для матриц, так и для массивов, хотя для массивов он всегда будет возвращать единицу. Пример его использования приведен в §2.5.3.

Поскольку в макросе `RDS_ARRAYROWS` отсутствует проверка указателя на данные матрицы на значение `NULL`, и он сразу обращается к этим данным, использовать его можно только после проверки на существование матрицы, например, макросом `RDS_ARRAYEXISTS` (стр. 339). При желании, вместо макросов, работающих непосредственно с данными переменной, для получения описания матрицы или массива можно использовать структуру `RDS_ARRAYACCESSDATA` (стр. 112) и функции, работающие с ней.

См. также:

`RDS_ARRAYCOLS` (стр. 337), `RDS_ARRAYEXISTS` (стр. 339),
`RDS_ARRAYDATA` (стр. 338), `RDS_ARRAYACCESSDATA` (стр. 112).

A.5.15.7. `rdsCopyVarArray` – копировать одну матрицу/массив в другую

Функция `rdsCopyVarArray` копирует все содержимое одной матрицы (то есть статической или динамической переменной блока, являющейся матрицей или массивом) в другую. Она также может использоваться для копирования в матрицу содержимого переменной произвольного типа, если фактическое значение последней в данный момент является матрицей, совместимой по типу.

```
BOOL RDSCALL rdsCopyVarArray(  
    LPVOID DestBaseAddr,    // Базовый адрес получателя  
    LPVOID SrcBaseAddr      // Базовый адрес источника  
);
```

Тип указателя на эту функцию:

`RDS_BpVpV`

Параметры:

`DestBaseAddr`

Указатель (может быть любого типа, в т.ч. и `void*`) на данные матрицы-получателя в дереве переменных блока, то есть указатель на область из восьми байтов, которые эта матрица занимает в дереве переменных (см. §2.5.3).

`SrcBaseAddr`

Указатель на данные переменной-источника (матрицы или переменной произвольного типа). Размещение переменной произвольного типа в памяти описывается в §2.5.6.

Возвращаемое значение:

`TRUE` – значение переменной скопировано успешно, `FALSE` – произошла ошибка (попытка копирования матриц несовместимых типов).

Примечания:

Эта функция копирует содержимое одной матрицы (массива) в другую, если типы их элементов совместимы. Матрица-получатель данных должна обязательно быть переменной (не важно, статической или динамической) типа “матрица” или “массив”, а матрица-источник может быть как матрицей/массивом, так и переменной произвольного типа – в этом случае копирование будет выполнено только в том случае, если ее фактическое значение на момент вызова функции является матрицей. После копирования старое содержимое матрицы, находящейся по указателю `DestBaseAddr`, будет потеряно, и матрица получит новый размер и новое содержимое. Если копируемая матрица была пустой, матрица-получатель будет очищена (получит размер `0x0`).

Если типы элементов источника и получателя совпадают, копирование выполняется быстрее всего. Если типы отличаются, РДС будет выполнять преобразование типа для каждого элемента. Например, если получатель – матрица строк, а источник – матрица вещественных чисел, каждое вещественное число при копировании будет преобразовано в строку.

Сервисные функции РДС одинаково работают с матрицами и с массивами, поэтому с помощью этой функции можно копировать матрицы в массивы и массивы в матрицы.

Пример использования функции `rdsCopyVarArray` приведен в §2.6.5.

См. также:

`rdsCopyRuntimeType` (стр. 322), `rdsCopyVarGeneral` (стр. 323).

A.5.15.8. `rdsGetVarArrayAccessData` – заполнить структуру описания матрицы/массива

Функция `rdsGetVarArrayAccessData` заполняет описанием указанной матрицы или массива структуру `RDS_ARRAYACCESSDATA` (стр. 112).

```
BOOL RDSCALL rdsGetVarArrayAccessData(  
    LPVOID BaseAddr,           // Базовый адрес переменной  
    RDS_PARRAYACCESSDATA pAccessData // Описание  
);
```

Тип указателя на эту функцию:

`RDS_BpVAd`

Параметры:

`BaseAddr`

Указатель (может быть любого типа, в т.ч. и `void*`) на данные матрицы в дереве переменных блока, то есть указатель на область из восьми байтов, которые эта матрица занимает в дереве переменных (см. §2.5.3).

`pAccessData`

Указатель на структуру `RDS_ARRAYACCESSDATA`, которую должна заполнить функция.

Возвращаемое значение:

`TRUE` – структура заполнена, `FALSE` – произошла ошибка (указанная в параметре `BaseAddr` переменная – не матрица и не массив).

Примечания:

Эта функция записывает в структуру по указателю `pAccessData` признак наличия данных в матрице `BaseAddr`, число ее строк и столбцов, размер элемента и указатель на первый элемент. Пример ее использования приведен в §2.5.3.

См. также:

`RDS_ARRAYACCESSDATA` (стр. 112), `rdsGetVarArrayParams` (стр. 344).

A.5.15.9. `rdsGetVarArrayParams` – получить размеры матрицы/массива и указатель на первый элемент

Функция `rdsGetVarArrayParams` возвращает указатель на первый элемент указанной матрицы или массива, а также, при необходимости, число строк и столбцов этой матрицы.

```
LPVOID RDSCALL rdsGetVarArrayParams (  
    LPVOID BaseAddr,      // Базовый адрес переменной  
    int *pRows,           // Возвращаемое число строк  
    int *pCols            // Возвращаемое число столбцов  
);
```

Тип указателя на эту функцию:

`RDS_pVpVpIpI`

Параметры:

`BaseAddr`

Указатель (может быть любого типа, в т.ч. и `void*`) на данные матрицы в дереве переменных блока, то есть указатель на область из восьми байтов, которые эта матрицы занимает в дереве переменных (см. §2.5.3).

`pRows`

Указатель на целую переменную, в которую функция запишет число строк в матрице. Если вызывающей программе не требуется знать число строк, в этом параметре можно передать `NULL`.

`pCols`

Указатель на целую переменную, в которую функция запишет число столбцов в матрице. Если вызывающей программе не требуется знать число столбцов, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Указатель произвольного типа (`void*`) на первый элемент в матрице, или `NULL`, если матрица пуста.

Примечания:

Эта функция устарела, в настоящее время вместо нее желательно использовать `rdsGetVarArrayAccessData` (стр. 343).

См. также:

`RDS_ARRAYACCESSDATA` (стр. 112), `rdsGetVarArrayAccessData` (стр. 343).

A.5.15.10. `rdsResizeVarArray` – изменить размер матрицы/массива

Функция `rdsResizeVarArray` изменяет число строк и столбцов указанной матрицы или массива с возможностью сохранения старого содержимого.


```

BOOL RDSCALL rdsResizeVarArray(
    LPVOID BaseAddr,      // Базовый адрес переменной
    int Rows,             // Новое число строк
    int Cols,             // Новое число столбцов
    BOOL KeepData,        // Сохранять содержимое
    RDS_PARRAYACCESSDATA pAccessData // Заполняемое описание
);

```

Тип указателя на эту функцию:

RDS_BpVIIIBAd

Параметры:

BaseAddr

Указатель (может быть любого типа, в т.ч. и void*) на данные матрицы в дереве переменных блока, то есть указатель на область из восьми байтов, которую эта матрицы занимает в дереве переменных (см. §2.5.3).

Rows, Cols

Новое число строк (Rows) и столбцов (Cols) в матрице. Для очистки матрицы можно передать ноль в обоих этих параметрах.

KeepData

TRUE – сохранять старое содержимое матрицы (если ее размер уменьшается, она будет обрезана справа и снизу, если увеличивается – дополнена ячейками со значением элемента по умолчанию). FALSE – после изменения размера заполнить всю матрицу значением по умолчанию.

pAccessData

Указатель на структуру описания матрицы RDS_ARRAYACCESSDATA (стр. 112) которую функция должна заполнить после изменения размера. Если описание матрицы не нужно вызывающей программе, в этом параметре можно передать NULL.

Возвращаемое значение:

TRUE – размер матрицы изменен, FALSE – произошла ошибка (указанная в параметре BaseAddr переменная – не матрица и не массив).

Примечания:

Эта функция изменяет размер матрицы по указателю BaseAddr. Пример ее использования приведен в §2.5.3.

См. также:

RDS_ARRAYACCESSDATA (стр. 112), rdsCopyVarArray (стр. 342).

A.5.15.11. rdsVarArrayIndexCheck – проверить допустимость индексов матрицы/массива

Функция rdsVarArrayIndexCheck проверяет допустимость значений индексов строки и столбца для указанной матрицы и выводит сообщение пользователю при необходимости.

```

BOOL RDSCALL rdsVarArrayIndexCheck(
    LPVOID BaseAddr,      // Базовый адрес переменной
    int Row,              // Индекс строки
    int Col,              // Индекс столбца
    DWORD Flags,          // Флаги RDS_VAIC_*
    LPSTR VarName,        // Имя переменной или NULL

```

```

        LPSTR BlockName          // Имя блока или NULL
    );

```

Тип указателя на эту функцию:

RDS_BpVIIDwSS

Параметры:

BaseAddr

Указатель (может быть любого типа, в т.ч. и void*) на данные матрицы в дереве переменных блока, то есть указатель на область из восьми байтов, которую эта матрицы занимает в дереве переменных (см. §2.5.3).

Row, Col

Индекс строки (Row) и столбца (Col) в матрице, нумерация начинается с нуля.

Flags

Один или несколько объединенных битовым ИЛИ флагов, управляющих поведением функции:

RDS_VAIC_SINGLE Указанная переменная – массив, а не матрица. Для массива вместо проверки индексов по отдельности произведение этих индексов будет сравниваться с общим числом элементов (обычная проверка для массивов, имеющих, с точки зрения пользователя, всего один индекс). Этот флаг также влияет на выводимое сообщение – если он взведен, в тексте будет использоваться слово “массив” вместо слова “матрица”.

RDS_VAIC_MESSAGE Выводить пользователю сообщение в случае недопустимости индексов.

RDS_VAIC_SINGLEMSG При взведенном RDS_VAIC_MESSAGE: выводить сообщение только один раз за расчет. До тех пор, пока расчет не будет остановлен или сброшен, функция не будет выводить новых сообщений пользователю.

RDS_VAIC_STOPCALC Останавливать расчет при недопустимых индексах.

VarName

Имя переменной для отображения в сообщении пользователю, или NULL, если имя переменной в сообщение вставлять не нужно.

BlockName

Имя блока, в котором произошла ошибка, для отображения сообщения пользователю, или NULL, если нужно использовать имя вызвавшего функцию блока.

Возвращаемое значение:

TRUE – индексы в пределах размера матрицы, FALSE – недопустимые индексы.

Примечания:

Доступ к элементам матриц и массивов в РДС производится по смещениям относительно базового адреса (см. §2.5.3) – это обеспечивает высокую скорость работы модели, но при этом, очевидно, никаких автоматических проверок допустимости индексов не производится. Функция rdsVarArrayIndexCheck позволяет организовать простейшую проверку индексов с выводом сообщения пользователю или без него. Такая проверка может, например, использоваться для организации классов доступа к матрицам и массивам в моделях, создаваемых модулями автокомпиляции (см. главу 4). При взведенном в параметре Flags флаге RDS_VAIC_SINGLE (при этом считается, что переменная – массив,

а не матрица), функция возвращает TRUE, если произведение параметров Row и Col больше или равно нулю и меньше произведения числа строк и числа столбцов матрицы. При сброшенном RDS_VAIC_SINGLE функция возвращает TRUE, если Row больше или равно нулю и меньше числа строк в матрице, и при этом Col больше или равно нулю и меньше числа столбцов в матрице.

При взведенном флаге RDS_VAIC_MESSAGE функция будет выводить сообщение вида “Выход индекса матрицы за диапазон: переменная *ИМЯ1*, блок *ИМЯ2*, размер $[N,M]$, обращение к ячейке $[r,c]$ ”, где *ИМЯ1* – значение параметра VarName (имя переменной), *ИМЯ2* – имя блока (значение параметра BlockName или имя вызвавшего функцию блока), *N* – число строк в матрице, *M* – число столбцов, *r* – значение параметра Row, *c* – значение параметра Col. Если переменная не матрица, а массив, текст сообщения немного изменяется. Сообщение выводится в отдельном окне через функцию rdsMessageBox (стр. 201).

Если взведен флаг RDS_VAIC_SINGLEMSG, при запущенном расчете сообщение будет выведено только один раз (при самом первом вызове функции для самого первого вызвавшего ее блока), все остальные сообщения будут пропускаться до тех пор, пока расчет не будет остановлен или сброшен. Это может быть полезно, если проверка производится в реакции на выполнение такта расчета RDS_BFM_MODEL (стр. 40) – если ошибка в значениях индексов будет повторяться каждый такт и при этом каждый раз будет выводиться сообщение, эти сообщения не дадут пользователю нормально работать.

См. также:

RDS_ARRAYCOLS (стр. 337), RDS_ARRAYROWS (стр. 341),
rdsGetVarArrayAccessData (стр. 343), rdsMessageBox (стр. 201).

A.5.16. Работа с динамическими переменными

Описываются функции, обеспечивающие создание, удаление и подписку на динамические переменные (см. §2.6).

A.5.16.1. rdsCreateAndSubscribeDV – создать динамическую переменную и подписаться на нее

Функция rdsCreateAndSubscribeDV создает динамическую переменную указанного типа в указанном блоке и подписывает на нее блок, вызвавший эту функцию.

```
RDS_PDYNVARLINK RDSCALL rdsCreateAndSubscribeDV(  
    int Block,           // Место создания (RDS_DV*)  
    LPSTR VarName,      // Имя переменной  
    LPSTR VarType,      // Тип переменной  
    BOOL Fixed,         // Запрет удаления  
    LPSTR Init          // Начальное значение  
);
```

Тип указателя на эту функцию:

RDS_DvISSBS

Параметры:

Block

Одна из констант, указывающих на блок, в котором динамическая переменная будет создана:

RDS_DVSELF Переменная будет создана в вызвавшем функцию блоке.

RDS_DVPARENT Переменная будет создана в родительской подсистеме вызвавшего функцию блока.

RDS_DVROOT Переменная будет создана в корневой подсистеме схемы.

VarName

Указатель на строку с именем создаваемой переменной.

VarType

Указатель на строку типа создаваемой переменной. Эта строка состоит из одного (для простых переменных) или нескольких (для структур и матриц) символов RDS_VARTYPE_* (см. стр. 49).

Fixed

TRUE, если удалять эту переменную разрешено только создавшему ее блоку, то есть блоку, вызвавшему эту функцию. FALSE, если удалить переменную может любой блок.

Init

Указатель на строку, содержащую описание значения переменной по умолчанию, составленное по правилам РДС (см. описание функции rdsSetBlockVarDefValueStr на стр. 334). В этом параметре можно передать NULL, если значение созданной переменной устанавливать не нужно.

Возвращаемое значение:

Указатель на созданную во внутренней памяти РДС структуру подписки RDS_DYNVARLINK (стр. 121) или NULL, если создать переменную не удалось (например, если переменная с таким именем уже есть в указанном блоке).

Примечания:

Эта функция создает новую динамическую переменную с именем VarName и типом VarType в вызвавшем функцию блоке (при Block==RDS_DVSELF), его родительской подсистеме (при Block==RDS_DVPARENT) или в корневой подсистеме (при Block==RDS_DVROOT). В параметре VarType передается строка типа переменной, составленная по правилам РДС: например, для вещественного числа – “D”, для матрицы целых чисел – “MI”, для структуры из двух вещественных чисел, как бы ни назывался тип этой структуры – “{DD}”. При успешном создании переменной функция сразу же подписывает на нее вызвавший блок и возвращает указатель на структуру подписки RDS_DYNVARLINK, используя поля которой модель блока будет обращаться к переменной.

Примеры использования функции rdsCreateAndSubscribeDV приведены в §2.6.3 и §2.6.5.

См. также:

RDS_DYNVARLINK (стр. 121), rdsSetBlockVarDefValueStr (стр. 334),
rdsDeleteDVByLink (стр. 349), rdsDeleteDynamicVar (стр. 350).

A.5.16.2. rdsCreateDynamicVar – создать динамическую переменную

Функция rdsCreateDynamicVar создает динамическую переменную указанного типа в указанном блоке.

```
LPVOID RDSCALL rdsCreateDynamicVar(  
    int Block,           // Место создания (RDS_DV*)  
    LPSTR VarName,       // Имя переменной  
    LPSTR VarType,       // Тип переменной  
    BOOL Fixed,          // Запрет удаления  
    LPSTR Init           // Начальное значение  
);
```

Тип указателя на эту функцию:

RDS_pVISSBS

Параметры:

Block

Одна из констант RDS_DV*, указывающих на блок, в котором динамическая переменная будет создана (см. описание функции rdsCreateAndSubscribeDV на стр. 347).

VarName

Указатель на строку с именем создаваемой переменной.

VarType

Указатель на строку типа создаваемой переменной. Эта строка состоит из одного (для простых переменных) или нескольких (для структур и матриц) символов RDS_VARTYPE_* (см. стр. 49).

Fixed

TRUE, если удалять эту переменную разрешено только создавшему ее блоку, то есть блоку, вызвавшему эту функцию. FALSE, если удалить переменную может любой блок.

Init

Указатель на строку, содержащую описание значения переменной по умолчанию, составленное по правилам РДС (см. описание функции rdsSetBlockVarDefValueStr на стр. 334). В этом параметре можно передать NULL, если значение созданной переменной устанавливать не нужно.

Возвращаемое значение:

Указатель на область данных созданной переменной или NULL, если создать переменную не удалось (например, если переменная с таким именем уже есть в указанном блоке).

Примечания:

Эта функция создает новую динамическую переменную с именем VarName и типом VarType в вызвавшем функцию блоке (при Block==RDS_DVSELF), его родительской подсистеме (при Block==RDS_DVPARENT) или в корневой подсистеме (при Block==RDS_DVROOT). В параметре VarType передается строка типа переменной, составленная по правилам РДС: например, для вещественного числа – “D”, для матрицы целых чисел – “MI”, для структуры из двух вещественных чисел, как бы ни назывался тип этой структуры – “{DD}”. Для полноценной работы с созданной переменной блок должен подписаться на нее, поэтому вместо этой функции сейчас чаще всего используется rdsCreateAndSubscribeDV (стр. 347).

См. также:

rdsCreateAndSubscribeDV (стр. 347), RDS_DYNVARLINK (стр. 121),
rdsSetBlockVarDefValueStr (стр. 334), rdsDeleteDynamicVar (стр. 350).

A.5.16.3. rdsDeleteDVByLink – удалить динамическую переменную и прекратить подписку на нее

Функция rdsDeleteDVByLink удаляет динамическую переменную, с которой связана указанная в параметрах структура подписки, и прекращает подписку блока на эту переменную.

```

BOOL RDSCALL rdsDeleteDVByLink(
    RDS_PDYNVARLINK pLink // Структура подписки
);

```

Тип указателя на эту функцию:

RDS_BDV

Параметр:

pLink

Указатель на структуру подписки на переменную RDS_DYNVARLINK (стр. 121).

Возвращаемое значение:

TRUE – переменная успешно удалена, FALSE – удаление невозможно.

Примечания:

Эта функция удаляет динамическую переменную по имеющейся структуре подписки не нее. При успешном удалении подписка прекращается. Удаление может быть невозможно, например, если переменную, удаление которой разрешено только создавшему ее блоку, пытается удалить какой-то другой блок, подписавшийся на нее.

При удалении блока или отключении от него модели все созданные в нем динамические переменные уничтожаются автоматически.

Пример использования функции rdsDeleteDVByLink приведен в §2.6.3.

См. также:

RDS_DYNVARLINK (стр. 121), rdsCreateAndSubscribeDV (стр. 347),
rdsDeleteDynamicVar (стр. 350).

A.5.16.4. rdsDeleteDynamicVar – удалить динамическую переменную

Функция rdsDeleteDynamicVar удаляет динамическую переменную с указанным именем в указанном блоке.

```

BOOL RDSCALL rdsDeleteDynamicVar(
    int Block,           // Блок (RDS_DV*)
    LPSTR VarName       // Имя переменной
);

```

Тип указателя на эту функцию:

RDS_BIS

Параметры:

Block

Одна из констант RDS_DV*, указывающих на блок, в котором удаляется динамическая переменная (см. описание функции rdsCreateAndSubscribeDV на стр. 347).

VarName

Указатель на строку с именем удаляемой переменной.

Возвращаемое значение:

TRUE – переменная успешно удалена, FALSE – удаление невозможно.

Примечания:

Эта функция позволяет удалить динамическую переменную с указанным в параметре VarName именем в вызвавшем функцию блоке (при Block==RDS_DVSELF), его

родительской подсистеме (при Block==RDS_DVPARENT) или в корневой подсистеме (при Block==RDS_DVROOT). Функция не может удалить переменную в произвольном блоке (любой доступ к динамическим переменным в РДС, включая их удаление, возможен только вверх по иерархии подсистем начиная с блока, модель которого вызывает функцию). Если блок, из модели которого вызвана функция, был подписан на удаленную переменную, эта подписка **не прекращается** (при повторном создании такой же переменной блок автоматически получит к ней доступ). Если нужно одновременно удалить переменную и прекратить подписку на нее, следует вызывать функцию rdsDeleteDVByLink (стр. 349).

При удалении блока или отключении от него модели все созданные в нем динамические переменные уничтожаются автоматически.

См. также:

rdsCreateDynamicVar (стр. 348), rdsDeleteDVByLink (стр. 349).

A.5.16.5. rdsEnumDynVarSubscribers – перебрать всех подписчиков переменной

Функция rdsEnumDynVarSubscribers вызывает пользовательскую функцию, указатель на который передается в ее параметрах, для каждого блока, подписанного на указанную динамическую переменную.

```
RDS_BHANDLE RDSCALL rdsEnumDynVarSubscribers(
    RDS_PDYNVARLINK pLink,      // Структура подписки
    RDS_BBhDvpV CallBack,       // Функция пользователя
    LPVOID Data                  // Параметр функции пользователя
);
```

Тип указателя на эту функцию:

RDS_BhDvCb5pV

Параметры:

pLink

Указатель на структуру подписки на переменную RDS_DYNVARLINK (стр. 121), подписчиков которой нужно перебрать.

CallBack

Указатель на пользовательскую функцию, которую нужно вызвать для каждого блока, подписанного на переменную из структуры pLink. Пользовательская функция должна иметь следующий вид:

```
BOOL RDSCALL имя_функции(
    RDS_BHANDLE block,
    RDS_PDYNVARLINK plink,
    LPVOID param)
```

В параметре block пользовательской функции передается идентификатор блока, для которого вызвана функция, в параметре plink – указатель на структуру подписки этого блока на данную переменную, в параметре param – параметр Data (см. ниже) без какой-либо обработки. Функция пользователя должна вернуть TRUE, если перебор блоков необходимо продолжить, и FALSE, если его нужно немедленно остановить.

Data

Параметр типа void*, передаваемый в пользовательскую функцию при вызове.

Возвращаемое значение:

Идентификатор блока, для которого функция пользователя вернула FALSE. Если функция пользователя всегда возвращала TRUE (перебраны все блоки, подписанные на переменную), вместо идентификатора блока возвращается NULL.

Примечания:

Функция `rdsEnumDynVarSubscribers` обычно используется для выполнения каких-либо однотипных действий над всеми блоками, подписавшимися на динамическую переменную. Для уведомления подписчиков об изменении переменной используется другая функция – `rdsNotifyDynVarSubscribers` (стр. 352). Функция перебирает блоки в порядке, определяемом внутренней логикой РДС, программист никак не может изменить этот порядок.

Параметр `Data`, имеющий тип `LPVOID` (произвольный указатель), передается в вызываемую функцию пользователя без изменений. Это единственный способ передать ей какие-либо данные: например, можно передать указатель на какую-либо структуру, а внутри пользовательской функции привести его к нужному типу и обращаться к полям этой структуры.

Пример использования функции `rdsEnumDynVarSubscribers` приведен в §2.12.8.

См. также:

`RDS_DYNVARLINK` (стр. 121), `rdsNotifyDynVarSubscribers` (стр. 352).

A.5.16.6. `rdsNotifyDynVarSubscribers` – уведомить подписчиков об изменении переменной

Функция `rdsNotifyDynVarSubscribers` вызывает модели всех блоков, подписанных на указанную переменную, для реакции на событие `RDS_BFM_DYNVARCHANGE` (стр. 34).

```
void RDSCALL rdsNotifyDynVarSubscribers(  
    RDS_PDYNVARLINK pLink // Структура подписки  
);
```

Тип указателя на эту функцию:

`RDS_VDv`

Параметр:

`pLink`

Указатель на структуру подписки на переменную `RDS_DYNVARLINK` (стр. 121).

Примечания:

Поскольку все блоки-подписчики обращаются к данным динамической переменной напрямую, РДС не может отследить ее изменение и уведомить о нем подписчиков автоматически. Чтобы все заинтересованные блоки узнали об изменении переменной, желательно, чтобы модель любого блока, записавшая в переменную новое значение, вызывала после этого функцию `rdsNotifyDynVarSubscribers` – при этом модели всех подписчиков будут вызваны с параметром `RDS_BFM_DYNVARCHANGE`.

Пример использования функции `rdsNotifyDynVarSubscribers` приведен в §2.6.3, необходимость уведомления подписчиков об изменениях рассматривается в §2.6.1.

См. также:

`RDS_DYNVARLINK` (стр. 121), `RDS_BFM_DYNVARCHANGE` (стр. 34).

A.5.16.7. rdsSubscribeToDynamicVar – создать подписку на динамическую переменную

Функция `rdsSubscribeToDynamicVar` подписывает вызвавший ее блок на указанную динамическую переменную.

```
RDS_PDYNVARLINK RDSCALL rdsSubscribeToDynamicVar(  
    int Block,           // Блок, содержащий переменную (RDS_DV*)  
    LPSTR VarName,      // Имя переменной  
    LPSTR VarType,      // Тип переменной  
    BOOL Search         // Искать по иерархии  
);
```

Тип указателя на эту функцию:

`RDS_DvISSB`

Параметры:

`Block`

Одна из констант, указывающая на блок, в котором ищется динамическая переменная для подписки:

`RDS_DVSELF` Переменная ищется в вызвавшем функцию блоке.

`RDS_DVPARENT` Переменная ищется в родительской подсистеме вызвавшего функцию блока.

`RDS_DVROOT` Переменная ищется в корневой подсистеме схемы.

`VarName`

Указатель на строку с именем переменной.

`VarType`

Указатель на строку типа переменной. Эта строка состоит из одного (для простых переменных) или нескольких (для структур и матриц) символов `RDS_VARTYPE_*` (см. стр. 49).

`Search`

`TRUE` – если в блоке `Block` нет переменной с именем `VarName` и типом `VarType`, функция будет искать переменную с этим именем и типом в родительской подсистеме этого блока, затем – в ее родительской подсистеме и т.д. до корневой подсистемы схемы. `FALSE` – искать переменную только в блоке `Block`.

Возвращаемое значение:

Указатель на созданную во внутренней памяти РДС структуру подписки `RDS_DYNVARLINK` (стр. 121) или `NULL`, если подписка невозможна (ошибка в одном из параметров функции).

Примечания:

Эта функция подписывает вызвавший ее блок на динамическую переменную с именем `VarName` и типом `VarType` в вызвавшем блоке (при `Block==RDS_DVSELF`), его родительской подсистеме (при `Block==RDS_DVPARENT`) или в корневой подсистеме (при `Block==RDS_DVROOT`). В параметре `VarType` передается строка типа переменной, составленная по правилам РДС: например, для вещественного числа – “D”, для матрицы целых чисел – “MI”, для структуры из двух вещественных чисел, как бы ни назывался тип этой структуры – “{DD}”.

Структура подписки `RDS_DYNVARLINK` создается во внутренней памяти РДС даже в том случае, если переменная с указанным именем и типом не найдена – при этом поле `Data` структуры получит значение `NULL`, а РДС будет наблюдать за динамическими переменными,

и, при создании переменной, отвечающей параметрам функции, немедленно заполнит поля структуры ее параметрами. Общие вопросы доступа к динамическим переменным и подписки на них рассматриваются в §2.6.1.

Пример использования функции `rdsSubscribeToDynamicVar` приведен в §2.6.2.

См. также:

`RDS_DYNVARLINK` (стр. 121), `rdsUnsubscribeFromDynamicVar` (стр. 354).

A.5.16.8. `rdsUnsubscribeFromDynamicVar` – прекратить подписку на динамическую переменную

Функция `rdsUnsubscribeFromDynamicVar` прекращает подписку блока, из модели которого она вызвана, на указанную динамическую переменную.

```
void RDSCALL rdsUnsubscribeFromDynamicVar(  
    RDS_PDYNVARLINK pLink // Структура подписки  
);
```

Тип указателя на эту функцию:

`RDS_VDv`

Параметр:

`pLink`

Указатель на структуру подписки на переменную `RDS_DYNVARLINK` (стр. 121), полученный в результате вызова функции `rdsSubscribeToDynamicVar` (стр. 353) или `rdsCreateAndSubscribeDV` (стр. 347).

Примечания:

После прекращения подписки на динамическую переменную внутренняя структура `RDS_DYNVARLINK`, указатель на которую передан в ее параметре, уничтожается. При удалении блока или отключении от него модели подписка прекращается автоматически.

Пример использования функции `rdsUnsubscribeFromDynamicVar` приведен в §2.6.2, общие вопросы подписки на динамические переменные рассматриваются в §2.6.1.

См. также:

`RDS_DYNVARLINK` (стр. 121), `rdsSubscribeToDynamicVar` (стр. 353),
`rdsCreateAndSubscribeDV` (стр. 347).

A.5.17. Системное меню и контекстное меню блока

Описываются функции, управляющие расширениями системного меню РДС и контекстного меню блока.

A.5.17.1. `rdsAdditionalContextMenuItem` – добавить временный пункт в контекстное меню блока

Функция `rdsAdditionalContextMenuItem` добавляет в контекстное меню блока, из модели которого она вызвана, временный (существующий только до закрытия меню) пункт с указанным текстом. Это устаревшая сервисная функция, сейчас вместо нее используется функция `rdsAdditionalContextMenuItemEx` (стр. 355), обладающая большими возможностями.

```
void RDSCALL rdsAdditionalContextMenuItem(  
    LPSTR Caption,    // Текст пункта меню  
    BOOL Enabled,    // Разрешенность
```

```

        int MenuFunc,      // Номер функции пункта меню
        int MenuData      // Данные пункта меню
    );

```

Тип указателя на эту функцию:

RDS_VSBII

Параметры:

Caption

Указатель на строку с текстом пункта меню – этот текст пользователь увидит в самом меню.

Enabled

TRUE – пункт меню изображается нормальным цветом и может быть выбран пользователем, FALSE – пункт меню серый и не может быть выбран.

MenuFunc

Номер функции пункта меню – при выборе пункта пользователем РДС копирует это число в поле Function структуры RDS_MENUFUNCDATA (стр. 63) без какой-либо обработки.

MenuData

Данные пункта меню – при выборе пункта пользователем РДС копирует это число в поле MenuData структуры RDS_MENUFUNCDATA без какой-либо обработки.

Примечания:

Эту функцию можно вызывать только из реакции модели блока на событие RDS_BFM_CONTEXTPOPUP (стр. 56), возникающее перед открытием контекстного меню блока. Она добавляет в это меню пункт с текстом Caption и связывает с ним пару целых чисел (MenuFunc, MenuData). При выборе пункта пользователем модель блока будет вызвана для реакции на событие RDS_BFM_MENUFUNCTION (стр. 63), и в ее параметре ExtParam будет передан указатель на структуру RDS_MENUFUNCDATA, в полях которой будут записаны числа, связанные с выбранным пунктом меню.

См. также:

RDS_BFM_MENUFUNCTION, RDS_MENUFUNCDATA (стр. 63),
 RDS_BFM_CONTEXTPOPUP (стр. 56),
 rdsAdditionalContextMenuItemEx (стр. 355).

A.5.17.2. rdsAdditionalContextMenuItemEx – добавить временный пункт в контекстное меню блока

Функция rdsAdditionalContextMenuItemEx добавляет в контекстное меню блока, из модели которого она вызвана, временный (существующий только до закрытия меню) пункт с указанным текстом.

```

void RDSCALL rdsAdditionalContextMenuItemEx(
    LPSTR Caption,      // Текст пункта меню
    DWORD Options,      // Флаги (RDS_MENU_*)
    int MenuFunc,       // Номер функции пункта меню
    int MenuData        // Данные пункта меню
);

```

Тип указателя на эту функцию:

RDS_VSDwII

Параметры:

Caption

Указатель на строку с текстом пункта меню – этот текст пользователь увидит в самом меню. Если в этом параметре передать NULL, вместо пункта меню будет создан разделитель (аналогично флагу RDS_MENU_DIVIDER, см. ниже).

Options

Один или несколько объединенных битовым ИЛИ флагов, задающих параметры пункта меню:

RDS_MENU_DISABLED	Выбор пункта меню будет запрещен, пункт будет изображаться серым цветом.
RDS_MENU_CHECKED	Слева от текста пункта меню будет находиться галочка.
RDS_MENU_DIVIDER	Вместо пункта меню будет изображаться горизонтальная черта-разделитель. Пользователь не может выбрать этот пункт. При установке этого флага другие флаги и параметры Caption, MenuFunc и MenuData игнорируются.

MenuFunc

Номер функции пункта меню – при выборе пункта пользователем РДС копирует это число в поле Function структуры RDS_MENUFUNCDDATA (стр. 63) без какой-либо обработки.

MenuData

Данные пункта меню – при выборе пункта пользователем РДС копирует это число в поле MenuData структуры RDS_MENUFUNCDDATA без какой-либо обработки.

Примечания:

Эту функцию можно вызывать только из реакции модели блока на событие RDS_BFM_CONTEXTPOPUP (стр. 56), возникающее перед открытием контекстного меню блока. Она добавляет в это меню пункт с текстом Caption и внешним видом, определяемым флагами в параметре Options, связывая с ним пару целых чисел (MenuFunc, MenuData). При выборе пункта пользователем модель блока будет вызвана для реакции на событие RDS_BFM_MENUFUNCTION (стр. 63), и в ее параметре ExtParam будет передан указатель на структуру RDS_MENUFUNCDDATA, в полях которой будут записаны числа, связанные с выбранным пунктом меню.

Примеры использования функции rdsAdditionalContextMenuItemEx приведены в §§2.12.6, 2.13.3 и 2.13.4.

См. также:

RDS_BFM_MENUFUNCTION, RDS_MENUFUNCDDATA (стр. 63),
RDS_BFM_CONTEXTPOPUP (стр. 56),
rdsRegisterContextMenuItemEx (стр. 361), rdsChangeMenuItem (стр. 356),
rdsEnableMenuItem (стр. 358).

A.5.17.3. rdsChangeMenuItem – изменить параметры пункта меню

Функция rdsChangeMenuItem изменяет параметры ранее созданного постоянного пункта системного меню РДС или контекстного меню блока.

```
BOOL RDSCALL rdsChangeMenuItem(  
    RDS_MENUITEM Item,          // Идентификатор пункта меню  
    LPSTR Caption,              // Текст пункта меню  
    DWORD Options,              // Флаги (RDS_MENU_*)
```

```

    int ShortCutKey,        // Код "горячей клавиши"
    DWORD ShiftFlags,      // флаги "горячей клавиши" (RDS_K*)
    int MenuFunc,          // Номер функции пункта меню
    int MenuData            // Данные пункта меню
);

```

Тип указателя на эту функцию:

RDS_BMhSDwIDwII

Параметры:

Item

Уникальный идентификатор пункта меню (RDS_MENUITEM), параметры которого изменяются.

Caption

Указатель на строку с новым текстом пункта меню – этот текст пользователь увидит в самом меню. Если в этом параметре передать NULL, текст пункта изменен не будет.

Options

Один или несколько объединенных битовым ИЛИ флагов, задающих параметры пункта меню:

RDS_MENU_DISABLED	Выбор пункта меню будет запрещен, пункт будет изображаться серым цветом.
RDS_MENU_CHECKED	Слева от текста пункта меню будет находиться галочка.
RDS_MENU_DIVIDER	Вместо пункта меню будет изображаться горизонтальная черта-разделитель. Пользователь не может выбрать этот пункт. При установке этого флага другие флаги и параметры Caption, ShortCutKey, ShiftFlags, MenuFunc и MenuData игнорируются.
RDS_MENU_HIDDEN	Пункт меню будет невидимым (скрытым от пользователя).
RDS_MENU_SHORTCUT	У пункта меню будет “горячая клавиша”, определяемая параметрами ShortCutKey и ShiftFlags.
RDS_MENU_UNIQUECAPTION	Если пункт с текстом Caption уже есть в меню, выполнение функции будет отменено.

ShortCutKey

Код “горячей клавиши” пункта меню (только если Item – пункт системного, а не контекстного меню). Это одна из стандартных констант VK_*, используемых в Windows API.

ShiftFlags

Набор битовых флагов RDS_K* (см. стр. 61), указывающих на сочетание “горячей клавиши” ShortCutKey со служебными клавишами Ctrl, Alt и Shift.

MenuFunc

Номер функции пункта меню – при выборе пункта пользователем РДС копирует это число в поле Function структуры RDS_MENUFUNCDATA (стр. 63) без какой-либо обработки.

MenuData

Данные пункта меню – при выборе пункта пользователем РДС копирует это число в поле MenuData структуры RDS_MENUFUNCDATA без какой-либо обработки.

Возвращаемое значение:

TRUE – параметры пункта меню изменены, FALSE – ошибка.

Примечания:

Эта функция позволяет изменять параметры постоянного пункта меню (не важно – контекстного или системного), если известен его уникальный идентификатор *Item*. Пример ее использования приведен в §2.12.7.

См. также:

rdsRegisterContextMenuEx (стр. 361),
rdsRegisterMenuItem (стр. 362), *rdsSetMenuItemOptions* (стр. 363),
rdsEnableMenuItem (стр. 358).

A.5.17.4. *rdsEnableMenuItem* – установить видимость и разрешенность пункта меню

Функция *rdsEnableMenuItem* изменяет видимость ранее созданного постоянного пункта системного меню РДС или контекстного меню блока, а так же разрешает или запрещает его выбор пользователем.

```
void RDSCALL rdsEnableMenuItem(  
    RDS_MENUITEM Item,          // Идентификатор пункта меню  
    BOOL Enabled,              // Разрешенность  
    BOOL Visible                // Видимость  
);
```

Тип указателя на эту функцию:

RDS_VMhBB

Параметры:

Item

Уникальный идентификатор пункта меню (*RDS_MENUITEM*), параметры которого изменяются.

Enabled

TRUE – пункт меню изображается нормальным цветом и может быть выбран пользователем, FALSE – пункт меню серый и не может быть выбран.

Visible

TRUE – пункт меню видим, FALSE – пункт меню скрыт от пользователя и не может быть выбран.

Примечания:

Эта функция позволяет изменять два параметра постоянного пункта меню (не важно – контекстного или системного), если известен его уникальный идентификатор *Item*. Если пункт меню нужно разрешить, запретить или скрыть, а другие его параметры оставить неизменными, использовать эту функцию удобнее, чем *rdsChangeMenuItem* (стр. 356).

См. также:

rdsRegisterContextMenuEx (стр. 361),
rdsRegisterMenuItem (стр. 362), *rdsSetMenuItemOptions* (стр. 363),
rdsChangeMenuItem (стр. 356).

A.5.17.5. rdsExecMenuItem – имитировать выбор пункта меню

Функция `rdsExecMenuItem` имитирует выбор пункта меню блока пользователем. Модель блока при этом вызывается для реакции на событие `RDS_BFM_MENUFUNCTION` (стр. 63).

```
void RDSCALL rdsExecMenuItem(  
    RDS_BHANDLE Block,      // Идентификатор блока  
    int MenuFunc,           // Номер функции пункта меню  
    int MenuData            // Данные пункта меню  
);
```

Тип указателя на эту функцию:

```
RDS_VBhII
```

Параметры:

`Block`

Уникальный идентификатор блока, для которого имитируется выбор пункта меню.

`MenuFunc`

Номер функции пункта меню – это число РДС копирует в поле `Function` структуры `RDS_MENUFUNCDDATA` (стр. 63) без какой-либо обработки.

`MenuData`

Данные пункта меню – это число РДС копирует в поле `MenuData` структуры `RDS_MENUFUNCDDATA` без какой-либо обработки.

Примечания:

Эта функция вызывает модель блока `Block` для реакции на событие `RDS_BFM_MENUFUNCTION`, как будто пользователь выбрал постоянный или временный пункт меню, с которым связана пара целых чисел (`MenuFunc`, `MenuData`). Модель вызванного блока не может отличить вызов из-за того, что пользователь действительно выбрал пункт меню, от вызова из-за функции `rdsExecMenuItem`.

Вызов модели блока всегда будет производиться в главном потоке РДС, даже если `rdsExecMenuItem` была вызвана в потоке расчета (в этом случае вызов будет отложен до завершения текущего такта).

Для успешной работы этой функции не обязательно, чтобы пункт меню, с которым связана пара чисел (`MenuFunc`, `MenuData`), существовал в действительности. Это позволяет модели блока, например, использовать `rdsExecMenuItem` для вызова самой себя в главном потоке из потока расчета. Допустим, существует некая функция `DoSomething()`, которую можно вызывать только в главном потоке РДС. Если ее нужно вызвать из реакции на такт расчета `RDS_BFM_MODEL`, этот вызов можно организовать так (приведен только внутренний оператор `switch` функции модели блока, см. стр. 26):

```
switch(CallMode)  
{ case RDS_BFM_MODEL:  
    if(какое-то условие)  
        rdsExecMenuItem(BlockData->Block,10000,0);  
    break;  
case RDS_BFM_MENUFUNCTION:  
    if(((RDS_PMENUFUNCDDATA)ExtParam)->Function  
        ==10000)  
        DoSomething();  
    break;  
}
```

В данном случае в такте расчета имитируется выбор пункта меню с номером функции 10000, а в реакции на этот пункт меню (которая всегда выполняется в главном потоке РДС) вызывается функция `DoSomething()`.

См. также:

`RDS_BFM_MENUFUNCTION`, `RDS_MENUFUNCDATA` (стр. 63),
`rdsctrlBlockMenuClick` (стр. 616).

A.5.17.6. `rdsRegisterContextMenuItem` – создать постоянный пункт контекстного меню блока

Функция `rdsRegisterContextMenuItem` добавляет в контекстное меню блока, из модели которого она вызвана, постоянный пункт с указанным текстом. Это устаревшая сервисная функция, сейчас вместо нее обычно используется функция `rdsRegisterContextMenuItemEx` (стр. 361), обладающая большими возможностями.

```
RDS_MENUITEM RDSCALL rdsRegisterContextMenuItem(  
    LPSTR Caption,    // Текст пункта меню  
    int MenuFunc,     // Номер функции пункта меню  
    int MenuData      // Данные пункта меню  
);
```

Тип указателя на эту функцию:

`RDS_MhSII`

Параметры:

`Caption`

Указатель на строку с текстом пункта меню – этот текст пользователь увидит в самом меню.

`MenuFunc`

Номер функции пункта меню – при выборе пункта пользователем РДС копирует это число в поле `Function` структуры `RDS_MENUFUNCDATA` (стр. 63) без какой-либо обработки.

`MenuData`

Данные пункта меню – при выборе пункта пользователем РДС копирует это число в поле `MenuData` структуры `RDS_MENUFUNCDATA` без какой-либо обработки.

Возвращаемое значение:

Идентификатор созданного пункта меню или `NULL` при возникновении какой-либо ошибки.

Примечания:

Эта функция создает новый пункт в контекстном меню блока и возвращает его идентификатор, который можно использовать в других сервисных функциях, работающих с меню. Созданный пункт будет находиться в меню до тех пор, пока он не будет удален функцией `rdsUnregisterMenuItem` (стр. 364), или до тех пор, пока от создавшего его блока не будет отключена модель.

Добавленный пункт будет иметь текст `Caption`, с пунктом будет связана пара целых чисел (`MenuFunc`, `MenuData`). При выборе этого пункта пользователем модель блока будет вызвана для реакции на событие `RDS_BFM_MENUFUNCTION` (стр. 63), и в ее параметре `ExtParam` будет передан указатель на структуру `RDS_MENUFUNCDATA`, в полях которой будут записаны числа, связанные с выбранным пунктом меню.

Пример использования функции `rdsRegisterContextMenuItem` приведен в §2.12.6.

См. также:

`RDS_BFM_MENUFUNCTION`, `RDS_MENUFUNCDATA` (стр. 63),
`rdsRegisterContextMenuItemEx` (стр. 361),
`rdsAdditionalContextMenuItemEx` (стр. 355),
`rdsUnregisterMenuItem` (стр. 364).

A.5.17.7. `rdsRegisterContextMenuItemEx` – создать постоянный пункт контекстного меню блока

Функция `rdsRegisterContextMenuItemEx` добавляет в контекстное меню блока, из модели которого она вызвана, постоянный пункт с указанным текстом.

```
RDS_MENUITEM RDSCALL rdsRegisterContextMenuItemEx (  
    LPSTR Caption,      // Текст пункта меню  
    DWORD Options,     // флаги (RDS_MENU_*)  
    int MenuFunc,      // Номер функции пункта меню  
    int MenuData        // Данные пункта меню  
);
```

Тип указателя на эту функцию:

`RDS_MhSDwII`

Параметры:

`Caption`

Указатель на строку с текстом пункта меню – этот текст пользователь увидит в самом меню. Если в этом параметре передать `NULL`, вместо пункта меню будет создан разделитель (аналогично флагу `RDS_MENU_DIVIDER`, см. ниже).

`Options`

Один или несколько объединенных битовым ИЛИ флагов, задающих параметры пункта меню:

<code>RDS_MENU_DISABLED</code>	Выбор пункта меню будет запрещен, пункт будет изображаться серым цветом.
<code>RDS_MENU_CHECKED</code>	Слева от текста пункта меню будет находиться галочка.
<code>RDS_MENU_DIVIDER</code>	Вместо пункта меню будет изображаться горизонтальная черта-разделитель. Пользователь не может выбрать этот пункт. При установке этого флага другие флаги кроме <code>RDS_MENU_HIDDEN</code> и параметры <code>Caption</code> , <code>MenuFunc</code> и <code>MenuData</code> игнорируются.
<code>RDS_MENU_HIDDEN</code>	Пункт меню будет невидимым (скрытым от пользователя).

`MenuFunc`

Номер функции пункта меню – при выборе пункта пользователем РДС копирует это число в поле `Function` структуры `RDS_MENUFUNCDATA` (стр. 63) без какой-либо обработки.

`MenuData`

Данные пункта меню – при выборе пункта пользователем РДС копирует это число в поле `MenuData` структуры `RDS_MENUFUNCDATA` без какой-либо обработки.

Возвращаемое значение:

Идентификатор созданного пункта меню или NULL при возникновении какой-либо ошибки.

Примечания:

Эта функция создает новый пункт в контекстном меню блока и возвращает его идентификатор, который можно использовать в других сервисных функциях, работающих с меню. Созданный пункт будет находиться в меню до тех пор, пока он не будет удален функцией `rdsUnregisterMenuItem` (стр. 364), или до тех пор, пока от создавшего его блока не будет отключена модель.

Добавленный пункт будет иметь текст `Caption`, его внешний вид будет определяться флагами в параметре `Options`, с пунктом будет связана пара целых чисел (`MenuFunc`, `MenuData`). При выборе этого пункта пользователем модель блока будет вызвана для реакции на событие `RDS_BFM_MENUFUNCTION` (стр. 63), и в ее параметре `ExtParam` будет передан указатель на структуру `RDS_MENUFUNCHDATA`, в полях которой будут записаны числа, связанные с выбранным пунктом меню.

См. также:

`RDS_BFM_MENUFUNCTION`, `RDS_MENUFUNCHDATA` (стр. 63),
`rdsRegisterContextMenuItem` (стр. 360),
`rdsUnregisterMenuItem` (стр. 364).

A.5.17.8. `rdsRegisterMenuItem` – создать пункт системного меню РДС

Функция `rdsRegisterMenuItem` добавляет новый пункт в меню РДС “Система | Дополнительно”.

```
RDS_MENUITEM RDSCALL rdsRegisterMenuItem(  
    LPSTR Caption,           // Текст пункта меню  
    DWORD Options,          // Флаги (RDS_MENU_*)  
    int ShortCutKey,         // Код "горячей клавиши"  
    DWORD ShiftFlags,       // Флаги "горячей клавиши" (RDS_K*)  
    int MenuFunc,           // Номер функции пункта меню  
    int MenuData            // Данные пункта меню  
);
```

Тип указателя на эту функцию:

`RDS_MhSDwIDwII`

Параметры:

`Caption`

Указатель на строку с текстом пункта меню – этот текст пользователь увидит в самом меню.

`Options`

Один или несколько объединенных битовым ИЛИ флагов, задающих параметры пункта меню:

<code>RDS_MENU_DISABLED</code>	Выбор пункта меню будет запрещен, он будет изображаться серым цветом.
<code>RDS_MENU_CHECKED</code>	Слева от текста пункта меню будет находиться галочка.
<code>RDS_MENU_HIDDEN</code>	Пункт меню будет невидимым (скрытым от пользователя).

RDS_MENU_SHORTCUT	У пункта меню будет “горячая клавиша”, определяемая параметрами ShortCutKey и ShiftFlags.
RDS_MENU_UNIQUECAPTION	Если пункт с текстом Caption уже есть в меню, новый пункт добавлен не будет.

ShortCutKey

Код “горячей клавиши” пункта меню. Это одна из стандартных констант VK_*, используемых в Windows API.

ShiftFlags

Набор битовых флагов RDS_K* (см. стр. 61), указывающих на сочетание “горячей клавиши” ShortCutKey со служебными клавишами Ctrl, Alt и Shift.

MenuFunc

Номер функции пункта меню – при выборе пункта пользователем РДС копирует это число в поле Function структуры RDS_MENUFUNCDATA (стр. 63) без какой-либо обработки.

MenuData

Данные пункта меню – при выборе пункта пользователем РДС копирует это число в поле MenuData структуры RDS_MENUFUNCDATA без какой-либо обработки.

Возвращаемое значение:

Уникальный идентификатор созданного пункта меню или NULL при невозможности добавления этого пункта.

Примечания:

Эта функция создает новый пункт в системном меню РДС и возвращает его идентификатор, который можно использовать в других сервисных функциях, работающих с меню. Созданный пункт будет находиться в меню до тех пор, пока он не будет удален функцией rdsUnregisterMenuItem (стр. 364), или до тех пор, пока от создавшего его блока не будет отключена модель.

Добавленный пункт будет иметь текст Caption, его внешний вид будет определяться флагами в параметре Options, с пунктом будет связана пара целых чисел (MenuFunc, MenuData). При выборе этого пункта пользователем модель блока будет вызвана для реакции на событие RDS_BFM_MENUFUNCTION (стр. 63), и в ее параметре ExtParam будет передан указатель на структуру RDS_MENUFUNCDATA, в полях которой будут записаны числа, связанные с выбранным пунктом меню.

Пример использования функции rdsRegisterMenuItem приведен в §2.12.7.

См. также:

RDS_BFM_MENUFUNCTION, RDS_MENUFUNCDATA (стр. 63),
 rdsChangeMenuItem (стр. 356), rdsSetMenuItemOptions (стр. 363),
 rdsEnableMenuItem (стр. 358), rdsUnregisterMenuItem (стр. 364).

A.5.17.9. rdsSetMenuItemOptions – установить флаги пункта меню

Функция rdsSetMenuItemOptions устанавливает флаги параметров ранее созданного постоянного пункта системного меню РДС или контекстного меню блока.

```
void RDSCALL rdsSetMenuItemOptions(  
    RDS_MENUITEM Item,           // Идентификатор пункта меню  
    DWORD Options                // флаги (RDS_MENU_*)  
);
```

Тип указателя на эту функцию:

RDS_VMhDw

Параметры:

Item

Уникальный идентификатор пункта меню (RDS_MENUITEM), параметры которого изменяются.

Options

Один или несколько объединенных битовым ИЛИ флагов, задающих параметры пункта меню (см. описание функции rdsChangeMenuItem на стр. 357 за исключением флага RDS_MENU_UNIQUECAPTION).

Примечания:

Эта функция позволяет изменять параметры постоянного пункта меню (не важно – контекстного или системного), если известен его уникальный идентификатор Item. Пример ее использования приведен в §2.12.6.

См. также:

rdsRegisterContextMenuEx (стр. 361),
rdsRegisterMenuItem (стр. 362), rdsChangeMenuItem (стр. 356),
rdsEnableMenuItem (стр. 358).

A.5.17.10. rdsUnregisterMenuItem – удалить постоянный пункт меню

Функция rdsUnregisterMenuItem удаляет ранее созданный постоянный пункт системного меню РДС или контекстного меню блока.

```
void RDSCALL rdsUnregisterMenuItem(  
    RDS_MENUITEM Item    // Идентификатор пункта меню  
);
```

Тип указателя на эту функцию:

RDS_VMh

Параметр:

Item

Уникальный идентификатор удаляемого пункта меню (RDS_MENUITEM).

Примечания:

Эта функция удаляет пункт меню (не важно, контекстного или системного) с идентификатором Item. Пример ее использования приведен в §2.12.6 и §2.12.7.

См. также:

rdsRegisterContextMenuEx (стр. 361),
rdsRegisterContextMenu (стр. 360), rdsRegisterMenuItem (стр. 362).

A.5.18. Графические функции

Описываются функции, позволяющие программно рисовать изображение блока в окне подсистемы (§2.10) и строить различные изображения на специальных панелях вспомогательного объекта создания окон (§2.7.3).

A.5.18.1. Применимость графических функций

Графические функции РДС могут вызываться только из функции модели блока и только в том случае, если модель в данный момент может что-либо рисовать. Фактически, есть всего две ситуации, в которых можно использовать графические функции:

- при программном рисовании внешнего вида блока в реакциях его модели на события RDS_BFM_DRAW (стр. 57) и RDS_BFM_DRAWADDITIONAL (стр. 59);
- при рисовании произвольных изображений на специальных панелях в модальных окнах, открываемых функцией rdsFORMShowModalServ (стр. 504).

Во всех остальных случаях вызовы графических функций игнорируются РДС. Программное рисование внешнего вида блока подробно рассматривается в §2.10, рисование на панелях модальных окон – в §2.7.3.

Во всех перечисленных выше случаях в функцию модели тем или иным способом передается контекст устройства Windows (device context, HDC), на котором необходимо нарисовать изображение, поэтому вместо графических функций РДС всегда можно использовать обычные графические функции Windows API. Функции РДС могут быть полезны тем, что они

- несколько упрощают выбор линий и заливки геометрических фигур;
- дают доступ к рисованию стандартных иконок РДС (rdsXGDrawStdIcon, стр. 368) и картинки блока (rdsXGDrawBlockPicture, стр. 367);
- в них не нужно указывать контекст устройства – РДС самостоятельно определяет, где рисовать изображение.

Вызовы графических функций РДС, графических функций Windows API и любых других функций можно совмещать в одной программе.

Во всех графических функциях РДС, как и во всех функциях Windows, координаты задаются в точках экрана, горизонтальная ось направлена слева направо, вертикальная – сверху вниз.

A.5.18.2. rdsXGArc – дуга эллипса или окружности

Функция rdsXGArc рисует дугу эллипса (окружности), если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGArc (  
    int Left,int Top,           // Левый верхний угол  
    int Right,int Bottom,      // Правый нижний угол  
    int x3,int y3,             // Первый радиус  
    int x4,int y4             // Второй радиус  
);
```

Тип указателя на эту функцию:

```
RDS_VIIIIIIII
```

Параметры:

Left, Top

Левый верхний угол (Left – горизонтальная координата, Top – вертикальная) прямоугольной области, заключающей в себя эллипс, дуга которого изображается.

Right, Bottom

Правый нижний угол (Right – горизонтальная координата, Bottom – вертикальная) прямоугольной области эллипса.

x3, y3

Горизонтальная (x3) и вертикальная (y3) координаты линии начала дуги (см. ниже).

x4, y4

Горизонтальная (x4) и вертикальная (y4) координаты линии конца дуги (см. ниже).

Примечания:

Эта функция рисует дугу эллипса, вписанного в прямоугольник (Left, Top) – (Right, Bottom). Дуга начинается от точки пересечения линии, соединяющей центр этого прямоугольника с точкой (x3, y3) и рисуется против часовой стрелки до точки пересечения линии, соединяющей центр прямоугольника с точкой (x4, y4). Рисование производится текущим выбранным стилем линии. Запомненная текущая точка рисования не изменяется.

См. также:

rdsXGSetPenStyle (стр. 382), rdsXGEllipse (стр. 369), rdsXGChord (стр. 366), rdsXGPie (стр. 374).

A.5.18.3. rdsXGChord – сегмент эллипса или окружности

Функция rdsXGChord рисует сегмент эллипса (окружности), если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGChord(  
    int Left, int Top,           // Левый верхний угол  
    int Right, int Bottom,      // Правый нижний угол  
    int x3, int y3,             // Первый радиус  
    int x4, int y4              // Второй радиус  
);
```

Тип указателя на эту функцию:

RDS_VIIIIIIII

Параметры:

Left, Top

Левый верхний угол (Left – горизонтальная координата, Top – вертикальная) прямоугольной области, заключающей в себя эллипс, сегмент которого изображается.

Right, Bottom

Правый нижний угол (Right – горизонтальная координата, Bottom – вертикальная) прямоугольной области эллипса.

x3, y3

Горизонтальная (x3) и вертикальная (y3) координаты линии начала сегмента (см. ниже).

x4, y4

Горизонтальная (x4) и вертикальная (y4) координаты линии конца сегмента (см. ниже).

Примечания:

Эта функция рисует сегмент эллипса, вписанного в прямоугольник (Left, Top) – (Right, Bottom). Дуга сегмента начинается от точки пересечения линии, соединяющей центр этого прямоугольника с точкой (x3, y3) и рисуется против часовой стрелки до точки

пересечения линии, соединяющей центр прямоугольника с точкой (x4,y4). Рисование производится текущими выбранными стилями линии и заливки. Запомненная текущая точка рисования не изменяется.

См. также:

rdsXGSetPenStyle (стр. 382), rdsXGSetBrushStyle (стр. 377),
rdsXGEllipse (стр. 369), rdsXGArc (стр. 365), rdsXGPie (стр. 374).

A.5.18.4. rdsXGDrawBlockPicture – рисование картинки блока

Функция rdsXGDrawBlockPicture рисует по указанным координатам векторную картинку блока, модель которого вызвала эту функцию (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGDrawBlockPicture(  
    int X,int Y,        // Координаты картинки  
    double Angle,      // Угол поворота  
    double Scale,      // Масштабный к-т  
    double Zoom,       // Увеличение  
    BOOL UseVars       // Использовать переменные  
);
```

Тип указателя на эту функцию:

RDS_VIIDDB

Параметры:

X, Y

Горизонтальная (X) и вертикальная (Y) координаты картинки (в этой точке будет находиться начало координат картинки блока со всеми ее элементами).

Angle

Угол поворота картинки в радианах. Положительное направление – против часовой стрелки.

Scale

Масштабный множитель картинки в долях единицы: 1.0 – исходный размер.

Zoom

Увеличение (общий масштаб) картинки в долях единицы (1.0 – масштаб 100%).

UseVars

TRUE – при рисовании учитывать связь элементов картинки с переменными блока,
FALSE – игнорировать связь с переменными.

Примечания:

Эта функция рисует векторную картинку блока, из модели которого она вызвана, размещая ее начало координат в точке (X,Y). Если для блока не задана картинка, вызов функции игнорируется.

Параметр Angle задает угол поворота картинки, параметры Scale и Zoom – ее масштаб: размеры масштабируемых элементов картинки умножаются на произведение Scale и Zoom, а размеры элементов, масштабирование которых запрещено – только на Zoom. Можно считать, что Zoom – это визуальный масштаб (аналогичный масштабу окна подсистемы), а Scale – множитель, используемый для увеличения и уменьшения картинки в каких-то других целях.

Параметр UseVars управляет связью элементов картинки с переменными блока. Если в нем передано FALSE, картинка будет нарисована без учета этих связей, как в режиме

редактирования. Если же в нем передано TRUE, то видимость, масштаб, поворот и цвет элементов могут зависеть от значений переменных, как в режимах моделирования и расчета.

После вызова функции `rdsXGDrawBlockPicture` запомненная текущая точка рисования может измениться.

A.5.18.5. `rdsXGDrawStdIcon` – рисование стандартной иконки

Функция `rdsXGDrawStdIcon` рисует одну из стандартных иконок РДС, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGDrawStdIcon(  
    int Left,int Top,        // Левый верхний угол иконки  
    DWORD Icon              // Идентификатор иконки (RDS_STDICON_*)  
);
```

Тип указателя на эту функцию:

`RDS_VIIDw`

Параметры:

`Left,Top`

Левый верхний угол (`Left` – горизонтальная координата, `Top` – вертикальная) изображаемой иконки.

`Icon`

Идентификатор изображаемой иконки – одна из констант `RDS_STDICON_*`:

<code>RDS_STDICON_BLOCK</code>	Маленькое изображение блока (белый квадрат)
<code>RDS_STDICON_DISABLEDCONN</code>	Иконка запрещения связи (знак “проезд запрещен”)
<code>RDS_STDICON_EYE</code>	Глаз (как в колонке видимости слоя редактора слоев)
<code>RDS_STDICON_GREENSQUARE</code>	Зеленый квадрат
<code>RDS_STDICON_PENCIL</code>	Карандаш (как в колонке разрешения изменения слоя в редакторе слоев)
<code>RDS_STDICON_REDCIRCEXCLAM</code>	Восклицательный знак в красном круге
<code>RDS_STDICON_REDGEAR</code>	Красная шестеренка
<code>RDS_STDICON_REDSQUARE</code>	Красный квадрат
<code>RDS_STDICON_REDTRIEXCLAM</code>	Восклицательный знак в красном треугольнике
<code>RDS_STDICON_RUN</code>	Знак запуска (черный треугольник вершиной вправо)
<code>RDS_STDICON_STOP</code>	Знак остановки (черный квадрат)
<code>RDS_STDICON_SYSTEM</code>	Маленькое изображение подсистемы (белый квадрат с квадратами внутри)
<code>RDS_STDICON_YELCIRCEXCLAM</code>	Восклицательный знак в желтом круге
<code>RDS_STDICON_YELLOWGEAR</code>	Желтая шестеренка
<code>RDS_STDICON_YELLOWQUESTION</code>	Вопросительный знак в желтом круге
<code>RDS_STDICON_YELLOW SQUARE</code>	Желтый квадрат

Примечания:

Эта функция рисует одну из стандартных иконок РДС, располагая ее левый верхний угол в точке (`Left,Top`). Чаще всего такие иконки выводятся поверх изображений блоков для индикации каких-либо проблем. На данный момент все стандартные иконки имеют размер 16x16 точек, однако, если вызывающей программе нужно знать размер иконки

(например, для выравнивания ее изображения по изображению блока), лучше всего воспользоваться функцией `rdsXGGetStdIconSize` (стр. 371).

Пример использования функции `rdsXGDrawStdIcon` приведен в §2.10.3.

См. также:

`rdsXGGetStdIconSize` (стр. 371).

A.5.18.6. `rdsXGEllipse` – эллипс или окружность

Функция `rdsXGEllipse` рисует эллипс (окружность), если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGEllipse(  
    int Left,int Top,        // Левый верхний угол  
    int Right,int Bottom    // Правый нижний угол  
);
```

Тип указателя на эту функцию:

`RDS_VIIII`

Параметры:

`Left,Top`

Левый верхний угол (`Left` – горизонтальная координата, `Top` – вертикальная) прямоугольной области, заключающей в себя эллипс.

`Right,Bottom`

Правый нижний угол (`Right` – горизонтальная координата, `Bottom` – вертикальная) прямоугольной области эллипса.

Примечания:

Эта функция рисует эллипс, вписанный в прямоугольник (`Left,Top`) – (`Right,Bottom`). Оси эллипса параллельны осям координат (эллипс не может быть повернутым). Рисование производится текущими выбранными стилями линии и заливки. Запомненная текущая точка рисования не изменяется. Пример использования функции приведен в §2.12.2.

См. также:

`rdsXGSetPenStyle` (стр. 382), `rdsXGSetBrushStyle` (стр. 377),
`rdsXGChord` (стр. 366), `rdsXGArc` (стр. 365), `rdsXGPie` (стр. 374).

A.5.18.7. `rdsXGFillRect` – заполненный прямоугольник

Функция `rdsXGFillRect` заполняет указанный прямоугольник текущей выбранной заливкой, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGFillRect(  
    int Left,int Top,        // Левый верхний угол  
    int Right,int Bottom    // Правый нижний угол  
);
```

Тип указателя на эту функцию:

`RDS_VIIII`

Параметры:

Left, Top

Левый верхний угол (Left – горизонтальная координата, Top – вертикальная) закрашиваемой прямоугольной области.

Right, Bottom

Правый нижний угол (Right – горизонтальная координата, Bottom – вертикальная) прямоугольной области.

Примечания:

Эта функция закрашивает прямоугольную область (Left, Top) – (Right, Bottom) цветом и стилем, выбранными функцией `rdsXGSetBrushStyle` (стр. 377). От функции рисования прямоугольника `rdsXGRectangle` (стр. 376) она отличается тем, что изображаемый ей прямоугольник не имеет рамки независимо от текущих установок стиля линии. Запомненная текущая точка рисования не изменяется. Пример использования функции приведен в §2.7.3 и §2.10.1.

См. также:

`rdsXGSetBrushStyle` (стр. 377), `rdsXGRectangle` (стр. 376),
`rdsXGInvertRect` (стр. 373).

A.5.18.8. `rdsXGFontSizeToHeight` – размер шрифта в высоту в точках

Функция `rdsXGFontSizeToHeight` переводит размер шрифта в типографских точках (points) в его высоту в точках экрана.

```
int RDSCALL rdsXGFontSizeToHeight(  
    int Size          // Размер шрифта  
);
```

Тип указателя на эту функцию:

RDS_II

Параметр:

Size

Размер шрифта в типографских точках.

Возвращаемое значение:

Высота шрифта в точках экрана.

Примечания:

Эта функция вычисляет высоту шрифта указанного размера в точках экрана, поэтому она зависит от настроек дисплея в Windows. Высота шрифта связана с его размером следующей формулой:

$$\text{высота} = \text{размер} \times \text{точек_на_дюйм} / 72,$$

где *точек_на_дюйм* – число точек экрана, приходящееся на один дюйм (именно этот параметр зависит от настроек Windows).

Эта функция применяется редко. Гораздо чаще используется `rdsXGGetTextSize` (стр. 371), позволяющая узнать высоту и ширину в точках экрана для конкретного текста, выведенного текущим выбранным шрифтом.

См. также:

`rdsXGGetTextSize` (стр. 371).

A.5.18.9. rdsXGGetStdIconSize – получить размеры стандартной иконки

Функция `rdsXGGetStdIconSize` возвращает размеры одной из стандартных иконок РДС.

```
BOOL RDSCALL rdsXGGetStdIconSize(  
    DWORD Icon,          // Идентификатор иконки (RDS_STDICON_*)  
    int *pWidth,         // Возвращаемая ширина  
    int *pHeight        // Возвращаемая высота  
);
```

Тип указателя на эту функцию:

`RDS_BDwpIpI`

Параметры:

`Icon`

Идентификатор изображаемой иконки – одна из констант `RDS_STDICON_*` (см. стр. 368).

`pWidth`

Указатель на целую переменную, в которую функция должна записать ширину иконки в точках экрана. Если вызывающей программе не нужна ширина иконки, в этом параметре можно передать `NULL`.

`pHeight`

Указатель на целую переменную, в которую функция должна записать высоту иконки в точках экрана. Если вызывающей программе не нужна высота иконки, в этом параметре можно передать `NULL`.

Возвращаемое значение:

`TRUE` – размеры иконки считаны, `FALSE` – в РДС нет иконки с идентификатором `Icon`.

Примечания:

Эта функция возвращает размеры одной из стандартных иконок, которую можно нарисовать вызовом `rdsXGDrawStdIcon` (стр. 368). На данный момент все стандартные иконки имеют размер 16x16 точек, но в будущих версиях РДС могут появиться и иконки другого размера, поэтому при рисовании не следует полагаться на заранее известный размер иконки.

Пример использования функции `rdsXGDrawStdIcon` приведен в §2.10.3.

См. также:

`rdsXGDrawStdIcon` (стр. 368).

A.5.18.10. rdsXGGetTextSize – получить размеры строки текста

Функция `rdsXGGetTextSize` возвращает ширину и высоту указанной строки текста в точках экрана, если вывести ее текущим установленным шрифтом. На момент вызова этой функции модель блока должна быть вызвана для рисования чего-либо (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGGetTextSize(  
    LPSTR Text,          // Строка  
    int *pWidth,         // Возвращаемая ширина  
    int *pHeight        // Возвращаемая высота  
);
```

Тип указателя на эту функцию:

RDS_VSpIpI

Параметры:

Text

Указатель на строку текста, размеры которой определяются.

pWidth

Указатель на целую переменную, в которую функция должна записать ширину строки в точках экрана. Если вызывающей программе не нужна ширина строки, в этом параметре можно передать NULL.

pHeight

Указатель на целую переменную, в которую функция должна записать высоту строки в точках экрана. Если вызывающей программе не нужна высота строки, в этом параметре можно передать NULL.

Примечания:

Эта функция возвращает размеры строки, если ее вывести текущим установленным шрифтом. Чаще всего она применяется для выравнивания выводимой строки относительно других графических объектов: сначала определяются размеры строки, затем вычисляются координаты ее левого верхнего угла, после чего строка выводится функциями rdsXGTextOut (стр. 384) или rdsXGTextRect (стр. 385).

Пример использования функции rdsXGGetTextSize приведен в §2.10.1.

См. также:

rdsXGSetFont (стр. 379), rdsXGSetFontByParStr (стр. 381),
rdsXGSetLogFont (стр. 381), rdsXGTextOut (стр. 384),
rdsXGTextRect (стр. 385).

A.5.18.11. rdsXGGetVisibleRect – получить координаты видимой области

Функция rdsXGGetVisibleRect возвращает координаты прямоугольной области рисования, видимой в данный момент, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGGetVisibleRect(  
    RECT *pRect    // Заполняемая структура координат  
);
```

Тип указателя на эту функцию:

RDS_VpR

Параметр:

pRect

Указатель на структуру описания прямоугольника RECT Windows API, в которую функция записывает координаты видимой области.

Примечания:

Эта функция возвращает координаты прямоугольной области, за пределами которой можно ничего не рисовать. При рисовании внешнего вида блока эта область соответствует видимой в окне части рабочего поля, при рисовании на панелях модальных окон – размерам панели и т.п. Модели блока не обязательно программно отсекают изображения за пределами этой области – это выполняется автоматически. Чаще всего функция

rdsXGGetVisibleRect используется в моделях блоков, занимающих большую площадь на рабочем поле и строящих сложные изображения (например, географические карты). Проверка на попадание элементов изображения блока в видимую область и пропуск рисования не попавших в нее позволяет существенно ускорить рисование, особенно при крупных масштабах окна подсистемы.

A.5.18.12. rdsXGInvertRect – инвертировать прямоугольник

Функция rdsXGInvertRect инвертирует цвета в указанном прямоугольнике, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGInvertRect(  
    int Left,int Top,      // Левый верхний угол  
    int Right,int Bottom // Правый нижний угол  
);
```

Тип указателя на эту функцию:

RDS_VIIII

Параметры:

Left,Top

Левый верхний угол (Left – горизонтальная координата, Top – вертикальная) инвертируемой прямоугольной области.

Right,Bottom

Правый нижний угол (Right – горизонтальная координата, Bottom – вертикальная) прямоугольной области.

Примечания:

Эта функция инвертирует (обращает) цвета всех точек в прямоугольной области (Left,Top)–(Right,Bottom). Инвертирование цветов может быть полезно при рисовании каких-либо маркеров выделения. Запомненная текущая точка рисования не изменяется.

См. также:

rdsXGRectangle (стр. 376), rdsXGFillRect (стр. 369).

A.5.18.13. rdsXGLineTo – отрезок прямой

Функция rdsXGLineTo рисует отрезок прямой от текущей точки рисования до указанной точки, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGLineTo(  
    int X,int Y          // Координаты конца отрезка  
);
```

Тип указателя на эту функцию:

RDS_VII

Параметры:

X, Y

Горизонтальная (X) и вертикальная (Y) координаты точки конца отрезка.

Примечания:

Эта функция строит отрезок прямой между текущей точкой рисования и точкой (X,Y). После этого точка (X,Y) становится новой текущей точкой рисования. Таким образом последовательные вызовы rdsXGLineTo рисуют ломаную линию, при этом самую первую

точку ломаной можно задать вызовом `rdsXGMoveTo` (стр. 374). Отрезок строится текущим, выбранным функцией `rdsXGSetPenStyle` (стр. 382), стилем линии.

Примеры использования функции `rdsXGLineTo` приведены в §2.7.3 и §2.10.1.

См. также:

`rdsXGMoveTo` (стр. 374), `rdsXGSetPenStyle` (стр. 382).

A.5.18.14. `rdsXGMoveTo` – установить текущую точку рисования

Функция `rdsXGMoveTo` устанавливает координаты текущей точки рисования, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGMoveTo(  
    int X,int Y      // Координаты точки  
);
```

Тип указателя на эту функцию:

`RDS_VII`

Параметры:

`X, Y`

Горизонтальная (`X`) и вертикальная (`Y`) координаты точки.

Примечания:

Эта функция делает текущей точкой рисования точку с координатами (`X,Y`). После этого вызовом `rdsXGLineTo` (стр. 373) можно нарисовать отрезок прямой, начинающийся в этой точке. Примеры использования функции `rdsXGMoveTo` приведены в §2.7.3 и §2.10.1.

См. также:

`rdsXGLineTo` (стр. 373).

A.5.18.15. `rdsXGPie` – сектор эллипса или окружности

Функция `rdsXGPie` рисует сектор эллипса (окружности), если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGPie(  
    int Left,int Top,      // Левый верхний угол  
    int Right,int Bottom, // Правый нижний угол  
    int x3,int y3,        // Первый радиус  
    int x4,int y4         // Второй радиус  
);
```

Тип указателя на эту функцию:

`RDS_VIIIIIIII`

Параметры:

`Left, Top`

Левый верхний угол (`Left` – горизонтальная координата, `Top` – вертикальная) прямоугольной области, заключающей в себя эллипс, сектор которого изображается.

`Right, Bottom`

Правый нижний угол (`Right` – горизонтальная координата, `Bottom` – вертикальная) прямоугольной области эллипса.

x3, y3

Горизонтальная (x3) и вертикальная (y3) координаты линии начала сектора (см. ниже).

x4, y4

Горизонтальная (x4) и вертикальная (y4) координаты линии конца сектора (см. ниже).

Примечания:

Эта функция рисует сектор эллипса, вписанного в прямоугольник (Left,Top) – (Right,Bottom). Сектор ограничен линиями, соединяющими центр этого прямоугольника с точками (x3,y3) и (x4,y4) и дугой эллипса между этими прямыми. Рисование производится текущими выбранными стилями линии и заливки. Запомненная текущая точка рисования не изменяется.

См. также:

rdsXGSetPenStyle (стр. 382), rdsXGSetBrushStyle (стр. 377),
rdsXGEllipse (стр. 369), rdsXGArc (стр. 365), rdsXGChord (стр. 366).

A.5.18.16. rdsXGPolygon – многоугольник

Функция rdsXGPolygon рисует многоугольник, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGPolygon(  
    POINT *pPoints, // Массив точек  
    int Count       // Число точек  
);
```

Тип указателя на эту функцию:

RDS_VpPI

Параметры:

pPoints

Массив (указатель на его первый элемент) структур Windows API POINT, в которых размещаются координаты точек многоугольника.

Count

Общее число точек в массиве pPoints.

Примечания:

Эта функция рисует замкнутый многоугольник, координаты точек которого переданы в массиве pPoints. Многоугольник автоматически замыкается отрезком, соединяющим точку pPoints[Count-1] с точкой pPoints[0]. Рисование производится текущими выбранными стилями линии и заливки. Запомненная текущая точка рисования не изменяется.

См. также:

rdsXGSetPenStyle (стр. 382), rdsXGSetBrushStyle (стр. 377),
rdsXGPolyline (стр. 375), rdsXGTriangle (стр. 385).

A.5.18.17. rdsXGPolyline – ломаная линия

Функция rdsXGPolyline рисует ломаную линию, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```

void RDSCALL rdsXGPolyline(
    POINT *pPoints, // Массив точек
    int Count       // Число точек
);

```

Тип указателя на эту функцию:

RDS_VpPI

Параметры:

pPoints

Массив (указатель на его первый элемент) структур Windows API POINT, в которых размещаются координаты точек ломаной линии.

Count

Общее число точек в массиве pPoints.

Примечания:

Эта функция рисует незамкнутую ломаную линию, координаты точек которой переданы в массиве pPoints. Для рисования замкнутой ломаной линии можно использовать функцию рисования многоугольника rdsXGPolygon (стр. 375), предварительно выключив заливку вызовом функции rdsXGSetBrushStyle (стр. 377). Рисование производится текущим выбранным стилем линии. Запомненная текущая точка рисования не изменяется.

См. также:

rdsXGSetPenStyle (стр. 382), rdsXGPolygon (стр. 375).

A.5.18.18. rdsXGRectangle – прямоугольник

Функция rdsXGRectangle рисует прямоугольник, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```

void RDSCALL rdsXGRectangle(
    int Left,int Top,      // Левый верхний угол
    int Right,int Bottom // Правый нижний угол
);

```

Тип указателя на эту функцию:

RDS_VIIII

Параметры:

Left,Top

Левый верхний угол (Left – горизонтальная координата, Top – вертикальная) прямоугольника.

Right,Bottom

Правый нижний угол (Right – горизонтальная координата, Bottom – вертикальная) прямоугольника.

Примечания:

Эта функция рисует прямоугольник (Left,Top) – (Right,Bottom). Рисование производится текущими выбранными стилями линии и заливки. Запомненная текущая точка рисования не изменяется. Пример использования функции приведен в §2.10.1.

См. также:

rdsXGSetPenStyle (стр. 382), rdsXGSetBrushStyle (стр. 377),
rdsXGFillRect (стр. 369), rdsXGInvertRect (стр. 373),
rdsXGRoundRect (стр. 377).

A.5.18.19. rdsXGRoundRect – скругленный прямоугольник

Функция rdsXGRoundRect рисует прямоугольник со скругленными углами, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGRoundRect(  
    int Left,int Top,        // Левый верхний угол  
    int Right,int Bottom,    // Правый нижний угол  
    int XRound,              // Ширина скругления  
    int YRound               // Высота скругления  
);
```

Тип указателя на эту функцию:

RDS_VIIIIIII

Параметры:

Left, Top

Левый верхний угол (Left – горизонтальная координата, Top – вертикальная) прямоугольника.

Right, Bottom

Правый нижний угол (Right – горизонтальная координата, Bottom – вертикальная) прямоугольника.

XRound

Ширина эллипса, используемого для скругления углов.

YRound

Высота эллипса, используемого для скругления углов.

Примечания:

Эта функция рисует прямоугольник (Left,Top) – (Right,Bottom), стороны которого сопрягаются эллипсами размером XRound x YRound. Рисование производится текущими выбранными стилями линии и заливки. Запомненная текущая точка рисования не изменяется.

См. также:

rdsXGSetPenStyle (стр. 382), rdsXGSetBrushStyle (стр. 377),
rdsXGRectangle (стр. 376).

A.5.18.20. rdsXGSetBrushStyle – установить стиль заливки

Функция rdsXGSetBrushStyle устанавливает текущий стиль заливки, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365). Новая заливка будет использоваться для всех геометрических фигур, рисуемых после этого вызова.

```
void RDSCALL rdsXGSetBrushStyle(  
    int Mask,                // Маска установки (RDS_GF*) или 0  
    int Style,              // Стиль заливки (RDS_GFS_*)  
    COLORREF Color         // Цвет заливки  
);
```

Тип указателя на эту функцию:

RDS_VIICr

Параметры:

Mask

Маска устанавливаемых параметров или 0, если нужно одновременно установить и стиль, и цвет заливки. В этом параметре могут передаваться следующие константы, объединенные битовым ИЛИ:

RDS_GFSTYLE Установка стиля заливки из параметра Style.

RDS_GFCOLOR Установка цвета заливки из параметра Color.

Style

Стиль заливки – одна из следующих констант:

RDS_GFS_EMPTY Нет заливки (внутренняя часть геометрических фигур будет прозрачной).

RDS_GFS_SOLID Сплошная заливка.

RDS_GFS_BDIAGONAL Диагональные линии слева снизу вправо вверх.

RDS_GFS_CROSS Вертикальные и горизонтальные линии (клетка).

RDS_GFS_DIAGCROSS Диагональная клетка.

RDS_GFS_FDIAGONAL Диагональные линии слева сверху вправо вниз.

RDS_GFS_HORIZONTAL Горизонтальные линии.

RDS_GFS_VERTICAL Вертикальные линии.

Color

Цвет заливки (COLORREF, см. стр. 24). В стиле RDS_GFS_EMPTY не используется.

Примечания:

Эта функция устанавливает текущий стиль заливки, который будет использоваться при рисовании замкнутых геометрических фигур. В параметре Style передается сам стиль, в параметре Color – цвет заливки. В параметре Mask передается набор битовых флагов, определяющих, какие из двух параметров заливки будут изменены. Если в Mask передано нулевое значение, будет изменен и стиль, и цвет. Например, чтобы установить сплошную красную заливку, можно сделать такой вызов:

```
rdsXGSetBrushStyle(0, RDS_GFS_SOLID, 0xff);
```

Для изменения цвета заливки на синий без изменения стиля можно сделать следующий вызов:

```
rdsXGSetBrushStyle(RDS_GFCOLOR, 0, 0xff0000);
```

Отключить заливку (сделать ее прозрачной) можно вызовом

```
rdsXGSetBrushStyle(0, RDS_GFS_EMPTY, 0);
```

Примеры использования функции rdsXGSetBrushStyle приведены в §2.7.3 и §2.10.1.

См. также:

rdsXGSetPenStyle (стр. 382), rdsXGSetFont (стр. 379).

A.5.18.21. rdsXGSetClipRect – установить область отсечения

Функция rdsXGSetClipRect устанавливает прямоугольную область отсечения рисования, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```

void RDSCALL rdsXGSetClipRect(
    RECT *pRect      // Описание области или NULL
);

```

Тип указателя на эту функцию:

RDS_VpR

Параметр:

pRect

Указатель на структуру описания прямоугольника RECT Windows API, в которой записаны координаты области отсечения. Если в этом параметре передано значение NULL, отсечение будет отменено.

Примечания:

Эта функция устанавливает координаты прямоугольной области, за пределами которой не будет производиться рисование. У всех геометрических фигур будут рисоваться только те части, которые попадают внутрь области. Если в параметре pRect передать не указатель на структуру, описывающую область, а NULL, отсечение будет отменено. Область отсечения может, например, использоваться при построении графиков, чтобы линия графика не выходила за пределы координатной сетки.

Если область отсечения должна иметь форму, отличную от прямоугольника, следует использовать соответствующие функции Windows API.

Пример использования функции приведен в §2.10.1.

A.5.18.22. rdsXGSetFont – установить шрифт

Функция rdsXGSetFont устанавливает параметры шрифта, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365). Этот шрифт будет использоваться при выводе всех последующих текстов.

```

void RDSCALL rdsXGSetFont(
    int Mask,           // Маска установки (RDS_GF*) или 0
    LPSTR Name,        // Имя шрифта
    int SizeHeight,    // Высота или размер
    COLORREF Color,    // Цвет
    int Charset,       // Набор символов
    int Escapement,    // Угол поворота
    BOOL Bold,         // Жирность
    BOOL Italic,       // Курсив
    BOOL Underline,    // Подчеркивание
    BOOL StrikeOut     // Перечеркивание
);

```

Тип указателя на эту функцию:

RDS_VISICrIIBBBB

Параметры:

Mask

Набор битовых флагов, указывающих, какие параметры шрифта нужно установить, или 0, если нужно установить все параметры сразу. Можно использовать следующие флаги:

RDS_GFNAME Установить имя шрифта, переданное в параметре Name.

RDS_GFSIZE	Установить размер шрифта в типографских точках (points), переданный в параметре SizeHeight. Не может использоваться одновременно с флагом RDS_GFHEIGHT.
RDS_GFHEIGHT	Установить высоту шрифта в точках экрана, переданную в параметре SizeHeight. Не может использоваться одновременно с флагом RDS_GFSIZE.
RDS_GFCHARSET	Установить набор символов шрифта, переданный в параметре Charset.
RDS_GFESCAPEMENT	Установить угол поворота шрифта в градусах, переданный в параметре Escapement.
RDS_GFCOLOR	Установить цвет шрифта, переданный в параметре Color.
RDS_GFBOLD	Установить жирность шрифта согласно параметру Bold.
RDS_GFITALIC	Установить курсив шрифта согласно параметру Italic.
RDS_GFUNDERLINE	Установить подчеркивание шрифта согласно параметру Underline.
RDS_GFSTRIKEOUT	Установить зачеркивание шрифта согласно параметру StrikeOut.

Все эти флаги совпадают с флагами, используемыми функцией `rdsWriteFontText` (стр. 282). Кроме приведенных выше флагов, в параметре `Mask` можно указывать следующие константы, объединяющие несколько флагов вместе:

RDS_GFFONTALLHEIGHT	Все указанные выше флаги, кроме RDS_GFSIZE (размер шрифта задается высотой в точках экрана).
RDS_GFFONTBASIC	Все указанные выше флаги, кроме RDS_GFSIZE (размер шрифта задается высотой в точках экрана) и RDS_GFESCAPEMENT (угол поворота не устанавливается).
RDS_GFFONTSTYLES	Жирность, курсив, зачеркивание и подчеркивание (флаги RDS_GFBOLD, RDS_GFITALIC, RDS_GFUNDERLINE и RDS_GFSTRIKEOUT).

Если в параметре `Mask` передать 0, будут установлены все параметры шрифта, высота его при этом будет считаться заданной в точках экрана, как при указании флага RDS_GFHEIGHT.

Name

Указатель на строку с именем шрифта.

SizeHeight

Размер шрифта в типографских точках (при указанном в `Mask` флаге RDS_GFSIZE) или в точках экрана (при указанном флаге RDS_GFHEIGHT).

Color

Цвет шрифта (COLORREF, см. стр. 24).

Charset

Набор символов шрифта.

Escapement

Угол поворота шрифта в градусах относительно горизонтали.

Bold

Шрифт жирный (TRUE) или обычный (FALSE).

Italic

Курсив (TRUE) или обычный шрифт (FALSE).

Underline

Шрифт подчеркнут (TRUE) или нет (FALSE).

StrikeOut

Шрифт перечеркнут (TRUE) или нет (FALSE).

Примечания:

Эта функция устанавливает параметры шрифта, которыми будут выводиться все текстовые строки во всех последующих вызовах. Пример использования функции rdsXGSetFont приведен в §2.12.3.

См. также:

rdsXGSetFontByParStr (стр. 381), rdsXGSetLogFont (стр. 381),
rdsWriteFontText (стр. 282).

A.5.18.23. rdsXGSetFontByParStr – установить шрифт по структуре описания

Функция rdsXGSetFontByParStr устанавливает параметры шрифта согласно переданной структуре RDS_SERVFONTPARAMS (стр. 135), если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365). Этот шрифт будет использоваться при выводе всех последующих текстов.

```
void RDSCALL rdsXGSetFontByParStr(  
    RDS_PSERVFONTPARAMS pPar,    // Структура описания  
    double Scale                  // Масштаб  
);
```

Тип указателя на эту функцию:

RDS_VpFsD

Параметры:

pPar

Указатель на структуру описания шрифта RDS_SERVFONTPARAMS.

Scale

Масштабный множитель размера шрифта.

Примечания:

Эта функция устанавливает параметры шрифта, читая их из полей структуры, переданной в параметре pPar. Этим шрифтом будут выводиться все текстовые строки во всех последующих вызовах. При установке размер шрифта умножается на параметр Scale, что позволяет легко менять масштаб текста без изменения полей структуры. Пример использования функции rdsXGSetFontByParStr приведен в §2.10.1.

См. также:

RDS_SERVFONTPARAMS (стр. 135), rdsXGSetFont (стр. 379),
rdsXGSetLogFont (стр. 381).

A.5.18.24. rdsXGSetLogFont – установить шрифт по структуре LOGFONT

Функция rdsXGSetLogFont устанавливает параметры шрифта согласно переданной структуре LOGFONT (стр. 24), если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365). Этот шрифт будет использоваться при выводе всех последующих текстов.

```
void RDSCALL rdsXGSetLogFont(  
    LOGFONT *pLogFont,    // Структура описания
```

```

        COLORREF Color           // Цвет шрифта
    );

```

Тип указателя на эту функцию:

```
RDS_VpLfCr
```

Параметры:

pLogFont

Указатель на структуру описания шрифта LOGFONT.

Color

Цвет шрифта (COLORREF, см. стр. 24).

Примечания:

Эта функция устанавливает параметры шрифта, читая их из полей структуры Windows API LOGFONT, переданной в параметре pPar. Этим шрифтом будут выводиться все текстовые строки во всех последующих вызовах. Поскольку в структуре LOGFONT не предусмотрено поле для описания цвета, цвет шрифта передается в параметре Color.

См. также:

rdsXGSetFontByParStr (стр. 381), rdsXGSetFont (стр. 379).

A.5.18.25. rdsXGSetPenStyle – установить стиль линии

Функция rdsXGSetPenStyle устанавливает текущий стиль линии, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365). Этот стиль будет использоваться для всех геометрических фигур, рисуемых после этого вызова.

```

void RDSCALL rdsXGSetPenStyle(
    int Mask,           // Маска установки (RDS_GF*) или 0
    int Style,          // Стиль линии (Windows API)
    int Width,          // Толщина линии
    COLORREF Color,     // Цвет линии
    int Mode            // Режим рисования
);

```

Тип указателя на эту функцию:

```
RDS_VIIICrI
```

Параметры:

Mask

Маска устанавливаемых параметров или 0, если нужно одновременно установить все параметры. В этом параметре могут передаваться следующие битовые флаги, объединенные битовым ИЛИ:

RDS_GFCOLOR Установить цвет линии из параметра Color.

RDS_GFMODE Установить режим рисования из параметра Mode.

RDS_GFSTYLE Установить стиль линии из параметра Style.

RDS_GFWIDTH Установить толщину линии из параметра Width.

Style

Одна из стандартных констант Windows API, задающих стиль линии:

PS_DASH Пунктирная линия.

PS_DASHDOT Линия из чередующихся отрезков и точек.

PS_DASHDOTDOT Линия из повторяющихся групп “отрезок-точка-точка”.

PS_DOT Линия, состоящая из точек.

PS_INSIDEFRAME	Специальный стиль сплошной линии, разрешающий Windows скорректировать размеры геометрической фигуры, ограниченной этой линией так, чтобы она уместилась в заданный прямоугольник.
PS_NULL	Невидимая линия (геометрические фигуры будут рисоваться без линий на границах).
PS_SOLID	Сплошная линия.

Width

Толщина линии в точках экрана. В Windows толщина более одной точки поддерживается только для сплошных линий.

Color

Цвет линии (COLORREF, см. стр. 24). В стиле линии PS_NULL не используется.

Mode

Одна из стандартных констант Windows API, задающих режим рисования линии. Режим рисования определяет логическую функцию, которая комбинирует биты цвета линии и цвета экрана для получения результирующего цвета. Если *pen* – бит цвета стиля линии (из параметра Color), *screen* – бит цвета экрана в данной точке, а *res* – результирующий (получившийся в результате рисования) цвет, то при помощи параметра Mode можно задать следующие операции:

R2_BLACK	$res = 0$ – точка экрана всегда будет черной
R2_WHITE	$res = 1$ – точка экрана всегда будет белой
R2_NOP	$res = screen$ – точка экрана не меняет цвет
R2_NOT	$res = NOT\ screen$ – точка экрана инвертируется
R2_COPYPEN	$res = pen$ – точка экрана принимает цвет линии
R2_NOTCOPYPEN	$res = NOT\ pen$
R2_MERGEPEENNOT	$res = (NOT\ screen)\ OR\ pen$
R2_MASKPENNOT	$res = (NOT\ screen)\ AND\ pen$
R2_MERGEENOTPEN	$res = (NOT\ pen)\ OR\ screen$
R2_MASKNOTPEN	$res = (NOT\ pen)\ AND\ screen$
R2_MERGEPEN	$res = pen\ OR\ screen$
R2_NOTMERGEPEN	$res = NOT\ (pen\ OR\ screen)$
R2_MASKPEN	$res = pen\ AND\ screen$
R2_NOTMASKPEN	$res = NOT\ (pen\ AND\ screen)$
R2_XORPEN	$res = pen\ XOR\ screen$
R2_NOTXORPEN	$res = NOT\ (pen\ XOR\ screen)$

Здесь словом NOT обозначена инверсия бита, OR – операция ИЛИ, AND – И, XOR – исключающее ИЛИ. Из всех перечисленных выше режимов чаще всего используются R2_COPYPEN (линия рисуется цветом, заданным в параметрах стиля) и R2_NOT (линия инвертирует точки экрана).

Примечания:

Эта функция устанавливает текущий стиль линии, который будет использоваться при рисовании замкнутых геометрических фигур. В параметре Mask передается набор битовых флагов, определяющих, какие из параметров стиля будут изменены. Примеры использования функции rdsXGSetPenStyle приведены в §2.7.3 и §2.10.1.

См. также:

rdsXGSetBrushStyle (стр. 377), rdsXGSetFont (стр. 379).

A.5.18.26. rdsXGSetPixel – точка

Функция `rdsXGSetPixel` устанавливает цвет точки с указанными координатами, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGSetPixel(  
    int X,int Y        // Координаты точки  
);
```

Тип указателя на эту функцию:

`RDS_VII`

Параметры:

`X, Y`

Горизонтальная (X) и вертикальная (Y) координаты точки, цвет которой изменяется.

Примечания:

Эта функция меняет цвет точки (X,Y) на текущий цвет линии, установленный вызовом `rdsXGSetPenStyle` (стр. 382). Геометрические фигуры следует, по возможности, рисовать другими графическими функциями – вызов `rdsXGSetPixel` может выполняться достаточно медленно.

См. также:

`rdsXGSetPenStyle` (стр. 382).

A.5.18.27. rdsXGTextOut – строка текста

Функция `rdsXGTextOut` выводит по указанным координатам указанную строку текста, если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGTextOut(  
    int X,int Y,        // Координаты  
    LPSTR Text          // Строка  
);
```

Тип указателя на эту функцию:

`RDS_VIIS`

Параметры:

`X, Y`

Горизонтальная (X) и вертикальная (Y) координаты левого верхнего угла выводимой строки.

`Text`

Указатель на выводимую строку.

Примечания:

Эта функция выводит строку, переданную в параметре `Text`, располагая левый верхний угол занимаемой ей области в точке (X,Y). Строка выводится текущим шрифтом, прямоугольная область под ней заполняется текущим цветом заливки. Функция выводит только одну строку, многострочные тексты не поддерживаются. После выполнения этой функции текущая точка рисования перемещается в правый верхний угол выведенной строки, поэтому `rdsXGTextOut` нельзя использовать внутри циклов, строящих ломаные линии функцией `rdsXGLineTo` (стр. 373).

Пример использования функции `rdsXGTextOut` приведен в §2.10.1.

См. также:

`rdsXGSetFont` (стр. 379), `rdsXGSetBrushStyle` (стр. 377),
`rdsXGTextRect` (стр. 385).

A.5.18.28. `rdsXGTextRect` – строка текста с отсечением

Функция `rdsXGTextRect` выводит по указанным координатам указанную строку, ограничивая ее прямоугольной областью. Модель блока при этом должна быть вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGTextRect(  
    int X,int Y,        // Координаты  
    LPSTR Text,        // Строка  
    RECT *pRect        // Область отсечения  
);
```

Тип указателя на эту функцию:

`RDS_VIISpR`

Параметры:

`X, Y`

Горизонтальная (X) и вертикальная (Y) координаты левого верхнего угла выводимой строки.

`Text`

Указатель выводимую на строку.

`pRect`

Указатель на структуру описания прямоугольника `RECT` Windows API, который будет использован для ограничения вывода строки.

Примечания:

Эта функция выводит строку, переданную в параметре `Text`, располагая левый верхний угол занимаемой ей области в точке (X,Y). Часть строки, вышедшая за пределы прямоугольной области `pRect`, выведена не будет. Строка выводится текущим шрифтом, прямоугольная область заполняется текущим цветом заливки. Функция выводит только одну строку, многострочные тексты не поддерживаются. Функция изменяет текущую точку рисования, поэтому `rdsXGTextRect` нельзя использовать внутри циклов, строящих ломаные линии функцией `rdsXGLineTo` (стр. 373).

См. также:

`rdsXGSetFont` (стр. 379), `rdsXGSetBrushStyle` (стр. 377),
`rdsXGTextOut` (стр. 384).

A.5.18.29. `rdsXGTriangle` – треугольник

Функция `rdsXGTriangle` рисует треугольник если модель блока вызвана для рисования (см. A.5.18.1 на стр. 365).

```
void RDSCALL rdsXGTriangle(  
    int x1,int y1,    // Первая вершина  
    int x2,int y2,    // Вторая вершина  
    int x3,int y3     // Третья вершина  
);
```

Тип указателя на эту функцию:

RDS_VIIIIII

Параметры:

x1, y1

Горизонтальная (x1) и вертикальная (y1) координаты первой вершины треугольника.

x2, y2

Горизонтальная (x2) и вертикальная (y2) координаты второй вершины треугольника.

x3, y3

Горизонтальная (x3) и вертикальная (y3) координаты третьей вершины треугольника.

Примечания:

Эта функция рисует треугольник с вершинами (x1,y1), (x2,y2) и (x3,y3). Рисование производится текущими выбранными стилями линии и заливки. Запомненная текущая точка рисования не изменяется.

См. также:

rdsXGSetPenStyle (стр. 382), rdsXGSetBrushStyle (стр. 377),
rdsXGPolygon (стр. 375).

А.5.19. Работа с временными файлами

Описываются функции, облегчающие использование временных (автоматически удаляемых) файлов.

А.5.19.1. rdsTMPCreateEmptyFile – создать временный файл

Функция rdsTMPCreateEmptyFile создает пустой временный файл и возвращает его имя.

```
LPSTR RDSCALL rdsTMPCreateEmptyFile(  
    int SetId,           // Набор временных файлов  
    LPSTR DesiredName    // Желаемое имя с путем  
);
```

Тип указателя на эту функцию:

RDS_SIS

Параметры:

SetId

Уникальный целый идентификатор набора временных файлов, к которому будет принадлежать этот файл.

DesiredName

Указатель на строку с желаемым именем файла. Имя должно содержать путь к файлу (возможно использование символических обозначений путей, см. стр. 189).

Возвращаемое значение:

Указатель на строку во внутренней памяти РДС, содержащую полное (с путем) имя созданного временного файла.

Примечания:

Эта функция создает пустой (нулевого размера) временный файл в папке, путь к которой берется из желаемого имени файла DesiredName. Если на диске в данный момент нет файла с именем DesiredName, файл получит это имя, если же такой файл уже есть,

функция подберет для файла уникальное имя с тем же расширением и разместит его в этой же папке. Например, результат вызова

```
char *name=rdsTMPCreateEmptyFile(setid,"c:\\rds\\temp.tmp");
```

будет зависеть от того, существует ли на момент вызова функции файл “c:\rds\temp.tmp”. Если такого файла нет, он будет создан, и функция вернет указатель на внутреннюю копию строки с его полным именем. Если файл уже есть, функция заменит часть имени файла, не относящуюся к пути и расширению, на уникальный для данной папки набор символов – например, “c:\rds\201114081443145.tmp”.

Созданный файл будет принадлежать к набору временных файлов SetId. Наборы временных файлов создаются функцией rdsTMPCreateFileSet (стр. 387), они нужны для того, чтобы можно было уничтожить все файлы в наборе одним вызовом rdsTMPDeleteFileSet (стр. 388). При выгрузке схемы из памяти все временные файлы, созданные РДС, уничтожаются автоматически.

Строка, указатель на которую возвращает функция, находится во внутренней памяти РДС (и будет находиться там, пока этот временный файл не будет удален из набора или пока весь набор не будет удален), поэтому вызывающей программе не нужно освобождать ее.

Пример использования функции rdsTMPCreateEmptyFile приведен в §4.4.

См. также:

rdsTMPCreateFileSet (стр. 387), rdsTMPDeleteFileSet (стр. 388),
rdsTMPRememberFileName (стр. 388).

A.5.19.2. rdsTMPCreateFileSet – создать набор временных файлов

Функция rdsTMPCreateFileSet создает в памяти РДС список временных файлов, в который можно добавлять файлы по одному, а затем, когда они станут не нужны, удалить все файлы из списка одним вызовом.

```
int RDSCALL rdsTMPCreateFileSet(void);
```

Тип указателя на эту функцию:

RDS_IV

Возвращаемое значение:

Уникальный идентификатор набора временных файлов, который можно использовать в других функциях работы с временными файлами.

Примечания:

Эта функция создает набор (внутренний список) временных файлов и возвращает его идентификатор. Файлы в набор можно добавлять функциями rdsTMPCreateEmptyFile (стр. 386) и rdsTMPRememberFileName (стр. 388). Когда набор временных файлов станет не нужен, его можно удалить функцией rdsTMPDeleteFileSet (стр. 388), при этом сами файлы из этого набора тоже будут удалены с диска. При выгрузке схемы из памяти автоматически стираются все временные файлы во всех созданных за время работы схемы наборах.

Пример использования функции rdsTMPCreateFileSet приведен в §4.4.

См. также:

rdsTMPDeleteFileSet (стр. 388), rdsTMPCreateEmptyFile (стр. 386),
rdsTMPRememberFileName (стр. 388).

A.5.19.3. rdsTMPDeleteFile – удалить временный файл

Функция `rdsTMPDeleteFile` удаляет ранее созданный временный файл с указанным именем.

```
void RDSCALL rdsTMPDeleteFile(  
    LPSTR FullName    // Полное имя (с путем)  
);
```

Тип указателя на эту функцию:

`RDS_VS`

Параметр:

`FullName`

Указатель на строку с полным именем удаляемого временного файла.

Примечания:

Эта функция удаляет файл с заданным полным именем из всех наборов временных файлов, в которых он присутствует. Файл удаляется и с диска, и из списка файлов в наборе. Если имя файла `FullName` не относится ни к одному набору временных файлов, такой файл не будет удален с диска.

См. также:

`rdsTMPDeleteFileSet` (стр. 388), `rdsTMPCreateEmptyFile` (стр. 386),
`rdsTMPRememberFileName` (стр. 388).

A.5.19.4. rdsTMPDeleteFileSet – удалить набор временных файлов

Функция `rdsTMPDeleteFileSet` удаляет ранее созданный набор временных файлов из памяти и стирает с диска все файлы, принадлежащие этому набору.

```
void RDSCALL rdsTMPDeleteFileSet(  
    int SetId          // Идентификатор набора  
);
```

Тип указателя на эту функцию:

`RDS_VI`

Параметр:

`SetId`

Уникальный целый идентификатор удаляемого набора временных файлов.

Примечания:

Эта функция стирает все файлы набора временных файлов с идентификатором `SetId`, после чего удаляет этот набор из памяти. Пример использования функции `rdsTMPCreateFileSet` приведен в §4.4.

См. также:

`rdsTMPCreateFileSet` (стр. 387), `rdsTMPCreateEmptyFile` (стр. 386),
`rdsTMPRememberFileName` (стр. 388).

A.5.19.5. rdsTMPRememberFileName – запомнить файл как временный

Функция `rdsTMPRememberFileName` добавляет к набору временных файлов новое имя.

```

LPSTR RDSCALL rdsTMPRememberFileName(
    int SetId,           // Набор временных файлов
    LPSTR Name           // Имя файла с путем
);

```

Тип указателя на эту функцию:

RDS_SIS

Параметры:

SetId

Уникальный целый идентификатор набора временных файлов, к которому будет принадлежать этот файл.

Name

Указатель на строку с именем файла. Имя должно содержать путь к файлу (возможно использование символических обозначений путей, см. стр. 189).

Возвращаемое значение:

Указатель на строку во внутренней памяти РДС, содержащую полное (с путем) имя временного файла.

Примечания:

Эта функция добавляет в набор SetId новое имя временного файла Name. Сам файл не создается – независимо от того, существует он в данный момент или нет, имя Name (с преобразованием символических обозначений путей к стандартным папкам в сами пути, если эти обозначения там встретятся) будет добавлено в набор, и файл с таким именем будет удален при вызове rdsTMPDeleteFileSet (стр. 388). Функция rdsTMPRememberFileName может быть полезна в тех случаях, когда временные файлы создаются какой-либо другой программой (например, компилятором), но их имена заранее известны.

Строка, указатель на которую возвращает функция, находится во внутренней памяти РДС (и будет находиться там, пока этот временный файл не будет удален из набора или пока весь набор не будет удален), поэтому вызывающей программе не нужно освобождать ее.

См. также:

rdsTMPCreateFileSet (стр. 387), rdsTMPDeleteFileSet (стр. 388),
rdsTMPCreateEmptyFile (стр. 386).

А.5.20. Сетевые функции

Описываются функции, позволяющие моделям блоков обмениваться данными по сети со схемами, загруженными в РДС на других машинах (§2.15).

А.5.20.1. rdsNetBroadcastData – передача данных всем блокам канала

Функция rdsNetBroadcastData передает указанные данные всем блокам, установившим соединение с указанным каналом сервера.

```

BOOL RDSCALL rdsNetBroadcastData(
    int ConnId,           // Идентификатор соединения
    DWORD Flags,         // флаги (RDS_NETSEND_*)
    int Id,              // Передаваемое целое (идентификатор)
    LPSTR String,        // Передаваемая строка
    LPVOID Buf,          // Передаваемая область памяти
);

```

```
        DWORD BufSize    // Размер передаваемой области
    );
```

Тип указателя на эту функцию:

RDS_BIDwISpVDw

Параметры:

ConnId

Уникальный идентификатор сетевого соединения, через которое будут переданы данные.

Flags

Набор битовых флагов, управляющих передачей:

RDS_NETSEND_NOWAIT После передачи данных РДС не будет ждать ответа от сервера перед передачей следующей порции данных (ожидание снижает нагрузку на сеть, но уменьшает скорость передачи).

RDS_NETSEND_SERVREPLY Получив эти данные, сервер должен ответить передавшему блоку – модель блока будет вызвана для реакции на событие RDS_BFM_NETDATAACCEPTED (стр. 85).

RDS_NETSEND_UDP Передавать данные по протоколу UDP, если это разрешено настройками РДС (по умолчанию данные передаются по протоколу TCP).

RDS_NETSEND_UPDATE Если в очереди на отправку данных уже есть данные, отправленные тем же блоком в тот же канал с тем же значением Id (см. ниже), старые данные будут выброшены из очереди.

Id

Передаваемое в канал целое число. При установленном в параметре Flags флаге RDS_NETSEND_UPDATE играет роль идентификатора передаваемых данных: устаревшие данные будут выброшены из очереди только в том случае, если их значение Id совпадает с Id в данном вызове.

String

Указатель на передаваемую в канал строку. Если строку передавать не нужно, этот параметр может быть равен NULL.

Buf

Указатель на начало передаваемой в канал области памяти с двоичными данными. Если двоичные данные передавать не нужно, этот параметр может быть равен NULL.

BufSize

Размер в байтах передаваемой области памяти по указателю Buf. При Buf==NULL значение этого параметра игнорируется.

Возвращаемое значение:

TRUE – данные успешно поставлены в очередь на передачу, FALSE – ошибка (например, нет соединения с идентификатором ConnId).

Примечания:

Эта функция используется для передачи целого числа Id, строки String и набора двоичных данных Buf всем блокам, подключившимся к какому-либо каналу передачи данных сервера (принципы обмена данных по сети и каналы передачи данных подробно

рассматриваются в §2.15.1). В параметре ConnId передается идентификатор сетевого соединения с конкретным каналом конкретного сервера, полученный с помощью функций rdsNetConnect (стр. 391) или rdsNetServer (стр. 394). Строку и двоичные данные передавать не обязательно – если они не нужны, соответствующим параметрам можно присвоить NULL.

В результате вызова этой функции данные, указанные в ее параметрах, ставятся в очередь для отправки на сервер. Сервер, получив эти данные, отправляет их всем блокам, подписавшимся на прием данных из канала, соответствующего соединению ConnId. После получения данных модели этих блоков вызываются для реакции на событие RDS_BFM_NETDATA RECEIVED (стр. 87) и в них передаются принятое целое число, строка и набор двоичных данных.

Помимо указанных в параметрах функции данных на сервер также передается идентификатор блока-отправителя (блока, из модели которого вызвана rdsNetBroadcastData), чтобы любой из блоков-получателей, при необходимости, мог в ответ вызовом rdsNetSendData (стр. 392) передать данные непосредственно отправителю, а не всем блокам канала.

Пример использования функции rdsNetBroadcastData приведен в §2.15.2.

См. также:

rdsNetConnect (стр. 391), rdsNetServer (стр. 394),
rdsNetSendData (стр. 392), RDS_BFM_NETDATA RECEIVED (стр. 87),
RDS_BFM_NETDATA ACCEPTED (стр. 85).

A.5.20.2. rdsNetCloseConnection – разорвать соединение

Функция rdsNetCloseConnection разрывает ранее установленное соединение с каналом передачи данных сервера.

```
void RDSCALL rdsNetCloseConnection(  
    int ConnId          // Идентификатор соединения  
);
```

Тип указателя на эту функцию:

RDS_VI

Параметр:

ConnId

Уникальный идентификатор разрываемого сетевого соединения.

Примечания:

Эта функция разрывает сетевое соединение с идентификатором ConnId, ранее установленное функциями rdsNetConnect (стр. 391) или rdsNetServer (стр. 394). Пример ее использования приведен в §2.15.2.

См. также:

rdsNetConnect (стр. 391), rdsNetServer (стр. 394),
RDS_BFM_NETDISCONNECT (стр. 88).

A.5.20.3. rdsNetConnect – установка сетевого соединения

Функция rdsNetConnect устанавливает соединение с указанным каналом передачи данных на указанном сервере.

```

int RDSCALL rdsNetConnect (
    LPSTR Host,      // Адрес сервера или NULL
    int Port,        // Номер порта или -1
    LPSTR Channel,   // Имя канала
    BOOL Receive     // Получение данных
);

```

Тип указателя на эту функцию:

RDS_ISISB

Параметры:

Host

Указатель на строку с именем или IP-адресом сервера. IP-адрес указывается в виде строки, например, “192.168.0.1”. Если нужно установить соединение с сервером по умолчанию, указанным в настройках РДС, в этом параметре передается NULL.

Port

Номер порта на сервере. Для использования номера порта по умолчанию, указанного в настройках РДС, в этом параметре передается -1.

Channel

Указатель на строку с именем канала передачи данных, связь с которым устанавливается. Если такого канала нет на сервере, он будет создан автоматически.

Receive

TRUE – блок, из модели которого вызвана функция, будет получать данные из канала.
FALSE – блок не будет получать данные. Передавать данные блок сможет в любом случае.

Возвращаемое значение:

Идентификатор созданного соединения (он будет использоваться во всех остальных сетевых функциях), или -1, если соединение установить не удалось.

Примечания:

Эта функция устанавливает соединение с каналом Channel сервера Host (сервер использует для подключений порт Port). Соединение устанавливается не сразу, об успешной его установке сигнализирует событие RDS_BFM_NETCONNECT (стр. 84). Если сервер с адресом Host недоступен, РДС будет повторять попытки соединения до тех пор, пока они не увенчаются успехом, либо пока соединение не будет разорвано вызовом rdsNetCloseConnection (стр. 391).

Если необходимо сделать сервером ту копию РДС, в которую загружена схема, модель блока которой вызвана в данный момент, следует воспользоваться функцией rdsNetServer (стр. 394).

Принципы обмена данных по сети и каналы передачи данных подробно рассматриваются в §2.15.1. Пример использования функции rdsNetConnect приведен в §2.15.2.

См. также:

rdsNetCloseConnection (стр. 391), rdsNetServer (стр. 394),
RDS_BFM_NETCONNECT (стр. 84).

A.5.20.4. rdsNetSendData – передача данных конкретному блоку канала

Функция rdsNetSendData передает указанные данные конкретному блоку, установившему соединение с указанным каналом сервера.


```

BOOL RDSCALL rdsNetSendData(
    int ConnId,        // Идентификатор соединения
    DWORD Flags,      // флаги (RDS_NETSEND_*)
    int Id,           // Передаваемое целое (идентификатор)
    LPSTR String,     // Передаваемая строка
    LPVOID Buf,       // Передаваемая область памяти
    DWORD BufSize,    // Размер передаваемой области
    RDS_NETSTATION Station, // Сетевой идентификатор машины
    RDS_NETBLOCK Block // Сетевой идентификатор блока
);

```

Тип указателя на эту функцию:

RDS_BIDwISpVDwNsNb

Параметры:

ConnId

Уникальный идентификатор сетевого соединения, через которое будут переданы данные.

Flags

Набор битовых флагов, управляющих передачей (те же, что и у функции rdsNetBroadcastData, см. стр. 390).

Id

Передаваемое в канал целое число. При установленном в параметре Flags флаге RDS_NETSEND_UPDATE играет роль идентификатора передаваемых данных: устаревшие данные будут выброшены из очереди только в том случае, если их значение Id совпадает с Id в данном вызове.

String

Указатель на передаваемую в канал строку. Если строку передавать не нужно, этот параметр может быть равен NULL.

Buf

Указатель на начало передаваемой в канал области памяти с двоичными данными. Если двоичные данные передавать не нужно, этот параметр может быть равен NULL.

BufSize

Размер в байтах передаваемой области памяти по указателю Buf. При Buf==NULL значение этого параметра игнорируется.

Station

Уникальный сетевой идентификатор (RDS_NETSTATION) машины, на которой запущена копия РДС, в которую загружена схема, блоку которой отправляются данные.

Block

Уникальный сетевой идентификатор (RDS_NETBLOCK) блока, которому отправляются данные. Этот идентификатор никак не связан с внутренним идентификатором этого блока RDS_BHANDLE.

Возвращаемое значение:

TRUE – данные успешно поставлены в очередь на передачу, FALSE – ошибка (например, нет соединения с идентификатором ConnId).

Примечания:

Эта функция используется для передачи целого числа `Id`, строки `String` и набора двоичных данных `Buf` конкретному блоку, подключившемуся к какому-либо каналу передачи данных сервера (принципы обмена данных по сети и каналы передачи данных подробно рассматриваются в §2.15.1). В параметре `ConnId` передается идентификатор сетевого соединения с конкретным каналом конкретного сервера, полученный с помощью функций `rdsNetConnect` (стр. 391) или `rdsNetServer` (стр. 394). Строку и двоичные данные передавать не обязательно – если они не нужны, соответствующим параметрам можно присвоить `NULL`.

Блок-получатель однозначно определяется парой параметров `Station` и `Block`: в первом передается идентификатор машины-получателя данных, во втором – идентификатор блока на этой машине. Эти специальные сетевые идентификаторы модель блока может узнать только из структуры `RDS_NETRECEIVEDDATA` (стр. 87), получив данные от этого блока. Таким образом, чтобы отправить данные какому-либо конкретному блоку, нужно сначала получить от него какие-либо данные. Функция `rdsNetSendData` в РДС предназначена для ответа на переданные данные: если какой-либо блок передает данные в канал функцией `rdsNetBroadcastData` (стр. 389), любой из принявших эти данные блоков может послать ответ непосредственно отправителю. Если логика работы схемы требует передачи данных какому-либо конкретному блоку на конкретной машине, для этого блока следует создать отдельный канал, в котором он будет единственным получателем. Число каналов передачи данных на сервере не ограничено.

В результате вызова `rdsNetSendData` данные, указанные в параметрах функции, ставятся в очередь для отправки на сервер. Сервер, получив эти данные, отправляет их на машину `Station`. После получения данных на этой машине модель блока `Block` будет вызвана для реакции на событие `RDS_BFM_NETDATAARECEIVED`.

См. также:

`rdsNetConnect` (стр. 391), `rdsNetServer` (стр. 394),
`rdsNetBroadcastData` (стр. 389), `RDS_BFM_NETDATAARECEIVED` (стр. 87),
`RDS_BFM_NETDATAACCEPTED` (стр. 85).

A.5.20.5. `rdsNetServer` – запуск сервера и установка соединения с ним

Функция `rdsNetServer` делает сервером копию РДС, в которой она вызвана, и устанавливает соединение с указанным каналом передачи данных на этом сервере.

```
int RDSCALL rdsNetServer(  
    int Port,           // Номер порта или -1  
    LPSTR Channel,     // Имя канала  
    BOOL Receive       // Получение данных  
);
```

Тип указателя на эту функцию:

`RDS_IISB`

Параметры:

`Port`

Номер порта, который будет использовать запущенный сервер. Для использования номера порта по умолчанию, указанного в настройках РДС, в этом параметре передается `-1`.

Channel

Указатель на строку с именем канала передачи данных, связь с которым устанавливается. Если такого канала нет на сервере, он будет создан автоматически.

Receive

TRUE – блок, из модели которого вызвана функция, будет получать данные из канала.
FALSE – блок не будет получать данные. Передавать данные блок сможет в любом случае.

Возвращаемое значение:

Идентификатор созданного соединения (он будет использоваться во всех остальных сетевых функциях), или -1, если соединение установить не удалось.

Примечания:

Эта функция включает функции сервера в копии РДС, в которую загружена схема с блоком, модель которого вызвала функцию. При этом она устанавливает соединение с каналом Channel этого запущенного сервера. В схеме достаточно иметь всего один блок, вызывающий rdsNetServer, все остальные блоки могут связываться с этим локальным сервером при помощи rdsNetConnect (стр. 391).

Принципы обмена данных по сети и каналы передачи данных подробно рассматриваются в §2.15.1. Пример использования функции rdsNetServer приведен в §2.15.2.

См. также:

rdsNetCloseConnection (стр. 391), rdsNetConnect (стр. 391),
RDS_BFM_NETCONNECT (стр. 84).

A.5.21. Функции поддержки внешнего управления

Описываются функции, используемые моделями блоков при управлении РДС из внешнего приложения (см. главу 3) для взаимодействия с этим приложением.

A.5.21.1. rdsExecutesRemoteOpsSet – регистрация блока как исполнителя операции внешнего управления

Функция rdsExecutesRemoteOpsSet регистрирует блок, модель которого ее вызвала, как исполнителя операции внешнего управления с указанным именем. Эта же функция может отменить такую регистрацию.

```
void RDSCALL rdsExecutesRemoteOpsSet(  
    LPSTR OpSetName,        // Имя операции  
    BOOL Yes                // Включить/выключить  
);
```

Тип указателя на эту функцию:

RDS_VSB

Параметры:

OpSetName

Указатель на строку с именем выполняемой блоком операции внешнего управления.

Yes

TRUE – зарегистрировать блок исполнителем операции OpSetName, FALSE – отменить эту регистрацию.

Примечания:

Эта функция регистрирует вызвавший ее блок как исполнителя операции внешнего управления с именем OpSetName (при Yes==TRUE) или отменяет эту регистрацию (при Yes==FALSE). Внешнее управляющее приложение может найти все зарегистрировавшие себя таким образом блоки при помощи функции rdscrtlFindOpSetProviders (стр. 628), а затем вызывать их функцией rdscrtlCallBlockFunctionEx (стр. 621).

Пример использования функции rdsExecutesRemoteOpsSet приведен в §3.3.

См. также:

rdscrtlFindOpSetProviders (стр. 628),
rdscrtlCallBlockFunctionEx (стр. 621).

A.5.21.2. rdsGetRemoteControllerName – получить имя внешней управляющей программы

Функция rdsGetRemoteControllerName возвращает имя внешней управляющей программы, если эта программа установила его функцией rdscrtlSetControllerName (стр. 639).

```
LPSTR RDSCALL rdsGetRemoteControllerName(void);
```

Тип указателя на эту функцию:

RDS_SV

Возвращаемое значение:

Указатель на строку во внутренней памяти РДС, содержащую имя внешней управляющей программы. Если программа не сообщила РДС свое имя, возвращается указатель на пустую строку. Если РДС в данный момент не работает под управлением внешней программы, возвращается NULL.

Примечания:

Внешняя управляющая программа может сообщить РДС свое имя вызовом функции rdscrtlSetControllerName из библиотеки RdsCtrl.dll. Функция модели не должна как-либо изменять эту строку. Имя может быть любой строкой символов, оно не связано с именем EXE-файла управляющей программы. Модели блоков могут анализировать это имя и выполнять разные действия в зависимости от того, какая программа ими управляет.

См. также:

rdscrtlSetControllerName (стр. 639), rdsHasRemoteController (стр. 397).

A.5.21.3. rdsGetRemoteControllerString – получить строку, установленную внешней программой

Функция rdsGetRemoteControllerString возвращает указатель на строку с указанным идентификатором, установленную внешней управляющей программой.

```
LPSTR RDSCALL rdsGetRemoteControllerString(  
    int ValueId          // Идентификатор строки  
);
```

Тип указателя на эту функцию:

RDS_SI

Параметр:

ValueId

Идентификатор запрашиваемой строки.

Возвращаемое значение:

Указатель на строку во внутренней памяти РДС, установленную управляющей программой при помощи функции `rdscrtlSetString` (стр. 641) из библиотеки `RdsCtrl.dll`. Функция модели не должна как-либо изменять эту строку.

Примечания:

Обычно установка строк функцией `rdscrtlSetString` используется управляющим приложением для задания каких-либо глобальных параметров, влияющих на все блоки схемы.

См. также:

`rdscrtlSetString` (стр. 641).

A.5.21.4. `rdsHasRemoteController` – проверка наличия внешнего управления

Функция `rdsHasRemoteController` проверяет, находится ли РДС под управлением внешней программы.

```
BOOL RDSCALL rdsHasRemoteController(void) ;
```

Тип указателя на эту функцию:

`RDS_BV`

Возвращаемое значение:

`TRUE` – РДС в данный момент управляется внешним приложением, `FALSE` – РДС работает самостоятельно.

См. также:

`rdsGetRemoteControllerName` (стр. 396).

A.5.21.5. `rdsRemoteControllerCall` – передача сообщения управляющей программе

Функция `rdsRemoteControllerCall` передает внешней управляющей программе целое число и строку.

```
int RDSCALL rdsRemoteControllerCall(  
    int MessageCode,          // Передаваемое число  
    LPSTR MessageString      // Передаваемая строка  
);
```

Тип указателя на эту функцию:

`RDS_IIS`

Параметры:

MessageCode

Передаваемое управляющей программе целое число.

MessageString

Указатель на передаваемую управляющей программе строку, или NULL, если строку передавать не нужно.

Возвращаемое значение:

0 – ошибка передачи, 1 – передача успешно выполнена.

Примечания:

Эта функция позволяет передать в управляющую программу число MessageCode и строку MessageString по инициативе модели блока, без каких-либо запросов от программы (в управляющей программе при этом наступает событие RDSCTRLEVENT_BLOCKMSG, стр. 593). Вместе с числом и строкой передается полное имя блока, модель которой передает данные, поэтому эту функцию можно вызывать только из моделей блоков.

Для того, чтобы управляющая программа могла получить переданные данные, в ней должна быть разрешена реакция на события (функция rdsctrlEnableEvents, стр. 665) и зарегистрирована реакция на получение сообщения от блока. Если реакция не событие зарегистрирована функциями rdsctrlRegisterEventStdCallback (стр. 668) или rdsctrlRegisterBlockMsgCallback (стр. 665), при получении данных будет вызвана указанная при регистрации функция управляющего приложения. Если же реакция зарегистрирована функцией rdsctrlRegisterEventMessage (стр. 667), при получении данных от блока указанному при регистрации окну будет направлено сообщение.

Пример использования функции rdsRemoteControllerCall приведен в §3.4.

См. также:

RDSCTRLEVENT_BLOCKMSG (стр. 593), rdsctrlEnableEvents (стр. 665),
rdsctrlRegisterBlockMsgCallback (стр. 665),
rdsctrlRegisterEventStdCallback (стр. 668),
rdsctrlRegisterEventMessage (стр. 667),
RDSCTRL_MSGEVENTDATA (стр. 586), RDSCTRL_BLOCKMSGDATA (стр. 583).

A.5.21.6. rdsRemoteReply – возврат строки управляющему приложению

Функция rdsRemoteReply служит для указания строки, которую нужно вернуть управляющему приложению при реакции модели блока на событие RDS_BFM_REMOTEMSG (стр. 42).

```
void RDSCALL rdsRemoteReply(  
    LPSTR String    // Возвращаемая строка  
);
```

Тип указателя на эту функцию:

RDS_VS

Параметр:

String

Указатель на строку, которую нужно вернуть управляющему приложению после завершения модели блока.

Примечания:

Эту функцию следует вызывать из функции модели блока в момент реакции на событие RDS_BFM_REMOTEMSG. Вызов ее из любых других реакций будет игнорироваться. С помощью этой функции модель передает в РДС строку, которая, после завершения

реакции, будет отправлена в управляющее приложение, где она будет использована в качестве результата выполнения функции `rdsctrlCallBlockFunction` (стр. 619) или `rdsctrlCallBlockFunctionEx` (стр. 621). Если модель блока не вызовет `rdsRemoteReply`, управляющему приложению будет передана пустая строка.

Примеры использования функции `rdsRemoteReply` приведены в §3.3 и в описании события `RDS_BFM_REMOTEMSG`.

См. также:

`RDS_BFM_REMOTEMSG` (стр. 42), `rdsctrlCallBlockFunction` (стр. 619),
`rdsctrlCallBlockFunctionEx` (стр. 621).

А.5.22. Общие функции вспомогательных объектов

Описываются функции общего назначения, использующиеся для работы с различными вспомогательными объектами РДС. Большинство команд вспомогательным объектам передается именно через эти функции.

А.5.22.1. Использование вспомогательных объектов РДС

Вспомогательные объекты РДС – это объекты, создаваемые различными сервисными функциями и принадлежащие блоку, модель которого вызвала эти функции создания. Вспомогательный объект будет существовать до тех пор, пока он не будет уничтожен функцией `rdsDeleteObject` (стр. 401) или пока от блока, которому он принадлежит, не будет отключена модель. Вспомогательные объекты создаются для разных целей и выполняют различные функции, но их объединяет общий интерфейс взаимодействия с моделью блока: все объекты имеют идентификатор типа `RDS_NOBJECT`, и для передачи им различных параметров и команд и получения от них значений используются одни и те же сервисные функции, работающие с объектом через его идентификатор. Работа с каждым типом вспомогательного объекта имеет свои особенности: некоторые объекты способны отслеживать различные события, происходящие со схемой (например, объект, содержащий список блоков, может автоматически выбрасывать из списка удаляемые блоки), некоторые, в дополнение к общим функциям установки и получения значений, имеют набор специализированных функций. На данный момент в РДС поддерживаются следующие типы вспомогательных объектов:

<i>Назначение</i>	<i>Функция создания</i>
Редактирование и создание связи или шины	<code>rdsCECreateEditor</code> (стр. 408)
Создание и поддержание в актуальном состоянии списка блоков и связей	<code>rdsBCLCreateList</code> (стр. 420)
Разбор текста	<code>rdsSTRCreateTextReader</code> (стр. 460)
Работа с текстом в формате INI-файлов Windows	<code>rdsINICreateTextHolder</code> (стр. 473)
Создание модальных окон с различными полями ввода	<code>rdsFORMCreate</code> (стр. 491)
Откат программных изменений параметров блока	<code>rdsBEUCreate</code> (стр. 536)
Создание окна с индикатором выполнения (progress bar)	<code>rdsPBARCreate</code> (стр. 538)
Работа с панелями в окне подсистемы	<code>rdsPANCreate</code> (стр. 545)

<i>Назначение</i>	<i>Функция создания</i>
Изменение структуры переменных блока	rdsVSCreateEditor (стр. 430)
Работа с текстом в формате CSV (значения, разделенные запятыми)	rdsCSVCreate (стр. 556)

A.5.22.2. rdsCommandObject – команда объекту

Функция `rdsCommandObject` передает указанному вспомогательному объекту команду с целым идентификатором. Это упрощенный вариант функции `rdsCommandObjectEx` (стр. 400).

```
BOOL RDSCALL rdsCommandObject (
    RDS_НОВЕКТ Object,    // Идентификатор объекта
    int ObjCmd             // Команда
);
```

Тип указателя на эту функцию:

`RDS_ВНОI`

Параметры:

`Object`

Идентификатор вспомогательного объекта РДС, который должен выполнить команду.

`ObjCmd`

Идентификатор команды. Разные объекты поддерживают разные команды, допустимые идентификаторы команд перечислены в описании каждого объекта.

Возвращаемое значение:

Логический результат выполнения команды. Если объект `Object` не поддерживает команду `ObjCmd`, функция возвращает `FALSE`. Если объект поддерживает команду, возвращаемое функцией значение зависит от выполнения данной команды.

Примечания:

Эта функция требует от объекта `Object` выполнения команды с идентификатором `ObjCmd`. Для каждого объекта описаны свои идентификаторы команд. Действия, выполняемые объектом по команде, зависят от назначения этого объекта: для списка блоков и связей, например, определена команда очистки списка, для объекта, работающего с модальными окнами – команда обновления окна, и т.п. Примеры использования функции `rdsCommandObject` приведены в §§2.7.3, 2.8.5, 2.16.2, 4.2 и др.

Вызов `rdsCommandObject(Object, ObjCmd)` полностью эквивалентен вызову `rdsCommandObjectEx(Object, ObjCmd, 0, NULL)`.

См. также:

Вспомогательные объекты (стр. 399), `rdsCommandObjectEx` (стр. 400), `rdsSetObjectDouble` (стр. 406), `rdsSetObjectInt` (стр. 407), `rdsSetObjectStr` (стр. 407).

A.5.22.3. rdsCommandObjectEx – команда объекту

Функция `rdsCommandObjectEx` передает указанному вспомогательному объекту команду с целым идентификатором и параметром и возвращает через один из параметров целый результат выполнения этой команды.


```

BOOL RDCALL rdsCommandObjectEx(
    RDS_НOBJECT Object,    // Идентификатор объекта
    int ObjCmd,            // Команда
    int CmdParam,          // Параметр команды
    int *pResult           // Возвращаемый результат
);

```

Тип указателя на эту функцию:

RDS_BHoIIPi

Параметры:

Object

Идентификатор вспомогательного объекта РДС, который должен выполнить команду.

ObjCmd

Идентификатор команды. Разные объекты поддерживают разные команды, допустимые идентификаторы команд перечислены в описании каждого объекта.

CmdParam

Параметр команды, если он нужен. Если у команды ObjCmd нет параметра, значение CmdParam игнорируется.

pResult

Указатель на целую переменную, в которую функция запишет целый результат выполнения команды, если команда ObjCmd возвращает целое число. Если результат команды не нужен вызывающей программе, в этом параметре можно передать NULL.

Возвращаемое значение:

Логический результат выполнения команды. Если объект Object не поддерживает команду ObjCmd, функция возвращает FALSE. Если объект поддерживает команду, возвращаемое функцией значение зависит от выполнения данной команды.

Примечания:

Эта функция требует от объекта Object выполнения команды с идентификатором ObjCmd. Для каждого объекта описаны свои идентификаторы команд. Действия, выполняемые объектом по команде, зависят от назначения этого объекта: для списка блоков и связей, например, определена команда очистки списка, для объекта, работающего с модальными окнами – команда обновления окна, и т.п. Вместе с командой может быть передан один целый параметр, зависящий от смысла команды. Если команда возвращает целое число, его можно получить, передав в параметре pResult указатель на целую переменную.

Если у команды нет параметра, и она не возвращает целое число, вместо rdsCommandObjectEx можно использовать более простую функцию rdsCommandObject (стр. 400).

Пример использования функции rdsCommandObjectEx приведен в §2.16.2.

См. также:

Вспомогательные объекты (стр. 399), rdsCommandObject (стр. 400), rdsSetObjectDouble (стр. 406), rdsSetObjectInt (стр. 407), rdsSetObjectStr (стр. 407).

A.5.22.4. rdsDeleteObject – удалить объект

Функция rdsDeleteObject удаляет вспомогательный объект РДС.

```
void RDSCALL rdsDeleteObject(
    RDS_HOBJECT Object    // Идентификатор объекта
);
```

Тип указателя на эту функцию:

RDS_VHo

Параметр:

Object

Идентификатор удаляемого объекта.

Примечания:

Эта функция удаляет вспомогательный объект Object вместе со всеми связанными с ним данными. Примеры ее использования приведены в §§2.7.2, 2.8.4, 2.8.5 и др.

См. также:

Вспомогательные объекты (стр. 399).

A.5.22.5. rdsGetObjectArray – получить массив из объекта

Функция rdsGetObjectArray возвращает указатель на заданный массив в заданном объекте, если такой массив существует.

```
LPVOID RDSCALL rdsGetObjectArray(
    RDS_HOBJECT Object,    // Идентификатор объекта
    int ObjOp,             // Команда (идентификатор массива)
    int OpParam,           // Дополнительный параметр
    int *pSize             // Возвращаемое число элементов
);
```

Тип указателя на эту функцию:

RDS_pVHoIIpI

Параметры:

Object

Идентификатор объекта, в котором нужно найти массив.

ObjOp

Идентификатор массива. Вместе с параметром OpParam однозначно определяет массив в объекте.

OpParam

Дополнительный идентификатор массива. Вместе с параметром ObjOp однозначно определяет массив в объекте.

pSize

Указатель на целую переменную, в которую функция запишет число элементов в массиве. Если это число не нужно вызывающей программе, в параметре можно передать NULL.

Возвращаемое значение:

Указатель общего вида (void*) на первый элемент массива, или NULL, если такого массива нет в объекте. Перед использованием этот указатель нужно будет привести к конкретному типу “указатель на тип элемента массива”.

Примечания:

Эта функция возвращает указатель на массив, содержащийся в объекте `Object`. Массив однозначно определяется парой целых чисел (`ObjOp`, `OpParam`) – конкретный смысл этих параметров зависит от назначения объекта `Object`. На данный момент только один вспомогательный объект РДС содержит в себе массивы – это объект для работы со списком блоков и связей, создаваемый функцией `rdsBCLCreateList` (стр. 420).

Пример использования функции `rdsGetObjectArray` приведен в §2.16.2.

См. также:

Вспомогательные объекты (стр. 399), `rdsBCLCreateList` (стр. 420),
`RDS_HBCL_BLOCKARRAY` (стр. 425), `RDS_HBCL_CONNARRAY` (стр. 427).

A.5.22.6. `rdsGetObjectDouble` – получить вещественное число

Функция `rdsGetObjectDouble` запрашивает вещественное число у вспомогательного объекта с указанным идентификатором.

```
double RDSCALL rdsGetObjectDouble(  
    RDS_HOBJECT Object,    // Идентификатор объекта  
    int ValueId,           // Идентификатор параметра  
    int ValueNum           // Номер параметра  
);
```

Тип указателя на эту функцию:

`RDS_DHoII`

Параметры:

`Object`

Идентификатор вспомогательного объекта РДС, вещественный параметр которого нужно получить.

`ValueId`

Идентификатор вещественного параметра (зависит от конкретного типа объекта `Object`).

`ValueNum`

Номер вещественного параметра, если в объекте несколько параметров с идентификатором `ValueId` (например, `ValueId` может быть идентификатором массива параметров, а `ValueNum` – индексом в этом массиве).

Возвращаемое значение:

Вещественное число, соответствующее сочетанию параметров (`ValueId`, `ValueNum`).

Примечания:

Эта функция возвращает какой-либо вещественный параметр объекта `Object`. Смысл этого параметра и допустимые значения `ValueId` и `ValueNum` зависят от конкретного типа объекта.

Если из-за особенностей используемого компилятора возврат типа `double` невозможен, вместо `rdsGetObjectDouble` можно использовать функцию `rdsGetObjectDoubleP` (стр. 404), возвращающую вещественное число через указатель.

Пример использования функции `rdsGetObjectDouble` приведен в §2.7.2.

См. также:

Вспомогательные объекты (стр. 399), `rdsGetObjectDoubleP` (стр. 404),
`rdsSetObjectDouble` (стр. 406).

A.5.22.7. `rdsGetObjectDoubleP` – получить вещественное число

Функция `rdsGetObjectDoubleP` запрашивает вещественное число у вспомогательного объекта с указанным идентификатором, возвращая его через указатель, переданный в параметре функции.

```
void RDSCALL rdsGetObjectDoubleP(  
    RDS_HOBJECT Object,    // Идентификатор объекта  
    int ValueId,           // Идентификатор параметра  
    int ValueNum,          // Номер параметра  
    double *pVal          // Возвращаемое значение  
);
```

Тип указателя на эту функцию:

`RDS_VHoIIPD`

Параметры:

`Object`

Идентификатор вспомогательного объекта РДС, вещественный параметр которого нужно получить.

`ValueId`

Идентификатор вещественного параметра (зависит от конкретного типа объекта `Object`).

`ValueNum`

Номер вещественного параметра, если в объекте несколько параметров с идентификатором `ValueId` (например, `ValueId` может быть идентификатором массива параметров, а `ValueNum` – индексом в этом массиве).

`pVal`

Указатель на вещественную переменную двойной точности (`double`), в которую функция запишет полученное значение.

Примечания:

Эта функция возвращает какой-либо вещественный параметр объекта `Object`. Смысл этого параметра и допустимые значения `ValueId` и `ValueNum` зависят от конкретного типа объекта.

См. также:

Вспомогательные объекты (стр. 399), `rdsGetObjectDouble` (стр. 403),
`rdsSetObjectDouble` (стр. 406).

A.5.22.8. `rdsGetObjectInt` – получить целое число

Функция `rdsGetObjectInt` запрашивает целое число у вспомогательного объекта с указанным идентификатором.

```
int RDSCALL rdsGetObjectInt(  
    RDS_HOBJECT Object,    // Идентификатор объекта  
    int ValueId,           // Идентификатор параметра
```

```

        int ValueNum           // Номер параметра
    );

```

Тип указателя на эту функцию:

RDS_IHoII

Параметры:

Object

Идентификатор вспомогательного объекта РДС, целый параметр которого нужно получить.

ValueId

Идентификатор целого параметра (зависит от конкретного типа объекта Object).

ValueNum

Номер целого параметра, если в объекте несколько параметров с идентификатором ValueId (например, ValueId может быть идентификатором массива параметров, а ValueNum – индексом в этом массиве).

Возвращаемое значение:

Целое число, соответствующее сочетанию параметров (ValueId, ValueNum).

Примечания:

Эта функция возвращает какой-либо целый параметр объекта Object. Смысл этого параметра и допустимые значения ValueId и ValueNum зависят от конкретного типа объекта.

Пример использования функции rdsGetObjectInt приведен в §2.7.2.

См. также:

Вспомогательные объекты (стр. 399), rdsSetObjectInt (стр. 407).

A.5.22.9. rdsGetObjectStr – получить строку

Функция rdsGetObjectStr запрашивает строку у вспомогательного объекта с указанным идентификатором.

```

LPSTR RDSCALL rdsGetObjectStr(
    RDS_HOBJECT Object,    // Идентификатор объекта
    int ValueId,           // Идентификатор параметра
    int ValueNum           // Номер параметра
);

```

Тип указателя на эту функцию:

RDS_SHoII

Параметры:

Object

Идентификатор вспомогательного объекта РДС, строку которого нужно получить.

ValueId

Идентификатор строки (зависит от конкретного типа объекта Object).

ValueNum

Номер строки, если в объекте несколько строк с идентификатором ValueId (например, ValueId может быть идентификатором массива строк, а ValueNum – индексом в этом массиве).

Возвращаемое значение:

Указатель на строку (`char*`), соответствующую сочетанию параметров (`ValueId`, `ValueNum`). Это может быть как строка во внутренней памяти объекта, так и динамически созданная строка – в последнем случае ее нужно освободить функцией `rdsFree` (стр. 187). Как именно устроена возвращаемая строка и нужно ли ее освободить, указывается в описаниях конкретных объектов.

Примечания:

Эта функция возвращает какую-либо строку объекта `Object`. Смысл этой строки и допустимые значения `ValueId` и `ValueNum` зависят от конкретного типа объекта.

Пример использования функции `rdsGetObjectStr` приведен в §2.7.4.

См. также:

Вспомогательные объекты (стр. 399), `rdsSetObjectStr` (стр. 407).

A.5.22.10. `rdsSetObjectDouble` – установить вещественное число

Функция `rdsSetObjectDouble` передает вещественное число во вспомогательный объект с указанным идентификатором.

```
void RDSCALL rdsSetObjectDouble(  
    RDS_HOBJECT Object,    // Идентификатор объекта  
    int ValueId,           // Идентификатор параметра  
    int ValueNum,          // Номер параметра  
    double Value           // Значение параметра  
);
```

Тип указателя на эту функцию:

`RDS_VHoIID`

Параметры:

`Object`

Идентификатор вспомогательного объекта РДС, вещественный параметр которого нужно установить.

`ValueId`

Идентификатор вещественного параметра (зависит от конкретного типа объекта `Object`).

`ValueNum`

Номер вещественного параметра, если в объекте несколько параметров с идентификатором `ValueId` (например, `ValueId` может быть идентификатором массива параметров, а `ValueNum` – индексом в этом массиве).

`Value`

Передаваемое в объект вещественное число, соответствующее сочетанию параметров (`ValueId`, `ValueNum`).

Примечания:

Эта функция устанавливает какой-либо вещественный параметр объекта `Object`. Смысл этого параметра и допустимые значения `ValueId` и `ValueNum` зависят от конкретного типа объекта.

Пример использования функции `rdsSetObjectDouble` приведен в §2.7.2.

См. также:

Вспомогательные объекты (стр. 399), `rdsGetObjectDouble` (стр. 403),
`rdsGetObjectDoubleP` (стр. 404).

A.5.22.11. `rdsSetObjectInt` – установить целое число

Функция `rdsSetObjectInt` передает целое число во вспомогательный объект с указанным идентификатором.

```
void RDSCALL rdsSetObjectInt(  
    RDS_HOBJECT Object,    // Идентификатор объекта  
    int ValueId,           // Идентификатор параметра  
    int ValueNum,          // Номер параметра  
    int Value              // Значение параметра  
);
```

Тип указателя на эту функцию:

`RDS_VHoIII`

Параметры:

`Object`

Идентификатор вспомогательного объекта РДС, целый параметр которого нужно установить.

`ValueId`

Идентификатор целого параметра (зависит от конкретного типа объекта `Object`).

`ValueNum`

Номер целого параметра, если в объекте несколько параметров с идентификатором `ValueId` (например, `ValueId` может быть идентификатором массива параметров, а `ValueNum` – индексом в этом массиве).

`Value`

Передаваемое в объект целое число, соответствующее сочетанию параметров (`ValueId`, `ValueNum`).

Примечания:

Эта функция устанавливает какой-либо целый параметр объекта `Object`. Смысл этого параметра и допустимые значения `ValueId` и `ValueNum` зависят от конкретного типа объекта.

Пример использования функции `rdsSetObjectInt` приведен в §2.7.2.

См. также:

Вспомогательные объекты (стр. 399), `rdsGetObjectInt` (стр. 404).

A.5.22.12. `rdsSetObjectStr` – установить строку

Функция `rdsSetObjectStr` передает строку во вспомогательный объект с указанным идентификатором.

```
void RDSCALL rdsSetObjectStr(  
    RDS_HOBJECT Object,    // Идентификатор объекта  
    int ValueId,           // Идентификатор параметра  
    int ValueNum,          // Номер параметра  
    LPSTR Value            // Значение параметра  
);
```

Тип указателя на эту функцию:

`RDS_VHoIIS`

Параметры:

`Object`

Идентификатор вспомогательного объекта РДС, строку которого нужно установить.

`ValueId`

Идентификатор строки (зависит от конкретного типа объекта `Object`).

`ValueNum`

Номер строки, если в объекте несколько строк с идентификатором `ValueId` (например, `ValueId` может быть идентификатором массива строк, а `ValueNum` – индексом в этом массиве).

`Value`

Передаваемая в объект строка, соответствующая сочетанию параметров (`ValueId`, `ValueNum`).

Примечания:

Эта функция устанавливает какую-либо строку объекта `Object`. Смысл и структура этой строки, а также допустимые значения `ValueId` и `ValueNum` зависят от конкретного типа объекта.

Пример использования функции `rdsSetObjectStr` приведен в §2.7.2.

См. также:

Вспомогательные объекты (стр. 399), `rdsGetObjectStr` (стр. 405).

А.5.23. Вспомогательный объект для изменения связей и шин

Описываются функции и команды вспомогательного объекта РДС, предназначенного для создания и изменения параметров связей и шин в схеме.

А.5.23.1. `rdsCECreateEditor` – создать объект-редактор связи/шины

Функция `rdsCECreateEditor` создает вспомогательный объект РДС, с помощью которого можно создать новую связь (шину) или изменить параметры и структуру существующей.

`RDS_НОВЕКТ RDSCALL rdsCECreateEditor(void);`

Тип указателя на эту функцию:

`RDS_HoV`

Возвращаемое значение:

Уникальный идентификатор созданного объекта.

Примечания:

Функция `rdsCECreateEditor` создает вспомогательный объект для редактирования связи и возвращает его идентификатор типа `RDS_НОВЕКТ`. С помощью различных сервисных функций РДС (как общих для всех типов вспомогательных объектов, так и специализированных для данного) в этот объект можно добавлять точки и отрезки, а также каналы для шины. Для того, чтобы создать по данным этого объекта новую связь (шину) или изменить существующую, используются функции `rdsCECreateConnBus` (стр. 415) и `rdsCEEditConnBus` (стр. 416) соответственно.

Созданный вспомогательный объект будет существовать до тех пор, пока схема не будет выгружена из памяти, или пока он не будет удален вызовом функции `rdsDeleteObject` (стр. 401).

Пример использования функции `rdsCECreateEditor` приведен в §2.16.2.

См. также:

Вспомогательные объекты (стр. 399), `rdsCECreateConnBus` (стр. 415), `rdsCEEditConnBus` (стр. 416), `rdsDeleteObject` (стр. 401).

A.5.23.2. `rdsCEAddBezier` – добавление кривой Безье

Функция `rdsCEAddBezier` добавляет в объект редактирования связи отрезок кривой Безье, соединяющий пару точек.

```
int RDSCALL rdsCEAddBezier(  
    RDS_HOBJECT Editor,    // Объект  
    int nFrom,              // Номер первой точки  
    int fromX, int fromY,   // Касательная первой точки  
    int nTo,                // Номер второй точки  
    int toX, int toY        // Касательная второй точки  
);
```

Тип указателя на эту функцию:

`RDS_IH0IIIIII`

Параметры:

`Editor`

Идентификатор вспомогательного объекта для редактирования связи, ранее созданного функцией `rdsCECreateEditor` (стр. 408).

`nFrom`

Номер первой из двух соединяемых отрезком точек во внутреннем наборе точек объекта.

`fromX, fromY`

Горизонтальное (`fromX`) и вертикальное (`fromY`) смещение управляющей точки (точки касательной) для точки связи с номером `nFrom`. Координаты указываются в точках экрана в масштабе 100%.

`nTo`

Номер второй из двух соединяемых отрезком точек во внутреннем наборе точек объекта.

`toX, toY`

Горизонтальное (`toX`) и вертикальное (`toY`) смещение управляющей точки (точки касательной) для точки связи с номером `nTo`. Координаты указываются в точках экрана в масштабе 100%.

Возвращаемое значение:

Номер добавленного отрезка в объекте или `-1` в случае ошибки (если `Editor` – объект не того типа).

Примечания:

Эта функция добавляет во внутренний набор объекта отрезок кривой Безье, соединяющий между собой точки связи (шины) с номерами `nFrom` и `nTo`. Номера точек в объекте возвращаются функциями `rdsCEAddInternalPoint` (стр. 413),

`rdsCEAddBlockPoint` (стр. 410) и `rdsCEAddBusPoint` (стр. 411) при создании этих точек. На момент добавления отрезка точки с номерами `nFrom` и `nTo` могут еще не существовать – главное, чтобы они существовали на момент фактического создания связи или шины по данным этого объекта.

Пара координат (`fromX, fromY`) задает смещение (в точках экрана) точки касательной кривой Безье относительно точки с номером `nFrom`. Если точка с этим номером имеет координаты (`x, y`), то касательная кривой в этой точке будет проходить через (`x, y`) – (`x+fromX, y+fromY`). Точно так же пара координат (`toX, toY`) задает смещение касательной в точке с номером `nTo`.

См. также:

```
rdsCECreateEditor (стр. 408), rdsCEAddLine (стр. 414),  
rdsCEAddInternalPoint (стр. 413), rdsCEAddBlockPoint (стр. 410),  
rdsCEAddBusPoint (стр. 411), RDS_LINEDESCRIPTION (стр. 129).
```

A.5.23.3. `rdsCEAddBlockPoint` – добавление точки соединения с блоком

Функция `rdsCEAddBlockPoint` добавляет в объект редактирования связи новую точку соединения с блоком.

```
int RDSCALL rdsCEAddBlockPoint(  
    RDS_HOBJECT Editor,    // Объект  
    RDS_BHANDLE Block,     // Блок  
    LPSTR VarName,         // Имя переменной блока  
    int x, int y,          // Относительные координаты точки  
    BOOL DisplayName       // Отображать имя переменной  
);
```

Тип указателя на эту функцию:

```
RDS_IHоBhSIIB
```

Параметры:

`Editor`

Идентификатор вспомогательного объекта для редактирования связи, ранее созданного функцией `rdsCECreateEditor` (стр. 408).

`Block`

Идентификатор блока, точка соединения с которым добавляется.

`VarName`

Указатель на строку с полным именем переменной блока `Block`, с которой соединяется добавляемая точка связи. Имя может включать в себя поля структур и элементы массивов и матриц, например, “`compl.Re`” или “`M1[2][4]`”.

`x, y`

Горизонтальная (`x`) и вертикальная (`y`) координаты добавляемой точки относительно точки привязки блока `Block` (начала координат векторной картинкой для блоков с картинкой или левого верхнего угла изображения блока для всех остальных блоков). Координаты указываются в точках экрана в масштабе 100%.

`DisplayName`

TRUE – отображать имя переменной рядом с точкой связи, FALSE – не отображать.

Возвращаемое значение:

Номер добавленной точки или -1 в случае ошибки (если Editor – объект не того типа). Эти номера точек используются в функциях создания отрезков связи rdsCEAddLine (стр. 414) и rdsCEAddBezier (стр. 409).

Примечания:

Эта функция добавляет во внутренний набор объекта новую точку связи, которая будет соединять ее с переменной VarName блока Block. Координаты точки указываются относительно точки привязки блока.

Пример использования функции rdsCEAddBlockPoint приведен в §2.16.2.

См. также:

rdsCECreateEditor (стр. 408), rdsCEAddLine (стр. 414),
rdsCEAddBezier (стр. 409), rdsCEAddInternalPoint (стр. 413),
rdsCEAddBusPoint (стр. 411), RDS_POINTDESCRIPTION (стр. 133).

A.5.23.4. rdsCEAddBusPoint – добавление точки соединения с шиной

Функция rdsCEAddBusPoint добавляет в объект редактирования связи новую точку соединения с шиной.

```
int RDSCALL rdsCEAddBusPoint(  
    RDS_HOBJECT Editor,    // Объект  
    RDS_CHANDLE Bus,      // Шина  
    LPSTR VarName,        // Имя переменной канала  
    BOOL Output,          // Источник данных  
    int x, int y,          // Координаты точки  
    BOOL DisplayName      // Отображать имя переменной  
);
```

Тип указателя на эту функцию:

RDS_IHoChSBIIB

Параметры:

Editor

Идентификатор вспомогательного объекта для редактирования связи, ранее созданного функцией rdsCECreateEditor (стр. 408).

Bus

Идентификатор шины, точка соединения с которой добавляется.

VarName

Указатель на строку с именем канала шины Bus, с которым соединяется добавляемая точка связи. Для связей, соединяемых с выходом канала, в имя могут входить поля структур и элементы массивов и матриц, например, “compl.Re” или “M1[2][4]”.

Output

TRUE – точка подключает связь к выходу канала шины (канал будет источником данных для данной связи), FALSE – точка подключает связь ко входу канала (канал будет получателем данных).

x, y

Горизонтальная (x) и вертикальная (y) координаты точки на рабочем поле. Координаты указываются в точках экрана в масштабе 100%.

DisplayName

TRUE – отображать имя переменной канала рядом с точкой связи, FALSE – не отображать.

Возвращаемое значение:

Номер добавленной точки или -1 в случае ошибки (если Editor – объект не того типа). Эти номера точек используются в функциях создания отрезков связи rdsCEAddLine (стр. 414) и rdsCEAddBezier (стр. 409).

Примечания:

Эта функция добавляет во внутренний набор объекта новую точку связи, которая будет соединять ее с каналом VarName шины Bus. Параметр Output определяет, будет эта точка получать данные из канала (TRUE) или записывать их в канал (FALSE). При Output==FALSE, когда связь передает данные в шину, в параметре VarName может передаваться только имя канала. При Output==TRUE, когда связь забирает данные из шины, в параметре VarName может также указываться имя внутреннего элемента сложной переменной канала (например, поле структуры или элемент матрицы).

См. также:

rdsCECreateEditor (стр. 408), rdsCEAddLine (стр. 414),
rdsCEAddBezier (стр. 409), rdsCEAddInternalPoint (стр. 413),
rdsCEAddBlockPoint (стр. 410), RDS_POINTDESCRIPTION (стр. 133).

A.5.23.5. rdsCEAddChannel – добавление канала шины

Функция rdsCEAddChannel добавляет в объект редактирования шины новый канал передачи данных.

```
BOOL RDSCALL rdsCEAddChannel(  
    RDS_HOBJECT Editor,    // Объект  
    LPSTR ChnName,        // Имя канала  
    char BaseVarType,      // Тип переменной  
    LPSTR StructType,      // Имя типа структуры  
    int ArrayDepth         // Вложенность матрицы  
);
```

Тип указателя на эту функцию:

RDS_BHoSCSI

Параметры:

Editor

Идентификатор вспомогательного объекта для редактирования связи (шины), ранее созданного функцией rdsCECreateEditor (стр. 408).

ChnName

Указатель на строку с именем добавляемого канала.

BaseVarType

Тип переменной канала – одна из констант RDS_VARTYPE_* (стр. 49) кроме RDS_VARTYPE_STRUCTEND (это служебная константа, не тип переменной) и RDS_VARTYPE_ARRAY (массивы задаются другим способом). Если нужно указать в качестве типа структуру, в BaseVarType передается RDS_VARTYPE_STRUCT, а в параметре StructType – имя типа структуры, под которым она зарегистрирована в РДС. Если нужно указать в качестве типа матрицу, в BaseVarType передается *min*

элемента матрицы, а в параметре `ArrayDepth` – вложенность (1 для матрицы, 2 для матрицы матриц и т.д., см. ниже).

`StructType`

Имя типа структуры, если в параметре `BaseVarType` передана константа `RDS_VARTYPE_STRUCT`.

`ArrayDepth`

Вложенность матрицы переменной канала: 0 – простая переменная типа `BaseVarType`, 1 – матрица типа `BaseVarType`, 2 – матрица матриц `BaseVarType` и т.п.

Возвращаемое значение:

`TRUE` – канал добавлен, `FALSE` – добавление невозможно (недопустимый тип, канал с таким именем уже есть и т.п.).

Примечания:

Эта функция добавляет во внутренний набор объекта новый канал передачи данных (эта информация будет использована только в том случае, если объект `Editor` будет использован для создания шины). Имя канала задается параметром `ChnName`, а тип – сочетанием параметров `BaseVarType`, `StructType` и `ArrayDepth`. Параметр `BaseVarType` задает базовый тип элемента (тип самой переменной или тип элемента матрицы), в параметре `StructType` передается имя типа структуры, если базовый тип – структура, а в параметре `ArrayDepth` – вложенность матриц (0 – простая переменная). Например:

<i>BaseVarType</i>	<i>StructType</i>	<i>Array Depth</i>	<i>Tun</i>
<code>RDS_VARTYPE_INT</code>	<code>NULL</code>	0	Целое число (<code>int</code>)
<code>RDS_VARTYPE_INT</code>	<code>NULL</code>	1	Матрица целых
<code>RDS_VARTYPE_INT</code>	<code>NULL</code>	2	Матрица матриц целых
<code>RDS_VARTYPE_STRUCT</code>	<code>"Complex"</code>	0	Структура “Complex”
<code>RDS_VARTYPE_STRUCT</code>	<code>"Complex"</code>	1	Матрица структур “Complex”

Такой же способ задания типа переменных используется и в функции `rdsVSAddVar` (стр. 432).

См. также:

`rdsCECreateEditor` (стр. 408), `rdsCECreateConnBus` (стр. 415),
`rdsCEEditConnBus` (стр. 416), `rdsVSAddVar` (стр. 432).

A.5.23.6. `rdsCEAddInternalPoint` – добавление промежуточной точки

Функция `rdsCEAddInternalPoint` добавляет в объект редактирования связи новую промежуточную точку.

```
int RDSCALL rdsCEAddInternalPoint(
    RDS_ОБЪЕКТ Editor,    // Объект
    int x, int y           // Координаты точки
);
```

Тип указателя на эту функцию:

`RDS_IHoII`

Параметры:

Editor

Идентификатор вспомогательного объекта для редактирования связи, ранее созданного функцией `rdsCECreateEditor` (стр. 408).

x, y

Горизонтальная (x) и вертикальная (y) координаты точки на рабочем поле. Координаты указываются в точках экрана в масштабе 100%.

Возвращаемое значение:

Номер добавленной точки или -1 в случае ошибки (если Editor – объект не того типа). Эти номера точек используются в функциях создания отрезков связи `rdsCEAddLine` (стр. 414) и `rdsCEAddBezier` (стр. 409).

Примечания:

Эта функция добавляет во внутренний набор объекта новую промежуточную точку связи. К этой точке может подходить несколько отрезков связи под разными углами. Если таких отрезков будет три и более, в точке будет изображаться узел, а сама точка будет считаться точкой разветвления.

См. также:

`rdsCECreateEditor` (стр. 408), `rdsCEAddLine` (стр. 414),
`rdsCEAddBezier` (стр. 409), `rdsCEAddBusPoint` (стр. 411),
`rdsCEAddBlockPoint` (стр. 410), `RDS_POINTDESCRIPTION` (стр. 133).

A.5.23.7. `rdsCEAddLine` – добавление отрезка прямой

Функция `rdsCEAddLine` добавляет в объект редактирования связи отрезок прямой, соединяющий пару точек.

```
int RDSCALL rdsCEAddLine(  
    RDS_HOBJECT Editor,    // Объект  
    int nFrom,              // Номер первой точки  
    int nTo                 // Номер второй точки  
);
```

Тип указателя на эту функцию:

`RDS_IHOII`

Параметры:

Editor

Идентификатор вспомогательного объекта для редактирования связи, ранее созданного функцией `rdsCECreateEditor` (стр. 408).

nFrom

Номер первой из двух соединяемых отрезком точек во внутреннем наборе точек объекта.

nTo

Номер второй из двух соединяемых отрезком точек во внутреннем наборе точек объекта.

Возвращаемое значение:

Номер добавленного отрезка в объекте или -1 в случае ошибки (если Editor – объект не того типа).

Примечания:

Эта функция добавляет во внутренний набор объекта отрезок прямой, соединяющий между собой точки связи (шины) с номерами nFrom и nTo. Номера точек в объекте возвращаются функциями rdsCEAddInternalPoint (стр. 413), rdsCEAddBlockPoint (стр. 410) и rdsCEAddBusPoint (стр. 411) при создании этих точек. На момент добавления отрезка точки с номерами nFrom и nTo могут еще не существовать – главное, чтобы они существовали на момент фактического создания связи или шины по данным этого объекта.

Пример использования функции rdsCEAddLine приведен в §2.16.2.

См. также:

rdsCECreateEditor (стр. 408), rdsCEAddBezier (стр. 409),
rdsCEAddInternalPoint (стр. 413), rdsCEAddBlockPoint (стр. 410),
rdsCEAddBusPoint (стр. 411), RDS_LINEDESCRIPTION (стр. 129).

А.5.23.8. rdsCECreateConnBus – создание связи или шины по данным объекта

Функция rdsCECreateConnBus создает новую связь или шину по данным указанного вспомогательного объекта.

```
RDS_CHANDLE RDSCALL rdsCECreateConnBus (  
    RDS_HOBJECT Editor,    // Объект  
    RDS_BHANDLE Parent,    // Подсистема  
    int Type,              // Связь или шина  
    int *pError            // Результат  
);
```

Тип указателя на эту функцию:

RDS_ChNoBhIpI

Параметры:

Editor

Идентификатор вспомогательного объекта для редактирования связи/шины, ранее созданного функцией rdsCECreateEditor (стр. 408).

Parent

Идентификатор подсистемы, внутри которой создается связь или шина.

Type

Тип создаваемого объекта: RDS_CTCONNECTION – создать связь, RDS_CTBUS – создать шину.

pError

Указатель на целую переменную, в которую функция запишет результат создания связи или шины (код ошибки):

RDS_HCE_ERR_OK

Операция выполнена без ошибок.

RDS_HCE_ERR_BADOBJECT

Editor не является идентификатором вспомогательного объекта для редактирования связи/шины.

RDS_HCE_ERR_INVBLKBUS

Подсистема с идентификатором Parent отсутствует в схеме.

RDS_HCE_ERR_BADLINE

Одна из линий связи ссылается на номер точки, отсутствующей в объекте (линия, соединяющая точки, добавлена в объект, а одна из этих точек – нет).

RDS_HCE_ERR_ALLOC Прочие ошибки.

Если вызывающей программе не нужен код ошибки, в параметре `pError` можно передать `NULL`.

Возвращаемое значение:

Идентификатор созданной связи или шины, или `NULL` в случае ошибки.

Примечания:

Эта функция создает в подсистеме `Parent` связь или шину (в зависимости от параметра `Type`) согласно набору точек и линий, содержащемуся во вспомогательном объекте `Editor`. При создании шины из объекта также читается набор каналов передачи данных.

При создании связей и шин РДС создает в памяти множество дополнительных служебных структур, обеспечивающих их работу. Если планируется последовательное создание нескольких связей или шин, имеет смысл предварительно вызвать функцию `rdsSetSystemUpdate` (стр. 172) с параметром `FALSE`, а после создания всех связей/шин вызвать ее же с параметром `TRUE`. Это приведет к тому, что все вспомогательные структуры будут обновляться не после создания каждой связи, а после завершения создания всех связей (в момент вызова функции с параметром `TRUE`), что может заметно ускорить работу РДС.

Пример использования функции `rdsCECreateConnBus` приведен в §2.16.2.

См. также:

`rdsCECreateEditor` (стр. 408), `rdsCEEditConnBus` (стр. 416),
`rdsSetSystemUpdate` (стр. 172).

A.5.23.9. `rdsCEEditConnBus` – изменение связи или шины по данным объекта

Функция `rdsCEEditConnBus` изменяет уже существующую связь или шину по данным указанного вспомогательного объекта.

```
BOOL RDSCALL rdsCEEditConnBus (  
    RDS_HOBJECT Editor,    // Объект  
    RDS_CHANDLE Conn,      // Существующая связь/шина  
    int *pError             // Результат  
);
```

Тип указателя на эту функцию:

RDS_BHoChI

Параметры:

`Editor`

Идентификатор вспомогательного объекта для редактирования связи/шины, ранее созданного функцией `rdsCECreateEditor` (стр. 408).

`Conn`

Идентификатор связи или шины, которую должна изменить функция.

`pError`

Указатель на целую переменную, в которую функция запишет результат изменения связи или шины (код ошибки):

RDS_HCE_ERR_OK Операция выполнена без ошибок.

RDS_HCE_ERR_BADOBJECT	Editor не является идентификатором вспомогательного объекта для редактирования связи/шины.
RDS_HCE_ERR_INVBLKBUS	Связь или шина с идентификатором Conn отсутствует в схеме.
RDS_HCE_ERR_BADLINE	Одна из линий связи ссылается на номер точки, отсутствующей в объекте (линия, соединяющая точки, добавлена в объект, а одна из этих точек – нет).
RDS_HCE_ERR_ALLOC	Прочие ошибки.

Если вызывающей программе не нужен код ошибки, в параметре pError можно передать NULL.

Возвращаемое значение:

TRUE – изменение выполнено, FALSE – ошибка.

Примечания:

Эта функция задает для связи или шины Conn набор точек и линий, содержащийся во вспомогательном объекте Editor. При изменении шины ей также присваивается набор каналов передачи данных из вспомогательного объекта, если он там задан. Если во вспомогательном объекте не создан набор каналов, каналы шины Conn не будут изменены.

См. также:

rdsCECreateEditor (стр. 408), rdsCECreateConnBus (стр. 415),
rdsSetSystemUpdate (стр. 172).

A.5.23.10. Команда RDS_HCE_RESET – очистка вспомогательного объекта

Команда RDS_HCE_RESET удаляет из вспомогательного объекта редактирования связи или шины наборы точек, отрезков и каналов передачи данных.

Вызов команды:

```
rdsCommandObject(Editor, RDS_HCE_RESET);  
или  
rdsCommandObjectEx(Editor, RDS_HCE_RESET, 0, NULL);
```

Параметр:

Editor

Идентификатор вспомогательного объекта (RDS_HOBJECT) для редактирования связи, ранее созданного функцией rdsCECreateEditor (стр. 408).

Примечания:

После очистки объекта он становится пустым, и в него можно снова добавлять точки, линии и каналы передачи данных для создания или изменения новой связи или шины. Для вызова команды используются функции общего назначения rdsCommandObject (стр. 400) и rdsCommandObjectEx (стр. 400).

Пример использования команды RDS_HCE_RESET приведен в §2.16.2.

См. также:

rdsCECreateEditor (стр. 408), rdsCommandObject (стр. 400),
rdsCommandObjectEx (стр. 400).

A.5.23.11. Макрос rdsCEClearEditor – очистка вспомогательного объекта

Макрос rdsCEClearEditor предназначен для очистки объекта редактирования связи.

```
rdsCEClearEditor(  
    editor          // Вспомогательный объект  
)
```

Определение:

```
#define rdsCEClearEditor(editor) \  
    rdsCommandObject((editor), RDS_HCE_RESET)
```

Параметр:

editor

Идентификатор вспомогательного объекта для редактирования связи, ранее созданного функцией rdsCECreateEditor (стр. 408).

Примечания:

Этот макрос включает в себя вызов команды RDS_HCE_RESET (стр. 418), его можно использовать для того, чтобы сделать текст программы более понятным.

См. также:

RDS_HCE_RESET (стр. 418).

A.5.23.12. Макрос rdsCECreateBus – создание шины

Макрос rdsCECreateBus предназначен для создания шины по данным объекта редактирования связи.

```
rdsCECreateBus(  
    editor,        // Вспомогательный объект  
    parent,        // Подсистема  
    perror         // Возвращаемый код ошибки  
)
```

Определение:

```
#define rdsCECreateBus(editor, parent, perror) \  
    rdsCECreateConnBus((editor), (parent), RDS_CTBUS, (perror))
```

Параметры:

editor

Идентификатор вспомогательного объекта (RDS_HOBJECT) для редактирования шины, ранее созданного функцией rdsCECreateEditor (стр. 408).

parent

Идентификатор подсистемы (RDS_BHANDLE), внутри которой создается шина.

perror

Указатель на целую переменную (int*), в которую функция запишет результат создания шины (см. описание функции rdsCECreateConnBus на стр. 415).

Примечания:

Этот макрос включает в себя вызов функции rdsCECreateConnBus, в который уже подставлена константа RDS_CTBUS, указывающая на создание именно шины, а не связи.

См. также:

`rdsCECreateConnBus` (стр. 415).

А.5.23.13. Макрос `rdsCECreateConnection` – создание связи

Макрос `rdsCECreateConnection` предназначен для создания связи по данным объекта редактирования связи.

```
rdsCECreateConnection(  
    editor,    // Вспомогательный объект  
    parent,    // Подсистема  
    perror     // Возвращаемый код ошибки  
)
```

Определение:

```
#define rdsCECreateConnection(editor,parent,perror) \  
    rdsCECreateConnBus((editor),(parent), \  
        RDS_CTCONNECTION,(perror))
```

Параметры:

`editor`

Идентификатор вспомогательного объекта (`RDS_HOBJECT`) для редактирования связи, ранее созданного функцией `rdsCECreateEditor` (стр. 408).

`parent`

Идентификатор подсистемы (`RDS_BHANDLE`), внутри которой создается связь.

`perror`

Указатель на целую переменную (`int*`), в которую функция запишет результат создания связи (см. описание функции `rdsCECreateConnBus` на стр. 415).

Примечания:

Этот макрос включает в себя вызов функции `rdsCECreateConnBus`, в который уже подставлена константа `RDS_CTCONNECTION`, указывающая на создание именно связи, а не шины.

См. также:

`rdsCECreateConnBus` (стр. 415).

А.5.23.14. Макросы `rdsCEEditConnection` и `rdsCEEditBus` – изменение связи и шины по данным объекта

Макросы `rdsCEEditConnection` и `rdsCEEditBus` являются синонимами функции `rdsCEEditConnBus` (стр. 416), введенными для лучшей читаемости текста программы. В настоящее время они практически не используются.

Определения:

```
#define rdsCEEditConnection    rdsCEEditConnBus  
#define rdsCEEditBus          rdsCEEditConnBus
```

См. также:

`rdsCEEditConnBus` (стр. 416).

А.5.24. Вспомогательный объект для работы со списком блоков и связей

Описываются функции и команды вспомогательного объекта РДС, предназначенного для создания и поддержания в актуальном состоянии списков блоков и связей схемы.

А.5.24.1. rdsBCLCreateList – создать объект для хранения списка блоков и связей

Функция rdsBCLCreateList создает вспомогательный объект РДС, во внутренних данных которого может храниться список идентификаторов блоков и связей схемы. Список может создаваться пустым или автоматически заполняться блоками и связями указанной подсистемы.

```
RDS_HOBJECT RDSCALL rdsBCLCreateList(  
    RDS_BHANDLE System,    // Подсистема  
    DWORD TypeMask,       // Набор типов объектов  
    BOOL Recurse          // Добавлять из вложенных подсистем  
);
```

Тип указателя на эту функцию:

RDS_HoBhDwB

Параметры:

System

Идентификатор подсистемы, блоки и связи которой нужно добавить в создаваемый список, либо NULL, если нужно создать пустой список.

TypeMask

Набор объединенных битовым ИЛИ констант типов блоков и связей (см. также стр. 113 и 120), которые нужно добавить в список из подсистемы System. Если в System передано значение NULL, параметр TypeMask игнорируется.

RDS_BTSYSTEM	подсистема
RDS_BTSIMPLEBLOCK	простой блок
RDS_BTINPUTBLOCK	внешний вход
RDS_BTOUTPUTBLOCK	внешний выход
RDS_BTBUSPORT	ввод шины
RDS_CTCONNECTION	связь
RDS_CTBUS	шина

Кроме указанных выше стандартных констант типов можно также использовать специальные объединяющие константы:

RDS_BTALLTYPES	все типы блоков
RDS_CTALLTYPES	связи и шины

Recurse

TRUE, если нужно добавлять в список не только блоки и связи, непосредственно находящиеся в подсистеме System, но и блоки и связи во всех ее вложенных подсистемах. FALSE, если нужно добавить только блоки и связи, непосредственно находящиеся в подсистеме System. Если в System передано значение NULL, параметр Recurse игнорируется.

Возвращаемое значение:

Уникальный идентификатор созданного объекта.

Примечания:

Эта функция создает вспомогательный объект для хранения списка блоков и связей и возвращает его идентификатор типа `RDS_HOBJECT`. Список может быть автоматически заполнен идентификаторами блоков и связей из подсистемы `System`, или оставлен пустым, если `System==NULL`. После создания списка в него могут добавляться отдельные блоки и связи функциями `rdsBCLAddBlock` (стр. 421) и `rdsBCLAddConn` (стр. 422) соответственно. Для доступа к массивам идентификаторов, хранящимся внутри объекта, служат команды `RDS_HBCL_BLOCKARRAY` (стр. 425) и `RDS_HBCL_CONNARRAY` (стр. 427). По умолчанию объект не отслеживает удаление блоков и связей, попавших в список – это можно изменить командой `RDS_HBCL_AUTODELETE` (стр. 424).

Созданный вспомогательный объект будет существовать до тех пор, пока схема не будет выгружена из памяти, или пока он не будет удален вызовом функции `rdsDeleteObject` (стр. 401).

Пример использования функции `rdsBCLCreateList` приведен в §2.16.2.

См. также:

Вспомогательные объекты (стр. 399), `rdsBCLAddBlock` (стр. 421), `rdsBCLAddConn` (стр. 422), `RDS_HBCL_BLOCKARRAY` (стр. 425), `RDS_HBCL_CONNARRAY` (стр. 427), `RDS_HBCL_AUTODELETE` (стр. 424).

A.5.24.2. `rdsBCLAddBlock` – добавление блока в список

Функция `rdsBCLAddBlock` добавляет в список блоков и связей блок с указанным идентификатором.

```
int RDSCALL rdsBCLAddBlock(  
    RDS_HOBJECT List,      // Список  
    RDS_BHANDLE Block,    // Блок  
    BOOL IgnoreDup         // Игнорировать повторы  
);
```

Тип указателя на эту функцию:

`RDS_IHoBhB`

Параметры:

`List`

Идентификатор вспомогательного объекта-списка, ранее созданного функцией `rdsBCLCreateList` (стр. 420).

`Block`

Идентификатор блока, который нужно добавить во внутренний массив объекта `List`.

`IgnoreDup`

`TRUE` – перед добавлением проверить, нет ли уже в списке блока с идентификатором `Block`. Если он есть, второй раз он добавлен не будет. `FALSE` – добавлять блок в любом случае, даже если он уже есть в списке (ускоряет работу при больших размерах списков, но может привести к повторам идентификаторов блоков в списке).

Возвращаемое значение:

Индекс добавленного блока во внутреннем массиве объекта `List`. Если в параметре `IgnoreDup` было передано значение `TRUE`, и блок с идентификатором `Block` уже находится в массиве, функция вернет индекс найденного идентификатора `Block`.

Примечания:

Эта функция позволяет добавлять в список отдельные блоки. Для определения общего числа блоков в списке можно использовать команду `RDS_HBCL_BLOCKCOUNT` (стр. 426), для получения указателя на внутренний массив идентификаторов блоков – команду `RDS_HBCL_BLOCKARRAY` (стр. 425). Функция удаления отдельных блоков из списка не предусмотрена: для того, чтобы выбросить блок из списка, следует вручную заменить в массиве его идентификатор на `NULL`.

Пример использования функции `rdsBCLAddBlock` приведен в §2.16.2.

См. также:

`rdsBCLCreateList` (стр. 420), `rdsBCLAddConn` (стр. 422),
`RDS_HBCL_BLOCKCOUNT` (стр. 426), `RDS_HBCL_BLOCKARRAY` (стр. 425),
`RDS_HBCL_CLEAR` (стр. 426).

A.5.24.3. `rdsBCLAddConn` – добавление связи или шины в список

Функция `rdsBCLAddConn` добавляет в список блоков и связей связь или шину с указанным идентификатором.

```
int RDSCALL rdsBCLAddConn(  
    RDS_HOBJECT List,      // Список  
    RDS_CHANDLE Conn,     // Связь  
    BOOL IgnoreDup         // Игнорировать повторы  
);
```

Тип указателя на эту функцию:

`RDS_IHoChB`

Параметры:

`List`

Идентификатор вспомогательного объекта-списка, ранее созданного функцией `rdsBCLCreateList` (стр. 420).

`Conn`

Идентификатор связи или шины, которую нужно добавить во внутренний массив объекта `List`.

`IgnoreDup`

`TRUE` – перед добавлением проверить, нет ли уже в списке связи (шины) с идентификатором `Conn`. Если она есть, второй раз она добавлена не будет. `FALSE` – добавлять связь в любом случае, даже если она уже есть в списке (ускоряет работу при больших размерах списков, но может привести к повторам идентификаторов связей и шин в списке).

Возвращаемое значение:

Индекс добавленной связи или шины во внутреннем массиве объекта `List`. Если в параметре `IgnoreDup` было передано значение `TRUE`, и связь с идентификатором `Conn` уже находится в массиве, функция вернет индекс найденного идентификатора `Conn`.

Примечания:

Эта функция позволяет добавлять в список отдельные связи и шины. Для определения общего числа связей и шин в списке следует использовать команду `RDS_HBCL_CONNCOUNT` (стр. 428), для получения указателя на внутренний массив идентификаторов связей и шин – команду `RDS_HBCL_CONNARRAY` (стр. 427). Функция удаления связей из списка не

предусмотрена: для того, чтобы выбросить связь или шину из списка, следует вручную заменить в массиве ее идентификатор на NULL.

Пример использования функции `rdsBCLAddConn` приведен в §2.16.2.

См. также:

`rdsBCLCreateList` (стр. 420), `rdsBCLAddBlock` (стр. 421),
`RDS_HBCL_CONNCOUNT` (стр. 428), `RDS_HBCL_CONNARRAY` (стр. 427),
`RDS_HBCL_CLEAR` (стр. 426).

A.5.24.4. `rdsBCLExecuteGroupSetDialog` – вызов окна групповой установки

Функция `rdsBCLExecuteGroupSetDialog` открывает окно для групповой установки параметров блоков и связей в указанном списке и, если пользователь нажмет в нем кнопку “OK”, устанавливает заданные в окне параметры для всех блоков и связей из списка.

```
void RDSCALL rdsBCLExecuteGroupSetDialog(  
    RDS_HOBJECT List,          // Список  
    DWORD Flags,              // Флаги (RDS_GS_*)  
    LPSTR Title,              // Заголовок окна  
    BOOL Undo                  // Разрешить отмену  
);
```

Тип указателя на эту функцию:

`RDS_VHoDwSB`

Параметры:

`List`

Идентификатор вспомогательного объекта-списка, ранее созданного функцией `rdsBCLCreateList` (стр. 420).

`Flags`

Объединенные битовым ИЛИ флаги запрета отдельных операций в окне групповой установки:

<code>RDS_GS_DISABLEAUTOCOMP</code>	Запрет изменения параметров автоматической компиляции модели блока
<code>RDS_GS_DISABLEBLOCKAPPEARANCE</code>	Запрет изменения параметров внешнего вида блока (изображение имени, точки привязки и т.п.)
<code>RDS_GS_DISABLEBUSPACK</code>	Запрет удаления пустых каналов шин
<code>RDS_GS_DISABLECONNAPPEARANCE</code>	Запрет изменения внешнего вида связи и шины
<code>RDS_GS_DISABLECONNSTATE</code>	Запрет включения/выключения связи
<code>RDS_GS_DISABLEDBLCLICK</code>	Запрет изменения реакции блока на двойной щелчок
<code>RDS_GS_DISABLEDLLFUN</code>	Запрет изменения функции модели блока
<code>RDS_GS_DISABLEDLLOPTIONS</code>	Запрет изменения флагов реакции функции модели блока
<code>RDS_GS_DISABLEDRAWTYPE</code>	Запрет изменения способа отображения внешнего вида блока (картинка, прямоугольник, программное рисование)

RDS_GS_DISABLEEDITORPARAMS	Запрет изменения параметров окна подсистемы
RDS_GS_DISABLELAYERCHANGE	Запрет изменения слоя блоков, связей и шин
RDS_GS_DISABLEPICTURE	Запрет изменения векторной картинки блока и ее параметров
RDS_GS_DISABLEREMARKS	Запрет изменения комментариев блоков
RDS_GS_DISABLESIZING	Запрет изменения параметров масштаба и размера блока
RDS_GS_DISABLEVARCHANGE	Запрет изменения переменных блока
RDS_GS_DISABLEVARVALUES	Запрет присвоения значений переменным блока

Если в параметре `Flags` будет передано нулевое значение (не установлен ни один флаг), будут разрешены все операции.

Title

Указатель на строку с заголовком, который должно иметь окно групповой установки. Если в этом параметре передать `NULL`, окно будет иметь стандартный заголовок “Параметры выделенных объектов”.

Undo

`TRUE`, если нужно разрешить пользователю в будущем отменить выполненную операцию групповой установки (он сможет сделать это, как обычно, выбрав пункт меню “Система | Отмена”). `FALSE`, если отмену нужно запретить.

Примечания:

Эта функция открывает стандартное окно групповой установки параметров РДС для блоков и связей, находящихся в списке `List`. В этом окне пользователь может одновременно изменить параметры всех этих блоков и связей.

См. также:

`rdsBCLCreateList` (стр. 420).

A.5.24.5. Команда RDS_HBCL_AUTODELETE – отслеживание удаления блоков и связей

Команда `RDS_HBCL_AUTODELETE` позволяет включать и выключать автоматическое выбрасывание из списка удаленных блоков и связей, а также считывать текущее состояние этой функции.

Вызов команды для установки:

```
int iOn=... // 1 - включить, 0 - выключить
rdsSetObjectInt(List, RDS_HBCL_AUTODELETE, 0, iOn);
```

Вызов команды для чтения:

```
int iOn=rdsGetObjectInt(List, RDS_HBCL_AUTODELETE, 0);
```

Параметры и результат:

List

Идентификатор вспомогательного объекта-списка (`RDS_HOBJECT`), ранее созданного функцией `rdsBCLCreateList` (стр. 420).

iOn

Целое число (int), указывающее на состояние функции отслеживания удаленных объектов: 1 – автоматическое выбрасывание удаленных объектов из списка включено, 0 – выключено.

Примечания:

При включенном автоматическом отслеживании РДС будет следить за удалением блоков и связей и автоматически заменять в списке идентификаторы удаленных объектов значением NULL. Это необходимо в тех случаях, когда заполнение списка и обращение к нему разнесены во времени, и в интервале между ними блоки и связи могут быть удалены пользователем или программно. Если не включить отслеживание удаления, в списке могут оказаться идентификаторы, которым больше не соответствует блок или связь, что приведет к ошибкам при использовании этих идентификаторов в сервисных функциях. Если же обращение к списку производится сразу после его заполнения (например, в одной и той же реакции модели блока), в автоматическом отслеживании удаления нет необходимости.

Пример использования команды RDS_HBCL_AUTODELETE приведен в §2.16.2.

См. также:

rdsBCLCreateList (стр. 420), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404).

A.5.24.6. Команда RDS_HBCL_BLOCKARRAY – получение массива блоков

Команда RDS_HBCL_BLOCKARRAY позволяет получить указатель на внутренний массив идентификаторов блоков в объекте-списке и общее число идентификаторов в этом массиве.

Вызов команды:

```
int iCount;  
RDS_BHANDLE *arrBlocks=(RDS_BHANDLE*) rdsGetObjectArray(  
    List,RDS_HBCL_BLOCKARRAY,0,&iCount);
```

Параметры и результат:

List

Идентификатор вспомогательного объекта-списка (RDS_NOBJECT), ранее созданного функцией rdsBCLCreateList (стр. 420).

iCount

Целая (int) переменная, в которую функция rdsGetObjectArray (стр. 402) запишет общее число блоков в списке. Вместо указателя на эту переменную можно передать NULL, если размер списка не нужен вызывающей программе.

arrBlocks

Указатель на первый элемент массива идентификаторов блоков. Функция rdsGetObjectArray возвращает указатель общего вида, поэтому перед использованием его нужно привести к типу “указатель на RDS_BHANDLE” (RDS_BHANDLE*).

Примечания:

Команда RDS_HBCL_BLOCKARRAY, переданная через сервисную функцию rdsGetObjectArray, возвращает указатель на первый элемент внутреннего массива идентификаторов блоков из объекта List. Если командой RDS_HBCL_AUTODELETE (стр. 424) включено отслеживание удаления блоков, некоторые элементы этого массива могут иметь значение NULL. В приведенной выше записи вызова команды общее число

элементов в массиве (включая элементы со значением NULL) записывается в переменную `iCount`. Идентификаторы блоков будут элементами массива `arrBlocks`: от `arrBlocks[0]` до `arrBlocks[iCount-1]`.

Следует помнить, что указатель на внутренний массив идентификаторов, возвращаемый функцией `rdsGetObjectArray`, можно использовать только до тех пор, пока вызовом функции `rdsBCLAddBlock` (стр. 421) в список не будет добавлен новый блок. Добавление нового блока может привести к отведению нового массива и уничтожению старого, поэтому после добавления блоков указатель на массив идентификаторов нужно получать заново.

Пример использования команды `RDS_HBCL_BLOCKARRAY` приведен в §2.16.2.

См. также:

`rdsBCLCreateList` (стр. 420), `rdsGetObjectArray` (стр. 402),
`RDS_HBCL_AUTODELETE` (стр. 424), `rdsBCLAddBlock` (стр. 421).

A.5.24.7. Команда `RDS_HBCL_BLOCKCOUNT` – число блоков в списке

Команда `RDS_HBCL_BLOCKCOUNT` возвращает общее число идентификаторов блоков во внутреннем массиве объекта-списка.

Вызов команды для чтения:

```
int iCount=rdsGetObjectInt(List,RDS_HBCL_BLOCKCOUNT,0);
```

Параметры и результат:

`List`

Идентификатор вспомогательного объекта-списка (`RDS_HOBJECT`), ранее созданного функцией `rdsBCLCreateList` (стр. 420).

`iCount`

Число блоков в списке (`int`).

Примечания:

В возвращаемое число входят также выброшенные из списка идентификаторы блоков (соответствующие им элементы массива имеют значение NULL). Фактически, возвращается общий размер внутреннего массива идентификаторов блоков внутри объекта-списка `List`.

См. также:

`rdsBCLCreateList` (стр. 420), `rdsGetObjectInt` (стр. 404),
`RDS_HBCL_BLOCKARRAY` (стр. 425).

A.5.24.8. Команда `RDS_HBCL_CLEAR` – очистка списка

Команда `RDS_HBCL_CLEAR` очищает список блоков и связей.

Вызов команды:

```
rdsCommandObject(List,RDS_HBCL_CLEAR);  
или  
rdsCommandObjectEx(List,RDS_HBCL_CLEAR,0,NULL);
```

Параметр:

`List`

Идентификатор вспомогательного объекта-списка (`RDS_HOBJECT`), ранее созданного функцией `rdsBCLCreateList` (стр. 420).

Примечания:

После очистки объекта-списка в него можно снова добавлять блоки и связи функциями `rdsBCLAddBlock` (стр. 421) и `rdsBCLAddConn` (стр. 422). Для вызова команды используются сервисные функции общего назначения `rdsCommandObject` (стр. 400) и `rdsCommandObjectEx` (стр. 400).

Пример использования команды `RDS_HBCL_CLEAR` приведен в §2.16.2.

См. также:

`rdsBCLCreateList` (стр. 420), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400).

A.5.24.9. Команда `RDS_HBCL_CONNARRAY` – получение массива связей/шин

Команда `RDS_HBCL_CONNARRAY` позволяет получить указатель на внутренний массив идентификаторов связей (шин) в объекте-списке и общее число идентификаторов в этом массиве.

Вызов команды:

```
int iCount;  
RDS_CHANDLE *arrConns=(RDS_CHANDLE*) rdsGetObjectArray(  
    List,RDS_HBCL_CONNARRAY,0,&iCount);
```

Параметры и результат:

`List`

Идентификатор вспомогательного объекта-списка (`RDS_HOBJECT`), ранее созданного функцией `rdsBCLCreateList` (стр. 420).

`iCount`

Целая (`int`) переменная, в которую функция `rdsGetObjectArray` (стр. 402) запишет общее число связей и шин в списке. Вместо указателя на эту переменную можно передать `NULL`, если размер списка не нужен вызывающей программе.

`arrConns`

Указатель на первый элемент массива идентификаторов связей и шин. Функция `rdsGetObjectArray` возвращает указатель общего вида, поэтому перед использованием его нужно привести к типу “указатель на `RDS_CHANDLE`” (`RDS_CHANDLE*`).

Примечания:

Команда `RDS_HBCL_CONNARRAY`, переданная через сервисную функцию `rdsGetObjectArray`, возвращает указатель на первый элемент внутреннего массива идентификаторов связей и шин из объекта `List`. Если командой `RDS_HBCL_AUTODELETE` (стр. 424) включено отслеживание удаления блоков и связей, некоторые элементы этого массива могут иметь значение `NULL`. В приведенной выше записи вызова команды общее число элементов в массиве (включая элементы со значением `NULL`) записывается в переменную `iCount`. Идентификаторы связей и шин будут элементами массива `arrConns`: от `arrConns[0]` до `arrConns[iCount-1]`.

Следует помнить, что указатель на внутренний массив идентификаторов, возвращаемый функцией `rdsGetObjectArray`, можно использовать только до тех пор, пока вызовом функции `rdsBCLAddConn` (стр. 422) в список не будет добавлена новая связь. Добавление новой связи может привести к отведению нового массива и уничтожению старого, поэтому после добавления связей и шин указатель на массив идентификаторов нужно получать заново.

Пример использования команды RDS_HBCL_CONNARRAY приведен в §2.16.2.

См. также:

rdsBCLCreateList (стр. 420), rdsGetObjectArray (стр. 402),
RDS_HBCL_AUTODELETE (стр. 424), rdsBCLAddConn (стр. 422).

A.5.24.10. Команда RDS_HBCL_CONNCOUNT – число связей и шин в списке

Команда RDS_HBCL_CONNCOUNT возвращает общее число идентификаторов связей и шин во внутреннем массиве объекта-списка.

Вызов команды для чтения:

```
int iCount=rdsGetObjectInt(List,RDS_HBCL_CONNCOUNT,0);
```

Параметры и результат:

List

Идентификатор вспомогательного объекта-списка (RDS_HOBJECT), ранее созданного функцией rdsBCLCreateList (стр. 420).

iCount

Число связей и шин в списке (int).

Примечания:

В возвращаемое число входят также выброшенные из списка идентификаторы связей и шин (соответствующие им элементы массива имеют значение NULL). Фактически, возвращается общий размер внутреннего массива идентификаторов связей внутри объекта-списка List.

См. также:

rdsBCLCreateList (стр. 420), rdsGetObjectInt (стр. 404),
RDS_HBCL_CONNARRAY (стр. 427).

A.5.24.11. Макрос rdsBCLGetBlockArray – получение массива блоков

Макрос rdsBCLGetBlockArray предназначен для получения указателя на внутренний массив идентификаторов блоков в объекте-списке и общего числа идентификаторов в этом массиве.

```
rdsBCLGetBlockArray(  
    list,          // Вспомогательный объект-список  
    pcount        // Возвращаемое число элементов  
)
```

Определение:

```
#define rdsBCLGetBlockArray(list,pcount) \  
    ((RDS_BHANDLE*)rdsGetObjectArray( \  
        (list),RDS_HBCL_BLOCKARRAY,0,(pcount)))
```

Параметры:

list

Идентификатор вспомогательного объекта-списка (RDS_HOBJECT), ранее созданного функцией rdsBCLCreateList (стр. 420).

pcount

Указатель на целую переменную (int*), в которую будет записано общее число блоков в списке. Может равняться NULL, если размер списка не нужен вызывающей программе.

Возвращаемое значение:

Указатель на первый элемент внутреннего массива идентификаторов блоков объекта list, приведенный к типу “указатель на RDS_BHANDLE”.

Примечания:

Этот макрос включает в себя вызов функции rdsGetObjectArray (стр. 402) для выполнения команды получения массива блоков RDS_HBCL_BLOCKARRAY (стр. 425) и приведение возвращенного указателя к типу RDS_BHANDLE*.

См. также:

RDS_HBCL_BLOCKARRAY (стр. 425), rdsGetObjectArray (стр. 402).

A.5.24.12. Макрос rdsBCLGetConnArray – получение массива связей и шин

Макрос rdsBCLGetConnArray предназначен для получения указателя на внутренний массив идентификаторов связей и шин в объекте-списке и общего числа идентификаторов в этом массиве.

```
rdsBCLGetConnArray(  
    list,          // Вспомогательный объект-список  
    pcount        // Возвращаемое число элементов  
)
```

Определение:

```
#define rdsBCLGetConnArray(list, pcount) \  
    ((RDS_CHANDLE*)rdsGetObjectArray( \  
        (list), RDS_HBCL_CONNARRAY, 0, (pcount)))
```

Параметры:

list

Идентификатор вспомогательного объекта-списка (RDS_HOBJECT), ранее созданного функцией rdsBCLCreateList (стр. 420).

pcount

Указатель на целую переменную (int*), в которую будет записано общее число связей и шин в списке. Может равняться NULL, если размер списка не нужен вызывающей программе.

Возвращаемое значение:

Указатель на первый элемент внутреннего массива идентификаторов связей объекта list, приведенный к типу “указатель на RDS_CHANDLE”.

Примечания:

Этот макрос включает в себя вызов функции rdsGetObjectArray (стр. 402) для выполнения команды получения массива связей и шин RDS_HBCL_CONNARRAY (стр. 427) и приведение возвращенного указателя к типу RDS_CHANDLE*.

См. также:

RDS_HBCL_CONNARRAY (стр. 427), rdsGetObjectArray (стр. 402).

A.5.25. Вспомогательный объект для изменения структуры переменных блока

Описываются функции и команды вспомогательного объекта РДС, предназначенного для работы со статическими переменными блока.

A.5.25.1. `rdsVSCreateEditor` – создать объект-редактор переменных

Функция `rdsVSCreateEditor` создает вспомогательный объект РДС, с помощью которого можно изменять структуру статических переменных блока.

```
RDS_НОБЪЕКТ RDSCALL rdsVSCreateEditor(void);
```

Тип указателя на эту функцию:

```
RDS_HoV
```

Возвращаемое значение:

Уникальный идентификатор созданного объекта.

Примечания:

Эта функция создает вспомогательный объект для редактирования статических переменных и возвращает его идентификатор типа `RDS_НОБЪЕКТ`. С помощью различных сервисных функций РДС (как общих для всех типов вспомогательных объектов, так и специализированных для данного) в этот объект можно добавлять переменные и менять их параметры. Для того, чтобы создать по данным этого объекта статические переменные блока, используется функция `rdsVSApplyToBlock` (стр. 436), для создания и регистрации в РДС нового типа структуры по данным объекта – функция `rdsVSInstallStruct` (стр. 442).

Созданный вспомогательный объект будет существовать до тех пор, пока схема не будет выгружена из памяти, или пока он не будет удален вызовом функции `rdsDeleteObject` (стр. 401).

Примеры использования функции `rdsVSCreateEditor` приведены в §2.16.1 и §4.3.

См. также:

Вспомогательные объекты (стр. 399), `rdsVSApplyToBlock` (стр. 436),
`rdsVSInstallStruct` (стр. 442), `rdsDeleteObject` (стр. 401).

A.5.25.2. `rdsVSAddAutoConn` – добавить связь с управляющей переменной

Функция `rdsVSAddAutoConn` добавляет в объект-редактор переменных связь основной переменной (входа или выхода) со вспомогательной (связанным сигналом для входа или управляющей переменной для выхода). Использование вспомогательных переменных в моделях блоков описывается в §2.5.7 и §2.5.8.

```
int RDSCALL rdsVSAddAutoConn(  
    RDS_НОБЪЕКТ Vars,           // Редактор переменных  
    LPSTR MainVarName,         // Основная переменная  
    LPSTR ConnVarName          // Вспомогательная переменная  
);
```

Тип указателя на эту функцию:

```
RDS_IHoSS
```

Параметры:

`Vars`

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

MainVarName

Указатель на строку с именем основной переменной (входа или выхода).

ConnVarName

Указатель на строку с именем вспомогательной переменной (связанного сигнала для входа или управляющей переменной для выхода).

Возвращаемое значение:

Номер добавленной связи или –1 в случае ошибки (такая связь уже существует).

Примечания:

Эта функция позволяет привязать в входе блока сигнальную переменную, которая будет автоматически взводиться при срабатывании связи, подключенной к этому входу, или привязать к выходу блока логическую или целую переменную, разрешающую срабатывание связи, идущей от этого выхода. В параметрах функции указывается имя входа или выхода (MainVarName) и имя связанной переменной (ConnVarName).

См. также:

rdsVSCreateEditor (стр. 430), rdsVSApplyToBlock (стр. 436),
RDS_HVAR_GETAUTOCONN (стр. 447), RDS_HVAR_GETAUTOCOUNT (стр. 447),
RDS_HVAR_GETAUTOMAIN (стр. 448).

A.5.25.3. rdsVSAddTypeRename – добавить переименование типов структур

Функция rdsVSAddTypeRename добавляет в объект-редактор переменных информацию об изменении названия типа указанной структуры.

```
int RDSCALL rdsVSAddTypeRename (  
    RDS_HOBJECT Vars,      // Редактор переменных  
    LPSTR OldStructType,   // Старое имя типа  
    LPSTR NewStructType    // Новое имя типа  
);
```

Тип указателя на эту функцию:

RDS_IHoSS

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

OldStructType

Указатель на строку с именем типа структуры до переименования.

NewStructType

Указатель на строку с именем типа структуры после переименования.

Возвращаемое значение:

Номер добавленной пары имен или –1 в случае ошибки (новое имя совпадает со старым или Vars – не объект-редактор переменных).

Примечания:

Эта функция запоминает в объекте-редакторе переменных Vars информацию о том, что структура OldStructType переименована в NewStructType. После этого при добавлении переменных в объект функциями rdsVSAddVar (стр. 432), rdsVSAddVarByTypeText (стр. 434), rdsVSAddVarByDescr (стр. 433) и

rdsVSCreateByDescr (стр. 437) все встреченные структуры типа OldStructType будут автоматически заменяться на NewStructType.

См. также:

rdsVSCreateEditor (стр. 430), RDS_HVAR_CLEARPYPEREN (стр. 444),
rdsVSAddVar (стр. 432), rdsVSAddVarByTypeText (стр. 434),
rdsVSAddVarByDescr (стр. 433), rdsVSCreateByDescr (стр. 437).

A.5.25.4. rdsVSAddVar – добавить переменную

Функция rdsVSAddVar добавляет переменную в объект-редактор.

```
int RDSCALL rdsVSAddVar(  
    RDS_HOBJECT Vars,          // Редактор переменных  
    int Index,                 // Номер переменной или -1  
    LPSTR VarName,             // Имя переменной  
    char BaseVarType,          // Тип переменной  
    LPSTR StructType,          // Имя типа структуры  
    DWORD Flags,               // Флаги переменной (RDS_VARFLAG_*)  
    int ArrayDepth,            // Вложенность матрицы  
    LPSTR DefVal               // Значение по умолчанию  
);
```

Тип указателя на эту функцию:

RDS_IHoISCSDwIS

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

Index

Начинающийся с нуля номер, который будет иметь эта переменная (уже существующие в наборе переменные начиная с этого номера сдвинутся на одну позицию вниз), или -1 для добавления переменной в конец списка.

VarName

Указатель на строку с именем переменной.

BaseVarType

Тип переменной – одна из констант RDS_VARTYPE_* (стр. 49) кроме RDS_VARTYPE_STRUCTEND (это служебная константа, не тип переменной) и RDS_VARTYPE_ARRAY (массивы задаются другим способом). Если нужно указать в качестве типа структуру, в BaseVarType передается RDS_VARTYPE_STRUCT, а в параметре StructType – имя типа структуры, под которым она зарегистрирована в РДС. Если нужно указать в качестве типа матрицу, в BaseVarType передается *тип элемента* матрицы, а в параметре ArrayDepth – вложенность (1 для матрицы, 2 для матрицы матриц и т.д., см. ниже).

StructType

Имя типа структуры, если в параметре BaseVarType передана константа RDS_VARTYPE_STRUCT.

Flags

Набор битовых флагов переменной (аналогично полю Flags структуры описания переменной RDS_VARDERSCRIPTION, стр. 138). Кроме стандартных флагов, можно также указать специальный битовый флаг RDS_VARFLAG_EXT_CHGNAME,

разрешающий автоматически изменить имя добавляемой переменной, если переменная с таким именем уже есть в наборе. Если переменная с именем VarName уже есть, и флаг RDS_VARFLAG_EXT_CHGNAME не указан в параметре Flags, новая переменная добавлена не будет и функция вернет -1.

ArrayDepth

Вложенность матрицы переменной: 0 – простая переменная типа BaseVarType, 1 – матрица типа BaseVarType, 2 – матрица матриц BaseVarType и т.п.

DefVal

Указатель на строку значения переменной по умолчанию (аналогично функции rdsSetBlockVarDefValueStr, стр. 334).

Возвращаемое значение:

Номер добавленной переменной или -1 в случае ошибки.

Примечания:

Эта функция добавляет в набор переменных объекта-редактора Vars новую переменную с именем VarName в позицию Index. Тип переменной задается сочетанием параметров BaseVarType, StructType и ArrayDepth. Параметр BaseVarType задает базовый тип элемента (тип самой переменной или тип элемента матрицы), в параметре StructType передается имя типа структуры, если базовый тип – структура, а в параметре ArrayDepth – вложенность матриц (0 – простая переменная). Например:

<i>BaseVarType</i>	<i>StructType</i>	<i>Array Depth</i>	<i>Тип</i>
RDS_VARTYPE_INT	NULL	0	Целое число (int)
RDS_VARTYPE_INT	NULL	1	Матрица целых
RDS_VARTYPE_INT	NULL	2	Матрица матриц целых
RDS_VARTYPE_STRUCT	"Complex"	0	Структура "Complex"
RDS_VARTYPE_STRUCT	"Complex"	1	Матрица структур "Complex"

Пример использования функции rdsVSAddVar приведен в §2.16.1.

См. также:

rdsVSCreateEditor (стр. 430), rdsVSAddVarByTypeText (стр. 434),
rdsVSAddVarByDescr (стр. 433), rdsVSCreateByDescr (стр. 437),
rdsSetBlockVarDefValueStr (стр. 334), типы переменных (стр. 49),
RDS_VARDESCRIPTION (стр. 138).

A.5.25.5. rdsVSAddVarByDescr – добавить переменную по строке описания

Функция rdsVSAddVarByDescr добавляет в объект-редактор новую переменную, соответствующую указанной строке описания.

```
int RDSCALL rdsVSAddVarByDescr
    RDS_HOBJECT Vars,      // Редактор переменных
    int Index,             // Номер переменной или -1
    LPSTR DescrString      // Строка описания
);
```

Тип указателя на эту функцию:

RDS_IHoIS

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

Index

Начинающийся с нуля номер, который будет иметь эта переменная (уже существующие в наборе переменные начиная с этого номера сдвинутся на одну позицию вниз), или `-1` для добавления переменной в конец списка.

DescrString

Указатель на строку с описанием переменной. Строка описания устроена точно так же, как строка, формируемая функцией `rdsCreateVarDescriptionString` при `StructFields==FALSE` (см. стр. 324).

Возвращаемое значение:

Номер добавленной переменной или `-1` в случае ошибки.

Примечания:

Эта функция добавляет в набор переменных объекта-редактора Vars новую переменную с описанием DescrString в позицию Index. Строка описания содержит тип, и имя переменной, ее значение по умолчанию и прочие параметры.

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsVSAddVar` (стр. 432),
`rdsVSAddVarByTypeText` (стр. 434), `rdsVSCreateByDescr` (стр. 437),
`rdsCreateVarDescriptionString` (стр. 324).

A.5.25.6. `rdsVSAddVarByTypeText` – добавить переменную по текстовому описанию типа

Функция `rdsVSAddVarByTypeText` добавляет в объект-редактор новую переменную, при этом ее тип задается текстом, понятным пользователю.

```
int RDSCALL rdsVSAddVarByTypeText
    RDS_HOBJECT Vars,      // Редактор переменных
    int Index,             // Номер переменной или -1
    LPSTR VarName,         // Имя переменной
    LPSTR VarTypeText,     // Текстовое описание типа
    DWORD Flags,           // Флаги переменной (RDS_VARFLAG_*)
    LPSTR DefVal           // Значение по умолчанию
);
```

Тип указателя на эту функцию:

`RDS_IHoISSDWS`

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

Index

Начинающийся с нуля номер, который будет иметь эта переменная (уже существующие в наборе переменные начиная с этого номера сдвинутся на одну позицию вниз), или `-1` для добавления переменной в конец списка.

VarName

Указатель на строку с именем переменной.

VarTypeText

Указатель на строку с текстовым описанием типа переменной (стр. 193), аналогичную формируемой функцией `rdsCreateVarTypeText` (стр. 325).

Flags

Набор битовых флагов переменной (аналогично полю `Flags` структуры описания переменной `RDS_VARDESCRIPTION`, стр. 138). В этой функции не используется флаг `RDS_VARFLAG_ONEINDEX`: будет переменная матрицей или массивом, определяется словами, использованными для задания ее типа в параметре `VarTypeText` (если первое слово в тексте – “массив”, переменная будет массивом, если “матрица” – матрицей).

DefVal

Указатель на строку значения переменной по умолчанию (аналогично функции `rdsSetBlockVarDefValueStr`, стр. 334).

Возвращаемое значение:

Номер добавленной переменной или –1 в случае ошибки.

Примечания:

Эта функция добавляет в набор переменных объекта-редактора `Vars` новую переменную с именем `VarName`, флагами `Flags` и значением по умолчанию `DefVal`. Тип переменной задается текстом `VarTypeText`.

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsVSAddVar` (стр. 432),
`rdsVSAddVarByDescr` (стр. 433), `rdsVSCreateByDescr` (стр. 437),
`rdsCreateVarDescriptionString` (стр. 324),
`rdsCreateVarTypeText` (стр. 325), `rdsSetBlockVarDefValueStr` (стр. 334),
названия типов переменных (стр. 193).

A.5.25.7. `rdsVSAddVarRename` – добавить переименование переменной

Функция `rdsVSAddVarRename` добавляет в объект-редактор переменных информацию об изменении имени переменной.

```
int RDSCALL rdsVSAddVarRename(  
    RDS_HOBJECT Vars,        // Редактор переменных  
    LPSTR OldVarName,        // Старое имя переменной  
    LPSTR NewVarName         // Новое имя переменной  
);
```

Тип указателя на эту функцию:

`RDS_IHoSS`

Параметры:

`Vars`

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

`OldVarName`

Указатель на строку с именем переменной до переименования.

NewVarName

Указатель на строку с именем переменной после переименования.

Возвращаемое значение:

Номер добавленного переименования или -1 в случае ошибки (новое имя совпадает со старым или Vars – не объект-редактор переменных).

Примечания:

Эта функция запоминает в объекте-редакторе переменных Vars информацию о том, что переменная OldVarName переименована в NewVarName. Эта информация будет использована при создании структуры статических переменных блока функцией rdsVSApplyToBlock (стр. 436): все связи, подключенные к OldVarName после замены структуры переменных будут подключены к NewVarName.

См. также:

rdsVSCreateEditor (стр. 430), rdsVSApplyToBlock (стр. 436).

A.5.25.8. rdsVSApplyToBlock – создать структуру переменных блока

Функция rdsVSApplyToBlock создает в указанном блоке структуру статических переменных согласно данным объекта-редактора.

```
BOOL RDSCALL rdsVSApplyToBlock(  
    RDS_HOBJECT Vars,      // Редактор переменных  
    RDS_BHANDLE Block,     // Блок  
    int *pRresult          // Результат операции  
);
```

Тип указателя на эту функцию:

RDS_BHObjpI

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

Block

Идентификатор простого блока, структуру статических переменных которого нужно заменить. Если в этом параметре передано значение NULL, структура переменных будет изменена у блока, модель которого вызвала функцию rdsVSApplyToBlock.

pRresult

Указатель на целую переменную, в которую функция запишет код ошибки. Если код ошибки не нужен вызывающей программе, в этом параметре можно передать NULL.

Код ошибки может принимать следующие значения:

RDS_HVAR_ROK	Структура переменных успешно создана.
RDS_HVAR_RVARCHCKERR	Модель блока Block не может работать с данной структурой переменных, то есть не прошла проверка типа переменных в событии RDS_BFM_VARCHCK (стр. 48).
RDS_HVAR_REMPTYVARSET	Набор переменных в объекте Vars пуст.
RDS_HVAR_RBADBLOCKTYPE	Блок Block – не простой, и ему нельзя назначить произвольную структуру переменных

RDS_HVAR_RNOBLKSIGNALS Первые две переменных в объекте *Vars* – не сигнальный вход и сигнальный выход, поэтому этот набор переменных не может быть назначен простому блоку.

Возвращаемое значение:

TRUE – структура переменных блока создана, FALSE – ошибка. Конкретный код ошибки возвращается через *pResult*.

Примечания:

Эта функция создает в блоке *Block* структуру статических переменных согласно объекту *Vars*. Прежняя структура переменных блока уничтожается, связи остаются подключенными к переменным с теми же именами. Если вызовами *rdsVSAddVarRename* (стр. 435) в *Vars* записаны какие-либо переименования переменных, связи переименованных переменных будут переключены на новые имена.

Примеры использования функции *rdsVSApplyToBlock* приведены в §2.16.1 и §4.4.

См. также:

rdsVSCreateEditor (стр. 430), *rdsVSAddVarRename* (стр. 435),
rdsVSInstallStruct (стр. 442), *rdsVSCreateFromBlock* (стр. 438).

A.5.25.9. *rdsVSCreateByDescr* – заполнить набор переменных по тексту описания

Функция *rdsVSCreateByDescr* заполняет объект-редактор списком переменных из указанного текстового описания.

```
BOOL RDSCALL rdsVSCreateByDescr
    RDS_HOBJECT Vars,           // Редактор переменных
    LPSTR DescrString          // Текст описания
);
```

Тип указателя на эту функцию:

RDS_BHoS

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией *rdsVSCreateEditor* (стр. 430).

DescrString

Указатель на текст с описанием всей структуры переменных. Текст описания структуры устроен точно так же, как строка, формируемая функцией *rdsCreateVarDescriptionString* при *StructFields==TRUE* (см. стр. 324).

Возвращаемое значение:

TRUE – набор переменных создан, FALSE – ошибка (например, в синтаксисе текстового описания).

Примечания:

Эта функция создает в объекте-редакторе *Vars* полный набор переменных по описанию *DescrString*. Текст описания содержит перечисление всех переменных с их именами, типами и прочими параметрами. Прежнее содержимое объекта *Vars* уничтожается.

Пример использования функции `rdsVSCreateByDescr` приведен в §2.16.1.

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsVSAddVarByDescr` (стр. 433),
`rdsCreateVarDescriptionString` (стр. 324).

A.5.25.10. `rdsVSCreateFromBlock` – считать структуру переменных блока

Функция `rdsVSCreateFromBlock` считывает в объект-редактор переменных структуру статических переменных указанного блока.

```
void RDCALL rdsVSCreateFromBlock(  
    RDS_HOBJECT Vars,      // Редактор переменных  
    RDS_BHANDLE Block,    // Блок  
    BOOL ReadAutoConn     // Считывать связи со вспомогательными  
);
```

Тип указателя на эту функцию:

`RDS_VHobhB`

Параметры:

`Vars`

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

`Block`

Идентификатор простого блока, структуру статических переменных которого нужно считать в объект `Vars`. Если в этом параметре передано значение `NULL`, будет считана структура переменных блока, модель которого вызвала функцию `rdsVSCreateFromBlock`.

`ReadAutoConn`

`TRUE` – считывать из блока связи между входами и связанными с ними сигналами и между выходами и их управляющими переменными (см. также функцию `rdsVSAddAutoConn`, стр. 430). `FALSE` – игнорировать такие связи, входы со связанными сигналами станут просто входами, выходы с управляющими переменными – просто выходами, при этом управляющие переменные и связанные сигналы останутся в структуре переменных как обычные переменные.

Примечания:

Эта функция считывает всю структуру статических переменных блока `Block` в объект `Vars`. Прежнее содержимое объекта `Vars` уничтожается.

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsVSAddAutoConn` (стр. 430),
`rdsVSApplyToBlock` (стр. 436).

A.5.25.11. `rdsVSExecuteEditor` – открыть окно редактора переменных

Функция `rdsVSExecuteEditor` вызывает стандартный редактор переменных РДС и позволяет пользователю изменить набор переменных во вспомогательном объекте.

```
BOOL RDCALL rdsVSExecuteEditor(  
    RDS_HOBJECT Vars,      // Редактор переменных  
    BOOL Extended,        // Расширенный редактор  
    DWORD Flags,          // Флаги (RDS_HVAR_F*)
```

```

        int MaxDepth,           // Max вложенность матриц
        LPSTR Caption          // Заголовок окна редактора
    );

```

Тип указателя на эту функцию:

RDS_BHoBDwIS

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

Extended

TRUE – вызвать окно редактирования переменных блока, FALSE – окно редактирования структуры (можно вводить только имя, тип и значение поля).

Flags

Набор битовых флагов, управляющих колонками окна редактора и разрешенными типами переменных:

RDS_HVAR_F1INDEX	Вместе с матрицами разрешить массивы (только при установленном RDS_HVAR_FARRAYS).
RDS_HVAR_FARRAYS	Разрешить матрицы.
RDS_HVAR_FARROFSTRUCT	Разрешить матрицы и массивы структур.
RDS_HVAR_FCHAR	Разрешить тип char.
RDS_HVAR_FDOUBLE	Разрешить тип double.
RDS_HVAR_FFLOAT	Разрешить тип float.
RDS_HVAR_FINT	Разрешить тип int.
RDS_HVAR_FLOGICAL	Разрешить логический тип.
RDS_HVAR_FNOOFFSET	Не показывать колонку смещения к переменной.
RDS_HVAR_FNOSTRUCTNAME	Запретить ввод имени всей структуры (только при Extended==FALSE).
RDS_HVAR_FRUNTIME	Разрешить произвольный тип.
RDS_HVAR_FSHORT	Разрешить тип short int.
RDS_HVAR_FSIGNAL	Разрешить сигнальный тип.
RDS_HVAR_FSTRING	Разрешить строки.
RDS_HVAR_FSTRUCT	Разрешить структуры.

Для удобства в “RdsDef.h” описано четыре дополнительных константы, представляющих собой объединение приведенных выше флагов для часто встречающихся случаев. Эти константы можно использовать в параметре функции вместо флагов:

RDS_HVAR_FALL	все типы переменных;
RDS_HVAR_FALLNS	все типы, кроме сигналов;
RDS_HVAR_FALLPLAIN	все простые типы, то есть все, кроме структур, матриц/массивов, строк и произвольных типов;
RDS_HVAR_FALLPLAINNS	все простые типы, кроме сигналов.

Используемые в этом параметре флаги частично совпадают с флагами функции `rdsListVarTypes` (стр. 162).

MaxDepth

Максимальная глубина вложенности матриц (1 – только матрицы простых переменных, 2 – матрицы матриц, 3 – матрицы матриц матриц и т.п.) или –1 для максимальной вложенности (на данный момент РДС поддерживает вложенность до пяти).

Caption

Указатель на строку заголовка окна редактора переменных или NULL для стандартного заголовка “Структура переменных”.

Возвращаемое значение:

TRUE – пользователь закрыл окно редактора кнопкой “ОК”, FALSE – пользователь отменил редактирование.

Примечания:

Эта функция позволяет пользователю изменить структуру переменных в объекте Vars при помощи стандартного редактора переменных РДС. В зависимости от значения Extended вызывается либо редактор переменных блока с колонками “вход/выход”, “пуск” и т.п., либо более простой редактор структуры, в котором можно вводить только имя, тип и значение для каждого поля и имя самой структуры, если не установлен флаг RDS_HVAR_FNOSTRUCTNAME.

Примеры использования функции rdsVSExecuteEditor приведены в §2.16.1 и §4.3.

См. также:

rdsVSCreateEditor (стр. 430), rdsListVarTypes (стр. 162),
rdsVSApplyToBlock (стр. 436), rdsVSInstallStruct (стр. 442).

A.5.25.12. rdsVSFindAutoConn – найти имя связанной переменной

Функция rdsVSFindAutoConn возвращает имя связанной переменной (связанного сигнала или управляющей переменной) для указанного имени главной.

```
LPSTR RDSCALL rdsVSFindAutoConn
    RDS_HOBJECT Vars,          // Редактор переменных
    LPSTR MainVarName         // Имя главной переменной
);
```

Тип указателя на эту функцию:

RDS_SHoS

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

MainVarName

Указатель на строку с именем главой переменной, для которой нужно найти связанную.

Возвращаемое значение:

Указатель на строку с именем связанной переменной во внутренней памяти объекта, или NULL, если для указанной главной переменной нет связанной.

Примечания:

Эта функция возвращает указатель на имя связанной переменной для главной переменной MainVarName (связанные переменные добавляются функцией rdsVSAddAutoConn, см. стр. 430). Имя находится во внутренней памяти объекта Vars, этим указателем можно пользоваться только до тех пор, пока содержимое объекта не будет изменено.

Использование связей вспомогательных переменных с основными в моделях блоков описывается в §2.5.7 и §2.5.8.

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsVSAddAutoConn` (стр. 430),
`RDS_HVAR_GETAUTOCONN` (стр. 447), `RDS_HVAR_GETAUTOCOUNT` (стр. 447),
`RDS_HVAR_GETAUTOMAIN` (стр. 448).

A.5.25.13. `rdsVSGetVarDefValueStr` – получить строку значения переменной по умолчанию

Функция `rdsVSGetVarDefValueStr` возвращает динамически сформированную строку со значением по умолчанию переменной с указанным номером в объекте-редакторе.

```
LPSTR RDSCALL rdsVSGetVarDefValueStr
    RDS_НОбъект Vars,      // Редактор переменных
    int Index,             // Номер переменной
    int *pLength           // Возвращаемая длина строки
);
```

Тип указателя на эту функцию:

`RDS_SHoS`

Параметры:

`Vars`

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

`Index`

Начинающийся с нуля номер переменной.

`pLength`

Указатель на целую переменную, в которую функция должна записать длину получившейся строки. Если вызывающей программе не нужна длина строки, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Указатель на созданную в динамической памяти строку со значением по умолчанию указанной переменной, или `NULL`, если такой переменной нет.

Примечания:

Динамическая строка, созданная функцией `rdsVSGetVarDefValueStr`, должна быть *обязательно* освобождена функцией `rdsFree` (стр. 187).

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsGetBlockVarDefValueStr` (стр. 330).

A.5.25.14. `rdsVSGetVarDescription` – получить описание переменной

Функция `rdsVSGetVarDescription` заполняет структуру `RDS_VARDESCRIPTION` (стр. 138) описанием указанной переменной объекта-редактора или описанием всего набора переменных этого объекта.

```
BOOL RDSCALL rdsVSGetVarDescription(
    RDS_НОбъект Vars,      // Редактор переменных
    int Index,             // Номер переменной или -1
);
```

```

        RDS_PVARDESCRIPTION pDescr // Заполняемая структура
    );

```

Тип указателя на эту функцию:

```
RDS_BHoIVd
```

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

Index

Начинающийся с нуля номер переменной или `-1`, если нужно заполнить `RDS_VARDESCRIPTION` описанием всего набора переменных объекта Vars как одной большой структуры.

pDescr

Указатель на структуру описания переменной `RDS_VARDESCRIPTION`, которую функция должна заполнить параметрами переменной с указанным номером.

Возвращаемое значение:

`TRUE` – структура описания заполнена, `FALSE` – ошибка (нет такой переменной или поле `servSize` структуры `RDS_VARDESCRIPTION` неверно инициализировано).

Примечания:

Эта функция позволяет получить описание одной из внутренних переменных объекта Vars или всего объекта как одной переменной структурного типа.

Пример использования функции `rdsVSGetVarDescription` приведен в §2.16.1.

См. также:

`rdsVSCreateEditor` (стр. 430), `RDS_VARDESCRIPTION` (стр. 138).

A.5.25.15. `rdsVSInstallStruct` – добавить структуру в общий список структур

Функция `rdsVSInstallStruct` добавляет набор переменных объекта-редактора в общий список структур РДС.

```

    BOOL RDSCALL rdsVSInstallStruct(
        RDS_HOBJECT Vars,      // Редактор переменных
        int *pRresult          // Результат операции
    );

```

Тип указателя на эту функцию:

```
RDS_BHopI
```

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

pRresult

Указатель на целую переменную, в которую функция запишет код ошибки. Если код ошибки не нужен вызывающей программе, в этом параметре можно передать `NULL`. Код ошибки может принимать следующие значения:

`RDS_HVAR_ROK` Структура успешно добавлена.

RDS_HVAR_ROKRENAMED	Структура добавлена, но ее имя изменено на другое.
RDS_HVAR_RNOTYPENAME	В объекте Vars не указано имя структуры (см. команду RDS_HVAR_SETTYPENAME, стр. 452).
RDS_HVAR_EMPTYVARSET	Набор переменных в объекте Vars пуст или Vars – не объект-редактор переменных.

Возвращаемое значение:

TRUE – структура добавлена, FALSE – ошибка. Конкретный код ошибки возвращается через pResult.

Примечания:

Эта функция позволяет зарегистрировать в общем списке структур РДС новую структуру, описание которой находится в объекте Vars. Если структура с таким именем уже есть в списке, данная структура будет добавлена с другим, автоматически выбранным именем. Имя структуры в объекте Vars в этом случае тоже изменится, новое имя можно будет считать командой RDS_HVAR_GETTYPENAME (стр. 449).

См. также:

rdsVSCreateEditor (стр. 430), RDS_HVAR_SETTYPENAME (стр. 452),
RDS_HVAR_GETTYPENAME (стр. 449).

A.5.25.16. rdsVSSetVarFlags – установить флаги переменной

Функция rdsVSSetVarFlags устанавливает битовые флаги переменной с указанным номером в объекте-редакторе.

```
void RDSCALL rdsVSSetVarFlags (
    RDS_HOBJECT Vars,      // Редактор переменных
    int Index,             // Номер переменной
    DWORD Flags,           // Флаги
    DWORD Mask             // Маска
);
```

Тип указателя на эту функцию:

RDS_VHoidDwDw

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

Index

Начинающийся с нуля номер переменной в объекте Vars.

Flags

Набор битовых флагов переменной RDS_VARFLAG_* (см. поле Flags структуры RDS_VARDDESCRIPTION, стр. 138).

Mask

Маска изменяемых битовых флагов (единичные биты в позиции тех флагов, которые нужно изменить в переменной Index согласно Flags).

Примечания:

Эта функция устанавливает и сбрасывает битовые флаги, определяющие поведение переменной с номером Index в объекте Vars. В параметре Flags передается целое число,

у которого в позициях, соответствующих взводимым флагам, будут единичные биты, а в позициях, соответствующих сбрасываемым – нулевые. При этом в параметре Mask должно быть передано целое число, у которого единичные биты соответствуют изменяемым (взводимым или сбрасываемым) флагам, а нулевые – флагам, остающимся неизменными.

Пример использования функции rdsVSSetVarFlags приведен в §2.16.1.

См. также:

RDS_VARDESCRIPTION (стр. 138), rdsSetBlockVarFlags (стр. 335),
RDS_HVAR_GETVARFLAGS (стр. 450).

A.5.25.17. rdsVSUsesStructType – используется ли структура в объекте

Функция rdsVSUsesStructType проверяет, используется ли структура с указанным именем типа где-нибудь в указанном объекте.

```
BOOL RDSCALL rdsVSUsesStructType  
    RDS_НОВЕКТ Vars,      // Редактор переменных  
    LPSTR StructName     // Имя типа структуры  
);
```

Тип указателя на эту функцию:

RDS_BHoS

Параметры:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

StructName

Указатель на строку с именем типа структуры.

Возвращаемое значение:

TRUE – структура с именем StructName используется в наборе переменных, FALSE – структура не используется.

Примечания:

Если структура с именем типа StructName является типом переменной в объекте Vars, типом поля какой-либо другой структуры в нем, или типом элемента какого-либо массива или матрицы в этом объекте, функция rdsVSUsesStructType вернет TRUE. Чаще всего эта функция используется в модулях автокомпиляции для поиска структур РДС, описания которых нужны для работы блока.

См. также:

rdsVSCreateEditor (стр. 430), rdsVarUsesStructType (стр. 336).

A.5.25.18. Команда RDS_HVAR_CLEARATYPEPEREN – очистка списка переименований структур

Команда RDS_HVAR_CLEARATYPEPEREN очищает в указанном объекте-редакторе список переименований структур, элементы которого были добавлены функцией rdsVSAddTypeRename (стр. 431). Команда RDS_HVAR_CLEARENAMES является синонимом этой команды (обе этих константы имеют одно значение).

Вызов команды:

```
rdsCommandObject (Vars, RDS_HVAR_CLEARVAREN) ;  
или  
rdsCommandObjectEx (Vars, RDS_HVAR_CLEARVAREN, 0, NULL) ;
```

Параметр:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsVSAddTypeRename` (стр. 431),
`RDS_HVAR_RESET` (стр. 451), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400).

А.5.25.19. Команда RDS_HVAR_CLEARVARREN – очистка списка переименований переменных

Команда `RDS_HVAR_CLEARVARREN` очищает в указанном объекте-редакторе список переименований переменных, элементы которого были добавлены функцией `rdsVSAddVarRename` (стр. 435).

Вызов команды:

```
rdsCommandObject (Vars, RDS_HVAR_CLEARVARREN) ;  
или  
rdsCommandObjectEx (Vars, RDS_HVAR_CLEARVARREN, 0, NULL) ;
```

Параметр:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsVSAddVarRename` (стр. 435),
`RDS_HVAR_RESET` (стр. 451), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400).

А.5.25.20. Команда RDS_HVAR_DELAUTO – удалить связь с управляющей переменной

Команда `RDS_HVAR_DELAUTO` удаляет из указанного объекта-редактора связь между основной и управляющей переменными, созданную функцией `rdsVSAddAutoConn` (стр. 430).

Вызов команды:

```
int iNum=... // Номер удаляемой связи  
BOOL bOk=rdsCommandObjectEx (Vars, RDS_HVAR_DELAUTO, iNum, NULL) ;
```

Параметры и результат:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

iNum

Целое число (int) – номер удаляемой связи переменных.

boK

Логический (BOOL) результат выполнения команды: TRUE – связь удалена, FALSE – в объекте Vars нет связи с номером iNum.

Примечания:

Связь между основной и вспомогательной переменными удаляется по номеру, номер связи возвращается функцией rdsVSAddAutoConn при создании этой связи. После удаления все связи с номерами, большими iNum, сдвигаются вверх на единицу: связь с номером iNum+1 становится связью с номером iNum, связь с номером iNum+2 становится связью с номером iNum+1 и т.д. Общее число связей можно узнать при помощи команды RDS_HVAR_GETAUTOCOUNT (стр. 447).

Использование связей вспомогательных переменных с основными в моделях блоков описывается в §2.5.7 и §2.5.8.

См. также:

rdsVSCreateEditor (стр. 430), rdsVSAddAutoConn (стр. 430),
RDS_HVAR_GETAUTOCONN (стр. 447), RDS_HVAR_GETAUTOCOUNT (стр. 447),
RDS_HVAR_GETAUTOMAIN (стр. 448), RDS_HVAR_RESET (стр. 451).

A.5.25.21. Команда RDS_HVAR_DELVAR – удалить переменную

Команда RDS_HVAR_DELVAR удаляет из указанного объекта-редактора переменную с заданным номером.

Вызов команды:

```
int iNum=... // Номер удаляемой переменной
BOOL boK=rdsCommandObjectEx(Vars,RDS_HVAR_DELVAR,iNum,NULL);
```

Параметры и результат:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

iNum

Целое число (int) – номер удаляемой переменной.

boK

Логический (BOOL) результат выполнения команды: TRUE – переменная удалена, FALSE – в объекте Vars нет переменной с номером iNum.

Примечания:

Переменная из объекта удаляется по номеру, номер переменной указывается при ее создании функциями rdsVSAddVar (стр. 432), rdsVSAddVarByDescr (стр. 433) и rdsVSAddVarByTypeText (стр. 434). При создании сразу всего набора переменных функциями rdsVSCreateByDescr (стр. 437) и rdsVSCreateFromBlock (стр. 438) переменные получают последовательные номера.

После удаления все переменные с номерами, большими iNum, сдвигаются вверх на единицу: переменная с номером iNum+1 становится получает номер iNum, переменная с номером iNum+2 получает номер iNum+1 и т.д.

Общее число переменных можно узнать при помощи команды RDS_HVAR_GETFIELDCOUNT (стр. 449).

См. также:

`rdsVSCreateEditor` (стр. 430), `RDS_HVAR_GETFIELDCOUNT` (стр. 449).

А.5.25.22. Команда `RDS_HVAR_GETAUTOCONN` – получить имя связанной переменной по номеру связи

Команда `RDS_HVAR_GETAUTOCONN` возвращает имя вспомогательной переменной из связи основных и вспомогательных переменных с заданным номером.

Вызов команды:

```
int iNum=... // Номер связи "основная-вспомогательная"  
char *strName=rdsGetObjectStr (Vars, RDS_HVAR_GETAUTOCONN, iNum) ;
```

Параметры и результат:

`Vars`

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

`iNum`

Целое число (`int`) – номер связи переменных.

`strName`

Указатель на строку (`char*`) во внутренней памяти РДС, в которой записано имя вспомогательной переменной, или `NULL`, если связи с номером `iNum` нет в объекте `Vars`.

Примечания:

Эта команда возвращает указатель на строку с именем вспомогательной переменной, задействованной в связи с номером `iNum`. Строка находится во внутренней памяти объекта `Vars`, этим указателем можно пользоваться только до тех пор, пока содержимое объекта не будет изменено. Связи основных и вспомогательных переменных создаются функцией `rdsVSAddAutoConn` (стр. 430), общее число таких связей можно узнать при помощи команды `RDS_HVAR_GETAUTOCOUNT` (стр. 447).

Использование связей вспомогательных переменных с основными в моделях блоков описывается в §2.5.7 и §2.5.8.

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsGetObjectStr` (стр. 405),
`rdsVSAddAutoConn` (стр. 430), `rdsVSFindAutoConn` (стр. 440),
`RDS_HVAR_GETAUTOMAIN` (стр. 448), `RDS_HVAR_GETAUTOCOUNT` (стр. 447).

А.5.25.23. Команда `RDS_HVAR_GETAUTOCOUNT` – получить число связей основных и вспомогательных переменных

Команда `RDS_HVAR_GETAUTOCOUNT` возвращает общее число связей основных и вспомогательных переменных в указанном объекте-редакторе.

Вызов команды:

```
int iNum=rdsGetObjectInt (Vars, RDS_HVAR_GETAUTOCOUNT, 0) ;
```

Параметры и результат:

`Vars`

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

iNum

Целое число (int) – общее число связей основных переменных со вспомогательными, хранящихся в объекте Vars.

Примечания:

Связи между основными и вспомогательными переменными создаются функцией rdsVSAddAutoConn (стр. 430). Использование этих связей в моделях блоков описывается в §2.5.7 и §2.5.8.

См. также:

rdsVSCreateEditor (стр. 430), rdsGetObjectInt (стр. 404),
rdsVSAddAutoConn (стр. 430), rdsVSFindAutoConn (стр. 440).

A.5.25.24. Команда RDS_HVAR_GETAUTOMAIN – получить имя основной переменной по номеру связи

Команда RDS_HVAR_GETAUTOMAIN возвращает имя основной переменной по номеру ее связи со вспомогательной.

Вызов команды:

```
int iNum=... // Номер связи "основная-вспомогательная"  
char *strName=rdsGetObjectStr (Vars,RDS_HVAR_GETAUTOMAIN,iNum);
```

Параметры и результат:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

iNum

Целое число (int) – номер связи переменных.

strName

Указатель на строку (char*) во внутренней памяти РДС, в которой записано имя основной переменной, или NULL, если связи с номером iNum нет в объекте Vars.

Примечания:

Эта команда возвращает указатель на строку с именем основной переменной, задействованной в связи с номером iNum. Строка находится во внутренней памяти объекта Vars, этим указателем можно пользоваться только до тех пор, пока содержимое объекта не будет изменено. Связи основных и вспомогательных переменных создаются функцией rdsVSAddAutoConn (стр. 430), общее число таких связей можно узнать при помощи команды RDS_HVAR_GETAUTOCOUNT (стр. 447).

Использование связей вспомогательных переменных с основными в моделях блоков описывается в §2.5.7 и §2.5.8.

См. также:

rdsVSCreateEditor (стр. 430), rdsGetObjectStr (стр. 405),
rdsVSAddAutoConn (стр. 430), rdsVSFindAutoConn (стр. 440),
RDS_HVAR_GETAUTOCOUNT (стр. 447), RDS_HVAR_GETAUTOCOUNT (стр. 447).

А.5.25.25. Команда RDS_HVAR_GETFIELDCOUNT – получить число переменных в объекте

Команда RDS_HVAR_GETFIELDCOUNT возвращает общее число переменных в указанном объекте-редакторе.

Вызов команды:

```
int iNum=rdsGetObjectInt (Vars,RDS_HVAR_GETFIELDCOUNT,0);
```

Параметры и результат:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

iNum

Целое число (int) – общее число переменных, хранящихся в объекте Vars.

См. также:

rdsVSCreateEditor (стр. 430), rdsGetObjectInt (стр. 404),
rdsVSAddVar (стр. 432), RDS_HVAR_DELVAR (стр. 446),
rdsVSGetVarDescription (стр. 441).

А.5.25.26. Команда RDS_HVAR_GETTYPENAME – получить имя типа всей структуры переменных

Команда RDS_HVAR_GETTYPENAME возвращает имя типа, которое присвоено всей структуре переменных объекта командой RDS_HVAR_SETTYPENAME (стр. 452).

Вызов команды:

```
char *strName=rdsGetObjectStr (Vars,RDS_HVAR_GETTYPENAME,0);
```

Параметры и результат:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

strName

Указатель на строку (char*) во внутренней памяти РДС, в которой записано имя типа всей структуры переменных объекта, или NULL, если структура переменных в объекте Vars пуста.

Примечания:

Имя типа структуры используется только при регистрации структуры переменных объекта Vars в общем списке структур РДС вызовом rdsVSInstallStruct (стр. 442) – именно под этим именем структура регистрируется в списке. Строка, указатель на которую возвращает команда RDS_HVAR_GETTYPENAME, находится во внутренней памяти объекта, этим указателем можно пользоваться только до тех пор, пока содержимое объекта не будет изменено.

См. также:

rdsVSCreateEditor (стр. 430), rdsGetObjectStr (стр. 405),
RDS_HVAR_SETTYPENAME (стр. 452), rdsVSInstallStruct (стр. 442).

А.5.25.27. Команда RDS_HVAR_GETTYPESTRING – получить строку типа всей структуры переменных

Команда RDS_HVAR_GETTYPESTRING возвращает строку типа всей структуры переменных объекта-редактора. Строка типа состоит из последовательности символов, каждый из которых соответствует типу переменной (см. стр. 49, §1.5).

Вызов команды:

```
char *strType=rdsGetObjectStr (Vars, RDS_HVAR_GETTYPESTRING, 0) ;
```

Параметры и результат:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

strType

Указатель на строку (char*) во внутренней памяти РДС, в которой находится сформированная командой строка типа структуры переменных, или NULL, если структура переменных в объекте Vars пуста.

Примечания:

Строка типа каждый раз формируется заново при выполнении команды RDS_HVAR_GETTYPESTRING. Она хранится во внутренней памяти объекта Vars, указателем на нее, возвращенным командой, можно пользоваться до следующего вызова этой же команды для этого объекта. При добавлении и удалении переменных тип всей структуры меняется, но строка типа, хранящаяся в объекте Vars, будет обновлена только при очередном вызове команды RDS_HVAR_GETTYPESTRING.

Пример использования этой команды приведен в §4.4.

См. также:

rdsVSCreateEditor (стр. 430), типы переменных (стр. 49),
rdsGetObjectStr (стр. 405).

А.5.25.28. Команда RDS_HVAR_GETVARFLAGS – получить флаги переменной

Команда RDS_HVAR_GETVARFLAGS возвращает флаги переменной с указанным номером в указанном объекте-редакторе.

Вызов команды:

```
int iNum=... // Номер переменной  
int iFlags=rdsGetObjectInt (Vars, RDS_HVAR_GETVARFLAGS, iNum) ;
```

Параметры и результат:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

iNum

Целое число (int) – номер переменной в объекте Vars.

iFlags

Целое число (int) – набор битовых флагов RDS_VARFLAG_* переменной с номером iNum в объекте Vars.

Примечания:

Эта команда возвращает те же самые флаги переменной, которые обычно записываются в поле `Flags` структуры `RDS_VARDESCRIPTION` (стр. 138). Эту структуру можно заполнить функцией `rdsVSGetVarDescription` (стр. 441), команда `RDS_HVAR_GETVARFLAGS` всего лишь предоставляет более простой доступ к флагам переменной.

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsGetObjectInt` (стр. 404),
`rdsVSGetVarDescription` (стр. 441), `RDS_VARDESCRIPTION` (стр. 138),
`rdsVSSetVarFlags` (стр. 443).

А.5.25.29. Команда `RDS_HVAR_GETVARRANK` – получить уровень всей структуры переменных

Команда `RDS_HVAR_GETVARRANK` возвращает уровень всей структуры переменных указанного объекта-редактора.

Вызов команды:

```
int iRank=rdsGetObjectInt (Vars, RDS_HVAR_GETVARRANK, 0);
```

Параметры и результат:

`Vars`

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

`iRank`

Целое число (`int`) – уровень структуры переменных объекта `Vars`, то есть максимальная вложенность элементов в этой структуре.

Примечания:

Эта команда возвращает то же самое значение, которое записывается в поле `Rank` структуры `RDS_VARDESCRIPTION` (стр. 138) при вызове функции `rdsVSGetVarDescription` (стр. 441) для всего набора переменных (то есть со значением `-1` вместо номера переменной). Уровень любой переменной в РДС – это максимальная вложенность ее элементов. Например, для любой простой переменной уровень будет равен нулю, для массива или матрицы вещественных чисел – единице (один вложенный элемент), для матрицы матриц целых – двум (внутри этой матрицы – еще одна матрица, внутри которой – целое число, то есть в переменной два элемента, вложенных один в другой). Для структуры переменных объекта-редактора уровень будет числом, на единицу большим максимального уровня всех переменных объекта, то есть если, например, структура будет состоять только из простых переменных, ее уровень будет равен единице.

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsGetObjectInt` (стр. 404),
`rdsVSGetVarDescription` (стр. 441), `RDS_VARDESCRIPTION` (стр. 138).

А.5.25.30. Команда `RDS_HVAR_RESET` – очистка объекта

Команда `RDS_HVAR_RESET` удаляет из указанного объекта-редактора переменных все переменные, связи и переименования.

Вызов команды:

```
rdsCommandObject (Vars, RDS_HVAR_RESET) ;  
или  
rdsCommandObjectEx (Vars, RDS_HVAR_RESET, 0, NULL) ;
```

Параметр:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400), `RDS_HVAR_CLEARATYPEPEREN` (стр. 444),
`RDS_HVAR_CLEARVARREN` (стр. 445).

А.5.25.31. Команда RDS_HVAR_SETTYPENAME – установить имя типа всей структуры переменных

Команда `RDS_HVAR_SETTYPENAME` присваивает всей структуре переменных объекта указанное имя типа.

Вызов команды:

```
char *strName=... // Имя типа структуры  
rdsSetObjectStr (Vars, RDS_HVAR_SETTYPENAME, 0, strName) ;
```

Параметры и результат:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

strName

Указатель на строку (`char*`), в которой записано имя типа всей структуры переменных объекта, или `NULL`, если имя типа нужно очистить.

Примечания:

Имя типа структуры используется только при регистрации структуры переменных объекта `Vars` в общем списке структур РДС вызовом `rdsVSInstallStruct` (стр. 442) – именно под этим именем структура регистрируется в списке. При создании структуры переменных блока функцией `rdsVSApplyToBlock` (стр. 436) имя типа структуры игнорируется.

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsSetObjectStr` (стр. 407),
`RDS_HVAR_GETTYPENAME` (стр. 449), `rdsVSInstallStruct` (стр. 442).

А.5.25.32. Команда RDS_HVAR_SETVARFLAGS – одновременно установить все флаги переменной

Команда `RDS_HVAR_SETVARFLAGS` устанавливает флаги переменной с указанным номером в указанном объекте-редакторе.

Вызов команды:

```
int iNum=...    // Номер переменной
int iFlags=...  // Набор флагов RDS_VARFLAG_*
rdsSetObjectInt (Vars, RDS_HVAR_SETVARFLAGS, iNum, iFlags);
```

Параметры и результат:

Vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

iNum

Целое число (int) – номер переменной в объекте Vars.

iFlags

Целое число (int) – набор битовых флагов `RDS_VARFLAG_*` (см. поле `Flags` структуры `RDS_VARESCRIPTION`, стр. 138) для переменной с номером `iNum` в объекте Vars.

Примечания:

Эта команда, как и функция `rdsVSSetVarFlags` (стр. 443), устанавливает битовые флаги переменной с номером `iNum`. В отличие от `rdsVSSetVarFlags`, команда не позволяет изменить только часть флагов. В параметрах команды не предусмотрена маска установки, поэтому переменная получает все флаги из параметра `iFlags`: единичные биты в параметре взведут соответствующие флаги, нулевые – сбросят. Фактически, вызов

```
rdsSetObjectInt (Vars, RDS_HVAR_SETVARFLAGS, iNum, iFlags);
```

полностью эквивалентен вызову функции

```
rdsVSSetVarFlags (Vars, iNum, iFlags, 0xFFFFFFFF);
```

См. также:

`rdsVSCreateEditor` (стр. 430), `rdsSetObjectInt` (стр. 407),
`rdsVSSetVarFlags` (стр. 443), `rdsVSAddVar` (стр. 432).

A.5.25.33. Макрос `rdsVSClearEditor` – очистка объекта

Макрос `rdsVSClearEditor` предназначен для полной очистки объекта-редактора переменных.

```
rdsVSClearEditor(
    vars    // Вспомогательный объект
)
```

Определение:

```
#define rdsVSClearEditor(vars) \
    rdsCommandObject((vars), RDS_HVAR_RESET)
```

Параметр:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

Примечания:

Этот макрос включает в себя вызов команды `RDS_HVAR_RESET` (стр. 451), его можно использовать для того, чтобы сделать текст программы более понятным.

См. также:

RDS_HVAR_RESET (стр. 451).

А.5.25.34. Макрос rdsVSClearTypeRenames – очистка списка переименований структур

Макрос rdsVSClearTypeRenames предназначен для очистки списка переименований типов структур.

```
rdsVSClearTypeRenames (
    vars                // Вспомогательный объект
)
```

Определение:

```
#define rdsVSClearTypeRenames(vars) \
    rdsCommandObject((vars), RDS_HVAR_CLEAR typeren)
```

Параметр:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

Примечания:

Этот макрос включает в себя вызов команды RDS_HVAR_CLEAR typeren (стр. 444).

См. также:

RDS_HVAR_CLEAR typeren (стр. 444).

А.5.25.35. Макрос rdsVSClearVarRenames – очистка списка переименований переменных

Макрос rdsVSClearVarRenames предназначен для очистки списка переименований переменных.

```
rdsVSClearVarRenames (
    vars                // Вспомогательный объект
)
```

Определение:

```
#define rdsVSClearVarRenames(vars) \
    rdsCommandObject((vars), RDS_HVAR_CLEAR varren)
```

Параметр:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

Примечания:

Этот макрос включает в себя вызов команды RDS_HVAR_CLEAR varren (стр. 445).

См. также:

RDS_HVAR_CLEAR varren (стр. 445).

A.5.25.36. Макрос rdsVSDeleteAutoConn – удалить связь между основной и управляющей переменными

Макрос rdsVSDeleteAutoConn предназначен для удаления внутренней связи основной переменной со вспомогательной.

```
rdsVSDeleteAutoConn(  
    vars,          // Вспомогательный объект  
    num            // Номер связи  
)
```

Определение:

```
#define rdsVSDeleteAutoConn(vars, num) \  
    rdsCommandObjectEx((vars), RDS_HVAR_DELAUTO, \  
        (num), NULL)
```

Параметры:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

num

Целое число (int) – номер удаляемой связи переменных.

Примечания:

Этот макрос включает в себя вызов команды RDS_HVAR_DELAUTO (стр. 445).

См. также:

RDS_HVAR_DELAUTO (стр. 445).

A.5.25.37. Макрос rdsVSDeleteVar – удалить переменную

Макрос rdsVSDeleteVar предназначен для удаления переменной с заданным номером из объекта-редактора.

```
rdsVSDeleteVar(  
    vars,          // Вспомогательный объект  
    num            // Номер переменной  
)
```

Определение:

```
#define rdsVSDeleteVar(vars, num) \  
    rdsCommandObjectEx((vars), RDS_HVAR_DELVAR, \  
        (num), NULL)
```

Параметры:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

num

Целое число (int) – номер удаляемой переменной.

Примечания:

Этот макрос включает в себя вызов команды RDS_HVAR_DELVAR (стр. 446).

См. также:

RDS_HVAR_DELVAR (стр. 446).

А.5.25.38. Макрос rdsVSGetAutoConn – получить имя связанной переменной по номеру связи

Макрос rdsVSGetAutoConn возвращает имя вспомогательной переменной из связи основных и вспомогательных переменных с заданным номером.

```
rdsVSGetAutoConn(  
    vars,          // Вспомогательный объект  
    num            // Номер связи  
)
```

Определение:

```
#define rdsVSGetAutoConn(vars, num) \  
    rdsGetObjectStr((vars), RDS_HVAR_GETAUTOCONN, (num))
```

Параметры:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

num

Целое число (int) – номер связи переменных.

Возвращаемое значение:

Указатель на строку (char*) во внутренней памяти РДС, в которой записано имя вспомогательной переменной, или NULL, если связи с номером num нет в объекте vars.

Примечания:

Этот макрос включает в себя вызов команды RDS_HVAR_GETAUTOCONN (стр. 447).

См. также:

RDS_HVAR_GETAUTOCONN (стр. 447).

А.5.25.39. Макрос rdsVSGetAutoCount – получить число связей основных и вспомогательных переменных

Макрос rdsVSGetAutoCount возвращает общее число связей основных и вспомогательных переменных в указанном объекте-редакторе.

```
rdsVSGetAutoCount(  
    vars          // Вспомогательный объект  
)
```

Определение:

```
#define rdsVSGetAutoCount(vars) \  
    rdsGetObjectInt((vars), RDS_HVAR_GETAUTOCOUNT, 0)
```

Параметр:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией rdsVSCreateEditor (стр. 430).

Возвращаемое значение:

Общее число связей основных переменных со вспомогательными, хранящихся в объекте `vars`.

Примечания:

Этот макрос включает в себя вызов команды `RDS_HVAR_GETAUTOCOUNT` (стр. 447).

См. также:

`RDS_HVAR_GETAUTOCOUNT` (стр. 447).

А.5.25.40. Макрос `rdsVSGetAutoMain` – получить имя основной переменной по номеру связи

Макрос `rdsVSGetAutoMain` возвращает имя основной переменной по номеру ее связи со вспомогательной.

```
rdsVSGetAutoMain(  
    vars,          // Вспомогательный объект  
    num            // Номер связи  
)
```

Определение:

```
#define rdsVSGetAutoMain(vars, num) \  
    rdsGetObjectStr((vars), RDS_HVAR_GETAUTOMAIN, (num))
```

Параметры:

`vars`

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

`num`

Целое число (`int`) – номер связи переменных.

Возвращаемое значение:

Указатель на строку (`char*`) во внутренней памяти РДС, в которой записано имя основной переменной, или `NULL`, если связи с номером `num` нет в объекте `vars`.

Примечания:

Этот макрос включает в себя вызов команды `RDS_HVAR_GETAUTOMAIN` (стр. 448).

См. также:

`RDS_HVAR_GETAUTOMAIN` (стр. 448).

А.5.25.41. Макрос `rdsVSGetFieldCount` – получить число переменных в объекте

Макрос `rdsVSGetFieldCount` возвращает общее число переменных в указанном объекте-редакторе.

```
rdsVSGetFieldCount(  
    vars          // Вспомогательный объект  
)
```

Определение:

```
#define rdsVSGetFieldCount(vars) \
    rdsGetObjectInt((vars), RDS_HVAR_GETFIELDCOUNT, 0)
```

Параметр:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

Возвращаемое значение:

Общее число переменных, хранящихся в объекте `vars`.

Примечания:

Этот макрос включает в себя вызов команды `RDS_HVAR_GETFIELDCOUNT` (стр. 449).

См. также:

`RDS_HVAR_GETFIELDCOUNT` (стр. 449).

A.5.25.42. Макрос `rdsVSGetStructName` – получить имя типа всей структуры переменных

Макрос `rdsVSGetStructName` возвращает имя типа, которое присвоено всей структуре переменных объекта.

```
rdsVSGetStructName(
    vars          // Вспомогательный объект
)
```

Определение:

```
#define rdsVSGetStructName(vars) \
    rdsGetObjectStr((vars), RDS_HVAR_GETTYPENAME, 0)
```

Параметр:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

Возвращаемое значение:

Указатель на строку (`char*`) во внутренней памяти РДС, в которой записано имя типа всей структуры переменных объекта, или `NULL`, если структура переменных в объекте `vars` пуста.

Примечания:

Этот макрос включает в себя вызов команды `RDS_HVAR_GETTYPENAME` (стр. 449).

См. также:

`RDS_HVAR_GETTYPENAME` (стр. 449).

A.5.25.43. Макрос `rdsVSGetStructRank` – получить уровень всей структуры переменных

Макрос `rdsVSGetStructRank` возвращает уровень всей структуры переменных указанного объекта-редактора.

```

rdsVSGetStructRank(
    vars                // Вспомогательный объект
)

```

Определение:

```

#define rdsVSGetStructRank(vars) \
    rdsGetObjectInt((vars), RDS_HVAR_GETVARRANK, 0)

```

Параметр:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

Возвращаемое значение:

Целое число – уровень структуры переменных объекта `vars`, то есть максимальная вложенность элементов в этой структуре.

Примечания:

Этот макрос включает в себя вызов команды `RDS_HVAR_GETVARRANK` (стр. 451).

См. также:

`RDS_HVAR_GETVARRANK` (стр. 451).

A.5.25.44. Макрос `rdsVSSetStructName` – установить имя типа всей структуры переменных

Макрос `rdsVSSetStructName` присваивает всей структуре переменных объекта указанное имя типа.

```

rdsVSSetStructName(
    vars,                // Вспомогательный объект
    name                 // Имя типа структуры
)

```

Определение:

```

#define rdsVSSetStructName(vars, name) \
    rdsSetObjectStr((vars), RDS_HVAR_SETTYPENAME, \
        0, (name))

```

Параметры:

vars

Идентификатор вспомогательного объекта-редактора переменных, ранее созданного функцией `rdsVSCreateEditor` (стр. 430).

name

Указатель на строку (`char*`), в которой записано имя типа всей структуры переменных объекта, или `NULL`, если имя типа нужно очистить.

Примечания:

Этот макрос включает в себя вызов команды `RDS_HVAR_SETTYPENAME` (стр. 452).

См. также:

`RDS_HVAR_SETTYPENAME` (стр. 452).

А.5.26. Вспомогательный объект для разбора текста

Описываются функции и команды вспомогательного объекта РДС, предназначенного для выделения из произвольного текста отдельных слов и сравнения их с ключевыми.

А.5.26.1. rdsSTRCreateTextReader – создать объект для разбора текста

Функция `rdsSTRCreateTextReader` создает вспомогательный объект РДС, с помощью которого можно разбирать текст, просматривая его слово за словом и находя ключевые слова.

```
RDS_НОВЕКТ RDSCALL rdsSTRCreateTextReader(  
    BOOL IgnoreCase           // Без учета регистра  
);
```

Тип указателя на эту функцию:

`RDS_НОВ`

Параметр:

`IgnoreCase`

`TRUE` – сравнивать слова текста с ключевыми без учета регистра символов, `FALSE` – с учетом.

Возвращаемое значение:

Уникальный идентификатор созданного объекта.

Примечания:

Эта функция создает вспомогательный объект для разбора текста и возвращает его идентификатор типа `RDS_НОВЕКТ`. С помощью различных сервисных функций РДС (как общих для всех типов вспомогательных объектов, так и специализированных для данного) в этот объект можно записать набор ключевых слов с целыми идентификаторами, а затем, передав в него указатель на начало разбираемого текста, получать из объекта слово за словом. При этом объект будет сравнивать извлекаемые из текста слова с ключевыми и возвращать идентификаторы опознанных ключевых слов.

Если в параметре `IgnoreCase` передано значение `TRUE`, при сравнении слов текста с ключевыми объект не будет учитывать регистр символов – например, и слово “WORD”, и слово “Word” будут считаться совпадающими с ключевым словом “word”. Этот параметр можно изменить уже после создания объекта командой `RDS_HSTR_IGNORECASE` (стр. 468).

Созданный вспомогательный объект будет существовать до тех пор, пока схема не будет выгружена из памяти, или пока он не будет удален вызовом функции `rdsDeleteObject` (стр. 401).

Пример использования функции `rdsSTRCreateTextReader` приведен в §2.8.4.

См. также:

Вспомогательные объекты (стр. 399), `rdsDeleteObject` (стр. 401),
`RDS_HSTR_SETTEXT` (стр. 470), `rdsSTRAddKeyword` (стр. 460),
`rdsSTRAddKeywordsArray` (стр. 461), `RDS_HSTR_IGNORECASE` (стр. 468).

А.5.26.2. rdsSTRAddKeyword – добавление ключевого слова

Функция `rdsSTRAddKeyword` добавляет в объект разбора текста ключевое слово с целым идентификатором. В процессе разбора текста при совпадении очередного слова с этим ключевым словом объект будет возвращать его идентификатор.

```

int RDSCALL rdsSTRAddKeyword(
    RDS_HOBJECT Parser,    // Объект
    LPSTR Keyword,        // Ключевое слово
    int Id                // Идентификатор
);

```

Тип указателя на эту функцию:

RDS_IHoSI

Параметры:

Parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией rdsSTRCreateTextReader (стр. 460).

Keyword

Указатель на строку с ключевым словом.

Id

Целый идентификатор ключевого слова Keyword.

Возвращаемое значение:

Если ключевого слова Keyword еще не было в объекте Parser, функция вернет переданный ей идентификатор Id. Если это слово уже было добавлено в объект, функция вернет идентификатор, который этому слову присвоили при первом добавлении. Если же в параметре Keyword передано значение NULL или указатель на пустую строку, или если Parser – не объект разбора текста, функция вернет -1.

Примечания:

Эта функция добавляет во внутренний набор ключевых слов объекта Parser новое слово Keyword и присваивает ему идентификатор Id. Если при извлечении слов из произвольного текста функцией rdsSTRGetWord (стр. 462) слово совпадет с Keyword, объект вернет вызвавшей программе значение Id.

См. также:

rdsSTRCreateTextReader (стр. 460), rdsSTRAddKeywordsArray (стр. 461), rdsSTRGetWord (стр. 462).

A.5.26.3. rdsSTRAddKeywordsArray – добавление набора ключевых слов

Функция rdsSTRAddKeywordsArray добавляет в объект разбора текста сразу несколько ключевых слов и присваивает им последовательные целые идентификаторы. В процессе разбора текста при совпадении очередного слова с одним из этих ключевых слов объект будет возвращать идентификатор, соответствующий найденному слову.

```

BOOL RDSCALL rdsSTRAddKeywordsArray(
    RDS_HOBJECT Parser,    // Объект
    LPSTR *Keywords,      // Массив ключевых слов
    int WordCount,        // Число слов в массиве
    int StartId           // Начальный идентификатор
);

```

Тип указателя на эту функцию:

RDS_BHopSII

Параметры:

`Parser`

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией `rdsSTRCreateTextReader` (стр. 460).

`Keywords`

Указатель на первый элемент массива указателей на строки (`char*`) ключевых слов. Массив либо завершается элементом `NULL`, либо число его элементов должно быть указано в параметре `WordCount`.

`WordCount`

Число ключевых слов в массиве `Keywords` или `-1`, если массив завершается элементом со значением `NULL`.

`StartId`

Целый идентификатор, который будет присвоен самому первому ключевому слову в массиве `Keywords` (`Keywords[0]`). Каждое следующее слово будет иметь идентификатор, на единицу больший предыдущего.

Возвращаемое значение:

`TRUE` – ключевые слова добавлены в объект, `FALSE` – ошибка (`Parser` не является объектом разбора текста или один из идентификаторов совпал с уже используемым в объекте).

Примечания:

Эта функция добавляет во внутренний набор ключевых слов объекта `Parser` ключевые слова из массива `Keywords` и присваивает им последовательные идентификаторы начиная с `StartId`. Если при извлечении слов из произвольного текста функцией `rdsSTRGetWord` (стр. 462) слово совпадет с одним из элементов массива `Keywords`, объект вернет вызвавшей программе соответствующее этому слову значение идентификатора.

Массив ключевых слов описывается как массив указателей на строки, каждая из которых представляет собой ключевое слово. Например, в результате выполнения программы

```
char *words[]={ "alpha", "beta", "gamma", NULL};  
RDS_ОБЪЕКТ obj=rdsSTRCreateTextReader(TRUE);  
rdsSTRAddKeywordsArray(obj, words, -1, 100);
```

слово “alpha” получит идентификатор 100, слово “beta” – 101, слово “gamma” – 102.

Пример использования функции `rdsSTRAddKeywordsArray` приведен в §2.8.4.

См. также:

`rdsSTRCreateTextReader` (стр. 460), `rdsSTRAddKeyword` (стр. 460),
`rdsSTRGetWord` (стр. 462).

A.5.26.4. `rdsSTRGetWord` – считать из текста очередное слово

Функция `rdsSTRGetWord` считывает очередное слово из текста, переданного в объект командой `RDS_HSTR_SETTEXT` (стр. 470), и возвращает указатель на это слово, указатель на следующую за словом часть текста, а также идентификатор этого слова, если оно совпало с одним из ключевых.

```
int RDSCALL rdsSTRGetWord(  
    RDS_ОБЪЕКТ Parser,    // Объект
```

```

LPSTR *pWord,           // Возвращаемый указатель на слово
LPSTR *pNext,           // Возвращаемый остаток текста
char *pWordType,        // Возвращаемый тип слова
BOOL Analyse            // Сравнивать с ключевыми
);

```

Тип указателя на эту функцию:

RDS_IHopSpSpCB

Параметры:

Parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией `rdsSTRCreateTextReader` (стр. 460).

pWord

Указатель типа `char**` на переменную, в которую будет записан указатель на извлеченное из текста слово, находящееся во внутренней памяти объекта `Parser`. Если вызывающей программе не нужно это слово, в этом параметре можно передать `NULL`.

pNext

Указатель типа `char**` на переменную, в которую будет записан указатель на следующую за извлеченным словом часть текста. Допускается обработать часть этого текста вручную, после чего снова передать объекту продолжение текста командой `RDS_HSTR_SETTEXT` (см. пример ниже). Если вызывающей программе не нужен указатель на продолжение текста, в этом параметре можно передать `NULL`.

pWordType

Указатель на переменную типа `char`, в которую будет записан вид считанного слова (аналогично параметру `pSym` функции `rdsGetTextWord`, см. стр. 291). Если вызывающей программе не нужен вид слова, в этом параметре можно передать `NULL`. От вида слова зависит его предварительная обработка внутри функции, от него также может зависеть реакция на это слово вызвавшей функцию программы. По указателю `pWordType` может быть записан один из следующих символов:

<i>*pWordType (код)</i>	<i>Значение</i>
0	Считан конец текста, в нем больше нет слов. Функция при этом вернет специальный идентификатор конца текста (по умолчанию – константу <code>RDS_HSTR_DEFENDOFTTEXT</code>).
10 (“\n”)	Считан конец строки, следующее слово находится на следующей строке. Извлеченное функцией слово (<code>*pWord</code>) при этом будет состоять из единственного символа “\n”, а результатом возврата функции будет специальный идентификатор конца строки (по умолчанию – <code>RDS_HSTR_DEFENDOFFLINE</code>).
Двойная кавычка (“ ”)	Считанное слово на самом деле является строкой в двойных кавычках (возможно, с пробелами), в которой непечатаемые символы заменены на их символические обозначения (см. стр. 193). Через параметр <code>pWord</code> при этом функция вернет строку без кавычек, в которой символические обозначения заменены на сами символы, в т.ч. и непечатаемые.

<i>*pWordType (код)</i>	<i>Значение</i>
Другие коды	Считано слово, состоящее из слитно набранных печатаемых символов. В *pWordType будет записан первый символ этого слова, а через параметр pWord функция вернет само слово.

Analyse

TRUE – сравнивать слова, извлекаемые из текста, с ключевыми словами, записанными в объект функциями rdsSTRAddKeyword (стр. 460) и rdsSTRAddKeywordsArray (стр. 461). FALSE – не сравнивать слова с ключевыми.

Возвращаемое значение:

Если в параметре Analyse передано TRUE и извлеченное из текста слово совпало с одним из ключевых слов объекта, функция вернет идентификатор этого ключевого слова. В противном случае результат функции будет следующим:

- Если достигнут конец текста (завершающий нулевой байт), функция вернет специальный идентификатор конца текста. По умолчанию это константа RDS_HSTR_DEFENDOFTEXT, но этот идентификатор может быть, при необходимости, изменен командой RDS_HSTR_ENDOFTEXTID (стр. 466).
- Если достигнут конец строки (символ перевода строки, байт с кодом 10), функция вернет специальный идентификатор конца строки. По умолчанию это константа RDS_HSTR_DEFENDOFLINE, но этот идентификатор может быть, при необходимости, изменен командой RDS_HSTR_ENDOFLINEID (стр. 466).
- Если сравнение с ключевыми словами не разрешено (Analyse==FALSE) или извлеченное слово не совпало ни с одним из ключевых, функция вернет специальный идентификатор неизвестного слова. По умолчанию это константа RDS_HSTR_DEFUNKNOWNWORD, но этот идентификатор может быть, при необходимости, изменен командой RDS_HSTR_UNKNOWNID (стр. 471).

Примечания:

Эта функция во многом похожа на функцию rdsGetTextWord (стр. 291), поэтому их параметры частично совпадают. Однако, в отличие от rdsGetTextWord, функция rdsSTRGetWord не только извлекает из текста слово, ограниченное пробелами или заключенное в кавычки, но и сравнивает его (если это разрешено параметром Analyse) с набором ключевых слов объекта. Это упрощает анализ извлеченного слова в вызвавшей программе.

Словом текста, как и в функции rdsGetTextWord, считается либо последовательность любых печатаемых символов, либо строка в двойных кавычках. Разделителями слов считаются пробелы и табуляции. Код перевода строки “\n” считается отдельным словом из одного символа. Нулевой байт, завершающий текст, тоже считается отдельным словом (пустой строкой).

Функция отдельно обрабатывает знак продолжения “+”, используемый в текстовом формате схем РДС. Обнаружив его, она пропускает следующий за ним перевод строки и повторение знака продолжения, считая следующую строку текста продолжением текущей строки и не возвращая отдельно слово конца строки “\n”. Таким образом, например, текст

```
word1 word2 +
+ word3 "word 4"
```

будет считаться одной строкой, состоящей из четырех слов: “word1”, “word2”, “word3” и “word 4”.

Строка в кавычках считается одним словом, какой бы длинной она ни была и сколько бы пробелов внутри ни содержала. В приведенном выше примере “word 4” будет считано функцией как одно слово, а не как два. Длинные строки тоже могут быть разбиты с помощью символов продолжения “+”, при этом функция автоматически соберет их вместе. Например, текст

```
word5 "abcd"+
+"efgh" word6
```

будет разобран на три слова: “word5”, “abcdefgh” и “word6”.

Работу функции `rdsSTRGetWord` можно комбинировать с общими функциями разбора текста (см. А.5.11), для этого можно использовать параметр `pNext`, через который возвращается указатель на следующий за извлеченным словом участок текста. Рассмотрим, например, следующий фрагмент программы:

```
// Создание объекта
RDS_НОВЫЙ_ОБЪЕКТ obj=rdsSTRCreateTextReader(TRUE);
// Занесение ключевых слов
rdsSTRAddKeyword(obj,"color",100);
rdsSTRAddKeyword(obj,"line",101);

...
char text[]="color rgb 0 0 0 line 1",*word,*next;
COLORREF col;
BOOL loop=TRUE;
// Установка разбираемого текста
rdsSetObjectStr(obj,RDS_HSTR_SETTEXT,0,text);
// Цикл обработки текста
while(loop)
    switch(rdsSTRGetWord(obj,NULL,&next,NULL,TRUE))
    { case RDS_HSTR_DEFENDOFTEXT: // Конец текста
        loop=FALSE; break;
      case 100: // Опознано слово "color"
        // Считываем и обрабатываем продолжение строки
        col=rdsReadColorText(next,&word);
        // Новый текст - со следующего слова
        rdsSetObjectStr(obj,RDS_HSTR_SETTEXT,0,word);
        break;
    }
    ...
}
```

В этом примере в объект `obj` добавлено два ключевых слова: “color” с идентификатором 100 и “line” с идентификатором 101. После передачи в объект указателя на начало разбираемого текста `text` командой `RDS_HSTR_SETTEXT`, в цикле `while(loop)` слова считываются из текста и разбираются в операторе `switch`. Если считано слово “color” (идентификатор 100), следующий за ним фрагмент текста передается в функцию `rdsReadColorText` (стр. 294), которая без участия объекта `obj` разберет фрагмент “rgb 0 0 0”, преобразует его в значение цвета `col` и запишет в переменную `word` указатель на следующий за этим фрагмент текста, т.е. на “line 1”. Этот указатель командой `RDS_HSTR_SETTEXT` будет передан в объект `obj` как новое начало текста, и разбор продолжится.

Другой пример использования функции `rdsSTRGetWord` приведен в §2.8.4.

См. также:

```
rdsSTRCreateTextReader (стр. 460), RDS_HSTR_SETTEXT (стр. 470),
RDS_HSTR_ENDOFTEXTID (стр. 466), RDS_HSTR_ENDOFLINEID (стр. 466),
```

RDS_HSTR_UNKNOWNID (стр. 471), rdsSTRAddKeyword (стр. 460),
rdsSTRAddKeywordsArray (стр. 461).

A.5.26.5. Команда RDS_HSTR_ENDOFLINEID – идентификатор конца строки

Команда RDS_HSTR_ENDOFLINEID позволяет получать и устанавливать специальный идентификатор, который функция rdsSTRGetWord (стр. 462) будет возвращать, считав из текста конец строки.

Вызов команды для установки:

```
int iId=... // Идентификатор  
rdsSetObjectInt (Parser, RDS_HSTR_ENDOFLINEID, 0, iId);
```

Вызов команды для чтения:

```
int iId=rdsGetObjectInt (Parser, RDS_HSTR_ENDOFLINEID, 0);
```

Параметры и результат:

Parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией rdsSTRCreateTextReader (стр. 460).

iId

Целый (int) идентификатор, используемый для сообщения о конце строки. По умолчанию – константа RDS_HSTR_DEFENDOFLINE со значением –2. Все значения по умолчанию специальных идентификаторов в объекте для разбора текста сделаны отрицательными, поэтому, если среди идентификаторов пользовательских ключевых слов нет отрицательных, значения по умолчанию можно не менять.

См. также:

rdsSTRCreateTextReader (стр. 460), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), rdsSTRGetWord (стр. 462).

A.5.26.6. Команда RDS_HSTR_ENDOFTEXTID – идентификатор конца текста

Команда RDS_HSTR_ENDOFTEXTID позволяет получать и устанавливать специальный идентификатор, который функция rdsSTRGetWord (стр. 462) будет возвращать, достигнув конца переданного объекту текста.

Вызов команды для установки:

```
int iId=... // Идентификатор  
rdsSetObjectInt (Parser, RDS_HSTR_ENDOFTEXTID, 0, iId);
```

Вызов команды для чтения:

```
int iId=rdsGetObjectInt (Parser, RDS_HSTR_ENDOFTEXTID, 0);
```

Параметры и результат:

Parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией rdsSTRCreateTextReader (стр. 460).

iId

Целый (int) идентификатор, используемый для сообщения о конце текста. По умолчанию – константа RDS_HSTR_DEFENDOFTTEXT со значением –3. Все значения по умолчанию специальных идентификаторов в объекте для разбора текста сделаны

отрицательными, поэтому, если среди идентификаторов пользовательских ключевых слов нет отрицательных, значения по умолчанию можно не менять.

См. также:

`rdsSTRCreateTextReader` (стр. 460), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404), `rdsSTRGetWord` (стр. 462).

A.5.26.7. Команда `RDS_HSTR_GETLASTWORD` – получить последнее считанное слово

Команда `RDS_HSTR_GETLASTWORD` возвращает последнее слово, считанное из текста функцией `rdsSTRGetWord` (стр. 462).

Вызов команды:

```
char *strWord=rdsGetObjectStr(Parser,RDS_HSTR_GETLASTWORD,0);
```

Параметры и результат:

`Parser`

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией `rdsSTRCreateTextReader` (стр. 460).

`strWord`

Указатель на строку (`char*`) во внутренней памяти объекта `Vars`, в которой записано слово, извлеченное из текста при последнем вызове функции `rdsSTRGetWord` для данного объекта.

Примечания:

Указатель, возвращенный командой, можно использовать только до следующего вызова `rdsSTRGetWord`.

См. также:

`rdsSTRCreateTextReader` (стр. 460), `rdsGetObjectStr` (стр. 405),
`rdsSTRGetWord` (стр. 462).

A.5.26.8. Команда `RDS_HSTR_GETRESTOFTEXT` – получить остаток текста

Команда `RDS_HSTR_GETRESTOFTEXT` возвращает остаток текста, оставшийся не разобранным после последнего вызова функции `rdsSTRGetWord` (стр. 462).

Вызов команды:

```
char *strRest=rdsGetObjectStr(Parser,RDS_HSTR_GETRESTOFTEXT,0);
```

Параметры и результат:

`Parser`

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией `rdsSTRCreateTextReader` (стр. 460).

`strRest`

Указатель (`char*`) на начало не разобранного остатка текста.

Примечания:

Указатель, возвращенный командой, указывает не на внутреннюю память объекта `Parser`, а на один из символов текста, переданного в объект командой

RDS_HSTR_SETTEXT (стр. 470). С этого места функция rdsSTRGetWord начнет искать начало слова при следующем вызове.

См. также:

rdsSTRCreateTextReader (стр. 460), rdsGetObjectStr (стр. 405),
rdsSTRGetWord (стр. 462), RDS_HSTR_SETTEXT (стр. 470).

А.5.26.9. Команда RDS_HSTR_IGNORECASE – учет регистра символов ключевых слов

Команда RDS_HSTR_IGNORECASE позволяет включать и выключать учет регистра символов при сравнении слов текста с ключевыми в функции rdsSTRGetWord (стр. 462).

Вызов команды для установки:

```
int iIgnoreCase=... // 1 - игнорировать регистр, 0 - учитывать  
rdsSetObjectInt (Parser, RDS_HSTR_IGNORECASE, 0, iIgnoreCase);
```

Вызов команды для чтения:

```
int iIgnoreCase=rdsGetObjectInt (Parser, RDS_HSTR_IGNORECASE, 0);
```

Параметры и результат:

Parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией rdsSTRCreateTextReader (стр. 460).

iIgnoreCase

Целое число (int), указывающее на способ поиска ключевых слов в тексте: 1 – без учета регистра, 0 – с учетом регистра.

Примечания:

Этот параметр можно задать сразу в момент создания объекта для разбора текста при вызове функции rdsSTRCreateTextReader.

См. также:

rdsSTRCreateTextReader (стр. 460), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), rdsSTRGetWord (стр. 462).

А.5.26.10. Команда RDS_HSTR_READDOUBLE – получить из текста вещественное число

Команда RDS_HSTR_READDOUBLE считывает из текста следующее слово и преобразует его в вещественное число.

Вызов команды:

```
int iSkip=... // 1 - пропускать переводы строк, 0 - нет  
double dValue=rdsGetObjectDouble (Parser, RDS_HSTR_READDOUBLE, iSkip);  
  
или  
rdsGetObjectDoubleP (Parser, RDS_HSTR_READDOUBLE, iSkip, &dValue);
```

Параметры и результат:

Parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией rdsSTRCreateTextReader (стр. 460).

iSkip

Целое число (int), указывающее на то, как команда будет реагировать на встретившиеся в тексте переводы строк: 1 – пропускать все переводы строк до тех пор, пока не будет считано полноценное слово; 0 – встретив перевод строки, вернуть нулевое значение.

dValue

Вещественное число (double), в которое преобразовано считанное из текста слово.

Примечания:

Эта команда считывает слово из текста при помощи функции `rdsSTRGetWord` (стр. 462), не сравнивая его с ключевыми словами, а затем преобразует его в вещественное число. Если `iSkip==1`, команда предварительно пропустит все встретившиеся переводы строк. Преобразование в вещественное число производится по правилам функции `rdsAtoD` (стр. 184). Если считанное слово не может быть преобразовано в вещественное число, команда вернет значение 0.0.

Пример использования команды `RDS_HSTR_READDOUBLE` приведен в §2.8.4.

См. также:

`rdsSTRCreateTextReader` (стр. 460), `rdsGetObjectDouble` (стр. 403),
`rdsGetObjectDoubleP` (стр. 404), `rdsSTRGetWord` (стр. 462),
`rdsAtoD` (стр. 184).

A.5.26.11. Команда `RDS_HSTR_READINT` – получить из текста целое число

Команда `RDS_HSTR_READINT` считывает из текста следующее слово и преобразует его в целое число.

Вызов команды:

```
int iSkip=... // 1 - пропускать переводы строк, 0 - нет
int iValue=rdsGetObjectInt(Parser,RDS_HSTR_READINT,iSkip);
```

Параметры и результат:

Parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией `rdsSTRCreateTextReader` (стр. 460).

iSkip

Целое число (int), указывающее на то, как команда будет реагировать на встретившиеся в тексте переводы строк: 1 – пропускать все переводы строк до тех пор, пока не будет считано полноценное слово; 0 – встретив перевод строки, вернуть нулевое значение.

iValue

Целое число (int), в которое преобразовано считанное из текста слово.

Примечания:

Эта команда считывает слово из текста при помощи функции `rdsSTRGetWord` (стр. 462), не сравнивая его с ключевыми словами, а затем преобразует его в целое число. Если `iSkip==1`, команда предварительно пропустит все встретившиеся переводы строк. Для преобразования используется функция `rdsAtoI` (стр. 184). Если считанное слово не может быть преобразовано в целое число, команда вернет значение 0.

Пример использования команды `RDS_HSTR_READINT` приведен в §2.8.4.

См. также:

`rdsSTRCreateTextReader` (стр. 460), `rdsGetObjectInt` (стр. 404),
`rdsSTRGetWord` (стр. 462), `rdsAtol` (стр. 184).

А.5.26.12. Команда RDS_HSTR_SETTEXT – передать в объект текст для разбора

Команда RDS_HSTR_SETTEXT передает в объект текст, из которого функция `rdsSTRGetWord` (стр. 462) будет последовательно извлекать слова.

Вызов команды:

```
char *strText=... // Текст для разбора на слова
rdsSetObjectStr(Parser,RDS_HSTR_SETTEXT,0,strText);
```

Параметры и результат:

`Parser`

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией `rdsSTRCreateTextReader` (стр. 460).

`strText`

Указатель на текст (`char*`), который будет разбираться объектом.

Примечания:

Объект `Parser` запоминает переданный ему указатель `strText`, но сам текст не копируется во внутреннюю память объекта. Таким образом, этот текст должен быть доступен все время работы объекта `Parser`. Если удалить этот текст (освободить динамическую строку, в которой он хранится, или завершить функцию, локальной переменной которой он является), объект попытается получить доступ к освобожденной памяти, что приведет к аварийному завершению работы РДС. Например, следующий фрагмент программы не сможет работать:

```
char *text=new char[100]; // Динамически отведенный массив
strcpy(text,"alpha beta gamma"); // Текст для разбора
// Создание объекта
RDS_ОБЪЕКТ obj=rdsSTRCreateTextReader(TRUE);
// Передача текста в объект
rdsSetObjectStr(obj,RDS_HSTR_SETTEXT,0,text);
delete[] text; // ОШИБКА! Текст используется объектом!
int id=rdsSTRGetWord(obj,NULL,NULL,NULL,TRUE); // Сбой!
```

В этом примере после оператора “`delete[] text`” указатель внутри объекта `obj` начинает ссылаться на область памяти с освобожденными данными, и следующий за ним вызов функции `rdsSTRGetWord` с большой вероятностью приведет к ошибке общей защиты.

Пример использования команды RDS_HSTR_SETTEXT приведен в §2.8.4.

См. также:

`rdsSTRCreateTextReader` (стр. 460), `rdsSetObjectStr` (стр. 407),
`rdsSTRGetWord` (стр. 462).

А.5.26.13. Команда RDS_HSTR_UNKNOWNID – идентификатор неопознанного слова

Команда RDS_HSTR_UNKNOWNID позволяет получать и устанавливать специальный идентификатор, который функция rdsSTRGetWord (стр. 462) будет возвращать, встретив слово, не совпадающее ни с одним из ключевых.

Вызов команды для установки:

```
int iId=... // Идентификатор
rdsSetObjectInt(Parser,RDS_HSTR_UNKNOWNID,0,iId);
```

Вызов команды для чтения:

```
int iId=rdsGetObjectInt(Parser,RDS_HSTR_UNKNOWNID,0);
```

Параметры и результат:

Parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией rdsSTRCreateTextReader (стр. 460).

iId

Целый (int) идентификатор, используемый для сообщения о неопознанном слове. По умолчанию – константа RDS_HSTR_DEFUNKNOWNWORD со значением –1. Все значения по умолчанию специальных идентификаторов в объекте для разбора текста сделаны отрицательными, поэтому, если среди идентификаторов пользовательских ключевых слов нет отрицательных, значения по умолчанию можно не менять.

См. также:

rdsSTRCreateTextReader (стр. 460), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), rdsSTRGetWord (стр. 462).

А.5.26.14. Макрос rdsSTRGetDoubleWord – получить из текста слово как вещественное число

Макрос rdsSTRGetDoubleWord считывает из текста следующее слово и преобразует его в вещественное число.

```
rdsSTRGetDoubleWord(
    parser,          // Вспомогательный объект
    skiplf           // Пропускать переводы строк
)
```

Определение:

```
#define rdsSTRGetDoubleWord(parser, skiplf) \
    rdsGetObjectDouble((parser),RDS_HSTR_READDOUBLE, \
        (skiplf))
```

Параметры:

parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией rdsSTRCreateTextReader (стр. 460).

skiplf

Целое число (int), указывающее на то, как команда будет реагировать на встретившиеся в тексте переводы строк: 1 – пропускать все переводы строк до тех

пор, пока не будет считано полноценное слово; 0 – встретив перевод строки, вернуть нулевое значение.

Возвращаемое значение:

Вещественное число (double), в которое преобразовано считанное из текста слово.

Примечания:

Этот макрос включает в себя вызов команды RDS_HSTR_READDOUBLE (стр. 468).

См. также:

RDS_HSTR_READDOUBLE (стр. 468).

A.5.26.15. Макрос rdsSTRGetIntWord – получить из текста слово как целое число

Макрос rdsSTRGetIntWord считывает из текста следующее слово и преобразует его в целое число.

```
rdsSTRGetIntWord(  
    parser,          // Вспомогательный объект  
    skipplf          // Пропускать переводы строк  
)
```

Определение:

```
#define rdsSTRGetIntWord(parser, skipplf) \  
    rdsGetObjectInt((parser), RDS_HSTR_READINT, \  
        (skipplf))
```

Параметры:

parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией rdsSTRCreateTextReader (стр. 460).

skipplf

Целое число (int), указывающее на то, как команда будет реагировать на встретившиеся в тексте переводы строк: 1 – пропускать все переводы строк до тех пор, пока не будет считано полноценное слово; 0 – встретив перевод строки, вернуть нулевое значение.

Возвращаемое значение:

Целое число (int), в которое преобразовано считанное из текста слово.

Примечания:

Этот макрос включает в себя вызов команды RDS_HSTR_READINT (стр. 469).

См. также:

RDS_HSTR_READINT (стр. 469).

A.5.26.16. Макрос rdsSTRSetTextToRead – передать в объект текст для разбора

Макрос rdsSTRSetTextToRead передает в объект текст, из которого функция rdsSTRGetWord (стр. 462) будет последовательно извлекать слова.


```

rdsSTRSetTextToRead(
    parser,          // Вспомогательный объект
    text             // Текст для разбора
)

```

Определение:

```

#define rdsSTRSetTextToRead(parser, text) \
    rdsSetObjectStr((parser), RDS_HSTR_SETTEXT, \
    0, (text))

```

Параметры:

parser

Идентификатор вспомогательного объекта для разбора текста, ранее созданного функцией `rdsSTRCreateTextReader` (стр. 460).

text

Указатель на текст (`char*`), который будет разбираться объектом.

Примечания:

Этот макрос включает в себя вызов команды `RDS_HSTR_SETTEXT` (стр. 470).

См. также:

`RDS_HSTR_SETTEXT` (стр. 470).

А.5.27. Вспомогательный объект для работы с текстом в формате INI-файла

Описываются функции и команды вспомогательного объекта РДС, предназначенного для разбора и формирования текста в стандартном формате INI-файлов Windows.

А.5.27.1. `rdsINICreateTextHolder` – создать объект для работы с текстом

Функция `rdsINICreateTextHolder` создает вспомогательный объект РДС, с помощью которого можно работать с текстом, хранящим значения различных параметров в формате INI-файлов Windows, то есть в виде “*имя=значение*”.

```

RDS_НОВЕКТ RDSCALL rdsINICreateTextHolder(
    BOOL IgnoreCase           // Без учета регистра
) ;

```

Тип указателя на эту функцию:

`RDS_НoB`

Параметр:

IgnoreCase

`TRUE` – искать в тексте названия параметров и секций без учета регистра символов, `FALSE` – с его учетом.

Возвращаемое значение:

Уникальный идентификатор созданного объекта.

Примечания:

Эта функция создает вспомогательный объект для работы с текстом и возвращает его идентификатор типа `RDS_НОВЕКТ`. Текст, с которым работает этот объект, разделен на секции, каждая из которых начинается с названия секции на отдельной строке в квадратных скобках. Внутри секции каждая строка имеет вид “*имя=значение*”. Например, текст

```
[Section1]
```

```

var1=valuе1
var2=0
[OtherSection]
var1=текстовая строка
var3=10

```

состоит из двух секций “Section1” и “OtherSection”, в первой из которых записаны значения параметров “var1” и “var2”, во второй – “var1” и “var3” (при этом “var1” в разных секциях имеет разное значение). С помощью различных сервисных функций РДС (как общих для всех типов вспомогательных объектов, так и специализированных для данного) в этот объект можно записывать значения параметров и считывать их из него, создавать и удалять секции и т.п. Объект может загружать текст из файла и записывать его в файл, а также передавать текст в РДС для использования его в качестве сохраняемых параметров блока при реакции на событие RDS_BFM_SAVETXT (стр. 54).

Если в параметре IgnoreCase передано значение TRUE, при поиске в тексте названий секций и параметров объект не будет учитывать регистр символов – например, секции “SECTION” и “Section” будут считаться совпадающими.

Созданный вспомогательный объект будет существовать до тех пор, пока схема не будет выгружена из памяти, или пока он не будет удален вызовом функции rdsDeleteObject (стр. 401).

Примеры использования функции rdsINICreateTextHolder приведены в §2.8.5, §4.2 и др.

См. также:

Вспомогательные объекты (стр. 399), rdsDeleteObject (стр. 401),
RDS_BFM_SAVETXT (стр. 54).

A.5.27.2. rdsINIOpenSection – установить текущую секцию

Функция rdsINIOpenSection устанавливает имя существующей секции, с которой будут работать все последующие вызовы команд чтения и записи параметров.

```

BOOL RDSCALL rdsINIOpenSection
    RDS_НОВЕКТ Ini,           // Объект
    LPSTR SectionName       // Имя секции
);

```

Тип указателя на эту функцию:

RDS_BHоS

Параметры:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

SectionName

Указатель на строку с именем секции.

Возвращаемое значение:

TRUE – секция SectionName найдена в тексте, FALSE – секция отсутствует.

Примечания:

Эта функция ищет в тексте, хранящемся в объекте Ini, секцию с именем SectionName, и, если она присутствует в тексте, устанавливает ее в качестве текущей. Все дальнейшие команды чтения и записи параметров будут работать с этой секцией текста. Если

секция с именем `SectionName` в тексте отсутствует, функция вернет `FALSE` и все дальнейшие команды чтения и записи будут игнорироваться.

Если логика работы программы требует, чтобы, в случае отсутствия секции, она была создана, следует использовать команду `RDS_HINI_CREATESECTION` (стр. 480).

Пример использования функции `rdsINIOpenSection` приведен в §2.8.5.

См. также:

`rdsINICreateTextHolder` (стр. 473), `RDS_HINI_CREATESECTION` (стр. 480),
`RDS_HINI_DELETESECTION` (стр. 481).

A.5.27.3. `rdsINIReadDouble` – получить вещественное значение параметра

Функция `rdsINIReadDouble` считывает значение указанного параметра из текущей секции объекта и возвращает его в виде вещественного числа.

```
double RDSCALL rdsINIReadDouble(  
    RDS_HOBJECT Ini,           // Объект  
    LPSTR Key,                 // Имя параметра  
    double DefValue           // Значение по умолчанию  
);
```

Тип указателя на эту функцию:

`RDS_DHoSD`

Параметры:

`Ini`

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

`Key`

Указатель на строку с именем параметра.

`DefValue`

Значение параметра по умолчанию (функция вернет это значение, если в текущей секции текста нет параметра с именем `Key`).

Возвращаемое значение:

Вещественное значение параметра `Key` из текущей секции объекта `Ini`.

Примечания:

Эта функция возвращает вещественное значение параметра с именем `Key` из текущей секции объекта `Ini`, если такой параметр есть в этой секции. Текущей считается секция, установленная функцией `rdsINIOpenSection` (стр. 474) или командой `RDS_HINI_CREATESECTION` (стр. 480). Если в секции нет параметра с указанным именем или если текущая секция не установлена, функция вернет значение `DefValue`. Преобразование строки значения в вещественное число производится по правилам функции `rdsAtoD` (стр. 184).

Если из-за особенностей используемого компилятора возврат типа `double` невозможен, вместо `rdsINIReadDouble` можно использовать функцию `rdsINIReadDoubleP` (стр. 476), возвращающую вещественное число через указатель.

Пример использования функции `rdsINIReadDouble` приведен в §2.8.5.

См. также:

rdsINICreateTextHolder (стр. 473), rdsINIWriteDouble (стр. 478),
rdsINIReadInt (стр. 477), rdsINIReadString (стр. 477),
rdsINIOpenSection (стр. 474), RDS_HINI_CREATESECTION (стр. 480),
rdsINIReadDoubleP (стр. 476), rdsAtoD (стр. 184).

A.5.27.4. rdsINIReadDoubleP – получить вещественное значение параметра

Функция rdsINIReadDoubleP считывает значение указанного параметра из текущей секции объекта и возвращает его в виде вещественного числа через переданный в параметрах функции указатель.

```
void RDSCALL rdsINIReadDoubleP(  
    RDS_HOBJECT Ini,           // Объект  
    LPSTR Key,                 // Имя параметра  
    double DefValue,           // Значение по умолчанию  
    double *pVal               // Возвращаемое значение  
);
```

Тип указателя на эту функцию:

RDS_VHoSDpD

Параметры:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

Key

Указатель на строку с именем параметра.

DefValue

Значение параметра по умолчанию (функция вернет это значение, если в текущей секции текста нет параметра с именем Key).

pVal

Указатель на вещественную переменную двойной точности (double), в которую функция запишет полученное значение.

Примечания:

Эта функция возвращает вещественное значение параметра с именем Key из текущей секции объекта Ini, если такой параметр есть в этой секции. Текущей считается секция, установленная функцией rdsINIOpenSection (стр. 474) или командой RDS_HINI_CREATESECTION (стр. 480). Если в секции нет параметра с указанным именем или если текущая секция не установлена, функция вернет значение DefValue. Преобразование строки значения в вещественное число производится по правилам функции rdsAtoD (стр. 184).

Функция rdsINIReadDoubleP отличается от rdsINIReadDouble (стр. 475) только тем, что возвращает значение через указатель pVal.

См. также:

rdsINICreateTextHolder (стр. 473), rdsINIWriteDouble (стр. 478),
rdsINIReadDouble (стр. 475), rdsAtoD (стр. 184).

A.5.27.5. rdsINIReadInt – получить целое значение параметра

Функция `rdsINIReadInt` считывает значение указанного параметра из текущей секции объекта и возвращает его в виде целого числа.

```
int RDSCALL rdsINIReadInt(  
    RDS_HOBJECT Ini,          // Объект  
    LPSTR Key,                // Имя параметра  
    int DefValue              // Значение по умолчанию  
);
```

Тип указателя на эту функцию:

`RDS_IHoSI`

Параметры:

`Ini`

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

`Key`

Указатель на строку с именем параметра.

`DefValue`

Значение параметра по умолчанию (функция вернет это значение, если в текущей секции текста нет параметра с именем `Key`).

Возвращаемое значение:

Целое значение параметра `Key` из текущей секции объекта `Ini`.

Примечания:

Эта функция возвращает целое значение параметра с именем `Key` из текущей секции объекта `Ini`, если такой параметр есть в этой секции. Текущей считается секция, установленная функцией `rdsINIOpenSection` (стр. 474) или командой `RDS_HINI_CREATESECTION` (стр. 480). Если в секции нет параметра с указанным именем или если текущая секция не установлена, функция вернет значение `DefValue`. Строка значения должна быть символьным представлением целого числа в десятичной системе счисления.

Пример использования функции `rdsINIReadInt` приведен в §2.8.5.

См. также:

`rdsINICreateTextHolder` (стр. 473), `rdsINIWriteInt` (стр. 479),
`rdsINIReadDouble` (стр. 475), `rdsINIReadString` (стр. 477),
`rdsINIOpenSection` (стр. 474), `RDS_HINI_CREATESECTION` (стр. 480).

A.5.27.6. rdsINIReadString – получить текст значения параметра

Функция `rdsINIReadString` считывает значение указанного параметра из текущей секции объекта и возвращает его в виде строки.

```
LPSTR RDSCALL rdsINIReadString(  
    RDS_HOBJECT Ini,          // Объект  
    LPSTR Key,                // Имя параметра  
    LPSTR DefValue,           // Значение по умолчанию  
    int *pLength              // Возвращаемая длина строки  
);
```

Тип указателя на эту функцию:

RDS_SHoSSpI

Параметры:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

Key

Указатель на строку с именем параметра.

DefValue

Указатель на строку с значением параметра по умолчанию (функция вернет этот указатель, если в текущей секции текста нет параметра с именем `Key`).

pLength

Указатель на целую переменную, в которую функция должна записать длину строки значения параметра. Если вызывающей программе не нужна длина строки, в этом параметре можно передать `NULL`.

Возвращаемое значение:

Указатель на строку значения параметра `Key` из текущей секции объекта `Ini`, находящуюся во внутренней памяти объекта. Вызывающая программа не должна изменять эту строку. Функция может вернуть `NULL`, если `Ini` – не объект для работы с текстом, или если вместо имени параметра передано значение `NULL`.

Примечания:

Эта функция возвращает указатель на строку значения параметра с именем `Key` в текущей секции объекта `Ini`, если такой параметр есть в этой секции. Текущей считается секция, установленная функцией `rdsINIOpenSection` (стр. 474) или командой `RDS_HINI_CREATESECTION` (стр. 480). Если в секции нет параметра с указанным именем или если текущая секция не установлена, функция вернет значение `DefValue`.

Пример использования функции `rdsINIReadString` приведен в §2.10.1.

См. также:

`rdsINICreateTextHolder` (стр. 473), `rdsINIWriteString` (стр. 480),
`rdsINIReadDouble` (стр. 475), `rdsINIReadInt` (стр. 477),
`rdsINIOpenSection` (стр. 474), `RDS_HINI_CREATESECTION` (стр. 480).

A.5.27.7. `rdsINIWriteDouble` – установить вещественное значение параметра

Функция `rdsINIWriteDouble` присваивает вещественное значение указанному параметру текущей секции объекта.

```
void RDSCALL rdsINIWriteDouble(  
    RDS_HOBJECT Ini,          // Объект  
    LPSTR Key,                // Имя параметра  
    double Value              // Значение  
);
```

Тип указателя на эту функцию:

RDS_VHoSD

Параметры:

Ini	Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией <code>rdsINICreateTextHolder</code> (стр. 473).
Key	Указатель на строку с именем параметра.
Value	Значение параметра.

Примечания:

Эта функция устанавливает вещественное значение параметра с именем `Key` в текущей секции объекта `Ini` (если такого параметра нет в текущей секции, он будет создан). Текущей считается секция, установленная функцией `rdsINIOpenSection` (стр. 474) или командой `RDS_HINI_CREATESECTION` (стр. 480).

Пример использования функции `rdsINIWriteDouble` приведен в §2.8.5.

См. также:

`rdsINICreateTextHolder` (стр. 473), `rdsINIReadDouble` (стр. 475),
`rdsINIWriteInt` (стр. 479), `rdsINIWriteString` (стр. 480),
`rdsINIOpenSection` (стр. 474), `RDS_HINI_CREATESECTION` (стр. 480).

A.5.27.8. `rdsINIWriteInt` – установить целое значение параметра

Функция `rdsINIWriteInt` присваивает целое значение указанному параметру текущей секции объекта.

```
void RDSCALL rdsINIWriteInt(  
    RDS_HOBJECT Ini,          // Объект  
    LPSTR Key,                // Имя параметра  
    int Value                 // Значение  
);
```

Тип указателя на эту функцию:

`RDS_VHoSI`

Параметры:

Ini	Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией <code>rdsINICreateTextHolder</code> (стр. 473).
Key	Указатель на строку с именем параметра.
Value	Значение параметра.

Примечания:

Эта функция устанавливает целое значение параметра с именем `Key` в текущей секции объекта `Ini` (если такого параметра нет в текущей секции, он будет создан). Текущей считается секция, установленная функцией `rdsINIOpenSection` (стр. 474) или командой `RDS_HINI_CREATESECTION` (стр. 480).

Пример использования функции `rdsINIWriteInt` приведен в §2.8.5.

См. также:

rdsINICreateTextHolder (стр. 473), rdsINIReadInt (стр. 477),
rdsINIWriteDouble (стр. 478), rdsINIWriteString (стр. 480),
rdsINIOpenSection (стр. 474), RDS_HINI_CREATESECTION (стр. 480).

A.5.27.9. rdsINIWriteString – установить текстовое значение параметра

Функция rdsINIWriteString присваивает строку указанному параметру текущей секции объекта.

```
void RDSCALL rdsINIWriteString(  
    RDS_HOBJECT Ini,          // Объект  
    LPSTR Key,                // Имя параметра  
    LPSTR Value               // Значение  
);
```

Тип указателя на эту функцию:

RDS_VHoSS

Параметры:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

Key

Указатель на строку с именем параметра.

Value

Указатель на строку со значением параметра.

Примечания:

Эта функция устанавливает текстовое значение параметра с именем Key в текущей секции объекта Ini (если такого параметра нет в текущей секции, он будет создан). Текущей считается секция, установленная функцией rdsINIOpenSection (стр. 474) или командой RDS_HINI_CREATESECTION (стр. 480).

Пример использования функции rdsINIWriteString приведен в §2.10.1.

См. также:

rdsINICreateTextHolder (стр. 473), rdsINIReadString (стр. 477),
rdsINIWriteDouble (стр. 478), rdsINIWriteInt (стр. 479),
rdsINIOpenSection (стр. 474), RDS_HINI_CREATESECTION (стр. 480).

A.5.27.10. Команда RDS_HINI_CREATESECTION – создать секцию

Команда RDS_HINI_CREATESECTION создает в тексте секцию с указанным именем и делает ее текущей.

Вызов команды:

```
char *strName=... // Имя секции  
rdsSetObjectStr(Ini, RDS_HINI_CREATESECTION, 0, strName);
```

Параметры и результат:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

strName

Указатель на строку (char*), в которой записано имя создаваемой секции.

Примечания:

Эта команда создает в тексте, хранящемся в объекте Ini, новую секцию с именем strName и делает ее текущей – все последующие вызовы чтения и записи параметров будут работать с этой секцией. Если секция с именем strName уже есть в тексте объекта, она просто становится текущей.

Пример использования команды RDS_HINI_CREATESECTION приведен в §2.8.5.

См. также:

rdsINICreateTextHolder (стр. 473), rdsSetObjectStr (стр. 407),
rdsINIOpenSection (стр. 474), RDS_HINI_DELETESECTION (стр. 481).

А.5.27.11. Команда RDS_HINI_DELETEKEYLAST – удалить параметр из текущей секции

Команда RDS_HINI_DELETEKEYLAST удаляет из текущей секции текста параметр с указанным именем.

Вызов команды:

```
char *strName=... // Имя параметра  
rdsSetObjectStr(Ini,RDS_HINI_DELETEKEYLAST,0,strName);
```

Параметры и результат:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

strName

Указатель на строку (char*), в которой записано имя удаляемого параметра.

Примечания:

Эта команда удаляет из текущей секции текста, хранящегося в объекте Ini, параметр с именем strName. Текущей считается секция, установленной функцией rdsINIOpenSection (стр. 474) или командой RDS_HINI_CREATESECTION (стр. 480).

См. также:

rdsINICreateTextHolder (стр. 473), rdsSetObjectStr (стр. 407),
RDS_HINI_DELETESECTION (стр. 481), rdsINIOpenSection (стр. 474),
RDS_HINI_CREATESECTION (стр. 480).

А.5.27.12. Команда RDS_HINI_DELETESECTION – удалить секцию

Команда RDS_HINI_DELETESECTION удаляет из текста секцию с указанным именем.

Вызов команды:

```
char *strName=... // Имя секции  
rdsSetObjectStr(Ini,RDS_HINI_DELETESECTION,0,strName);
```

Параметры и результат:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

strName

Указатель на строку (`char*`), в которой записано имя удаляемой секции.

Примечания:

Эта команда удаляет из текста, хранящегося в объекте `Ini`, секцию с именем `strName` вместе со всеми находящимися в ней параметрами. Если эта секция была текущей, после ее удаления объект не будет иметь текущей секции и все вызовы чтения и записи параметров будут игнорироваться.

См. также:

`rdsINICreateTextHolder` (стр. 473), `rdsSetObjectStr` (стр. 407),
`rdsINIOpenSection` (стр. 474), `RDS_HINI_CREATESECTION` (стр. 480).

A.5.27.13. Команда `RDS_HINI_GETLASTERROR` – получить результат последней операции

Команда `RDS_HINI_GETLASTERROR` возвращает успешность последней выполненной объектом операции.

Вызов команды:

```
BOOL bError=rdsCommandObject(Ini,RDS_HINI_GETLASTERROR);  
или  
BOOL bError=rdsCommandObjectEx(Ini,RDS_HINI_GETLASTERROR,0,NULL);
```

Параметры и результат:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

bError

Логический (`BOOL`) результат выполнения команды: `TRUE` – при выполнении последней операции возникли ошибки, `FALSE` – последняя операция выполнена успешно.

Примечания:

Эта команда используется в тех случаях, когда в какой-либо другой команде работы с объектом не предусмотрен возврат успешности выполнения. Например, для того, чтобы узнать, удалось ли загрузить в объект текст из файла командой `RDS_HINI_LOADFILE` (стр. 482), сразу после нее нужно выполнить команду `RDS_HINI_GETLASTERROR`.

Пример использования команды `RDS_HINI_GETLASTERROR` приведен в §4.2.

См. также:

`rdsINICreateTextHolder` (стр. 473), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400).

A.5.27.14. Команда `RDS_HINI_LOADFILE` – загрузить текст из файла

Команда `RDS_HINI_LOADFILE` загружает в объект текст из файла с указанным именем.

Вызов команды:

```
char *strFileName=... // Имя файла  
rdsSetObjectStr (Ini, RDS_HINI_LOADFILE, 0, strFileName);
```

Параметры и результат:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

strFileName

Указатель на строку (`char*`), в которой записано имя файла, из которого нужно загрузить текст. Имя файла может содержать как полный путь, так и сокращенный с символическими обозначениями стандартных путей (см. стр. 189).

Примечания:

Эта команда загружает в объект текст из файла с указанным именем. Успешность загрузки можно узнать, выполнив команду `RDS_HINI_GETLASTERROR` (стр. 482).

Пример использования команды `RDS_HINI_LOADFILE` приведен в §4.2.

См. также:

`rdsINICreateTextHolder` (стр. 473), `rdsSetObjectStr` (стр. 407),
`RDS_HINI_SAVEFILE` (стр. 484), `RDS_HINI_SETTEXT` (стр. 485),
`RDS_HINI_GETLASTERROR` (стр. 482).

А.5.27.15. Команда `RDS_HINI_RESET` – очистка текста

Команда `RDS_HINI_RESET` очищает текст в указанном объекте, удаляя из него все секции со всеми их параметрами.

Вызов команды:

```
rdsCommandObject (Ini, RDS_HINI_RESET);  
или  
rdsCommandObjectEx (Ini, RDS_HINI_RESET, 0, NULL);
```

Параметр:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

См. также:

`rdsINICreateTextHolder` (стр. 473), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400), `RDS_HINI_LOADFILE` (стр. 482),
`RDS_HINI_SETTEXT` (стр. 485).

А.5.27.16. Команда `RDS_HINI_SAVEBLOCKTEXT` – передать текст параметров блока в РДС

Команда `RDS_HINI_SAVEBLOCKTEXT` передает в РДС текст, содержащийся в объекте, для добавления к сохраняемым параметрам блока при реакции модели этого блока на событие `RDS_BFM_SAVETXT` (стр. 54).

Вызов команды:

```
rdsCommandObject (Ini, RDS_HINI_SAVEBLOCKTEXT);  
или  
int iNewLine=... // 1 - с новой строки, 0 - без новой строки  
rdsCommandObjectEx (Ini, RDS_HINI_SAVEBLOCKTEXT, iNewLine, NULL);
```

Параметры:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

iNewLine

Целое (`int`) значение, указывающее на необходимость добавить перед текстом объекта перевод строки (1) или передать текст в РДС без изменений (0). При вызове этой команды функцией `rdsCommandObject` (стр. 400) перевод строки не добавляется.

Примечания:

Эта команда передает сформированный в объекте текст в РДС, где он будет добавлен к сохраняемым в текстовом виде параметрам блока. Если на момент вызова команды `RDS_HINI_SAVEBLOCKTEXT` сохраняемый текст уже был сформирован вызовами функций, описанных в А.5.10, текст из объекта `Ini` будет добавлен в конец этого текста.

Команду `RDS_HINI_SAVEBLOCKTEXT` можно вызывать только из модели блока в момент ее реакции на событие сохранения данных в текстовом виде `RDS_BFM_SAVETXT`. Во всех остальных случаях команда будет проигнорирована.

Пример использования команды `RDS_HINI_SAVEBLOCKTEXT` приведен в §2.8.5.

См. также:

`rdsINICreateTextHolder` (стр. 473), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400), `RDS_BFM_SAVETXT` (стр. 54).

А.5.27.17. Команда `RDS_HINI_SAVEFILE` – записать текст в файл

Команда `RDS_HINI_SAVEFILE` записывает текст, хранящийся в объекте, в файл с указанным именем.

Вызов команды:

```
char *strFileName=... // Имя файла  
rdsSetObjectStr (Ini, RDS_HINI_SAVEFILE, 0, strFileName);
```

Параметры:

Ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

strFileName

Указатель на строку (`char*`), в которой записано имя файла, в который нужно записать текст. Имя файла может содержать как полный, так и сокращенный путь с символическими обозначениями стандартных папок (см. стр. 189).

Примечания:

Эта команда записывает текст из объекта в файл с указанным именем. Успешность записи можно узнать, выполнив команду `RDS_HINI_GETLASTERROR` (стр. 482).

Пример использования команды `RDS_HINI_SAVEFILE` приведен в §4.2.

См. также:

`rdsINICreateTextHolder` (стр. 473), `rdsSetObjectStr` (стр. 407),
`RDS_HINI_LOADFILE` (стр. 482), `RDS_HINI_GETLASTERROR` (стр. 482).

A.5.27.18. Команда `RDS_HINI_SETTEXT` – занести текст в объект

Команда `RDS_HINI_SETTEXT` копирует указанный текст во внутреннюю память объекта.

Вызов команды:

```
char *strText=... // Текст в формате INI-файла
rdsSetObjectStr (ini, RDS_HINI_SETTEXT, 0, strText);
```

Параметры:

`ini`

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

`strText`

Указатель на текст (`char*`) в формате INI-файла, который должен быть скопирован в объект.

Примечания:

Эта команда копирует текст `strText` во внутреннюю память объекта `ini`. После этого можно обращаться к секциям и параметрам этого текста соответствующими сервисными функциями. Текст, содержащийся в объекте до выполнения этой команды, теряется.

Пример использования команды `RDS_HINI_SETTEXT` приведен в §2.8.5.

См. также:

`rdsINICreateTextHolder` (стр. 473), `rdsSetObjectStr` (стр. 407),
`RDS_HINI_LOADFILE` (стр. 482), `RDS_HINI_RESET` (стр. 483).

A.5.27.19. Макрос `rdsINIClearText` – очистка объекта

Макрос `rdsINIClearText` очищает текст в объекте.

```
rdsINIClearText (
    ini          // Вспомогательный объект
)
```

Определение:

```
#define rdsINIClearText (ini) \
    rdsCommandObject ((ini), RDS_HINI_RESET)
```

Параметр:

`ini`

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

Примечания:

Этот макрос включает в себя вызов команды `RDS_HINI_RESET` (стр. 483).

См. также:

`RDS_HINI_RESET` (стр. 483).

A.5.27.20. Макрос rdsINICreateSection – создать секцию

Макрос rdsINICreateSection создает в тексте секцию с указанным именем и делает ее текущей.

```
rdsINICreateSection(  
    ini,           // Вспомогательный объект  
    sectionname    // Имя секции  
)
```

Определение:

```
#define rdsINICreateSection(ini, sectionname) \  
    rdsSetObjectStr((ini), RDS_HINI_CREATESECTION, \  
        0, (sectionname))
```

Параметры:

ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

sectionname

Указатель на строку (char*) названия секции.

Примечания:

Макрос rdsINICreateSection включает в себя вызов команды RDS_HINI_CREATESECTION (стр. 480).

См. также:

RDS_HINI_CREATESECTION (стр. 480).

A.5.27.21. Макрос rdsINIDeleteSection – удалить секцию

Макрос rdsINIDeleteSection удаляет из текста секцию с указанным именем.

```
rdsINIDeleteSection(  
    ini,           // Вспомогательный объект  
    sectionname    // Имя секции  
)
```

Определение:

```
#define rdsINIDeleteSection(ini, sectionname) \  
    rdsSetObjectStr((ini), RDS_HINI_DELETESECTION, \  
        0, (sectionname))
```

Параметры:

ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

sectionname

Указатель на строку (char*) названия секции.

Примечания:

Макрос rdsINIDeleteSection включает в себя вызов команды RDS_HINI_DELETESECTION (стр. 481).

См. также:

RDS_HINI_DELETESECTION (стр. 481).

A.5.27.22. Макрос rdsINIDeleteValue – удалить параметр из текущей секции

Макрос rdsINIDeleteValue удаляет из текущей секции текста параметр с указанным именем.

```
rdsINIDeleteValue(  
    ini,          // Вспомогательный объект  
    key           // Имя параметра  
)
```

Определение:

```
#define rdsINIDeleteValue(ini, key) \  
    rdsSetObjectStr((ini), RDS_HINI_DELETEKEYLAST, \  
        0, (key))
```

Параметры:

ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

key

Указатель на строку (char*) с именем удаляемого параметра.

Примечания:

Макрос rdsINIDeleteValue включает в себя вызов команды RDS_HINI_DELETEKEYLAST (стр. 481).

См. также:

RDS_HINI_DELETEKEYLAST (стр. 481).

A.5.27.23. Макрос rdsINILoadFile – загрузить текст из файла

Макрос rdsINILoadFile загружает в объект текст из файла с указанным именем.

```
rdsINILoadFile(  
    ini,          // Вспомогательный объект  
    filename      // Имя файла  
)
```

Определение:

```
#define rdsINILoadFile(ini, filename) \  
    rdsSetObjectStr((ini), RDS_HINI_LOADFILE, \  
        0, (filename))
```

Параметры:

ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

filename

Указатель на строку (char*), в которой записано имя файла, из которого нужно загрузить текст. Имя файла может содержать как полный путь, так и сокращенный с символическими обозначениями путей (см. стр. 189).

Примечания:

Этот макрос включает в себя вызов команды RDS_HINI_LOADFILE (стр. 482).

См. также:

RDS_HINI_LOADFILE (стр. 482).

A.5.27.24. Макрос rdsINIReadBool – получить логическое значение параметра

Макрос rdsINIReadBool считывает указанный параметр из текущей секции объекта и возвращает его в виде логического значения.

```
rdsINIReadBool(  
    ini,          // Вспомогательный объект  
    key,          // Имя параметра  
    defval       // Значение по умолчанию  
)
```

Определение:

```
#define rdsINIReadBool(ini, key, defval) \  
    (rdsINIReadInt((ini), (key), (defval) ? 1 : 0) != 0)
```

Параметры:

ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

key

Указатель на строку (char*) с именем параметра.

defval

Логическое (BOOL) значение параметра по умолчанию (функция вернет это значение, если в текущей секции текста нет параметра с именем key).

Возвращаемое значение:

Логическое (BOOL) значение параметра key из текущей секции объекта ini.

Примечания:

Этот макрос включает в себя вызов функции rdsINIReadInt (стр. 477), в котором считанное целое значение параметра сравнивается с нулем. Таким образом, нулевое значение параметра будет считаться ложью, не нулевое – истиной.

См. также:

rdsINIReadInt (стр. 477), rdsINIWriteBool (стр. 490).

A.5.27.25. Макрос rdsINISaveBlockText – передать текст параметров блока в РДС

Макрос rdsINISaveBlockText передает в РДС текст, содержащийся в объекте, для добавления к сохраняемым параметрам блока при реакции модели этого блока на событие RDS_BFM_SAVETXT (стр. 54).

```
rdsINISaveBlockText(  
    ini,                // Вспомогательный объект  
    newline             // Перевод строки перед текстом  
)
```

Определение:

```
#define rdsINISaveBlockText(ini, newline) \  
    rdsCommandObjectEx((ini), RDS_HINI_SAVEBLOCKTEXT, \  
        (newline)?1:0, NULL)
```

Параметры:

ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

newline

Логическое или целое значение, указывающее на необходимость добавить перед текстом объекта перевод строки (1, TRUE) или передать текст в РДС без изменений (0, FALSE).

Примечания:

Этот макрос включает в себя вызов команды RDS_HINI_SAVEBLOCKTEXT (стр. 483).

См. также:

RDS_HINI_SAVEBLOCKTEXT (стр. 483).

A.5.27.26. Макрос rdsINISaveFile – записать текст в файл

Макрос rdsINISaveFile записывает текст, хранящийся в объекте, в файл с указанным именем.

```
rdsINISaveFile(  
    ini,                // Вспомогательный объект  
    filename            // Имя файла  
)
```

Определение:

```
#define rdsINISaveFile(ini, filename) \  
    rdsSetObjectStr((ini), RDS_HINI_SAVEFILE, \  
        0, (filename))
```

Параметры:

ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

filename

Указатель на строку (char*), в которой записано имя файла, в который нужно записать текст. Имя файла может содержать как полный путь, так и сокращенный с символическими обозначениями путей (см. стр. 189).

Примечания:

Этот макрос включает в себя вызов команды RDS_HINI_SAVEFILE (стр. 484).

См. также:

RDS_HINI_SAVEFILE (стр. 484).

A.5.27.27. Макрос rdsINISetText – занести текст в объект

Макрос rdsINISetText копирует указанный текст во внутреннюю память объекта.

```
rdsINISetText(  
    ini,          // Вспомогательный объект  
    text         // Текст  
)
```

Определение:

```
#define rdsINISetText(ini, text) \  
    rdsSetObjectStr((ini), RDS_HINI_SETTEXT, 0, (text))
```

Параметры:

ini

Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией rdsINICreateTextHolder (стр. 473).

text

Указатель на текст (char*) в формате INI-файла, который должен быть скопирован в объект.

Примечания:

Этот макрос включает в себя вызов команды RDS_HINI_SETTEXT (стр. 485).

См. также:

RDS_HINI_SETTEXT (стр. 485).

A.5.27.28. Макрос rdsINIWriteBool – установить логическое значение параметра

Макрос rdsINIWriteBool присваивает логическое значение указанному параметру текущей секции объекта, преобразуя его в целое.

```
rdsINIWriteBool(  
    ini,          // Вспомогательный объект  
    key,          // Имя параметра  
    val          // Значение  
)
```

Определение:

```
#define rdsINIWriteBool(ini, key, val) \  
    rdsINIWriteInt((ini), (key), (val)?1:0)
```

Параметры:

ini
Идентификатор вспомогательного объекта для работы с текстом, ранее созданного функцией `rdsINICreateTextHolder` (стр. 473).

key
Указатель на строку (`char*`) с именем параметра.

val
Логическое (`BOOL`) значение параметра.

Примечания:

Этот макрос включает в себя вызов функции `rdsINIWriteInt` (стр. 477), в котором параметру присваивается единица, если `val==TRUE`, и ноль, если `val==FALSE`. Таким образом, для хранения логического значения используется целое.

См. также:

`rdsINIWriteInt` (стр. 479), `rdsINIReadBool` (стр. 488).

А.5.28. Вспомогательный объект для работы с модальными окнами

Описываются функции и команды вспомогательного объекта РДС, предназначенного для открытия модальных окон с различными полями ввода.

А.5.28.1. `rdsFORMCreate` – создать объект для работы с окном

Функция `rdsFORMCreate` создает вспомогательный объект РДС, позволяющий открыть модальное окно с различными полями для ввода параметров.

```
RDS_HOBJECT RDSCALL rdsFORMCreate(  
    BOOL Tabbed,          // У окна есть вкладки  
    int Width,            // Ширина окна или -1  
    int Height,           // Высота окна или -1  
    LPSTR Caption        // Заголовок окна  
);
```

Тип указателя на эту функцию:

`RDS_HoBIIS`

Параметры:

Tabbed
`TRUE` – внутри окна будет несколько вкладок, каждая со своими полями ввода.
`FALSE` – окно не будет иметь вкладок.

Width
Ширина окна в точках экрана, или `-1`, если ширину нужно автоматически подобрать так, чтобы все поля ввода уместились в окно.

Height
Высота окна в точках экрана, или `-1`, если высоту нужно автоматически подобрать так, чтобы все поля ввода уместились в окно.

Caption
Указатель на строку с заголовком окна.

Возвращаемое значение:

Уникальный идентификатор созданного объекта.

Примечания:

Эта функция создает вспомогательный объект, с помощью которого можно открывать модальные окна для редактирования различных параметров. Поля ввода различных типов добавляются в объект функцией `rdsFORMAddEdit` (стр. 492). Для открытия окна используется команда `RDS_FORM_SHOWMODAL` (стр. 506) или сервисные функции `rdsFORMShowModalEx` (стр. 503) и `rdsFORMShowModalServ` (стр. 504).

Созданный вспомогательный объект будет существовать до тех пор, пока схема не будет выгружена из памяти, или пока он не будет удален вызовом функции `rdsDeleteObject` (стр. 401).

Обычно работа с модальным окном при помощи данного вспомогательного объекта строится по следующему сценарию:

- объект создается вызовом `rdsFORMCreate`;
- вызовами `rdsFORMAddEdit` в объект добавляются поля ввода;
- различными командами – `RDS_FORMVAL_VALUE` (стр. 525) и другими – в поля ввода записываются значения;
- окно объекта открывается командой `RDS_FORM_SHOWMODAL`, вызовом `rdsFORMShowModalEx` или `rdsFORMShowModalServ`;
- если пользователь закрыл окно кнопкой “ОК”, измененные значения считываются из полей ввода;
- объект удаляется вызовом `rdsDeleteObject`.

Примеры использования функции `rdsFORMCreate` приведены в §2.7.2, §2.10.1 и др.

См. также:

Вспомогательные объекты (стр. 399), `rdsDeleteObject` (стр. 401),
`rdsFORMAddEdit` (стр. 492), `rdsFORMShowModalEx` (стр. 503),
`rdsFORMShowModalServ` (стр. 504), `RDS_FORM_SHOWMODAL` (стр. 506).

A.5.28.2. `rdsFORMAddEdit` – добавить поле ввода

Функция `rdsFORMAddEdit` добавляет в объект для работы с модальным окном новое поле ввода.

```
void RDSCALL rdsFORMAddEdit(  
    RDS_HOBJECT Win,           // Объект  
    int TabId,                 // Идентификатор вкладки  
    int CtrlId,                 // Идентификатор поля  
    DWORD Type,                // Тип (RDS_FORMCTRL_* | RDS_FORMFLAG_*)  
    LPSTR Caption,             // Заголовок поля  
    int Width                  // Ширина поля или -1  
);
```

Тип указателя на эту функцию:

`RDS_VHObjIDwSI`

Параметры:

`Win`

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

`TabId`

Целый идентификатор вкладки окна или боковой панели (см. стр. 502), на которую будет добавлено поле. Вкладки создаются вызовами `rdsFORMAddTab` (стр. 501). Если в окне нет ни вкладок, ни боковой панели, параметр игнорируется.

CtrlId

Произвольный целый идентификатор, который будет присвоен созданному полю ввода. Вся дальнейшая работа с этим полем будет вестись по этому идентификатору.

Type

Одна из констант `RDS_FORMCTRL_*`, указывающая тип создаваемого поля ввода, объединенная битовым ИЛИ с флагами поля `RDS_FORMFLAG_*`. Типы и флаги полей подробно рассматриваются в А.5.28.3 (стр. 494). В окно может быть добавлено поле ввода одного из следующих типов:

<code>RDS_FORMCTRL_BUTTON</code>	Кнопка, реакция на которую определяется функцией обратного вызова (стр. 494).
<code>RDS_FORMCTRL_CHECKBOX</code>	Одиночный флаг (стр. 495).
<code>RDS_FORMCTRL_COLOR</code>	Кнопка выбора цвета (стр. 495).
<code>RDS_FORMCTRL_COMBOEDIT</code>	Поле ввода с выпадающим списком готовых вариантов заполнения (стр. 495).
<code>RDS_FORMCTRL_COMBOLIST</code>	Выпадающий список без возможности ручного ввода значения (стр. 495).
<code>RDS_FORMCTRL_DIRDIALOG</code>	Кнопка выбора папки (стр. 496).
<code>RDS_FORMCTRL_DISPLAY</code>	Индикация значения (стр. 496).
<code>RDS_FORMCTRL_EDIT</code>	Обычное поле ввода на одну строку текста (стр. 496).
<code>RDS_FORMCTRL_FONTSELECT</code>	Кнопка выбора шрифта (стр. 496).
<code>RDS_FORMCTRL_HOTKEY</code>	Поле ввода для сочетания клавиш (стр. 496).
<code>RDS_FORMCTRL_LABEL</code>	Надпись без поля ввода (стр. 497).
<code>RDS_FORMCTRL_LISTANDEDIT</code>	Поле ввода с дополнительным выпадающим списком (стр. 497).
<code>RDS_FORMCTRL_MULTILINE</code>	Поле ввода нескольких строк текста (стр. 497).
<code>RDS_FORMCTRL_NONVISUAL</code>	Скрытое поле (стр. 498).
<code>RDS_FORMCTRL_OPENDIALOG</code>	Кнопка с диалогом открытия файла (стр. 498).
<code>RDS_FORMCTRL_PAINTBOX</code>	Область программного рисования (стр. 498).
<code>RDS_FORMCTRL_RADIOBUTTON</code>	Флаг, связанный с другими флагами – включен может быть только один (стр. 499).
<code>RDS_FORMCTRL_RANGEEDIT</code>	Два поля ввода для задания диапазона значений (стр. 499).
<code>RDS_FORMCTRL_SAVEDIALOG</code>	Кнопка с диалогом сохранения файла (стр. 499).
<code>RDS_FORMCTRL_UPDOWN</code>	Поле ввода со стрелками вверх и вниз (стр. 500).
Для установки и чтения значения полей ввода используется команда <code>RDS_FORMVAL_VALUE</code> (стр. 525), но для некоторых типов полей существуют дополнительные команды: например, для поля ввода со стрелками можно установить шаг изменения значения, для поля с выпадающим списком – список вариантов и т.п. Константа типа поля ввода <code>RDS_FORMCTRL_*</code> в параметре <code>Type</code> может объединяться с флагами <code>RDS_FORMFLAG_*</code> , модифицирующими внешний вид поля:	
<code>RDS_FORMFLAG_CHECK</code>	Слева от заголовка поля находится флаг, управляющий разрешением и запрещением этого поля (стр. 500).
<code>RDS_FORMFLAG_CHECKRADIO</code> (только вместе с флагом <code>RDS_FORMFLAG_CHECK</code>).	Разрешающий флаг слева от заголовка поля входит в группу взаимоисключающих флагов (стр. 501).
<code>RDS_FORMFLAG_DISABLED</code>	Поле ввода запрещено и отображается серым цветом (стр. 501).

RDS_FORMFLAG_LCENTER	Заголовок поля ввода выровнен по центру (стр. 501).
RDS_FORMFLAG_LINE	Под полем располагается рельефная линия (стр. 501).
RDS_FORMFLAG_LRIGHT	Заголовок поля ввода выровнен по правому краю (стр. 501).

Caption

Указатель на строку с заголовком поля ввода. В большинстве полей заголовок размещается слева от самого поля и, если не установлены флаги RDS_FORMFLAG_LRIGHT или RDS_FORMFLAG_LCENTER, выровнен по левому краю. В полях типа RDS_FORMCTRL_CHECKBOX и RDS_FORMCTRL_RADIOBUTTON заголовок размещается справа от галочки, по которой может щелкать пользователь, и всегда выровнен по левому краю. В полях типа RDS_FORMCTRL_MULTILINE и RDS_FORMCTRL_PAINTBOX заголовок размещается над полем (поле занимает всю ширину окна). Если для поля задан флаг RDS_FORMFLAG_CHECK, дополнительная галочка будет размещаться слева от заголовка.

Width

Ширина поля в точках экрана или -1 для ширины по умолчанию. Для некоторых типов полей ввода (RDS_FORMCTRL_MULTILINE, RDS_FORMCTRL_PAINTBOX, RDS_FORMCTRL_CHECKBOX, RDS_FORMCTRL_RADIOBUTTON) ширина устанавливается автоматически и значение Width игнорируется.

Примечания:

Эта функция добавляет в объект Win на вкладку TabId поле ввода типа Type с заголовком Caption и идентификатором CtrlId. Поля добавляются сверху вниз, то есть каждый следующий вызов rdsFORMAddEdit добавляет поле на вкладку TabId непосредственно под последним добавленным полем. Идентификаторы полям можно присваивать произвольно – главное, чтобы в одном окне не было двух полей с одинаковыми идентификаторами.

Константы RDS_FORMFLAG_* позволяют изменить выравнивание заголовка поля ввода, а также добавить слева от этого заголовка флаг, который будет разрешать или запрещать ввод данных в поле.

См. также:

rdsFORMCreate (стр. 491), rdsFORMAddTab (стр. 501),
RDS_FORMVAL_VALUE (стр. 525), RDS_FORMVAL_ENABLED (стр. 511),
rdsFORMEnableSidePanel (стр. 502), типы и флаги полей ввода (стр. 494).

A.5.28.3. Типы и флаги полей ввода

Типы и флаги полей ввода указываются при добавлении полей во вспомогательный объект для работы с окнами функцией rdsFORMAddEdit (стр. 492).

Типы полей ввода

Тип поля ввода – это одна из констант RDS_FORMCTRL_*, передающихся в младшем байте параметра Type функции добавления поля ввода rdsFORMAddEdit. На данный момент в РДС поддерживаются следующие типы полей ввода:

RDS_FORMCTRL_BUTTON

Кнопка, реакция на которую определяется функцией обратного вызова (пример ее



использования приведен в §2.16.1). Для открытия окна с такими кнопками нужно использовать функцию `rdsFORMShowModalServ` (стр. 504), только она поддерживает функцию обратного вызова, которая может реагировать на нажатия кнопок (событие `RDS_FORMSERVEVENT_CLICK`, см. стр. 128). Кнопка поддерживает следующие команды:

- `RDS_FORMVAL_VALUE` (стр. 525) – установка и чтение текста на кнопке;
- `RDS_FORMVAL_ENABLED` (стр. 511) – разрешение/запрещение кнопки;
- `RDS_FORMVAL_CHECK` (стр. 510) – управление дополнительным разрешающим флагом, если при создании кнопки был указан флаг `RDS_FORMFLAG_CHECK` (стр. 500).

`RDS_FORMCTRL_CHECKBOX`

Одиночный флаг, не связанный с другими (пример такого поля приведен в §2.13.6). Пользователь может включить или выключить его.



Поле поддерживает следующие команды:

- `RDS_FORMVAL_VALUE` (стр. 525) – установка и чтение значения флага (0 – флаг выключен, 1 – флаг включен);
- `RDS_FORMVAL_ENABLED` (стр. 511) – разрешение/запрещение поля.

`RDS_FORMCTRL_COLOR`

Кнопка выбора цвета (пример такого поля приведен в §2.10.1). При ее нажатии открывается стандартный диалог выбора цвета Windows.



Текущий выбранный цвет изображается на самой кнопке. Поле поддерживает следующие команды:

- `RDS_FORMVAL_VALUE` (стр. 525) – установка и чтение цвета в виде целого числа;
- `RDS_FORMVAL_ENABLED` (стр. 511) – разрешение/запрещение поля;
- `RDS_FORMVAL_CHECK` (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг `RDS_FORMFLAG_CHECK` (стр. 500).

`RDS_FORMCTRL_COMBOEDIT`

Поле ввода с выпадающим списком готовых вариантов заполнения. Пользователь может ввести значение поля с клавиатуры или выбрать одно из значений в выпадающем списке. Поле поддерживает следующие команды:



- `RDS_FORMVAL_VALUE` (стр. 525) – установка и чтение значения поля;
- `RDS_FORMVAL_ENABLED` (стр. 511) – разрешение/запрещение поля;
- `RDS_FORMVAL_LIST` (стр. 515) – установка списка вариантов (строки списка разделяются символом перевода строки “\n” с кодом 10);
- `RDS_FORMVAL_CHECK` (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг `RDS_FORMFLAG_CHECK` (стр. 500).

`RDS_FORMCTRL_COMBOLIST`

Выпадающий список без возможности ручного ввода значения (пример такого поля приведен в §2.7.2). Поле поддерживает следующие команды:



- `RDS_FORMVAL_VALUE` (стр. 525) – установка и чтение номера выбранного в списке варианта или чтение значения поля: функции `rdsGetObjectStr` (стр. 405) и `rdsGetObjectDouble` (стр. 403) возвращают само значение поля, функция `rdsGetObjectInt` (стр. 404) возвращает номер выбранного варианта, все три функции `rdsSetObjectInt` (стр. 407), `rdsSetObjectDouble` (стр. 406) и `rdsSetObjectStr` (стр. 407) устанавливают номер выбранного варианта;

- RDS_FORMVAL_ITEMINDEX (стр. 513) – установка и чтение номера выбранного в списке варианта;
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля;
- RDS_FORMVAL_LIST (стр. 515) – установка списка вариантов (строки списка разделяются символом перевода строки “\n” с кодом 10);
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

RDS_FORMCTRL_DIRDIALOG

Кнопка выбора папки (пример такого поля приведен в §4.2). При нажатии на кнопку открывается стандартный диалог Windows для выбора папки. Пользователь также может ввести имя папки вручную, поддерживаются символические обозначения стандартных путей РДС (стр. 189). Поле поддерживает следующие команды:

Заголовок поля	Имя папки	...
----------------	-----------	-----

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение имени папки;
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля;
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

RDS_FORMCTRL_DISPLAY

Индикация значения без возможности редактирования пользователем (пример такого поля приведен в §2.7.4). Поле поддерживает единственную команду:

Заголовок поля	Значение
----------------	----------

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения поля.

RDS_FORMCTRL_EDIT

Обычное поле для ввода одной строки текста (пример такого поля приведен в §2.7.2). Поле поддерживает следующие команды:

Заголовок поля	Значение
----------------	----------

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения поля;
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля;
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

RDS_FORMCTRL_FONTSELECT

Кнопка выбора шрифта. При нажатии на нее открывается стандартный диалог выбора шрифта Windows. Пример такого поля приведен в §2.10.1. Поле поддерживает следующие команды:

Заголовок поля	...
----------------	-----

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения поля: в качестве значения используется текст описания шрифта, формируемый функцией `rdsStructToFontText` (стр. 279) и разбираемый функцией `rdsFontTextToStruct` (стр. 290);
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля;
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

RDS_FORMCTRL_HOTKEY

Поле ввода для сочетания клавиш (пример такого поля приведен в §2.12.4). Поместив курсор в это поле, пользователь может нажать

Заголовок поля	Ctrl + Alt + F3
----------------	-----------------


произвольное сочетание клавиш, после чего название этого сочетания автоматически отобразится в самом поле. Клавиша Backspace очищает поле. Значением поля является код клавиши и сочетание флагов, указывающих состояние служебных клавиш Ctrl, Alt и Shift. Поле поддерживает следующие команды:

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения кода клавиши;
- RDS_FORMVAL_HKSHIFTS (стр. 512) – установка и чтение состояния клавиш Ctrl, Alt и Shift;
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля;
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

RDS_FORMCTRL_LABEL

Надпись без поля ввода (пример ее использования приведен в §2.10.1). Обычно такая надпись используется в качестве заголовка группы полей ввода, следующих за ней. Надпись поддерживает единственную команду:

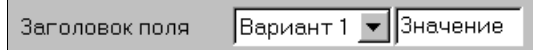
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля.



RDS_FORMCTRL_LISTANDEDIT

Поле ввода, слева от которого размещается выпадающий список с фиксированным набором вариантов. В этом списке могут находиться, например, единицы измерения значения из основного поля ввода, названия параметров, которые можно ввести в основное поле и т.п. Если окно открыто функциями rdsFORMShowModalServ (стр. 504) или rdsFORMShowModalEx (стр. 503), в их функциях обратного вызова можно разрешать или запрещать основное поле ввода в зависимости от варианта, выбранного в выпадающем списке. Поле поддерживает следующие команды:

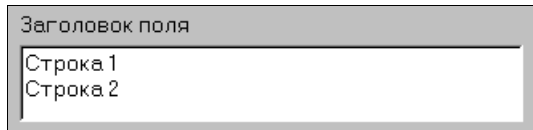
- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения основного поля ввода;
- RDS_FORMVAL_LIST (стр. 515) – установка списка вариантов (строки списка разделяются символом перевода строки “\n” с кодом 10);
- RDS_FORMVAL_AUXLISTITEM (стр. 508) – установка и чтение номера выбранного в выпадающем списке варианта;
- RDS_FORMVAL_AUXLISTWIDTH (стр. 509) – установка ширины выпадающего списка в точках экрана. Параметр width в функции rdsFORMAddEdit (стр. 492) задает ширину только основного поля ввода;
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение всего поля ввода (выпадающего списка вместе с основным полем ввода);
- RDS_FORMVAL_2NDEDITENABLED (стр. 506) – отдельное разрешение/запрещение основного поля ввода;
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).



RDS_FORMCTRL_MULTILINE

Поле для ввода нескольких строк текста (пример такого поля приведен в §2.13.6). Параметр Width функции rdsFORMAddEdit (стр. 492) для этого поля игнорируется – поле занимает всю ширину окна, а его заголовок размещается не слева от поля, а над ним. Поле поддерживает следующие команды:

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения поля;



- RDS_FORMVAL_MLHEIGHT (стр. 516) – установка высоты поля в точках экрана (отрицательное значение высоты автоматически расширит поле до нижней границы окна);
- RDS_FORMVAL_MLRETURNS (стр. 517) – реакция поля на нажатие пользователем клавиши Enter: 1 – при нажатии Enter в поле записывается перевод строки, 0 – нажатие Enter, как обычно, эквивалентно закрытию окна кнопкой “ОК” (поведение поля по умолчанию);
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля;
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

RDS_FORMCTRL_NONVISUAL

Скрытое поле, никак не отображающееся в окне, но позволяющее хранить какие-либо данные (пример такого поля приведен в §2.16.1). Эти данные можно считывать, например, в функции обратного вызова функций открытия окна `rdsFORMShowModalServ` (стр. 504) и `rdsFORMShowModalEx` (стр. 503) – другого способа передать произвольные данные в функцию обратного вызова нет. Поле поддерживает единственную команду:

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения поля.

RDS_FORMCTRL_OPENDIALOG

Кнопка открытия файла (пример приведен в §2.16.2). При нажатии на кнопку появляется стандартный диалог открытия файла Windows.



Пользователь также может ввести имя файла вручную, поддерживаются символические обозначения стандартных путей РДС (стр. 189). Поле поддерживает следующие команды:

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения поля;
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля;
- RDS_FORMVAL_LIST (стр. 515) – установка списка шаблонов имен файлов диалога (см. ниже);
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

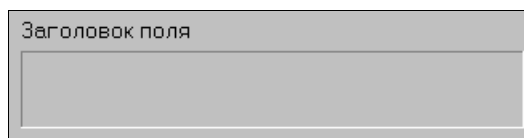
Список шаблонов, устанавливаемый командой RDS_FORMVAL_LIST, имеет следующую структуру:

- одна строка списка содержит один шаблон, строки списка разделяются символом перевода строки “\n” с кодом 10;
- шаблон состоит из названия, видимого пользователю, за которым следует символ вертикальной черты “|” и набор масок имен файлов, разделенных точкой с запятой.

Например, если требуется указать два шаблона, один из которых будет отображать в диалоге только текстовые файлы с расширениями “txt” и “log”, а другой – все файлы, командой RDS_FORMVAL_LIST нужно передать в поле строку “Текстовые файлы|*.txt;*.log\nВсе файлы|*.*”.

RDS_FORMCTRL_PAINTBOX

Область, в которой можно программно рисовать функцией обратного вызова (пример использования приведен в §2.7.3). Для открытия окна с областями программного рисования нужно использовать сервисную функцию `rdsFORMShowModalServ` (стр. 504), только она поддерживает функцию обратного вызова, которая может рисовать в таких областях по событию RDS_FORMSERVEVENT_DRAW (см. стр. 128). Поле поддерживает следующие команды:




- RDS_FORMVAL_PBHEIGHT (стр. 518) – установка высоты поля в точках экрана (отрицательное значение высоты автоматически расширит поле до нижней границы окна);
- RDS_FORMVAL_PBBEVEL (стр. 517) – установка рельефной рамки вокруг области рисования: 1 – рамка есть (поведение по умолчанию), 0 – рамки нет.
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

Это поле не реагирует на действия пользователя, поэтому включение дополнительного разрешающего флага для него не имеет особого смысла. Тем не менее, этот флаг поддерживается и может управляться командой RDS_FORMVAL_CHECK.

RDS_FORMCTRL_RADIOBUTTON

Флаг, связанный с другими флагами этой же вкладки или боковой панели (см. стр. 502). Вместе с дополнительными разрешающими флагами



RDS_FORMFLAG_CHECKRADIO (стр. 501) такие флаги в пределах одной вкладки или боковой панели окна образуют группу, в которой может быть включен только один флаг. При включении какого-либо флага в такой группе все остальные автоматически выключаются. Поле поддерживает следующие команды:

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения флага (0 – флаг выключен, 1 – флаг включен);
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля.

RDS_FORMCTRL_RANGEEDIT

Два поля ввода для задания диапазона значений (пример приведен в §2.10.1). Значения в полях устанавливаются независимо, никаких



внутренних проверок не производится. В оба поля можно вводить как числовые, так и символьные значения. Фактически, RDS_FORMCTRL_RANGEEDIT можно использовать просто как двойное поле ввода. Поле поддерживает следующие команды:

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения первого поля (начала диапазона);
- RDS_FORMVAL_RANGEMAX (стр. 520) – установка и чтение значения второго поля (конца диапазона);
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение всего поля ввода (обоих его полей одновременно);
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

Ширина, указываемая в параметре Width функции функции rdsFORMAddEdit (стр. 492), для поля этого типа задает общую ширину двух полей вместе с разделяющим их многоточием. Ширина левого и правого поля вычисляется автоматически таким образом, чтобы все двойное поле имело ширину Width.

RDS_FORMCTRL_SAVEDIALOG

Кнопка сохранения файла (пример такого поля приведен в §2.13.6). При нажатии на кнопку



появляется стандартный диалог сохранения файла Windows. Пользователь также может ввести имя файла вручную, поддерживаются символические обозначения стандартных путей РДС (стр. 189). Поле поддерживает следующие команды:

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения поля;
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля;

- RDS_FORMVAL_LIST (стр. 515) – установка списка шаблонов имен файлов диалога (см. ниже);
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

Список шаблонов, устанавливаемый командой RDS_FORMVAL_LIST, имеет следующую структуру:

- одна строка списка содержит один шаблон, строки списка разделяются символом перевода строки “\n” с кодом 10;
- шаблон состоит из названия, видимого пользователю, за которым следует символ вертикальной черты “|” и набор масок имен файлов, разделенных точкой с запятой.

Например, если требуется указать два шаблона, один из которых будет отображать в диалоге только текстовые файлы с расширениями “txt” и “log”, а другой – все файлы, командой RDS_FORMVAL_LIST нужно передать в поле строку “Текстовые файлы|*.txt;*.log\nВсе файлы|*.*”.

RDS_FORMCTRL_UPDOWN

Поле ввода со стрелками вверх и вниз, позволяющими изменять значение с фиксированным шагом (пример приведен в §2.10.1). Поле может работать как с целыми, так и с вещественными значениями. Поддерживаются следующие команды:

- RDS_FORMVAL_VALUE (стр. 525) – установка и чтение значения поля;
- RDS_FORMVAL_UPDOWNINC (стр. 521) – установка шага изменения значения поля при нажатии на кнопки со стрелками;
- RDS_FORMVAL_UPDOWNMIN (стр. 524) – установка минимального значения поля, ниже которого его нельзя будет изменить стрелками (эту проверку можно отключить, см. ниже);
- RDS_FORMVAL_UPDOWNMAX (стр. 522) – установка максимального значения поля, выше которого его нельзя будет изменить стрелками (эту проверку можно отключить, см. ниже);
- RDS_FORMVAL_ENABLED (стр. 511) – разрешение/запрещение поля;
- RDS_FORMVAL_CHECK (стр. 510) – управление дополнительным разрешающим флагом, если при создании поля был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

Проверка максимального и минимального значений поля, устанавливаемых командами RDS_FORMVAL_UPDOWNMIN и RDS_FORMVAL_UPDOWNMAX, по умолчанию отключена. После ее включения ее можно снова отключить, передав в качестве минимального или максимального значения либо пустую строку при помощи функции rdsSetObjectStr (стр. 407), либо специальное значение-индикатор математической ошибки (возвращаемое функцией rdsGetHugeDouble, см. стр. 157) при помощи функции rdsSetObjectDouble (стр. 406).

Флаги полей ввода

Флаги поля ввода – это набор констант RDS_FORMFLAG_*, объединенных битовым ИЛИ и передающихся в старших трех байтах параметра Type функции добавления поля ввода rdsFORMAddEdit (стр. 492). На данный момент в РДС поддерживаются следующие флаги полей ввода:

RDS_FORMFLAG_CHECK

Дополнительный разрешающий флаг (“галочка”) слева от заголовка данного поля. Этот флаг не может сочетаться с полями типа RDS_FORMCTRL_CHECKBOX (стр. 495) и RDS_FORMCTRL_RADIOBUTTON (стр. 499). Для программного управления состоянием этого флага используется команда RDS_FORMVAL_CHECK (стр. 510). Фактически, этот флаг позволяет объединить в одном

поле ввода два: например, значение разрешающего флага можно использовать как признак наличия или отсутствия какого-либо параметра, а значение самого поля ввода – как значение этого параметра. Пример использования флага приведен в §2.15.3.

RDS_FORMFLAG_CHECKRADIO

Разрешающий флаг слева от заголовка поля связан с другими флагами вкладки (только при установленном флаге RDS_FORMFLAG_CHECK).

<input checked="" type="radio"/> Заголовок поля	<input type="text" value="Значение"/>
---	---------------------------------------

Вместе с полями ввода RDS_FORMCTRL_RADIOBUTTON (стр. 499) такие флаги в пределах одной вкладки или боковой панели окна образуют группу, в которой может быть включен только один флаг. При включении какого-либо флага в такой группе все остальные автоматически выключаются.

Если для данного поля не установлен флаг RDS_FORMFLAG_CHECK, флаг RDS_FORMFLAG_CHECKRADIO игнорируется.

RDS_FORMFLAG_DISABLED

Поле ввода запрещено и отображается серым цветом. Позже оно может быть разрешено командой RDS_FORMVAL_ENABLED (стр. 511).

Заголовок поля	<input type="text" value="Значение"/>
----------------	---------------------------------------

RDS_FORMFLAG_LCENTER

Заголовок поля ввода выровнен по центру (только если не установлен флаг RDS_FORMFLAG_CHECK). По умолчанию все заголовки выравниваются по левому краю.

Заголовок слева	<input type="text" value="Значение"/>
Заголовок по центру	<input type="text" value="Значение"/>

RDS_FORMFLAG_LINE

Рельефная горизонтальная линия под данным полем ввода (пример использования флага приведен в §2.10.1). Обычно такие линии используют для визуального разделения групп полей ввода, связанных по смыслу.

Заголовок поля	<input type="text" value="Значение"/>
----------------	---------------------------------------

RDS_FORMFLAG_LRIGHT

Заголовок поля ввода выровнен по правому краю (только если не установлен флаг RDS_FORMFLAG_CHECK). По умолчанию все заголовки выравниваются по левому краю.

Заголовок слева	<input type="text" value="Значение"/>
Заголовок справа	<input type="text" value="Значение"/>

A.5.28.4. rdsFORMAddTab – добавить вкладку

Функция rdsFORMAddTab добавляет в объект для работы с модальным окном новую вкладку.

```
void RDSCALL rdsFORMAddTab(
    RDS_HOBJECT Win,      // Объект
    int TabId,            // Идентификатор вкладки
    LPSTR Caption         // Заголовок вкладки
);
```

Тип указателя на эту функцию:

RDS_VH○IS

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

TabId

Произвольный целый идентификатор, который будет присвоен добавляемой вкладке. Этот идентификатор будет использоваться для добавления полей ввода на вкладку функцией `rdsFORMAddEdit` (стр. 492).

Caption

Указатель на строку с заголовком вкладки.

Примечания:

Эта функция добавляет в объект `Win` новую вкладку с идентификатором `TabId`. Вкладки можно добавлять в окно только в том случае, если при создании объекта вызовом `rdsFORMCreate` в параметре `Tabbed` было передано `TRUE`. Вкладки добавляются слева направо, то есть каждый следующий вызов `rdsFORMAddTab` добавляет вкладку справа от последней добавленной. Идентификаторы вкладкам можно присваивать произвольно – главное, чтобы в одном окне не было двух вкладок с одинаковыми идентификаторами, и чтобы идентификатор вкладки не совпадал с идентификатором боковой панели окна, указанном в вызове `rdsFORMEnableSidePanel` (стр. 502).

Пример использования функции `rdsFORMAddTab` приведен в §2.10.1.

См. также:

`rdsFORMCreate` (стр. 491), `rdsFORMAddEdit` (стр. 492),
`rdsFORMEnableSidePanel` (стр. 502).

A.5.28.5. `rdsFORMEnableSidePanel` – включить боковую панель

Функция `rdsFORMEnableSidePanel` включает в окне, с которым работает объект, дополнительную панель слева от основных полей ввода.

```
void RDSCALL rdsFORMEnableSidePanel (  
    RDS_HOBJECT Win,           // Объект  
    int TabId,                 // Идентификатор панели  
    int Width                  // Ширина панели или -1  
);
```

Тип указателя на эту функцию:

`RDS_VHObj`

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

TabId

Произвольный целый идентификатор, который будет присвоен панели. Этот идентификатор будет использоваться вместо идентификатора вкладки для добавления полей ввода на эту панель функцией `rdsFORMAddEdit` (стр. 492).

Width

Ширина панели в точках экрана или `-1` для автоматического вычисления ширины (ширина будет подобрана так, чтобы на панели умещались все добавленные на нее поля ввода).

Примечания:

Эта функция включает в объекте Win боковую панель с идентификатором TabId. Боковая панель может быть только одна. Идентификатор панели не должен совпадать ни с одним из идентификаторов вкладок окна, если они есть.

Пример использования функции rdsFORMEnableSidePanel приведен в §2.7.3.

См. также:

rdsFORMCreate (стр. 491), rdsFORMAddEdit (стр. 492),
rdsFORMAddTab (стр. 501).

A.5.28.6. rdsFORMShowModalEx – открыть окно с функцией обратного вызова

Функция rdsFORMShowModalEx открывает модальное окно, описанное указанным вспомогательным объектом, и не возвращает управления вызвавшей программе до закрытия окна. При изменении пользователем значения любого поля вызывается указанная в параметрах функция.

```
BOOL RDSCALL rdsFORMShowModalEx(  
    RDS_НОБЪЕКТ Win,          // Объект  
    RDS_ВНo CheckFunc        // Функция реакции на изменения  
);
```

Тип указателя на эту функцию:

RDS_ВНoCb3

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

CheckFunc

Указатель на функцию, которую нужно вызывать при любом изменении полей ввода, или NULL, если такая функция не нужна. Функция должна иметь следующий вид:

```
void RDSCALL имя_функции(RDS_НОБЪЕКТ win);
```

В параметре win этой функции передается идентификатор вспомогательного объекта, поле ввода в окне которого изменилось.

Возвращаемое значение:

TRUE – пользователь закрыл окно кнопкой “ОК”, FALSE – кнопкой “Отмена”.

Примечания:

Эта функция открывает окно, описанное объектом Win, и ждет, пока пользователь не закроет его. Пользователь может изменить любые поля, ввод в которые не запрещен. Если в параметре CheckFunc передан указатель на функцию обратного вызова, при изменении любого поля эта функция будет автоматически вызываться. В функцию обратного вызова передается идентификатор объекта-окна, и в ней можно, например, управлять разрешенностью каких-либо полей ввода в зависимости от значений, введенных в другие. Внутри функции обратного вызова нельзя определить, какое именно поле ввода изменилось – если эта информация требуется для организации желаемого пользовательского интерфейса, вместо функции rdsFORMShowModalEx следует использовать rdsFORMShowModalServ (стр. 504), поскольку ее функция обратного вызова обладает расширенными возможностями.

Пример использования функции rdsFORMShowModalEx приведен в §2.7.2.

См. также:

rdsFORMCreate (стр. 491), rdsFORMAddEdit (стр. 492),
rdsFORMAddTab (стр. 501), rdsFORMShowModalServ (стр. 504),
RDS_FORM_SHOWMODAL (стр. 506).

A.5.28.7. rdsFORMShowModalServ – открыть окно с расширенной функцией обратного вызова

Функция rdsFORMShowModalServ открывает модальное окно, описанное указанным вспомогательным объектом, и не возвращает управления вызвавшей программе до закрытия окна. В ответ на различные действия, выполняемые пользователем в окне, вызывается указанная в параметрах функция.

```
BOOL RDSCALL rdsFORMShowModalServ(  
    RDS_HOBJECT Win,           // Объект  
    RDS_VHOFsfid ServFunc    // Функция реакции  
);
```

Тип указателя на эту функцию:

RDS_BHOCb6

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

ServFunc

Указатель на функцию, которую нужно вызывать в ответ на действия в окне, или NULL, если такая функция не нужна. Функция должна иметь следующий вид:

```
void RDSCALL имя_функции(RDS_HOBJECT win,  
                          RDS_PFORMSERVFUNCDATA pdata);
```

В параметре win этой функции передается идентификатор вспомогательного объекта, которому принадлежит окно, а в параметре pdata – указатель на структуру RDS_FORMSERVFUNCDATA (стр. 128), описывающую произошедшее событие.

Возвращаемое значение:

TRUE – пользователь закрыл окно кнопкой “ОК”, FALSE – кнопкой “Отмена”.

Примечания:

Эта функция открывает окно, описанное объектом Win, и ждет, пока пользователь не закроет его. Пользователь может изменить любые поля, ввод в которые не запрещен. Если в параметре ServFunc передан указатель на функцию обратного вызова, при различных событиях в окне эта функция будет автоматически вызываться. В функцию обратного вызова передается идентификатор объекта-окна и указатель на структуру RDS_FORMSERVFUNCDATA, в полях которой содержится информация о произошедшем событии, идентификатор поля ввода, с которым связано это событие и другая необходимая информация. На данный момент поддерживается реакция на три события: изменение значения поля ввода, нажатие кнопки (для полей типа RDS_FORMCTRL_BUTTON) и программное рисование (для полей типа RDS_FORMCTRL_PAINTBOX). Реагировать на нажатия кнопок и программно рисовать изображения в окне можно, только если окно открыто функцией rdsFORMShowModalServ, поскольку только в ее функции обратного вызова предусмотрена реакция на такие события.

Пример использования функции rdsFORMShowModalServ приведен в §2.7.3.

См. также:

`rdsFORMCreate` (стр. 491), `RDS_FORMSERVFUNCDATA` (стр. 128),
`rdsFORMAddEdit` (стр. 492), `rdsFORMAddTab` (стр. 501),
`rdsFORMShowModalEx` (стр. 503), `RDS_FORM_SHOWMODAL` (стр. 506).

А.5.28.8. Команда `RDS_FORM_CLEAR` – очистить объект

Команда `RDS_FORM_CLEAR` удаляет из объекта все поля ввода, вкладки и боковую панель.

Вызов команды:

```
rdsCommandObject (Win, RDS_FORM_CLEAR) ;  
или  
rdsCommandObjectEx (Win, RDS_FORM_CLEAR, 0, NULL) ;
```

Параметр:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

См. также:

`rdsFORMCreate` (стр. 491), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400), `rdsFORMAddEdit` (стр. 492),
`rdsFORMAddTab` (стр. 501), `rdsFORMEnableSidePanel` (стр. 502).

А.5.28.9. Команда `RDS_FORM_INVALIDATE` – обновить окно

Команда `RDS_FORM_INVALIDATE` принудительно перерисовывает открытое модальное окно.

Вызов команды:

```
rdsCommandObject (Win, RDS_FORM_INVALIDATE) ;  
или  
rdsCommandObjectEx (Win, RDS_FORM_INVALIDATE, 0, NULL) ;
```

Параметр:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

Примечания:

Эта команда используется в тех случаях, когда модальное окно содержит поля типа `RDS_FORMCTRL_PAINTBOX`, на которых функция обратного вызова (см. `rdsFORMShowModalServ`, стр. 504) программно рисует какие-либо изображения. Если изображение связано со значениями полей ввода, при их изменении пользователем функция обратного вызова может дать объекту команду `RDS_FORM_INVALIDATE` для обновления изображения.

Пример использования этой команды приведен в §2.7.3.

См. также:

`rdsFORMCreate` (стр. 491), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400), `rdsFORMAddEdit` (стр. 492),
`rdsFORMShowModalServ` (стр. 504).

А.5.28.10. Команда RDS_FORM_SHOWMODAL – открыть окно

Команда RDS_FORM_SHOWMODAL открывает модальное окно, описанное указанным вспомогательным объектом, и не возвращает управления вызвавшей программе до закрытия окна.

Вызов команды:

```
BOOL bOk=rdsCommandObject (Win,RDS_FORM_SHOWMODAL) ;  
или  
BOOL bOk=rdsCommandObjectEx (Win,RDS_FORM_SHOWMODAL, 0, NULL) ;
```

Параметр и результат:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

bOk

TRUE – пользователь закрыл окно кнопкой “ОК”, FALSE – кнопкой “Отмена”.

Примечания:

В отличие от сервисных функций rdsFORMShowModalEx (стр. 503) и rdsFORMShowModalServ (стр. 504), в этой команде не поддерживается функция обратного вызова, поэтому до закрытия окна вызвавшая программа никак не сможет отреагировать на сделанные пользователем изменения.

См. также:

rdsFORMCreate (стр. 491), rdsCommandObject (стр. 400),
rdsCommandObjectEx (стр. 400), rdsFORMShowModalEx (стр. 503),
rdsFORMShowModalServ (стр. 504).

А.5.28.11. Команда RDS_FORMVAL_2NDEDITENABLED – разрешение основного поля в двойном поле ввода со списком

Команда RDS_FORMVAL_2NDEDITENABLED разрешает или запрещает основное поле ввода в двойном поле типа RDS_FORMCTRL_LISTANDEDIT (стр. 497).

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода  
int iEnable=... // 1 - разрешить, 0 - запретить  
rdsSetObjectInt (Win,iCtrlId,  
RDS_FORMVAL_2NDEDITENABLED,iEnable) ;  
или  
int iCtrlId=... // Идентификатор поля ввода  
double dEnable=... // >0.0 - разрешить, <=0.0 - запретить  
rdsSetObjectDouble (Win,iCtrlId,  
RDS_FORMVAL_2NDEDITENABLED,dEnable) ;
```

Вызов команды для чтения:

```
int iCtrlId=... // Идентификатор поля ввода  
int iEnable=rdsGetObjectInt (Win,iCtrlId,  
RDS_FORMVAL_2NDEDITENABLED) ;  
или
```

```
int iCtrlId=... // Идентификатор поля ввода
double dEnable=rdsGetObjectDouble(Win,iCtrlId,
                                   RDS_FORMVAL_2NDEDITENABLED);
```

Параметры и результат:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

iEnable

Целое число, указывающее на запрещение (нулевое значение) и разрешение (отличное от нуля значение) основного поля ввода.

dEnable

Вещественное число, указывающее на запрещение (нулевое или отрицательное значение) и разрешение (положительное значение) основного поля ввода.

Примечания:

Эта команда управляет разрешением основного (правого) поля ввода в двойном поле, состоящем из собственно поля ввода справа и выпадающего списка слева. На нее реагируют только поля ввода типа `RDS_FORMCTRL_LISTANDEDIT`. Вместе с командой `RDS_FORMVAL_ENABLED` (стр. 511) эта команда позволяет оперативно управлять состоянием компонентов двойного поля:

<i>Команда</i> <i>RDS_FORMVAL_ENABLED</i>	<i>Команда</i> <i>RDS_FORMVAL_2NDEDITENABLED</i>	<i>Состояние поля</i>
разрешено	разрешено	И выпадающий список, и поле ввода разрешены
разрешено	запрещено	Выпадающий список разрешен, поле ввода запрещено
запрещено	не важно	И выпадающий список, и поле ввода запрещены

Чаще всего эту команду используют в функции обратного вызова функций открытия окна `rdsFORMShowModalEx` (стр. 503) и `rdsFORMShowModalServ` (стр. 504) для запрещения поля ввода при выборе отдельных вариантов в выпадающем списке.

Для передачи команды полю ввода можно использовать как целые функции `rdsSetObjectInt` (стр. 407) и `rdsGetObjectInt` (стр. 404), так и вещественные `rdsSetObjectDouble` (стр. 406) и `rdsGetObjectDouble` (стр. 403). Чаще всего используются именно целые функции – принимаемое и возвращаемое ими целое число трактуется по обычным правилам языка C: нулевое значение считается логической ложью (поле запрещено), все остальные – логической истиной (поле разрешено).

См. также:

`RDS_FORMCTRL_LISTANDEDIT` (стр. 497), `rdsFORMCreate` (стр. 491), `rdsFORMAddEdit` (стр. 492), `rdsSetObjectInt` (стр. 407), `rdsGetObjectInt` (стр. 404), `rdsSetObjectDouble` (стр. 406), `rdsGetObjectDouble` (стр. 403), `RDS_FORMVAL_ENABLED` (стр. 511), `RDS_FORMVAL_CHECK` (стр. 510).

А.5.28.12. Команда RDS_FORMVAL_AUXLISTITEM – номер варианта выпадающего списка в двойном поле ввода со списком

Команда RDS_FORMVAL_AUXLISTITEM устанавливает или возвращает номер варианта, выбранного в выпадающем списке двойного поля ввода типа RDS_FORMCTRL_LISTANDEDIT (стр. 497).

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода
int iItemNum=... // Номер выбранного пункта
rdsSetObjectInt(Win, iCtrlId,
                RDS_FORMVAL_AUXLISTITEM, iItemNum);

или

int iCtrlId=... // Идентификатор поля ввода
double dItemNum=... // Номер выбранного пункта
rdsSetObjectDouble(Win, iCtrlId,
                  RDS_FORMVAL_AUXLISTITEM, dItemNum);

или

int iCtrlId=... // Идентификатор поля ввода
char *sItemNum=... // Номер выбранного пункта в виде строки
rdsSetObjectStr(Win, iCtrlId,
                RDS_FORMVAL_AUXLISTITEM, sItemNum);
```

Вызов команды для чтения:

```
int iCtrlId=... // Идентификатор поля ввода
int iItemNum=rdsGetObjectInt(Win, iCtrlId,
                             RDS_FORMVAL_AUXLISTITEM);

или

int iCtrlId=... // Идентификатор поля ввода
double dItemNum=rdsGetObjectDouble(Win, iCtrlId,
                                   RDS_FORMVAL_AUXLISTITEM);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

iItemNum

Целый номер выбранного в выпадающем списке пункта (пункты нумеруются начиная с нуля). Значение –1 указывает на то, что в списке не выбран ни один пункт.

dItemNum

Номер выбранного в выпадающем списке пункта (пункты нумеруются начиная с нуля) в виде вещественного (double) числа. Значение –1 указывает на то, что в списке не выбран ни один пункт.

sItemNum

Указатель на строку, в которой записан номер выбранного в выпадающем списке пункта (пункты нумеруются начиная с нуля). Строка “–1” указывает на то, что в списке не выбран ни один пункт.

Примечания:

Эта команда управляет номером пункта, выбранного в выпадающем списке в составе двойного поля ввода типа RDS_FORMCTRL_LISTANDEDIT.

Для передачи команды полю ввода можно использовать как целые функции rdsSetObjectInt (стр. 407) и rdsGetObjectInt (стр. 404), так и вещественные rdsSetObjectDouble (стр. 406) и rdsGetObjectDouble (стр. 403). Поскольку номер пункта – целое число, чаще всего используются именно целые функции. Для установки номера выбранного пункта можно также использовать строковую функцию rdsSetObjectStr (стр. 407), при этом строка будет преобразована в целое число по правилам функции rdsAtol (стр. 184). Чтение номера выбранного пункта в виде строки не предусмотрено.

См. также:

RDS_FORMCTRL_LISTANDEDIT (стр. 497), rdsFORMCreate (стр. 491),
rdsFORMAddEdit (стр. 492), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), rdsSetObjectDouble (стр. 406),
rdsGetObjectDouble (стр. 403), rdsSetObjectStr (стр. 407).

A.5.28.13. Команда RDS_FORMVAL_AUXLISTWIDTH – ширина выпадающего списка в двойном поле ввода со списком

Команда RDS_FORMVAL_AUXLISTWIDTH устанавливает ширину в точках экрана для выпадающего списка в составе двойного поля ввода типа RDS_FORMCTRL_LISTANDEDIT (стр. 497).

Вызов команды:

```
int iCtrlId=... // Идентификатор поля ввода
int iWidth=...  // Ширина выпадающего списка
rdsSetObjectInt(Win,iCtrlId,
                RDS_FORMVAL_AUXLISTWIDTH,iWidth);

или

int iCtrlId=... // Идентификатор поля ввода
double dWidth=... // Ширина выпадающего списка
rdsSetObjectDouble(Win,iCtrlId,
                  RDS_FORMVAL_AUXLISTWIDTH,dWidth);

или

int iCtrlId=... // Идентификатор поля ввода
char *sWidth=... // Ширина выпадающего списка в виде строки
rdsSetObjectStr(Win,iCtrlId,
                RDS_FORMVAL_AUXLISTWIDTH,sWidth);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

iWidth

Ширина выпадающего списка в точках экрана.

dWidth

Ширина выпадающего списка в точках экрана в виде вещественного числа.

sWidth

Указатель на строку, в которой записана ширина выпадающего списка в точках экрана.

Примечания:

Эта команда задает ширину выпадающего списка в составе двойного поля ввода типа RDS_FORMCTRL_LISTANDEDIT. Ее имеет смысл передавать полю ввода только до открытия окна, поскольку в момент открытия окна размеры всех объектов в нем фиксируются до тех пор, пока окно не будет закрыто.

Для передачи команды полю ввода можно использовать любую из функций rdsSetObjectInt (стр. 407), rdsSetObjectDouble (стр. 406) и rdsSetObjectStr (стр. 407). Чаще всего используется целая функция rdsSetObjectInt, поскольку ширина списка – целое число. При использовании rdsSetObjectStr переданная строка будет преобразована в целое число по правилам функции rdsAtoI (стр. 184).

См. также:

RDS_FORMCTRL_LISTANDEDIT (стр. 497), rdsFORMCreate (стр. 491),
rdsFORMAddEdit (стр. 492), rdsSetObjectInt (стр. 407),
rdsSetObjectDouble (стр. 406), rdsSetObjectStr (стр. 407).

A.5.28.14. Команда RDS_FORMVAL_CHECK – управление разрешающим флагом поля ввода

Команда RDS_FORMVAL_CHECK служит для включения или выключения дополнительного разрешающего флага поля ввода, а также для получения значения этого флага. Команда выполняется, только если при создании поля ввода был указан флаг RDS_FORMFLAG_CHECK (стр. 500).

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода
int iEnable=... // 1 - разрешить, 0 - запретить
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_CHECK,iEnable);

или

int iCtrlId=... // Идентификатор поля ввода
double dEnable=... // >0.0 - разрешить, <=0.0 - запретить
rdsSetObjectDouble(Win,iCtrlId,RDS_FORMVAL_CHECK,dEnable);
```

Вызов команды для чтения:

```
int iCtrlId=... // Идентификатор поля ввода
int iEnable=rdsGetObjectInt(Win,iCtrlId,RDS_FORMVAL_CHECK);

или

int iCtrlId=... // Идентификатор поля ввода
double dEnable=rdsGetObjectDouble(Win,iCtrlId,RDS_FORMVAL_CHECK);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

iEnable

Целое число, указывающее на включение (отличное от нуля значение) или выключение (нулевое значение) разрешающего флага.

dEnable

Вещественное число, указывающее на включение (положительное значение) или выключение (нулевое или отрицательное значение) разрешающего флага.

Примечания:

Эта команда управляет разрешающим флагом поля ввода. При включенном флаге поле будет разрешено и пользователь сможет вводить в него данные, при выключенном – запрещено и заблокировано (при этом оно обычно изображается серым цветом).

Для передачи команды полю ввода можно использовать как целые функции rdsSetObjectInt (стр. 407) и rdsGetObjectInt (стр. 404), так и вещественные rdsSetObjectDouble (стр. 406) и rdsGetObjectDouble (стр. 403). Чаще всего используются именно целые функции – принимаемое и возвращаемое ими целое число трактуется по обычным правилам языка C: нулевое значение считается логической ложью (флаг выключен, поле запрещено), все остальные – логической истиной (флаг включен, поле разрешено).

Не следует путать команду управления дополнительным флагом разрешения поля RDS_FORMVAL_CHECK с командой разрешения всего поля RDS_FORMVAL_ENABLED (стр. 511): первая включает и выключает дополнительный флаг, разрешая или запрещая поле (разрешенность самого флага при этом не меняется), а вторая разрешает или запрещает все поле целиком вместе с дополнительным флагом.

Пример использования команды RDS_FORMVAL_CHECK приведен в §2.15.3.

См. также:

RDS_FORMFLAG_CHECK (стр. 500), rdsFORMCreate (стр. 491),
rdsFORMAddEdit (стр. 492), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), rdsSetObjectDouble (стр. 406),
rdsGetObjectDouble (стр. 403), RDS_FORMVAL_ENABLED (стр. 511).

A.5.28.15. Команда RDS_FORMVAL_ENABLED – разрешение и запрещение всего поля ввода

Команда RDS_FORMVAL_ENABLED разрешает или запрещает все поле ввода целиком, со всеми его компонентами и дополнительным флагом, если он есть. С помощью этой команда можно также узнать, разрешено ли указанное поле в данный момент.

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода
int iEnable=... // 1 - разрешить, 0 - запретить
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_ENABLED,iEnable);
```

или

```
int iCtrlId=... // Идентификатор поля ввода
double dEnable=... // >0.0 - разрешить, <=0.0 - запретить
rdsSetObjectDouble(Win,iCtrlId,RDS_FORMVAL_ENABLED,dEnable);
```

Вызов команды для чтения:

```
int iCtrlId=... // Идентификатор поля ввода
int iEnable=rdsGetObjectInt(Win,iCtrlId,RDS_FORMVAL_ENABLED);

или

int iCtrlId=... // Идентификатор поля ввода
double dEnable=rdsGetObjectDouble(Win,iCtrlId,RDS_FORMVAL_ENABLED);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

iEnable

Целое число, указывающее на разрешение (отличное от нуля значение) или запрещение (нулевое значение) поля.

dEnable

Вещественное число, указывающее на разрешение (положительное значение) или запрещение (нулевое или отрицательное значение) поля.

Примечания:

Эта команда управляет разрешением всего поля ввода. В разрешенное поле пользователь может вводить данные, запрещенное поле заблокировано (при этом оно обычно изображается серым цветом).

Для передачи команды полю ввода можно использовать как целые функции `rdsSetObjectInt` (стр. 407) и `rdsGetObjectInt` (стр. 404), так и вещественные `rdsSetObjectDouble` (стр. 406) и `rdsGetObjectDouble` (стр. 403). Чаще всего используются именно целые функции – принимаемое и возвращаемое ими целое число трактуется по обычным правилам языка C: нулевое значение считается логической ложью (поле запрещено), все остальные – логической истиной (поле разрешено).

Чаще всего эта команда используется в функции обратного вызова функций открытия окна `rdsFORMShowModalEx` (стр. 503) и `rdsFORMShowModalServ` (стр. 504) для разрешения и запрещения одних поля ввода в зависимости от значений, введенных в другие. Пример ее использования приведен в §2.7.2.

См. также:

`rdsFORMCreate` (стр. 491), `rdsFORMAddEdit` (стр. 492),
`rdsSetObjectInt` (стр. 407), `rdsGetObjectInt` (стр. 404),
`rdsSetObjectDouble` (стр. 406), `rdsGetObjectDouble` (стр. 403).

A.5.28.16. Команда RDS_FORMVAL_HKSHIFTS – состояние Ctrl, Alt и Shift в поле ввода кода клавиши

Команда `RDS_FORMVAL_HKSHIFTS` устанавливает или считывает флаги состояния служебных клавиш Ctrl, Alt и Shift в поле для ввода кода клавиши `RDS_FORMCTRL_HOTKEY` (стр. 496).

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода
int iShiftFlags=... // Состояние клавиш (флаги RDS_K*)
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_HKSHIFTS,iShiftFlags);
```

Вызов команды для чтения:

```
int iCtrlId=... // Идентификатор поля ввода
int iShiftFlags=rdsGetObjectInt(Win,iCtrlId,RDS_FORMVAL_HKSHIFTS);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

iShiftFlags

Набор битовых флагов `RDS_KSHIFT`, `RDS_KALT` и `RDS_KCTRL` (см. стр. 61), указывающих на то, какие из клавиш `Ctrl`, `Alt` и `Shift` нажаты вместе с основной клавишей.

Примечания:

Эта команда может применяться только к полю ввода кода клавиши типа `RDS_FORMCTRL_HOTKEY`. Пример ее использования приведен в §2.12.4.

См. также:

`RDS_FORMCTRL_HOTKEY` (стр. 496), `rdsFORMCreate` (стр. 491),
`rdsFORMAddEdit` (стр. 492), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404).

A.5.28.17. Команда `RDS_FORMVAL_ITEMINDEX` – номер варианта, выбранного в выпадающем списке с фиксированными вариантами

Команда `RDS_FORMVAL_ITEMINDEX` устанавливает или возвращает номер варианта, выбранного в фиксированном выпадающем списке типа `RDS_FORMCTRL_COMBOLIST` (стр. 495). Для выпадающего списка типа `RDS_FORMCTRL_COMBOEDIT` (стр. 495), в котором пользователь может не только выбирать вариант из списка, но и вводить значение с клавиатуры, эту команду использовать нельзя, поскольку, в общем случае, значение поля ввода может не совпадать ни с одним из вариантов списка. Для дополнительного списка в двойном поле `RDS_FORMCTRL_LISTANDEDIT` (стр. 497) эта команда тоже не используется, поскольку для получения и установки номера варианта в этом списке предусмотрена специальная команда `RDS_FORMVAL_AUXLISTITEM` (стр. 508).

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода
int iItemNum=... // Номер выбранного пункта
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_ITEMINDEX,iItemNum);
```

или

```
int iCtrlId=... // Идентификатор поля ввода
double dItemNum=... // Номер выбранного пункта
rdsSetObjectDouble(Win,iCtrlId,RDS_FORMVAL_ITEMINDEX,dItemNum);
```

```

или

int iCtrlId=... // Идентификатор поля ввода
char *sItemNum=... // Номер выбранного пункта в виде строки
rdsSetObjectStr(Win,iCtrlId,RDS_FORMVAL_ITEMINDEX,sItemNum);

```

Вызов команды для чтения:

```

int iCtrlId=... // Идентификатор поля ввода
int iItemNum=rdsGetObjectInt(Win,iCtrlId,RDS_FORMVAL_ITEMINDEX);

или

int iCtrlId=... // Идентификатор поля ввода
double dItemNum=rdsGetObjectDouble(Win,iCtrlId,
                                   RDS_FORMVAL_ITEMINDEX);

или

int iCtrlId=... // Идентификатор поля ввода
char *sItemNum=rdsGetObjectStr(Win,iCtrlId,RDS_FORMVAL_ITEMINDEX);

```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

iItemNum

Целый номер выбранного в выпадающем списке пункта (пункты нумеруются начиная с нуля). Значение `-1` указывает на то, что в списке не выбран ни один пункт.

dItemNum

Номер выбранного в выпадающем списке пункта (пункты нумеруются начиная с нуля) в виде вещественного (`double`) числа. Значение `-1` указывает на то, что в списке не выбран ни один пункт.

sItemNum

Указатель на строку, в которой записан номер выбранного в выпадающем списке пункта (пункты нумеруются начиная с нуля). Строка `"-1"` указывает на то, что в списке не выбран ни один пункт. При получении номера пункта функцией `rdsGetObjectStr` в данном случае возвращается указатель на строку во внутренней памяти объекта Win, этот указатель будет действителен до тех пор, пока выбранный пункт не изменится.

Примечания:

Эта команда управляет номером пункта, выбранного в выпадающем списке поля ввода типа `RDS_FORMCTRL_COMBOLIST`.

Для передачи команды полю ввода можно использовать целые функции `rdsSetObjectInt` (стр. 407) и `rdsGetObjectInt` (стр. 404), вещественные `rdsSetObjectDouble` (стр. 406) и `rdsGetObjectDouble` (стр. 403), а также строковые `rdsSetObjectStr` (стр. 407) и `rdsGetObjectStr` (стр. 405). При использовании строковых функций строка преобразуется в целое число по правилам функции `rdsAtol` (стр. 184). Пример использования команды приведен в §2.13.6.

См. также:

RDS_FORMCTRL_COMBOLIST (стр. 495), rdsFORMCreate (стр. 491),
rdsFORMAddEdit (стр. 492), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), rdsSetObjectDouble (стр. 406),
rdsGetObjectDouble (стр. 403), rdsSetObjectStr (стр. 407),
rdsGetObjectStr (стр. 405).

A.5.28.18. Команда RDS_FORMVAL_LIST – установка списка вариантов

Команда RDS_FORMVAL_LIST служит для установки списка вариантов выпадающего списка в полях ввода типа RDS_FORMCTRL_COMBOLIST (стр. 495), RDS_FORMCTRL_COMBOEDIT (стр. 495) и RDS_FORMCTRL_LISTANDEDIT (стр. 497), а также для установки списка шаблонов имен файлов для кнопок вызова диалогов открытия и закрытия файла RDS_FORMCTRL_OPENDIALOG (стр. 498) и RDS_FORMCTRL_SAVEDIALOG (стр. 499) соответственно.

Вызов команды:

```
int iCtrlId=... // Идентификатор поля ввода
char *sList=... // Строка списка
rdsSetObjectStr(Win,iCtrlId,RDS_FORMVAL_LIST,sList);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

sList

Указатель на строку, содержащую список вариантов или список шаблонов имен файлов.

Примечания:

Эта команда передает список строк в поля ввода, которым такой список необходим для работы. В зависимости от типа поля этот список устроен по-разному, но его строки всегда разделяются символом перевода строки “\n” с кодом 10. Для полей ввода с выпадающими списками (RDS_FORMCTRL_COMBOLIST, RDS_FORMCTRL_COMBOEDIT, RDS_FORMCTRL_LISTANDEDIT) каждая строка списка содержит один вариант: например, для выпадающего списка с вариантами “Ручной”, “Полуавтоматический” и “Автоматический” передается строка “Ручной\nПолуавтоматический\nАвтоматический”. Для полей, которым требуются шаблоны имен файлов, каждая строка списка содержит один шаблон, который состоит из названия, видимого пользователю, символа вертикальной черты “|” и набора масок имен файлов, разделенных точкой с запятой. Например, если требуется указать два шаблона, один из которых будет отображать в диалоге только текстовые файлы с расширениями “txt” и “log”, а другой – все файлы, в поле ввода передается строка “Текстовые файлы|*.txt;*.log\nВсе файлы|*.*”.

Примеры использования команды RDS_FORMVAL_LIST приведены в §2.7.2 (для поля с выпадающим списком) и в §2.13.6 (для задания шаблонов имен файлов).

См. также:

RDS_FORMCTRL_COMBOLIST (стр. 495), RDS_FORMCTRL_COMBOEDIT (стр. 495),
RDS_FORMCTRL_LISTANDEDIT (стр. 497),
RDS_FORMCTRL_OPENDIALOG (стр. 498), RDS_FORMCTRL_SAVEDIALOG (стр. 499),
rdsFORMCreate (стр. 491), rdsFORMAddEdit (стр. 492),
rdsSetObjectStr (стр. 407).

А.5.28.19. Команда RDS_FORMVAL_MLHEIGHT – высота многострочного поля ввода

Команда RDS_FORMVAL_MLHEIGHT устанавливает высоту в точках экрана для поля ввода нескольких строк типа RDS_FORMCTRL_MULTILINE (стр. 497).

Вызов команды:

```
int iCtrlId=... // Идентификатор поля ввода
int iHeight=... // Высота поля
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_MLHEIGHT,iHeight);

или

int iCtrlId=... // Идентификатор поля ввода
double dHeight=... // Высота поля
rdsSetObjectDouble(Win,iCtrlId,RDS_FORMVAL_MLHEIGHT,dHeight);

или

int iCtrlId=... // Идентификатор поля ввода
char *sHeight=... // Высота поля в виде строки
rdsSetObjectStr(Win,iCtrlId,RDS_FORMVAL_MLHEIGHT,sHeight);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

iHeight

Высота поля в точках экрана или –1, если нужно, чтобы поле заняло по высоте всю оставшуюся часть окна.

dHeight

Высота поля в точках экрана в виде вещественного числа или –1.0, если нужно, чтобы поле заняло по высоте всю оставшуюся часть окна.

sHeight

Указатель на строку, в которой записана высота поля в точках экрана. Строка “–1” указывает на то, что поле должно занять по высоте всю оставшуюся часть окна.

Примечания:

Эта команда задает высоту поля ввода типа RDS_FORMCTRL_MULTILINE. Высоту имеет смысл передавать полю ввода только до открытия окна, поскольку в момент его открытия размеры всех объектов в нем фиксируются до тех пор, пока окно не будет закрыто.

Для передачи команды полю ввода можно использовать любую из функций rdsSetObjectInt (стр. 407), rdsSetObjectDouble (стр. 406) и rdsSetObjectStr

(стр. 407). Чаще всего используется целая функция `rdsSetObjectInt`, поскольку высота поля – целое число. При использовании `rdsSetObjectStr` переданная строка будет преобразована в целое число по правилам функции `rdsAtoI` (стр. 184).

Пример использования команды `RDS_FORMVAL_MLHEIGHT` приведен в §2.13.6.

См. также:

`RDS_FORMCTRL_MULTILINE` (стр. 497), `rdsFORMCreate` (стр. 491),
`rdsFORMAddEdit` (стр. 492), `rdsSetObjectInt` (стр. 407),
`rdsSetObjectDouble` (стр. 406), `rdsSetObjectStr` (стр. 407).

A.5.28.20. Команда `RDS_FORMVAL_MLRETURNS` – управление реакцией многострочного поля ввода на клавишу Enter

Команда `RDS_FORMVAL_MLRETURNS` разрешает или запрещает вставку перевода строки в многострочное поле ввода типа `RDS_FORMCTRL_MULTILINE` (стр. 497) по нажатию клавиши Enter.

Вызов команды:

```
int iCtrlId=... // Идентификатор поля ввода
int iEnable=... // 1 - разрешить вставку, 0 - запретить
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_MLRETURNS,iEnable);
```

Параметры:

`Win`

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

`iCtrlId`

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

`iEnable`

Целое число, указывающее на разрешение вставки перевода строки по клавише Enter (отличное от нуля значение) или запрещение такой вставки (нулевое значение).

Примечания:

Обычно нажатие клавиши Enter в открытом модальном окне приводит к автоматическому нажатию кнопки окна по умолчанию (в данном случае – кнопки “OK”). Для многострочного поля ввода типа `RDS_FORMCTRL_MULTILINE` командой `RDS_FORMVAL_MLRETURNS` можно разрешить вместо этого вставку перевода строки в поле. По умолчанию для вставки перевода строки используется сочетание клавиш Ctrl+Enter.

Пример использования команды приведен в §4.3.

См. также:

`RDS_FORMCTRL_MULTILINE` (стр. 497), `rdsFORMCreate` (стр. 491),
`rdsFORMAddEdit` (стр. 492), `rdsSetObjectInt` (стр. 407).

A.5.28.21. Команда `RDS_FORMVAL_PBBEVEL` – установка рамки вокруг области рисования

Команда `RDS_FORMVAL_PBBEVEL` включает или выключает рамку вокруг области программного рисования `RDS_FORMCTRL_PAINTBOX` (стр. 498). По умолчанию рамка включена.

Вызов команды:

```
int iCtrlId=... // Идентификатор поля ввода
int iEnable=... // 1 - включить рамку, 0 - выключить
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_PBBEVEL,iEnable);

или

int iCtrlId=... // Идентификатор поля ввода
char *sEnable=... // "1" - включить рамку, "0" - выключить
rdsSetObjectStr(Win,iCtrlId,RDS_FORMVAL_PBBEVEL,sEnable);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

iEnable

Целое число, указывающее на разрешение (отличное от нуля значение) или запрещение (нулевое значение) отображения рамки.

sEnable

Указатель на строку, в которой записано целое число, указывающее на разрешение (отличное от нуля значение) или запрещение (нулевое значение) отображения рамки. Пустая строка запрещает отображение рамки.

Примечания:

Признак наличия рамки имеет смысл передавать полю ввода только до открытия окна, поскольку в момент открытия окна общий вид всех объектов, включая наличие рамок, в нем фиксируются до тех пор, пока окно не будет закрыто.

Для передачи команды полю ввода можно использовать функции `rdsSetObjectInt` (стр. 407) и `rdsSetObjectStr` (стр. 407). Чаще всего используется целая функция `rdsSetObjectInt`, при этом переданное в нее целое число трактуется по обычным правилам языка C: нулевое значение считается логической ложью (рамка выключена), все остальные – логической истиной (рамка включена). При использовании `rdsSetObjectStr` переданная строка будет преобразована в целое число по правилам функции `rdsAtoI` (стр. 184).

См. также:

`RDS_FORMCTRL_PAINTBOX` (стр. 498), `rdsFORMCreate` (стр. 491),
`rdsFORMAddEdit` (стр. 492), `rdsSetObjectInt` (стр. 407),
`rdsSetObjectStr` (стр. 407).

А.5.28.22. Команда `RDS_FORMVAL_PBHEIGHT` – высота области программного рисования

Команда `RDS_FORMVAL_PBHEIGHT` устанавливает высоту в точках экрана для области программного рисования типа `RDS_FORMCTRL_PAINTBOX` (стр. 498).

Вызов команды:

```
int iCtrlId=... // Идентификатор поля ввода
int iHeight=... // Высота поля
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_PBHEIGHT,iHeight);

или

int iCtrlId=... // Идентификатор поля ввода
double dHeight=... // Высота поля
rdsSetObjectDouble(Win,iCtrlId,RDS_FORMVAL_PBHEIGHT,dHeight);

или

int iCtrlId=... // Идентификатор поля ввода
char *sHeight=... // Высота поля в виде строки
rdsSetObjectStr(Win,iCtrlId,RDS_FORMVAL_PBHEIGHT,sHeight);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

iCtrlId

Целый идентификатор области рисования, присвоенный ей при вызове функции `rdsFORMAddEdit` (стр. 492).

iHeight

Высота области в точках экрана или `-1`, если нужно, чтобы область заняла по высоте всю оставшуюся часть окна.

dHeight

Высота области в точках экрана в виде вещественного числа или `-1.0`, если нужно, чтобы область заняла по высоте всю оставшуюся часть окна.

sHeight

Указатель на строку, в которой записана высота области в точках экрана. Строка `"-1"` указывает на то, что область должна занять по высоте всю оставшуюся часть окна.

Примечания:

Эта команда задает высоту области программного рисования, то есть поля ввода типа `RDS_FORMCTRL_PAINTBOX`. Высоту имеет смысл передавать полю ввода только до открытия окна, поскольку в момент открытия окна размеры всех объектов в нем фиксируются до тех пор, пока окно не будет закрыто.

Для передачи команды полю ввода можно использовать любую из функций `rdsSetObjectInt` (стр. 407), `rdsSetObjectDouble` (стр. 406) и `rdsSetObjectStr` (стр. 407). Чаще всего используется целая функция `rdsSetObjectInt`, поскольку высота поля – целое число. При использовании `rdsSetObjectStr` переданная строка будет преобразована в целое число по правилам функции `rdsAtoI` (стр. 184).

Пример использования команды `RDS_FORMVAL_PBHEIGHT` приведен в §2.7.3.

См. также:

`RDS_FORMCTRL_PAINTBOX` (стр. 498), `rdsFORMCreate` (стр. 491),
`rdsFORMAddEdit` (стр. 492), `rdsSetObjectInt` (стр. 407),
`rdsSetObjectDouble` (стр. 406), `rdsSetObjectStr` (стр. 407).

А.5.28.23. Команда RDS_FORMVAL_RANGEMAX – значение поля ввода конца диапазона

Команда RDS_FORMVAL_RANGEMAX устанавливает или возвращает значение второго (правого) поля в двойном поле ввода RDS_FORMCTRL_RANGEEDIT (стр. 499), которое чаще всего используется для задания значения конца диапазона. Для установки или получения значения левого поля (начала диапазона) используется обычная команда RDS_FORMVAL_VALUE (стр. 525).

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода
int iValue=... // Значение
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_RANGEMAX,iValue);

или

int iCtrlId=... // Идентификатор поля ввода
double dValue=... // Значение
rdsSetObjectDouble(Win,iCtrlId,RDS_FORMVAL_RANGEMAX,dValue);

или

int iCtrlId=... // Идентификатор поля ввода
char *sValue=... // Значение в виде строки
rdsSetObjectStr(Win,iCtrlId,RDS_FORMVAL_RANGEMAX,sValue);
```

Вызов команды для чтения:

```
int iCtrlId=... // Идентификатор поля ввода
int iValue=rdsGetObjectInt(Win,iCtrlId,RDS_FORMVAL_RANGEMAX);

или

int iCtrlId=... // Идентификатор поля ввода
double dValue=rdsGetObjectDouble(Win,iCtrlId,
                                RDS_FORMVAL_RANGEMAX);

или

int iCtrlId=... // Идентификатор поля ввода
char *sValue=rdsGetObjectStr(Win,iCtrlId,RDS_FORMVAL_RANGEMAX);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

iValue

Значение поля в виде целого числа.

dValue

Значение поля в виде вещественного (double) числа.

sValue

Указатель на строку, в которой записано значение поля. При получении значения функцией rdsGetObjectStr в данном случае возвращается указатель на строку во внутренней памяти объекта Win, этот указатель будет действителен до тех пор, пока значение поля не изменится.

Примечания:

Эта команда служит для доступа к правому полю двойного поля ввода RDS_FORMCTRL_RANGEEDIT. Для передачи команды полю ввода можно использовать целые функции rdsSetObjectInt (стр. 407) и rdsGetObjectInt (стр. 404), вещественные rdsSetObjectDouble (стр. 406) и rdsGetObjectDouble (стр. 403), а также строковые rdsSetObjectStr (стр. 407) и rdsGetObjectStr (стр. 405) – тип используемых функций зависит только от того, в каком виде удобнее передавать значение полю или получать его из поля. Пример использования команды приведен в §2.10.1.

См. также:

RDS_FORMCTRL_RANGEEDIT (стр. 499), rdsFORMCreate (стр. 491),
rdsFORMAddEdit (стр. 492), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), rdsSetObjectDouble (стр. 406),
rdsGetObjectDouble (стр. 403), rdsSetObjectStr (стр. 407),
rdsGetObjectStr (стр. 405), RDS_FORMVAL_VALUE (стр. 525).

A.5.28.24. Команда RDS_FORMVAL_UPDOWNINC – шаг изменения поля ввода со стрелками

Команда RDS_FORMVAL_UPDOWNINC устанавливает или возвращает значение шага, с которым будет изменяться значение поля ввода типа RDS_FORMCTRL_UPDOWN (стр. 500) при нажатии кнопок со стрелками.

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода
int iValue=... // Значение шага
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_UPDOWNINC,iValue);

или

int iCtrlId=... // Идентификатор поля ввода
double dValue=... // Значение шага
rdsSetObjectDouble(Win,iCtrlId,RDS_FORMVAL_UPDOWNINC,dValue);

или

int iCtrlId=... // Идентификатор поля ввода
char *sValue=... // Значение шага в виде строки
rdsSetObjectStr(Win,iCtrlId,RDS_FORMVAL_UPDOWNINC,sValue);
```

Вызов команды для чтения:

```
int iCtrlId=... // Идентификатор поля ввода
int iValue=rdsGetObjectInt(Win,iCtrlId,RDS_FORMVAL_UPDOWNINC);

или

int iCtrlId=... // Идентификатор поля ввода
double dValue=rdsGetObjectDouble(Win,iCtrlId,
                                RDS_FORMVAL_UPDOWNINC);

или

int iCtrlId=... // Идентификатор поля ввода
char *sValue=rdsGetObjectStr(Win,iCtrlId,RDS_FORMVAL_UPDOWNINC);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

iValue

Значение шага изменения в виде целого числа.

dValue

Значение шага изменения в виде вещественного (double) числа.

sValue

Указатель на строку, в которой записано значение шага изменения. При получении значения функцией `rdsGetObjectStr` в данном случае возвращается указатель на строку во внутренней памяти объекта Win, этот указатель будет действителен до тех пор, пока шаг не будет изменен.

Примечания:

Эта команда устанавливает или получает шаг изменения поля `RDS_FORMCTRL_UPDOWN` с кнопками-стрелками. Для передачи команды полю ввода можно использовать целые функции `rdsSetObjectInt` (стр. 407) и `rdsGetObjectInt` (стр. 404), вещественные `rdsSetObjectDouble` (стр. 406) и `rdsGetObjectDouble` (стр. 403), а также строковые `rdsSetObjectStr` (стр. 407) и `rdsGetObjectStr` (стр. 405). Хотя значение шага может устанавливаться и строковой функцией `rdsSetObjectStr`, само поле может работать только с целыми или вещественными значениями, поэтому в передаваемой строке должно быть записано какое-либо число.

Пример использования команды `RDS_FORMVAL_UPDOWNINC` приведен в §2.10.1.

См. также:

`RDS_FORMCTRL_UPDOWN` (стр. 500), `rdsFORMCreate` (стр. 491),
`rdsFORMAddEdit` (стр. 492), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404), `rdsSetObjectDouble` (стр. 406),
`rdsGetObjectDouble` (стр. 403), `rdsSetObjectStr` (стр. 407),
`rdsGetObjectStr` (стр. 405).

A.5.28.25. Команда `RDS_FORMVAL_UPDOWNMAX` – максимальное значение поля ввода со стрелками

Команда `RDS_FORMVAL_UPDOWNMAX` устанавливает или возвращает максимальное значение поля ввода типа `RDS_FORMCTRL_UPDOWN` (стр. 500). Нажатием кнопок со стрелками нельзя будет установить в поле значение, большее переданного этой командой.

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода
int iMax=... // Значение
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_UPDOWNMAX,iMax);

или

int iCtrlId=... // Идентификатор поля ввода
double dMax=... // Значение
rdsSetObjectDouble(Win,iCtrlId,RDS_FORMVAL_UPDOWNMAX,dMax);
```

```

или

int iCtrlId=... // Идентификатор поля ввода
char *sMax=... // Значение в виде строки
rdsSetObjectStr(Win,iCtrlId,RDS_FORMVAL_UPDOWNMAX,sMax);

```

Вызов команды для чтения:

```

int iCtrlId=... // Идентификатор поля ввода
int iMax=rdsGetObjectInt(Win,iCtrlId,RDS_FORMVAL_UPDOWNMAX);

или

int iCtrlId=... // Идентификатор поля ввода
double dMax=rdsGetObjectDouble(Win,iCtrlId,RDS_FORMVAL_UPDOWNMAX);

или

int iCtrlId=... // Идентификатор поля ввода
char *sMax=rdsGetObjectStr(Win,iCtrlId,RDS_FORMVAL_UPDOWNMAX);

```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

iMax

Максимальное значение поля в виде целого числа.

dMax

Максимальное значение поля в виде вещественного (double) числа.

sMax

Указатель на строку, в которой записано максимальное значение поля. При получении значения функцией `rdsGetObjectStr` в данном случае возвращается указатель на строку во внутренней памяти объекта Win, этот указатель будет действителен до тех пор, пока максимальное значение не будет изменено.

Примечания:

Эта команда устанавливает или получает максимально допустимое значение поля `RDS_FORMCTRL_UPDOWN`, до которого его можно увеличить кнопками-стрелками (вручную пользователь может ввести в это поле любое значение, в том числе и превышающее заданное максимальное). Для передачи команды полю ввода можно использовать целые функции `rdsSetObjectInt` (стр. 407) и `rdsGetObjectInt` (стр. 404), вещественные `rdsSetObjectDouble` (стр. 406) и `rdsGetObjectDouble` (стр. 403), а также строковые `rdsSetObjectStr` (стр. 407) и `rdsGetObjectStr` (стр. 405). Хотя максимальное значение может устанавливаться и строковой функцией `rdsSetObjectStr`, само поле может работать только с целыми или вещественными значениями, поэтому в передаваемой строке должно быть записано какое-либо число.

По умолчанию проверка максимального значений поля отключена, передача в поле любого числового значения командой `RDS_FORMVAL_UPDOWNMAX` включает ее. Для того, чтобы снова отключить проверку, следует передать в качестве нового максимального значения либо пустую строку при помощи функции `rdsSetObjectStr`, либо специальное значение-индикатор математической ошибки (возвращаемое функцией `rdsGetHugeDouble`, см. стр. 157) при помощи функции `rdsSetObjectDouble`.

Пример использования команды RDS_FORMVAL_UPDOWNMAX приведен в §2.10.1.

См. также:

RDS_FORMCTRL_UPDOWN (стр. 500), rdsFORMCreate (стр. 491),
rdsFORMAddEdit (стр. 492), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), rdsSetObjectDouble (стр. 406),
rdsGetObjectDouble (стр. 403), rdsSetObjectStr (стр. 407),
rdsGetObjectStr (стр. 405).

А.5.28.26. Команда RDS_FORMVAL_UPDOWNMIN – минимальное значение поля ввода со стрелками

Команда RDS_FORMVAL_UPDOWNMIN устанавливает или возвращает минимальное значение поля ввода типа RDS_FORMCTRL_UPDOWN (стр. 500). Нажатием кнопок со стрелками нельзя будет установить в поле значение, меньшее переданного этой командой.

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода
int iMin=... // Значение
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_UPDOWNMIN,iMin);

или

int iCtrlId=... // Идентификатор поля ввода
double dMin=... // Значение
rdsSetObjectDouble(Win,iCtrlId,RDS_FORMVAL_UPDOWNMIN,dMin);

или

int iCtrlId=... // Идентификатор поля ввода
char *sMin=... // Значение в виде строки
rdsSetObjectStr(Win,iCtrlId,RDS_FORMVAL_UPDOWNMIN,sMin);
```

Вызов команды для чтения:

```
int iCtrlId=... // Идентификатор поля ввода
int iMin=rdsGetObjectInt(Win,iCtrlId,RDS_FORMVAL_UPDOWNMIN);

или

int iCtrlId=... // Идентификатор поля ввода
double dMin=rdsGetObjectDouble(Win,iCtrlId,RDS_FORMVAL_UPDOWNMIN);

или

int iCtrlId=... // Идентификатор поля ввода
char *sMin=rdsGetObjectStr(Win,iCtrlId,RDS_FORMVAL_UPDOWNMIN);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

iMin

Минимальное значение поля в виде целого числа.

dMin

Минимальное значение поля в виде вещественного (double) числа.

sMin

Указатель на строку, в которой записано минимальное значение поля. При получении значения функцией `rdsGetObjectStr` в данном случае возвращается указатель на строку во внутренней памяти объекта Win, этот указатель будет действителен до тех пор, пока минимальное значение не будет изменено.

Примечания:

Эта команда устанавливает или получает минимально допустимое значение поля `RDS_FORMCTRL_UPDOWN`, до которого его можно уменьшить кнопками-стрелками (вручную пользователь может ввести в это поле любое значение, в том числе и меньшее заданного минимального). Для передачи команды полю ввода можно использовать целые функции `rdsSetObjectInt` (стр. 407) и `rdsGetObjectInt` (стр. 404), вещественные `rdsSetObjectDouble` (стр. 406) и `rdsGetObjectDouble` (стр. 403), а также строковые `rdsSetObjectStr` (стр. 407) и `rdsGetObjectStr` (стр. 405). Хотя минимальное значение может устанавливаться и строковой функцией `rdsSetObjectStr`, само поле может работать только с целыми или вещественными значениями, поэтому в передаваемой строке должно быть записано какое-либо число.

По умолчанию проверка минимальных значений поля отключена, передача в поле любого числового значения командой `RDS_FORMVAL_UPDOWNMIN` включает ее. Для того, чтобы снова отключить проверку, следует передать в качестве нового минимального значения либо пустую строку при помощи функции `rdsSetObjectStr`, либо специальное значение-индикатор математической ошибки (возвращаемое функцией `rdsGetHugeDouble`, см. стр. 157) при помощи функции `rdsSetObjectDouble`.

Пример использования команды `RDS_FORMVAL_UPDOWNMIN` приведен в §2.10.1.

См. также:

`RDS_FORMCTRL_UPDOWN` (стр. 500), `rdsFORMCreate` (стр. 491),
`rdsFORMAddEdit` (стр. 492), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404), `rdsSetObjectDouble` (стр. 406),
`rdsGetObjectDouble` (стр. 403), `rdsSetObjectStr` (стр. 407),
`rdsGetObjectStr` (стр. 405).

A.5.28.27. Команда `RDS_FORMVAL_VALUE` – значение поля

Команда `RDS_FORMVAL_VALUE` устанавливает или возвращает значение поля ввода. Возможные значения и способ их установки и получения зависят от конкретных типов полей ввода.

Вызов команды для установки:

```
int iCtrlId=... // Идентификатор поля ввода
int iValue=...  // Значение
rdsSetObjectInt(Win,iCtrlId,RDS_FORMVAL_VALUE,iValue);

или

int iCtrlId=... // Идентификатор поля ввода
double dValue=... // Значение
rdsSetObjectDouble(Win,iCtrlId,RDS_FORMVAL_VALUE,dValue);

или
```

```
int iCtrlId=... // Идентификатор поля ввода
char *sValue=... // Значение в виде строки
rdsSetObjectStr(Win,iCtrlId,RDS_FORMVAL_VALUE,sValue);
```

Вызов команды для чтения:

```
int iCtrlId=... // Идентификатор поля ввода
int iValue=rdsGetObjectInt(Win,iCtrlId,RDS_FORMVAL_VALUE);

или

int iCtrlId=... // Идентификатор поля ввода
double dValue=rdsGetObjectDouble(Win,iCtrlId,
                                RDS_FORMVAL_VALUE);

или

int iCtrlId=... // Идентификатор поля ввода
char *sValue=rdsGetObjectStr(Win,iCtrlId,RDS_FORMVAL_VALUE);
```

Параметры:

Win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

iCtrlId

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

iValue

Значение поля в виде целого числа.

dValue

Значение поля в виде вещественного (double) числа.

sValue

Указатель на строку, в которой записано значение поля. При получении значения функцией `rdsGetObjectStr` в данном случае возвращается указатель на строку во внутренней памяти объекта Win, этот указатель будет действителен до тех пор, пока значение поля не изменится.

Примечания:

Для передачи команды полю ввода можно использовать целые функции `rdsSetObjectInt` (стр. 407) и `rdsGetObjectInt` (стр. 404), вещественные `rdsSetObjectDouble` (стр. 406) и `rdsGetObjectDouble` (стр. 403), а также строковые `rdsSetObjectStr` (стр. 407) и `rdsGetObjectStr` (стр. 405) – тип используемых функций зависит только от того, в каком виде удобнее передавать значение полю или получать его из поля.

При использовании команды для разных типов полей ввода следует учитывать следующие особенности:

- для полей ввода `RDS_FORMCTRL_EDIT` (стр. 496), `RDS_FORMCTRL_COMBOEDIT` (стр. 495), `RDS_FORMCTRL_DISPLAY` (стр. 496), `RDS_FORMCTRL_MULTILINE` (стр. 497), `RDS_FORMCTRL_UPDOWN` (стр. 500), `RDS_FORMCTRL_NONVISUAL` (стр. 498) значение записывается в поле ввода и считывается оттуда непосредственно (все эти поля имеют в своем составе единственное поле ввода);
- для поля ввода `RDS_FORMCTRL_COMBOLIST` (стр. 495) значением, в зависимости от использованной функции, является либо номер выбранного в выпадающем списке варианта (варианты нумеруются начиная с нуля), либо его текст: функции

`rdsGetObjectStr` и `rdsGetObjectDouble` возвращают само значение поля (текст варианта), функция `rdsGetObjectInt` возвращает номер варианта, все три функции `rdsSetObjectInt`, `rdsSetObjectDouble` и `rdsSetObjectStr` устанавливают номер варианта;

- для полей-переключателей типа `RDS_FORMCTRL_CHECKBOX` (стр. 495) и `RDS_FORMCTRL_RADIOBUTTON` (стр. 499) значением всегда является целое число – ноль, если флаг выключен, и единица (любое ненулевое значение), если флаг включен;
- для двойного поля ввода диапазона `RDS_FORMCTRL_RANGEEDIT` (стр. 499) значением является содержимое левого поля ввода (начала диапазона), для доступа к значению правого поля используется команда `RDS_FORMVAL_RANGEMAX` (стр. 520);
- для полей открытия и сохранения файла `RDS_FORMCTRL_OPENDIALOG` (стр. 498) и `RDS_FORMCTRL_SAVEDIALOG` (стр. 499), а также для поля выбора папки `RDS_FORMCTRL_DIRDIALOG` (стр. 496) значением является имя файла (папки) с путем, в котором пути к стандартным папкам РДС заменены на их символические обозначения (стр. 189);
- для поля выбора цвета `RDS_FORMCTRL_COLOR` (стр. 495) значением является выбранный цвет в виде целого числа, то есть стандартный тип Windows `COLORREF` (стр. 24), приведенный к типу `int`;
- для поля выбора шрифта `RDS_FORMCTRL_FONTSELECT` (стр. 496) значением является текст описания шрифта, подобный формируемому функцией `rdsStructToFontText` (стр. 279) и разбираемому функцией `rdsFontTextToStruct` (стр. 290);
- для двойного поля ввода `RDS_FORMCTRL_LISTANDEDIT` (стр. 497) значением является содержимое основного (правого) поля ввода, а для доступа к номеру варианта, выбранного в списке слева от него, используется команда `RDS_FORMVAL_AUXLISTITEM` (стр. 508);
- для поля ввода кода клавиши `RDS_FORMCTRL_HOTKEY` (стр. 496) значением является сам код клавиши (или ноль, если код не введен) без флагов состояния служебных клавиш `Ctrl`, `Alt` и `Shift` – для доступа к этим флагам используется команда `RDS_FORMVAL_HKSHIFTS` (стр. 512);
- для кнопки `RDS_FORMCTRL_BUTTON` (стр. 494) значением является текст на кнопке, пользователь не может его изменить;
- поля `RDS_FORMCTRL_LABEL` (стр. 497) и `RDS_FORMCTRL_PAINTBOX` (стр. 498) не имеют значения, поэтому вызов для них команды `RDS_FORMVAL_VALUE` не имеет смысла.

Пример использования команды `RDS_FORMVAL_VALUE` приведен в §2.7.2, §2.7.4 и др.

См. также:

`rdsFORMCreate` (стр. 491), `rdsFORMAddEdit` (стр. 492), типы и флаги полей ввода (стр. 494), `rdsSetObjectInt` (стр. 407), `rdsGetObjectInt` (стр. 404), `rdsSetObjectDouble` (стр. 406), `rdsGetObjectDouble` (стр. 403), `rdsSetObjectStr` (стр. 407), `rdsGetObjectStr` (стр. 405).

А.5.28.28. Макрос `rdsFORMClear` – очистка объекта

Макрос `rdsFORMClear` удаляет из объекта все поля ввода, вкладки и боковую панель.

```
rdsFORMClear(  
    win           // Вспомогательный объект-окно  
)
```

Определение:

```
#define rdsFORMClear(win) \
    rdsCommandObject((win), RDS_FORM_CLEAR)
```

Параметр:

win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

Примечания:

Этот макрос включает в себя вызов команды `RDS_FORM_CLEAR` (стр. 505).

См. также:

`RDS_FORM_CLEAR` (стр. 505).

А.5.28.29. Макрос `rdsFORMEnableControl` – разрешение и запрещение всего поля ввода

Макрос `rdsFORMEnableControl` разрешает или запрещает все поле ввода целиком, со всеми его компонентами и дополнительным флагом, если он есть.

```
rdsFORMEnableControl(
    win,          // Вспомогательный объект-окно
    ctrlid,       // Идентификатор поля ввода
    enabled       // Разрешить/запретить
)
```

Определение:

```
#define rdsFORMEnableControl(win, ctrlid, enabled) \
    rdsSetObjectInt((win), (ctrlid), RDS_FORMVAL_ENABLED, \
        (enabled)?1:0)
```

Параметры:

win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

ctrlid

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

enabled

TRUE – разрешить поле ввода, FALSE – запретить.

Примечания:

Этот макрос включает в себя вызов команды `RDS_FORMVAL_ENABLED` (стр. 511).

См. также:

`RDS_FORMVAL_ENABLED` (стр. 511).

А.5.28.30. Макрос `rdsFORMGetBool` – получение целого значения поля ввода и преобразование его в логическое

Макрос `rdsFORMGetBool` возвращает истину, если значение указанного поля ввода отлично от нуля, и ложь, если оно равно нулю.


```

rdsFORMGetBool(
    win,          // Вспомогательный объект-окно
    ctrlid       // Идентификатор поля ввода
)

```

Определение:

```

#define rdsFORMGetBool(win,ctrlid) \
    (rdsGetObjectInt((win),(ctrlid),RDS_FORMVAL_VALUE) !=0)

```

Параметры:

win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

ctrlid

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

Возвращаемое значение:

Значение поля ввода ctrlid, приведенное к логическому типу.

Примечания:

Этот макрос включает в себя вызов команды RDS_FORMVAL_VALUE (стр. 525) с использованием функции rdsGetObjectInt (стр. 404). Возвращенное функцией целое значение поля ввода сравнивается с нулем.

См. также:

RDS_FORMVAL_VALUE (стр. 525).

A.5.28.31. Макрос rdsFORMGetDouble – получение вещественного значения поля ввода

Макрос rdsFORMGetDouble возвращает значение указанного поля ввода в виде вещественного числа.

```

rdsFORMGetDouble(
    win,          // Вспомогательный объект-окно
    ctrlid       // Идентификатор поля ввода
)

```

Определение:

```

#define rdsFORMGetDouble(win,ctrlid) \
    rdsGetObjectDouble((win),(ctrlid),RDS_FORMVAL_VALUE)

```

Параметры:

win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

ctrlid

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

Возвращаемое значение:

Вещественное значение поля ввода ctrlid.

Примечания:

Этот макрос включает в себя вызов команды RDS_FORMVAL_VALUE (стр. 525) с использованием функции rdsGetObjectDouble (стр. 403).

См. также:

RDS_FORMVAL_VALUE (стр. 525).

А.5.28.32. Макрос rdsFORMGetEnableCheck – получение значения дополнительного разрешающего флага поля ввода

Макрос rdsFORMGetEnableCheck возвращает истину, если дополнительный разрешающий флаг указанного поля ввода включен.

```
rdsFORMGetEnableCheck(  
    win,          // Вспомогательный объект-окно  
    ctrlid       // Идентификатор поля ввода  
)
```

Определение:

```
#define rdsFORMGetEnableCheck(win,ctrlid) \  
    (rdsGetObjectInt((win),(ctrlid),RDS_FORMVAL_CHECK)!=0)
```

Параметры:

win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

ctrlid

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

Возвращаемое значение:

Значение разрешающего флага поля ввода ctrlid, приведенное к логическому типу.

Примечания:

Этот макрос включает в себя вызов команды RDS_FORMVAL_CHECK (стр. 510) с использованием функции rdsGetObjectInt (стр. 404). Возвращенное функцией целое значение поля ввода сравнивается с нулем.

См. также:

RDS_FORMVAL_CHECK (стр. 510).

А.5.28.33. Макрос rdsFORMGetInt – получение целого значения поля ввода

Макрос rdsFORMGetInt возвращает значение указанного поля ввода в виде целого числа.

```
rdsFORMGetInt(  
    win,          // Вспомогательный объект-окно  
    ctrlid       // Идентификатор поля ввода  
)
```

Определение:

```
#define rdsFORMGetInt(win,ctrlid) \  
    rdsGetObjectInt((win),(ctrlid),RDS_FORMVAL_VALUE)
```

Параметры:

`win`

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

`ctrlid`

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

Возвращаемое значение:

Целое значение поля ввода `ctrlid`.

Примечания:

Этот макрос включает в себя вызов команды `RDS_FORMVAL_VALUE` (стр. 525) с использованием функции `rdsGetObjectInt` (стр. 404).

См. также:

`RDS_FORMVAL_VALUE` (стр. 525).

A.5.28.34. Макрос `rdsFORMGetString` – получение значения поля ввода в виде строки

Макрос `rdsFORMGetString` возвращает значение указанного поля ввода в виде строки.

```
rdsFORMGetString(  
    win,          // Вспомогательный объект-окно  
    ctrlid       // Идентификатор поля ввода  
)
```

Определение:

```
#define rdsFORMGetString(win,ctrlid) \  
    rdsGetObjectStr((win),(ctrlid),RDS_FORMVAL_VALUE)
```

Параметры:

`win`

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

`ctrlid`

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

Возвращаемое значение:

Указатель на строку значения поля ввода `ctrlid` во внутренней памяти объекта `win`.

Примечания:

Этот макрос включает в себя вызов команды `RDS_FORMVAL_VALUE` (стр. 525) с использованием функции `rdsGetObjectStr` (стр. 405). Указатель, возвращенный макросом, будет действительным до следующего изменения значения поля ввода.

См. также:

`RDS_FORMVAL_VALUE` (стр. 525).

А.5.28.35. Макрос `rdsFORMSetBool` – установка целого значения поля ввода как логического

Макрос `rdsFORMSetBool` заносит в указанное поле ввода логическое значение в виде целого (1 – истина, 0 – ложь).

```
rdsFORMSetBool(  
    win,          // Вспомогательный объект-окно  
    ctrlid,       // Идентификатор поля ввода  
    value         // Значение  
)
```

Определение:

```
#define rdsFORMSetBool(win,ctrlid,value) \  
    rdsSetObjectInt((win),(ctrlid),RDS_FORMVAL_VALUE, \  
        (value)?1:0)
```

Параметры:

`win`

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

`ctrlid`

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

`value`

Логическое значение поля ввода (TRUE или FALSE).

Примечания:

Этот макрос включает в себя вызов команды `RDS_FORMVAL_VALUE` (стр. 525) с использованием функции `rdsSetObjectInt` (стр. 407). Логическое значение TRUE записывается в поле ввода как единица, FALSE – как ноль.

См. также:

`RDS_FORMVAL_VALUE` (стр. 525).

А.5.28.36. Макрос `rdsFORMSetComboList` – установка списка вариантов

Макрос `rdsFORMSetComboList` заносит в указанное поле ввода список вариантов для выпадающего списка или список шаблонов имен файлов для кнопок открытия и сохранения файла.

```
rdsFORMSetComboList(  
    win,          // Вспомогательный объект-окно  
    ctrlid,       // Идентификатор поля ввода  
    list          // Список  
)
```

Определение:

```
#define rdsFORMSetComboList(win,ctrlid,list) \  
    rdsSetObjectStr((win),(ctrlid),RDS_FORMVAL_LIST, \  
        (list))
```

Параметры:

`win`

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

`ctrlid`

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

`list`

Указатель на строку со списком вариантов для полей `RDS_FORMCTRL_COMBOEDIT` (стр. 495), `RDS_FORMCTRL_COMBOLIST` (стр. 495) и `RDS_FORMCTRL_LISTANDEDIT` (стр. 497) или со списком шаблонов имен файлов для полей ввода `RDS_FORMCTRL_OPENDIALOG` (стр. 498) и `RDS_FORMCTRL_SAVEDIALOG` (стр. 499). Устройство такой строки рассматривается в описаниях соответствующих типов полей ввода.

Примечания:

Этот макрос включает в себя вызов команды `RDS_FORMVAL_LIST` (стр. 515) с использованием функции `rdsSetObjectStr` (стр. 407).

См. также:

`RDS_FORMVAL_LIST` (стр. 515).

A.5.28.37. Макрос `rdsFORMSetDouble` – установка вещественного значения поля ввода

Макрос `rdsFORMSetDouble` заносит в указанное поле ввода вещественное значение.

```
rdsFORMSetDouble(  
    win,          // Вспомогательный объект-окно  
    ctrlid,       // Идентификатор поля ввода  
    value         // Значение  
)
```

Определение:

```
#define rdsFORMSetDouble(win,ctrlid,value) \  
    rdsSetObjectDouble((win),(ctrlid),RDS_FORMVAL_VALUE, \  
        (value))
```

Параметры:

`win`

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

`ctrlid`

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

`value`

Вещественное значение поля ввода.

Примечания:

Этот макрос включает в себя вызов команды `RDS_FORMVAL_VALUE` (стр. 525) с использованием функции `rdsSetObjectDouble` (стр. 406).

См. также:

RDS_FORMVAL_VALUE (стр. 525).

A.5.28.38. Макрос rdsFORMSetEnableCheck – установка значения дополнительного разрешающего флага поля ввода

Макрос rdsFORMSetEnableCheck устанавливает дополнительный разрешающий флаг указанного поля ввода.

```
rdsFORMSetEnableCheck(  
    win,          // Вспомогательный объект-окно  
    ctrlid,       // Идентификатор поля ввода  
    enabled       // Разрешение/запрещение  
)
```

Определение:

```
#define rdsFORMSetEnableCheck(win,ctrlid,enabled) \  
    rdsSetObjectInt((win),(ctrlid),RDS_FORMVAL_CHECK, \  
        (enabled)?1:0)
```

Параметры:

win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

ctrlid

Целый идентификатор поля ввода, присвоенный ему при вызове функции rdsFORMAddEdit (стр. 492).

enabled

TRUE – включить флаг (поле будет разрешено), FALSE – выключить (поле будет запрещено).

Примечания:

Этот макрос включает в себя вызов команды RDS_FORMVAL_CHECK (стр. 510) с использованием функции rdsSetObjectInt (стр. 407). Логическое значение TRUE передается в поле ввода как единица, FALSE – как ноль.

См. также:

RDS_FORMVAL_CHECK (стр. 510).

A.5.28.39. Макрос rdsFORMSetInt – установка целого значения поля ввода

Макрос rdsFORMSetInt заносит в указанное поле ввода целое значение.

```
rdsFORMSetInt(  
    win,          // Вспомогательный объект-окно  
    ctrlid,       // Идентификатор поля ввода  
    value         // Значение  
)
```

Определение:

```
#define rdsFORMSetInt(win,ctrlid,value) \  
    rdsSetObjectInt((win),(ctrlid),RDS_FORMVAL_VALUE, \  
        (value))
```

Параметры:

win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

ctrlid

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

value

Целое значение поля ввода.

Примечания:

Этот макрос включает в себя вызов команды `RDS_FORMVAL_VALUE` (стр. 525) с использованием функции `rdsSetObjectInt` (стр. 407).

См. также:

`RDS_FORMVAL_VALUE` (стр. 525).

A.5.28.40. Макрос `rdsFORMSetMultilineHeight` – установка высоты многострочного поля ввода

Макрос `rdsFORMSetMultilineHeight` задает высоту в точках экрана для указанного поля ввода типа `RDS_FORMCTRL_MULTILINE` (стр. 497).

```
rdsFORMSetMultilineHeight(  
    win,          // Вспомогательный объект-окно  
    ctrlid,       // Идентификатор поля ввода  
    height        // Высота  
)
```

Определение:

```
#define rdsFORMSetMultilineHeight(win,ctrlid,height) \  
    rdsSetObjectInt((win),(ctrlid),RDS_FORMVAL_MLHEIGHT, \  
        (height))
```

Параметры:

win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией `rdsFORMCreate` (стр. 491).

ctrlid

Целый идентификатор поля ввода, присвоенный ему при вызове функции `rdsFORMAddEdit` (стр. 492).

height

Высота поля ввода в точках экрана.

Примечания:

Этот макрос включает в себя вызов команды `RDS_FORMVAL_MLHEIGHT` (стр. 516) с использованием функции `rdsSetObjectInt` (стр. 407).

См. также:

`RDS_FORMVAL_MLHEIGHT` (стр. 516).

A.5.28.41. Макрос rdsFORMShowModal – открытие модального окна

Макрос rdsFORMShowModal открывает модальное окно, описываемое указанным вспомогательным объектом.

```
rdsFORMShowModal(  
    win          // Вспомогательный объект-окно  
)
```

Определение:

```
#define rdsFORMShowModal(win) \  
    rdsCommandObject((win), RDS_FORM_SHOWMODAL)
```

Параметр:

win

Идентификатор вспомогательного объекта для работы с модальным окном, ранее созданного функцией rdsFORMCreate (стр. 491).

Возвращаемое значение:

TRUE – пользователь закрыл окно кнопкой “ОК”, FALSE – кнопкой “Отмена”.

Примечания:

Этот макрос включает в себя вызов команды RDS_FORM_SHOWMODAL (стр. 506).

См. также:

RDS_FORM_SHOWMODAL (стр. 506).

A.5.29. Вспомогательный объект для отмены редактирования параметров блока

Описываются функции и команды вспомогательного объекта РДС, предназначенного для запоминания параметров блока до и после их изменения в общем наборе операций РДС. Это изменение параметров пользователь сможет, при желании, отменить.

A.5.29.1. rdsBEUCreate – создать объект для отмены редактирования параметров блока

Функция rdsBEUCreate создает вспомогательный объект РДС, позволяющий позже вернуть параметры блока в состояние на момент создания этого объекта.

```
RDS_HOBJECT RDSCALL rdsBEUCreate(  
    RDS_BHANDLE Block    // Блок  
) ;
```

Тип указателя на эту функцию:

RDS_HoBh

Параметр:

Block

Идентификатор блока, параметры которого запоминаются.

Возвращаемое значение:

Уникальный идентификатор созданного объекта.

Примечания:

Эта функция создает вспомогательный объект, который будет содержать два полных набора параметров блока: до изменения и после изменения. С помощью этого объекта можно

добавить операцию какого-либо программного изменения параметров блока в общий список выполненных пользователем операций. Пользователь сможет отменить эту операцию, выбрав пункт меню “Система | Отменить” или нажав Ctrl+Z, а также возратить отмененную операцию, выбрав пункт меню “Система | Возврат” или нажав Ctrl+R.

Созданный вспомогательный объект будет существовать до тех пор, пока схема не будет выгружена из памяти, или пока он не будет удален вызовом функции `rdsDeleteObject` (стр. 401).

Параметры блока до изменения запоминаются в момент создания объекта. Обычно работа с таким объектом строится по следующему сценарию:

- объект создается вызовом `rdsBEUCreate`, при этом все параметры блока, идентификатор которого передан в функцию, сохраняются во внутренней памяти объекта;
- параметры блока программно изменяются каким-либо образом;
- при помощи команды `RDS_BEU_STORECHANGED` (стр. 537) прежние параметры блока из памяти объекта и измененные параметры этого же блока передаются в общий набор запомненных операций РДС – теперь изменение может быть отменено пользователем;
- объект удаляется вызовом `rdsDeleteObject` – поскольку параметры блока сохраняются в момент создания объекта, для повторного использования этот объект непригоден.

При сохранении параметров блока до и после изменения модель этого блока вызывается для реакции на событие `RDS_BFM_SAVEBIN` (стр. 53) и `RDS_BFM_SAVETXT` (стр. 54) для того, чтобы она могла сохранить параметры из личной области данных блока. При этом в качестве причины сохранения данных указывается `RDS_LS_SAVEUNDO` (см. вызов функции `rdsGetSystemInt` с параметром `RDS_GSISAVELOADACTION`, стр. 159).

Следует помнить, что при изменении параметров блока в функции настройки (событие `RDS_BFM_SETUP`, стр. 71) добавление параметров блока до и после изменения в общий набор запомненных операций производится автоматически, и использовать этот вспомогательный объект не следует.

См. также:

Вспомогательные объекты (стр. 399), `rdsDeleteObject` (стр. 401),
`RDS_BEU_STORECHANGED` (стр. 537), `RDS_BFM_SAVEBIN` (стр. 53),
`RDS_BFM_SAVETXT` (стр. 54).

A.5.29.2. Команда `RDS_BEU_STORECHANGED` – запись измененных параметров блока

Команда `RDS_BEU_STORECHANGED` записывает измененные параметры блока в общий набор запомненных операций РДС.

Вызов команды:

```
BOOL bOk=rdsCommandObject (Undo, RDS_BEU_STORECHANGED) ;  
или  
BOOL bOk=rdsCommandObjectEx (Undo, RDS_BEU_STORECHANGED, 0, NULL) ;
```

Параметры и результат:

Undo

Идентификатор вспомогательного объекта, ранее созданного функцией `rdsBEUCreate` (стр. 536).

bOk

TRUE – параметры записаны, FALSE – ошибка (отмена операций запрещена в настройках РДС или блок, для параметров которого создан объект, удален).

Примечания:

Эта команда записывает текущие значения параметров блока, идентификатор которого ранее был передан в функцию `rdsBEUCreate` при создании объекта, в набор запомненных операций РДС. После этого пользователь сможет отменить изменение параметров через пункт “Система | Отменить” главного меню РДС.

См. также:

`rdsBEUCreate` (стр. 536), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400).

А.5.29.3. Макрос `rdsBEUStore` – запись измененных параметров блока

Макрос `rdsBEUStore` записывает измененные параметры блока в общий набор запомненных операций РДС.

```
rdsBEUStore(  
    undo          // Вспомогательный объект  
)
```

Определение:

```
#define rdsBEUStore(undo) \  
    rdsCommandObject((undo), RDS_BEU_STORECHANGED)
```

Параметр:

`undo`

Идентификатор вспомогательного объекта, ранее созданного функцией `rdsBEUCreate` (стр. 536).

Возвращаемое значение:

`TRUE` – параметры записаны, `FALSE` – ошибка (отмена операций запрещена в настройках РДС или блок, для параметров которого создан объект, удален).

Примечания:

Этот макрос включает в себя вызов команды `RDS_BEU_STORECHANGED` (стр. 537).

См. также:

`RDS_BEU_STORECHANGED` (стр. 537).

А.5.30. Вспомогательный объект для вывода индикатора выполнения

Описываются функции и команды вспомогательного объекта РДС, предназначенного для вывода на экран индикатора хода выполнения какой-либо длительной операции (progress bar).

А.5.30.1. `rdsPBARCreate` – создать объект для вывода индикатора выполнения

Функция `rdsPBARCreate` создает вспомогательный объект РДС, позволяющий показывать пользователю горизонтальный индикатор (progress bar), обычно символизирующий ход выполнения какой-либо операции.

```
RDS_HOBJECT RDSCALL rdsPBARCreate(  
    int Max,          // Максимум значения  
    LPSTR Caption     // Заголовок индикатора  
);
```

Тип указателя на эту функцию:

RDS_HoIS

Параметры:

Max

Максимальное значение индикатора (его позиция изменяется от нуля до этого значения).

Caption

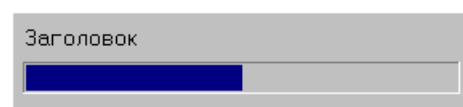
Указатель на строку с заголовком индикатора или NULL, если заголовка нет или он будет задан позже.

Возвращаемое значение:

Уникальный идентификатор созданного объекта.

Примечания:

Эта функция создает вспомогательный объект, который будет отображать в отдельном окне горизонтальный, увеличивающийся вправо индикатор. Размеры индикатора и его окна фиксированы и определяются РДС. Значение позиции индикатора может изменяться от нуля до параметра Max, при этом нулевому значению соответствует пустой (не заполненный) индикатор, а значению Max – индикатор, заполненный из конца в конец. Обычно в параметре Max передают число элементарных шагов какой-либо длительной операции, а в его позиции в процессе выполнения этой операции отражают число уже выполненных шагов. Если одновременно показывается несколько индикаторов, их окна автоматически выстраиваются друг рядом с другом.



Сразу после создания объекта индикатор не отображается, его нужно вывести на экран командой RDS_PBAR_SHOW (стр. 543). Скрыть индикатор с экрана, не уничтожая объект, можно командой RDS_PBAR_HIDE (стр. 540). Созданный вспомогательный объект будет существовать до тех пор, пока схема не будет выгружена из памяти, или пока он не будет удален вызовом функции rdsDeleteObject (стр. 401) (при удалении объекта индикатор автоматически исчезнет с экрана).

Чаще всего работа с объектом для отображения индикатора выполнения строится по следующему сценарию:

- перед выполнением длительной операции объект создается вызовом rdsPBARCreate, в этот момент задается максимум и заголовок индикатора;
- индикатор выводится на экран командой RDS_PBAR_SHOW;
- в процессе выполнения операции командами RDS_PBAR_POSITION (стр. 541) или RDS_PBAR_ADDTOPOS (стр. 540) позиция индикатора изменяется, отображая ход выполнения этой операции (к моменту ее окончания индикатор должен быть полностью заполнен);
- объект удаляется вызовом rdsDeleteObject, при этом индикатор исчезает с экрана.

Если нужно последовательно выполнить несколько операций, вместо удаления объекта в конце очередной из них можно сбросить индикатор в исходное, не заполненное состояние командой RDS_PBAR_RESET (стр. 542), изменить его максимум и заголовок командами RDS_PBAR_MAX (стр. 541) и RDS_PBAR_SETCAPTION (стр. 543), после чего начать следующую операцию, увеличивая позицию индикатора согласно ходу ее выполнения.

См. также:

Вспомогательные объекты (стр. 399), `rdsDeleteObject` (стр. 401),
`RDS_PBAR_SHOW` (стр. 543), `RDS_PBAR_POSITION` (стр. 541),
`RDS_PBAR_ADDTOPOS` (стр. 540).

A.5.30.2. Команда `RDS_PBAR_ADDTOPOS` – добавить число к текущей позиции индикатора

Команда `RDS_PBAR_ADDTOPOS` добавляет указанное число к текущей позиции индикатора выполнения.

Вызов команды:

```
int iDelta=... // Добавка к позиции индикатора  
rdsSetObjectInt(Bar,RDS_PBAR_ADDTOPOS,0,iDelta);
```

Параметры:

`Bar`

Идентификатор вспомогательного объекта для работы с индикатором, ранее созданного функцией `rdsPBARCreate` (стр. 538).

`iDelta`

Целое число, на которое нужно увеличить текущую позицию индикатора.

Примечания:

Эта команда добавляет к текущей позиции индикатора число `iDelta` и немедленно отображает изменения на экране. Ее использование в некоторых случаях удобнее непосредственной установки текущей позиции индикатора. Если, например, выполняемая операция, ход которой нужно отображать, состоит из заранее известного числа шагов, это число шагов можно установить как максимум индикатора при его создании, а после выполнения каждого шага вызывать команду `RDS_PBAR_ADDTOPOS` для продвижения индикатора на один шаг.

См. также:

`rdsPBARCreate` (стр. 538), `rdsSetObjectInt` (стр. 407).

A.5.30.3. Команда `RDS_PBAR_HIDE` – убрать индикатор с экрана

Команда `RDS_PBAR_HIDE` скрывает индикатор, относящийся к указанному объекту (сам объект при этом не уничтожается).

Вызов команды:

```
rdsCommandObject(Bar,RDS_PBAR_HIDE);  
или  
rdsCommandObjectEx(Bar,RDS_PBAR_HIDE,0,NULL);
```

Параметр:

`Bar`

Идентификатор вспомогательного объекта для работы с индикатором, ранее созданного функцией `rdsPBARCreate` (стр. 538).

Примечания:

Для возврата скрытого индикатора на экран следует использовать команду `RDS_PBAR_SHOW` (стр. 543).

См. также:

`rdsPBARCreate` (стр. 538), `rdsCommandObject` (стр. 400),
`rdsCommandObjectEx` (стр. 400), `RDS_PBAR_SHOW` (стр. 543).

А.5.30.4. Команда `RDS_PBAR_MAX` – максимальное значение индикатора

Команда `RDS_PBAR_MAX` позволяет получить или установить максимальное значение индикатора, то есть значение его позиции, при котором он будет заполнен из конца в конец.

Вызов команды для установки:

```
int iMax=... // Максимальное значение позиции
rdsSetObjectInt (Bar, RDS_PBAR_MAX, 0, iMax);
```

Вызов команды для чтения:

```
int iMax=rdsGetObjectInt (Bar, RDS_PBAR_MAX, 0);
```

Параметры и результат:

`Bar`

Идентификатор вспомогательного объекта для работы с индикатором, ранее созданного функцией `rdsPBARCreate` (стр. 538).

`iMax`

Целое число, соответствующее значению позиции полностью заполненного индикатора.

Примечания:

Эта команда позволяет изменить максимум индикатора уже после его создания.

См. также:

`rdsPBARCreate` (стр. 538), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404).

А.5.30.5. Команда `RDS_PBAR_POSITION` – текущая позиция индикатора

Команда `RDS_PBAR_POSITION` позволяет считывать и устанавливать текущую позицию индикатора, определяющую степень его заполненности.

Вызов команды для установки:

```
int iPos=... // Позиция индикатора
rdsSetObjectInt (Bar, RDS_PBAR_POSITION, 0, iPos);
```

Вызов команды для чтения:

```
int iPos=rdsGetObjectInt (Bar, RDS_PBAR_POSITION, 0);
```

Параметры и результат:

`Bar`

Идентификатор вспомогательного объекта для работы с индикатором, ранее созданного функцией `rdsPBARCreate` (стр. 538).

`iPos`

Целое значение текущей позиции индикатора.

Примечания:

Эта команда позволяет установить индикатор в новую произвольную позицию (изменения немедленно отразятся в окне этого индикатора). Если число шагов операции, ход

которой нужно отражать на индикаторе, очень велико, прямая установка может оказаться удобнее пошагового изменения командой RDS_PBAR_ADDTOPOS (стр. 540). При большом количестве элементарных операций, каждая из которых занимает немного процессорного времени, суммарные накладные расходы на обновление индикатора на каждом шаге могут оказаться больше времени выполнения самой операции. В таких случаях удобнее обновлять индикатор не на каждом шаге, а с заданным интервалом по времени. Этого можно добиться, например, циклом следующего вида:

```
int Total=...           // Общее число шагов операции
DWORD LastUpdate;       // Время последнего обновления индикатора
RDS_HOBJECT Bar=rdsPBARCreate(Total, "Выполнение...");
rdsCommandObject(Bar, RDS_PBAR_SHOW); // Показать индикатор
LastUpdate=GetTickCount();
for(int i=0; i<Total; i++)
{ DWORD time=GetTickCount(); // Текущее время
  if(time-LastUpdate>1000) // Прошло > 1 сек
  { LastUpdate=time; // Новое время обновления
    rdsSetObjectInt(Bar, RDS_PBAR_POSITION, 0, i);
  }
  ... какие-то действия ...
} // for(int i=0; ...)
rdsDeleteObject(Bar); // Удаление индикатора
```

Здесь для определения интервала времени используется функция Windows API GetTickCount, возвращающая время в миллисекундах, прошедшее с момента загрузки системы.

См. также:

rdsPBARCreate (стр. 538), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404).

A.5.30.6. Команда RDS_PBAR_RESET – сбросить индикатор

Команда RDS_PBAR_RESET сбрасывает индикатор, устанавливая его текущую позицию в ноль.

Вызов команды:

```
rdsCommandObject(Bar, RDS_PBAR_RESET);
или
rdsCommandObjectEx(Bar, RDS_PBAR_RESET, 0, NULL);
```

Параметр:

Bar

Идентификатор вспомогательного объекта для работы с индикатором, ранее созданного функцией rdsPBARCreate (стр. 538).

Примечания:

Если окно индикатора присутствует на экране, изменения отразятся в нем немедленно. Вместо команды RDS_PBAR_RESET можно использовать RDS_PBAR_POSITION (стр. 541) с нулевым параметром. Максимум и заголовок индикатора не изменяются.

См. также:

rdsPBARCreate (стр. 538), rdsCommandObject (стр. 400),
rdsCommandObjectEx (стр. 400), RDS_PBAR_POSITION (стр. 541).

A.5.30.7. Команда RDS_PBAR_SETCAPTION – установить заголовок индикатора

Команда RDS_PBAR_SETCAPTION позволяет изменить заголовок индикатора уже после его создания.

Вызов команды:

```
char *sCaption=... // Заголовок  
rdsSetObjectStr (Bar, RDS_PBAR_SETCAPTION, 0, sCaption);
```

Параметры:

Bar

Идентификатор вспомогательного объекта для работы с индикатором, ранее созданного функцией rdsPBARCreate (стр. 538).

sCaption

Указатель на строку с новым заголовком индикатора или NULL для пустого заголовка.

См. также:

rdsPBARCreate (стр. 538), rdsSetObjectStr (стр. 407).

A.5.30.8. Команда RDS_PBAR_SHOW – показать индикатор

Команда RDS_PBAR_SHOW выводит на экран индикатор, соответствующий указанному объекту.

Вызов команды:

```
rdsCommandObject (Bar, RDS_PBAR_SHOW);  
или  
rdsCommandObjectEx (Bar, RDS_PBAR_SHOW, 0, NULL);
```

Параметр:

Bar

Идентификатор вспомогательного объекта для работы с индикатором, ранее созданного функцией rdsPBARCreate (стр. 538).

Примечания:

Сразу после создания функцией rdsPBARCreate индикатор невидим. Для того, чтобы его окно появилось на экране, индикатору нужно дать команду RDS_PBAR_SHOW.

См. также:

rdsPBARCreate (стр. 538), rdsCommandObject (стр. 400),
rdsCommandObjectEx (стр. 400).

A.5.30.9. Макрос rdsPBARIncrement – увеличить текущую позицию индикатора

Макрос rdsPBARIncrement добавляет указанное число к текущей позиции индикатора выполнения.

```
rdsPBARIncrement (  
    bar,          // Вспомогательный объект-индикатор  
    delta         // Добавляемое значение  
)
```

Определение:

```
#define rdsPBARIncrement (bar, delta) \
    rdsSetObjectInt ((bar), RDS_PBAR_ADDTOPOS, 0, (delta))
```

Параметры:

bar

Идентификатор вспомогательного объекта для работы с индикатором, ранее созданного функцией rdsPBARCreate (стр. 538).

delta

Целое число, на которое нужно увеличить текущую позицию индикатора.

Примечания:

Этот макрос включает в себя вызов команды RDS_PBAR_ADDTOPOS (стр. 540).

См. также:

RDS_PBAR_ADDTOPOS (стр. 540).

А.5.30.10. Макрос rdsPBARSetPos – установить текущую позицию индикатора

Макрос rdsPBARSetPos позволяет устанавливать текущую позицию индикатора, определяющую степень его заполненности.

```
rdsPBARSetPos (
    bar,          // Вспомогательный объект-индикатор
    pos           // Позиция
)
```

Определение:

```
#define rdsPBARSetPos (bar, pos) \
    rdsSetObjectInt ((bar), RDS_PBAR_POSITION, 0, (pos))
```

Параметры:

bar

Идентификатор вспомогательного объекта для работы с индикатором, ранее созданного функцией rdsPBARCreate (стр. 538).

pos

Целое значение устанавливаемой позиции индикатора.

Примечания:

Этот макрос включает в себя вызов команды RDS_PBAR_POSITION (стр. 541).

См. также:

RDS_PBAR_POSITION (стр. 541).

А.5.30.11. Макрос rdsPBARShow – показать или скрыть индикатор

Макрос rdsPBARShow позволяет управлять видимостью окна индикатора на экране.

```
rdsPBARShow (
    bar,          // Вспомогательный объект-индикатор
    show          // Показать (TRUE) или скрыть (FALSE)
)
```


Определение:

```
#define rdsPBARShow(bar, show) \
    rdsCommandObject((bar), \
        (show)?RDS_PBAR_SHOW:RDS_PBAR_HIDE)
```

Параметры:

bar

Идентификатор вспомогательного объекта для работы с индикатором, ранее созданного функцией `rdsPBARCreate` (стр. 538).

show

TRUE – показать окно индикатора, FALSE – скрыть его.

Примечания:

Этот макрос передает в индикатор команду `RDS_PBAR_SHOW` (стр. 543) или `RDS_PBAR_HIDE` (стр. 540) в зависимости от параметра *show*.

См. также:

`RDS_PBAR_SHOW` (стр. 543), `RDS_PBAR_HIDE` (стр. 540).

А.5.31. Вспомогательный объект для панелей в окне подсистемы

Описываются функции и команды вспомогательного объекта РДС, предназначенного для создания внутри окна подсистемы независимых панелей, являющихся самостоятельными оконными объектами Windows (см. §2.10.4).

А.5.31.1. `rdsPANCreate` – создать объект для работы с панелью

Функция `rdsPANCreate` создает вспомогательный объект РДС, позволяющий модели блока открывать в окне родительской подсистемы этого блока независимо перемещаемую панель с произвольным содержимым.

```
RDS_HOBJECT RDSCALL rdsPANCreate(
    int Order,           // Близость к переднему плану
    int Left, int Top,   // Левый верхний угол
    int Width, int Height, // Размеры
    int Flags,           // Флаги (RDS_PAN_F_*)
    LPSTR Caption        // Заголовок или NULL
);
```

Тип указателя на эту функцию:

`RDS_HoIIIIIIIS`

Параметры:

Order

Целое число, определяющее близость панели к переднему плану (панели одного и того же блока с большим значением *Order* будут перекрывать панели с меньшим значением, перекрытие панелей разных блоков определяется взаимным расположением самих блоков).

Left, Top

Координаты левого верхнего угла (*Left* – горизонтальная, *Top* – вертикальная) панели в окне подсистемы в точках экрана в масштабе 100%.

Width, Height

Ширина (*Width*) и высота (*Height*) панели в точках экрана в масштабе 100%.

Flags

Набор битовых флагов, определяющих поведение и внешний вид панели:

RDS_PAN_F_BORDER	Панель имеет рельефную рамку.
RDS_PAN_F_CAPTION	У панели есть полоса заголовка в стиле окон Windows (только вместе с флагом RDS_PAN_F_BORDER).
RDS_PAN_F_HIDDEN	Панель создается невидимой. Видимость панели можно включить позже командой RDS_PAN_VISIBLE (стр. 554).
RDS_PAN_F_MOVEABLE	Пользователь может перемещать панель, перетаскивая ее мышью за заголовок (только вместе с флагом RDS_PAN_F_CAPTION).
RDS_PAN_F_NOBUTTON	В полосе заголовки панели нет кнопки закрытия (только вместе с флагом RDS_PAN_F_CAPTION). По умолчанию эта кнопка есть, и нажатие на нее скрывает панель.
RDS_PAN_F_PAINTMSG	Блок, модель которого создала панель, получает сообщения о необходимости перерисовать ее.
RDS_PAN_F_SCALABLE	Панель меняет свои размеры вместе с масштабом подсистемы.
RDS_PAN_F_SIZEABLE	Пользователь может изменять размеры панели, перетаскивая углы и стороны ее рамки (только вместе с флагом RDS_PAN_F_BORDER).

Caption

Указатель на строку, которая будет отображаться в полосе заголовка панели, или NULL, если панели не нужен текстовый заголовок.

Возвращаемое значение:

Уникальный идентификатор созданного объекта.

Примечания:

Эта функция создает вспомогательный объект, который будет отвечать за работу с панелью в окне родительской подсистемы блока, модель которого вызвала функцию. В отличие от других вспомогательных объектов, этот объект привязан к блоку, модель которого его создала – панель будет отображаться в родительской подсистеме именно этого блока, его модель будет получать уведомления о различных действиях с панелью через событие RDS_BFM_BLOCKPANEL (стр. 55) и т.п.

При создании объекта указываются координаты и размеры панели, ее близость к переднему плану и заголовок. Все эти параметры, кроме близости к переднему плану, могут быть позже изменены соответствующими командами.

Работа с панелями подробно описана в §2.10.4. Как правило, взаимодействие модели блока с объектом панели строится по следующему сценарию:

- вспомогательный объект создается функцией `rdSPANCreate`, при этом, если окно родительской подсистемы вызвавшего блока закрыто или при создании был указан флаг RDS_PAN_F_HIDDEN, сама панель не создается;
- как только окно подсистемы блока будет открыто (или как только видимость панели будет включена), РДС создаст внутри окна панель и уведомит об этом модель блока, передав ей дескриптор оконного объекта созданной панели типа `HWND` в параметрах события RDS_BFM_BLOCKPANEL;
- реагируя на уведомление о создании панели, модель блока размещает в ней поля ввода или другие объекты Windows;
- пока панель видима и окно подсистемы открыто, РДС уведомляет модель блока о перемещениях панели, изменении ее размера и необходимости перерисовки (если задан

флаг `RDS_PAN_F_PAINTMSG`) при помощи все того же события `RDS_BFM_BLOCKPANEL`;

- при закрытии окна подсистемы или скрытии панели РДС уведомит об этом модель блока, которая должна уничтожить все объекты Windows, созданные ей внутри панели, затем уничтожит оконный объект панели и будет ждать следующего открытия окна, чтобы снова создать в нем панель;
- когда панель перестанет быть нужна модели, модель удалит вспомогательный объект функцией `rdsDeleteObject`, при этом, если панель, связанная с этим объектом, была видима, перед уничтожением объекта она закроется автоматически (с уведомлением об этом модели блока).

Чаще всего панели используются для размещения в окнах подсистемы дополнительных объектов, не относящихся к РДС: стандартных полей ввода, областей вывода внешних библиотек и т.п. В §2.10.4 приводится пример построения на панели трехмерного изображения средствами библиотеки OpenGL.

См. также:

Вспомогательные объекты (стр. 399), `rdsDeleteObject` (стр. 401), `RDS_BFM_BLOCKPANEL` (стр. 55).

A.5.31.2. `rdsPANGetDescr` – получить описание панели

Функция `rdsPANGetDescr` заполняет переданную структуру описания панели.

```
BOOL RDCALL rdsPANGetDescr(  
    RDS_HOBJECT Panel,           // Панель  
    RDS_PPANDESCRIPTION pDescr // Структура описания  
);
```

Тип указателя на эту функцию:

`RDS_BHoPnd`

Параметры:

`Panel`

Идентификатор вспомогательного объекта-панели, ранее созданного функцией `rdsPANCreate` (стр. 545).

`pDescr`

Указатель на структуру описания панели `RDS_PANDESCRIPTION` (стр. 131), которую функция должна заполнить.

Возвращаемое значение:

`TRUE` – структура описания панели заполнена, `FALSE` – ошибка (`Panel` не является идентификатором объекта-панели или поле `servSize` структуры по указателю `pDescr` неверно инициализировано).

Примечания:

При помощи этой функции модель блока обычно получает различные параметры панели, включая идентификатор созданного для нее оконного объекта, если он есть. Пример использования функции `rdsPANGetDescr` приведен в §2.10.4.

См. также:

`rdsPANCreate` (стр. 545), `RDS_PANDESCRIPTION` (стр. 131).

А.5.31.3. Команда RDS_PAN_CAPTION – заголовок панели

Команда RDS_PAN_CAPTION устанавливает или возвращает заголовок панели.

Вызов команды для установки:

```
char *sCaption=... // Строка заголовка  
rdsSetObjectStr (Panel, RDS_PAN_CAPTION, 0, sCaption);
```

Вызов команды для чтения:

```
char *sCaption=rdsGetObjectStr (Panel, RDS_PAN_CAPTION, 0);
```

Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией rdsPANCreate (стр. 545).

sCaption

Указатель на строку с заголовком панели. При получении заголовка функция rdsGetObjectStr возвращает указатель на строку во внутренней памяти объекта, он будет действительным до следующего изменения заголовка.

Примечания:

Эта команда управляет строкой, которая отображается в полосе заголовка панели. Если у панели нет рамки или полосы заголовка, строка заголовка отображаться не будет, но она все равно будет храниться во внутренней памяти объекта, и ее можно будет получать и устанавливать.

См. также:

```
rdsPANCreate (стр. 545), rdsSetObjectStr (стр. 407),  
rdsGetObjectStr (стр. 405).
```

А.5.31.4. Команда RDS_PAN_CLIENTHEIGHT – высота внутренней, доступной для модели, части панели

Команда RDS_PAN_CLIENTHEIGHT позволяет считывать и устанавливать высоту внутренней части панели, то есть области внутри панели, в которой модель блока может размещать свои объекты. При наличии у панели рамки и полосы заголовка размеры внутренней части этой панели будут меньше ее общих размеров.

Вызов команды для установки:

```
int iClHeight=... // Высота внутренней части  
rdsSetObjectInt (Panel, RDS_PAN_CLIENTHEIGHT, 0, iClHeight);
```

Вызов команды для чтения:

```
int iClHeight=rdsGetObjectInt (Panel, RDS_PAN_CLIENTHEIGHT, 0);
```

Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией rdsPANCreate (стр. 545).

iClHeight

Высота внутренней части панели в точках экрана в масштабе 100%.

Примечания:

Если модель блока размещает на панели какие-либо объекты, она должна ориентироваться на размеры внутренней части этой панели, а не на общие ее размеры, поскольку часть панели может быть занята заголовком и рамкой.

См. также:

`rdsPANCreate` (стр. 545), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404), `RDS_PAN_CLIENTWIDTH` (стр. 549),
`RDS_PAN_HEIGHT` (стр. 550).

А.5.31.5. Команда `RDS_PAN_CLIENTWIDTH` – ширина внутренней, доступной для модели, части панели

Команда `RDS_PAN_CLIENTWIDTH` позволяет считывать и устанавливать ширину внутренней части панели, то есть области внутри панели, в которой модель блока может размещать свои объекты. При наличии у панели рамки и полосы заголовка размеры внутренней части этой панели будут меньше ее общих размеров.

Вызов команды для установки:

```
int iClWidth=... // Ширина внутренней части
rdsSetObjectInt (Panel, RDS_PAN_CLIENTWIDTH, 0, iClWidth);
```

Вызов команды для чтения:

```
int iClWidth=rdsGetObjectInt (Panel, RDS_PAN_CLIENTWIDTH, 0);
```

Параметры и результат:

`Panel`

Идентификатор вспомогательного объекта-панели, ранее созданного функцией `rdsPANCreate` (стр. 545).

`iClWidth`

Ширина внутренней части панели в точках экрана в масштабе 100%.

Примечания:

Если модель блока размещает на панели какие-либо объекты, она должна ориентироваться на размеры внутренней части этой панели, а не на общие ее размеры, поскольку часть панели может быть занята заголовком и рамкой.

См. также:

`rdsPANCreate` (стр. 545), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404), `RDS_PAN_CLIENTHEIGHT` (стр. 548),
`RDS_PAN_WIDTH` (стр. 555).

А.5.31.6. Команда `RDS_PAN_FLAGS` – флаги панели

Команда `RDS_PAN_FLAGS` позволяет считывать и устанавливать флаги панели (`RDS_PAN_F_*`, см. стр. 545), определяющие внешний вид и поведение панели.

Вызов команды для установки:

```
int iFlags=... // Флаги
rdsSetObjectInt (Panel, RDS_PAN_FLAGS, 0, iFlags);
```

Вызов команды для чтения:

```
int iFlags=rdsGetObjectInt (Panel, RDS_PAN_FLAGS, 0);
```

Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией `rdsPANCreate` (стр. 545).

iFlags

Набор битовых флагов панели.

Примечания:

При открытой родительской подсистеме блока, модель которого передает команду объекту, изменение флага `RDS_PAN_F_HIDDEN` будет приводить к открытию и закрытию панели согласно значению флага.

См. также:

`rdsPANCreate` (стр. 545), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404).

A.5.31.7. Команда `RDS_PAN_HEIGHT` – общая высота панели

Команда `RDS_PAN_HEIGHT` позволяет считывать и устанавливать общую высоту панели, в которую может входить рамка и полоса заголовка, если они у панели есть.

Вызов команды для установки:

```
int iHeight=... // Высота
rdsSetObjectInt (Panel, RDS_PAN_HEIGHT, 0, iHeight);
```

Вызов команды для чтения:

```
int iHeight=rdsGetObjectInt (Panel, RDS_PAN_HEIGHT, 0);
```

Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией `rdsPANCreate` (стр. 545).

iHeight

Общая высота панели в точках экрана в масштабе 100% (в нее входят и элементы оформления панели).

Примечания:

Если модель блока размещает на панели какие-либо объекты, она должна ориентироваться на размеры внутренней части этой панели, а не на общие ее размеры, поскольку часть панели может быть занята заголовком и рамкой.

Пример использования команды `RDS_PAN_HEIGHT` приведен в §2.10.4.

См. также:

`rdsPANCreate` (стр. 545), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404), `RDS_PAN_CLIENTHEIGHT` (стр. 548),
`RDS_PAN_WIDTH` (стр. 555).

A.5.31.8. Команда `RDS_PAN_LEFT` – горизонтальная координата панели

Команда `RDS_PAN_LEFT` позволяет считывать и устанавливать горизонтальную координату левого верхнего угла панели.

Вызов команды для установки:

```
int iLeft=... // Координата  
rdsSetObjectInt (Panel, RDS_PAN_LEFT, 0, iLeft);
```

Вызов команды для чтения:

```
int iLeft=rdsGetObjectInt (Panel, RDS_PAN_LEFT, 0);
```

Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией rdsPANCreate (стр. 545).

iLeft

Горизонтальная координата левого верхнего угла панели в точках экрана в масштабе 100%.

Примечания:

Вместе с командой RDS_PAN_TOP (стр. 554) эта команда позволяет узнавать текущее положение панели и перемещать ее внутри окна подсистемы. Пример использования команды приведен в §2.10.4.

См. также:

rdsPANCreate (стр. 545), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), RDS_PAN_TOP (стр. 554).

A.5.31.9. Команда RDS_PAN_MAXCLHEIGHT – максимальная высота внутренней части панели

Команда RDS_PAN_MAXCLHEIGHT позволяет считывать и устанавливать максимально допустимую высоту внутренней части панели, то есть области внутри панели, в которой модель блока может размещать свои объекты.

Вызов команды для установки:

```
int iMaxHeight=... // Высота внутренней части  
rdsSetObjectInt (Panel, RDS_PAN_MAXCLHEIGHT, 0, iMaxHeight);
```

Вызов команды для чтения:

```
int iMaxHeight=rdsGetObjectInt (Panel, RDS_PAN_MAXCLHEIGHT, 0);
```

Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией rdsPANCreate (стр. 545).

iMaxHeight

Максимально допустимая высота внутренней части панели в точках экрана, или -1, если высоту внутренней части ограничивать не нужно.

Примечания:

При установленном ограничении на высоту внутренней части панели ее размер нельзя будет сделать больше указанного значения. Если на момент установки ограничения размер панели превышает его, размер уменьшится до заданного значения.

См. также:

`rdsPANCreate` (стр. 545), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404), `RDS_PAN_CLIENTHEIGHT` (стр. 548),
`RDS_PAN_MINCLHEIGHT` (стр. 552), `RDS_PAN_MAXCLWIDTH` (стр. 552),
`RDS_PAN_MINCLWIDTH` (стр. 553).

A.5.31.10. Команда `RDS_PAN_MAXCLWIDTH` – максимальная ширина внутренней части панели

Команда `RDS_PAN_MAXCLWIDTH` позволяет считывать и устанавливать максимально допустимую ширину внутренней части панели, то есть области внутри панели, в которой модель блока может размещать свои объекты.

Вызов команды для установки:

```
int iMaxWidth=... // Ширина внутренней части
rdsSetObjectInt (Panel, RDS_PAN_MAXCLWIDTH, 0, iMaxWidth);
```

Вызов команды для чтения:

```
int iMaxWidth=rdsGetObjectInt (Panel, RDS_PAN_MAXCLWIDTH, 0);
```

Параметры и результат:

`Panel`

Идентификатор вспомогательного объекта-панели, ранее созданного функцией `rdsPANCreate` (стр. 545).

`iMaxWidth`

Максимально допустимая ширина внутренней части панели в точках экрана, или –1, если ширину внутренней части ограничивать не нужно.

Примечания:

При установленном ограничении на ширину внутренней части панели ее размер нельзя будет сделать больше указанного значения. Если на момент установки ограничения размер панели превышает его, размер уменьшится до заданного значения.

См. также:

`rdsPANCreate` (стр. 545), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404), `RDS_PAN_CLIENTHEIGHT` (стр. 548),
`RDS_PAN_MINCLHEIGHT` (стр. 552), `RDS_PAN_MAXCLHEIGHT` (стр. 551),
`RDS_PAN_MINCLWIDTH` (стр. 553).

A.5.31.11. Команда `RDS_PAN_MINCLHEIGHT` – минимальная высота внутренней части панели

Команда `RDS_PAN_MINCLHEIGHT` позволяет считывать и устанавливать минимально допустимую высоту внутренней части панели, то есть области внутри панели, в которой модель блока может размещать свои объекты.

Вызов команды для установки:

```
int iMinHeight=... // Высота внутренней части
rdsSetObjectInt (Panel, RDS_PAN_MINCLHEIGHT, 0, iMinHeight);
```

Вызов команды для чтения:

```
int iMinHeight=rdsGetObjectInt (Panel, RDS_PAN_MINCLHEIGHT, 0);
```


Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией `rdsPANCreate` (стр. 545).

iMinHeight

Минимально допустимая высота внутренней части панели в точках экрана, или `-1`, если высоту внутренней части ограничивать не нужно.

Примечания:

При установленном ограничении на высоту внутренней части панели ее размер нельзя будет сделать меньше указанного значения. Если на момент установки ограничения размер панели меньше него, размер увеличится до заданного значения.

См. также:

`rdsPANCreate` (стр. 545), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404), `RDS_PAN_CLIENTHEIGHT` (стр. 548),
`RDS_PAN_MAXCLHEIGHT` (стр. 551), `RDS_PAN_MAXCLWIDTH` (стр. 552),
`RDS_PAN_MINCLWIDTH` (стр. 553).

A.5.31.12. Команда `RDS_PAN_MINCLWIDTH` – минимальная ширина внутренней части панели

Команда `RDS_PAN_MINCLWIDTH` позволяет считывать и устанавливать минимально допустимую ширину внутренней части панели, то есть области внутри панели, в которой модель блока может размещать свои объекты.

Вызов команды для установки:

```
int iMinWidth=... // Ширина внутренней части
rdsSetObjectInt(Panel,RDS_PAN_MINCLWIDTH,0,iMinWidth);
```

Вызов команды для чтения:

```
int iMinWidth=rdsGetObjectInt(Panel,RDS_PAN_MINCLWIDTH,0);
```

Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией `rdsPANCreate` (стр. 545).

iMinWidth

Минимально допустимая ширина внутренней части панели в точках экрана, или `-1`, если ширину внутренней части ограничивать не нужно.

Примечания:

При установленном ограничении на ширину внутренней части панели ее размер нельзя будет сделать меньше указанного значения. Если на момент установки ограничения размер панели меньше него, размер увеличится до заданного значения.

См. также:

`rdsPANCreate` (стр. 545), `rdsSetObjectInt` (стр. 407),
`rdsGetObjectInt` (стр. 404), `RDS_PAN_CLIENTHEIGHT` (стр. 548),
`RDS_PAN_MAXCLHEIGHT` (стр. 551), `RDS_PAN_MAXCLWIDTH` (стр. 552),
`RDS_PAN_MINCLHEIGHT` (стр. 552).

A.5.31.13. Команда RDS_PAN_TOP – вертикальная координата панели

Команда RDS_PAN_TOP позволяет считывать и устанавливать вертикальную координату левого верхнего угла панели.

Вызов команды для установки:

```
int iTop=... // Координата
rdsSetObjectInt (Panel, RDS_PAN_TOP, 0, iTop);
```

Вызов команды для чтения:

```
int iTop=rdsGetObjectInt (Panel, RDS_PAN_TOP, 0);
```

Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией rdsPANCreate (стр. 545).

iTop

Вертикальная координата левого верхнего угла панели в точках экрана в масштабе 100%.

Примечания:

Вместе с командой RDS_PAN_LEFT (стр. 550) эта команда позволяет узнавать текущее положение панели и перемещать ее внутри окна подсистемы. Пример использования команды приведен в §2.10.4.

См. также:

rdsPANCreate (стр. 545), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), RDS_PAN_LEFT (стр. 550).

A.5.31.14. Команда RDS_PAN_VISIBLE – видимость панели

Команда RDS_PAN_VISIBLE позволяет показывать и скрывать панель, а также узнавать, скрыта ли она в данный момент.

Вызов команды для установки:

```
int iShow=... // 1 - показать, 0 - скрыть
rdsSetObjectInt (Panel, RDS_PAN_VISIBLE, 0, iShow);
```

Вызов команды для чтения:

```
int iShow=rdsGetObjectInt (Panel, RDS_PAN_VISIBLE, 0);
```

Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией rdsPANCreate (стр. 545).

iShow

Целое число (int), указывающее на состояние панели: 1 – панель показывается, 0 – панель скрыта.

Примечания:

Эта команда не всегда управляет непосредственной видимостью панели. Если окно подсистемы с блоком-владельцем панели открыто, этой командой можно показывать и скрывать панель и узнавать, видима ли она для пользователя в данный момент. Однако, если

окно подсистемы закрыто, эта команда переключает и возвращает *желаемое* состояние панели, то есть состояние, в которое панель перейдет при открытии окна. Если, например, включить видимость панели в закрытом окне, а потом открыть окно, панель автоматически покажется внутри окна, при этом даже при закрытом окне, когда пользователь не видит панели, команда RDS_PAN_VISIBLE будет возвращать для данной панели единицу.

Если вызывающей программе нужно узнать, открыта ли в действительности данная панель, следует вызвать функцию rdsPANGetDescr (стр. 547) для заполнения структуры описания панели RDS_PANDESCRIPTION (стр. 131) и сравнить дескриптор окна панели (поле Handle) с NULL: если дескриптор не нулевой, значит, для панели создан оконный объект, то есть панель физически существует в открытом окне подсистемы.

Пример использования команды RDS_PAN_VISIBLE приведен в §2.10.4.

См. также:

```
rdsPANCreate (стр. 545), rdsSetObjectInt (стр. 407),  
rdsGetObjectInt (стр. 404), rdsPANGetDescr (стр. 547).
```

A.5.31.15. Команда RDS_PAN_WIDTH – общая ширина панели

Команда RDS_PAN_WIDTH позволяет считывать и устанавливать общую ширину панели, в которую может входить рамка, если они у панели есть.

Вызов команды для установки:

```
int iWidth=... // Ширина  
rdsSetObjectInt (Panel, RDS_PAN_WIDTH, 0, iWidth);
```

Вызов команды для чтения:

```
int iWidth=rdsGetObjectInt (Panel, RDS_PAN_WIDTH, 0);
```

Параметры и результат:

Panel

Идентификатор вспомогательного объекта-панели, ранее созданного функцией rdsPANCreate (стр. 545).

iWidth

Общая ширина панели в точках экрана в масштабе 100% (в нее входят и элементы оформления панели).

Примечания:

Если модель блока размещает на панели какие-либо объекты, она должна ориентироваться на размеры внутренней части этой панели, а не на общие ее размеры, поскольку часть панели может быть занята заголовком и рамкой.

Пример использования команды RDS_PAN_WIDTH приведен в §2.10.4.

См. также:

```
rdsPANCreate (стр. 545), rdsSetObjectInt (стр. 407),  
rdsGetObjectInt (стр. 404), RDS_PAN_CLIENTWIDTH (стр. 549),  
RDS_PAN_HEIGHT (стр. 550).
```

А.5.32. Вспомогательный объект для работы с форматом CSV

Описываются функции и команды вспомогательного объекта РДС, предназначенного для создания и разбора текста в формате CSV, то есть набора текстовых значений, разделенных запятыми.

А.5.32.1. `rdsCSVCreate` – создать объект для работы с текстом в формате CSV

Функция `rdsCSVCreate` создает вспомогательный объект РДС, с помощью которого можно работать с текстом формата CSV (comma separated values), то есть с текстом, каждая строка которого содержит набор значений, отделенных друг от друга запятой или другим символом-разделителем.

```
RDS_ОБЪЕКТ RDSCALL rdsCSVCreate(void);
```

Тип указателя на эту функцию:

```
RDS_НоV
```

Возвращаемое значение:

Уникальный идентификатор созданного объекта.

Примечания:

Эта функция создает вспомогательный объект, с помощью которого можно как разбирать текст в формате CSV, так и формировать его. Такие тексты удобны для простого хранения различных матриц и списков. Элементами текста, с которыми работает объект, являются строки, поэтому если нужно работать с целыми или вещественными числами, преобразовывать строки в числа и числа в строки нужно вручную. Элементы разделяются запятыми. Если элемент текста содержит пробелы, он должен быть заключен в двойные кавычки, если элемент содержит двойные кавычки, символ двойной кавычки в нем удваивается. Например, текст

```
Строка1, "alpha, beta", "строка"с"кавычкой", "Строка 2"  
10.0,      20.0, 30.0  
1, , "2 3"
```

будет состоять из следующих элементов:

	<i>Столбец 0</i>	<i>Столбец 1</i>	<i>Столбец 2</i>	<i>Столбец 3</i>
<i>Строка 0</i>	Строка1	alpha, beta	строка"с"кавычкой	Строка 2
<i>Строка 1</i>	10.0	20.0	30.0	
<i>Строка 2</i>	1		2 3	

При желании, командой `RDS_CSV_DELIMITERCHAR` (стр. 560) вместо запятой можно установить другой символ-разделитель, а командой `RDS_CSV_QUOTECHAR` (стр. 566) заменить на другой символ двойную кавычку, используемую для выделения строк с пробелами.

При разборе текста можно загрузить в объект весь текст из какого-либо файла (команда `RDS_CSV_LOADFROMFILE`, стр. 563), считывать его из файла построчно (команда `RDS_CSV_STRFROMFILE`, стр. 567) или передать в объект уже загруженный в память текст (команда `RDS_CSV_TEXT`, стр. 568). При этом функцией `rdsCSVGetItem` (стр. 557) можно получить элемент текста, находящийся в заданной строке и заданном столбце. При формировании текста можно записывать элементы в заданную строку и столбец текста функцией `rdsCSVSetItem` (стр. 558), а уже сформированный текст записывать в файл целиком (команда `RDS_CSV_SAVETOFILE`, стр. 566) или построчно (команда

RDS_CSV_STRTOFIELD, стр. 567), а также получать его динамически сформированную копию (команда RDS_CSV_TEXT).

Созданный вспомогательный объект будет существовать до тех пор, пока схема не будет выгружена из памяти, или пока он не будет удален вызовом функции rdsDeleteObject (стр. 401).

Пример использования функции rdsCSVCreate приведен в §2.16.2.

См. также:

Вспомогательные объекты (стр. 399), rdsDeleteObject (стр. 401).

A.5.32.2. rdsCSVGetItem – получить элемент текста

Функция rdsCSVGetItem возвращает элемент текста CSV с указанными номерами строки и столбца.

```
LPSTR RDSCALL rdsCSVGetItem(  
    RDS_HOBJECT Csv,          // Объект  
    int Line,                 // Номер строки  
    int Col                   // Номер столбца  
);
```

Тип указателя на эту функцию:

RDS_SHoII

Параметры:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

Line

Номер строки (строки нумеруются, начиная с нуля).

Col

Номер столбца (столбцы нумеруются, начиная с нуля).

Возвращаемое значение:

Указатель на строку значения с индексами (Line,Col) во внутренней памяти объекта. Вызывающая программа не должна изменять эту строку. Если значения с такими номерами строки и столбца нет в тексте, функция возвращает указатель на пустую строку. Результатом возврата этой функции не может быть NULL, поэтому его можно использовать в любых строковых функциях без дополнительных проверок.

Примечания:

Элемент текста с указанными индексами всегда возвращается в виде строки. Преобразовывать эту строку в число, если это требуется по смыслу разбираемого текста, должна вызывающая программа.

Пример использования функции rdsCSVGetItem приведен в §2.16.2.

См. также:

rdsCSVCreate (стр. 556), rdsCSVSetItem (стр. 558),
RDS_CSV_LINECOUNT (стр. 563), RDS_CSV_MAXCOLUMNS (стр. 564).

A.5.32.3. rdsCSVSetItem – установить элемент текста

Функция `rdsCSVSetItem` устанавливает элемент текста CSV с указанными номерами строки и столбца.

```
void RDSCALL rdsCSVSetItem(  
    RDS_HOБJEKT Csv,          // Объект  
    int Line,                 // Номер строки  
    int Col,                  // Номер столбца  
    LPSTR Value               // Значение элемента  
);
```

Тип указателя на эту функцию:

`RDS_VHоIIS`

Параметры:

`Csv`

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией `rdsCSVCreate` (стр. 556).

`Line`

Номер строки (строки нумеруются, начиная с нуля).

`Col`

Номер столбца (столбцы нумеруются, начиная с нуля).

`Value`

Указатель на строку, которую нужно поместить в позицию текста (`Line,Col`). Если в тексте недостаточно строк или в строке недостаточно значений, текст автоматически будет дополнен пустыми элементами перед позицией (`Line,Col`).

Примечания:

Эта функция успешно выполнится независимо от того, сколько строк в данный момент находится в тексте внутри объекта и сколько элементов находится в каждой строке. Например, если для пустого, не содержащего ни одного элемента текста вызвать функцию `rdsCSVSetItem(Csv,2,1,"ABCD")`, в текст будет добавлено две пустые строки с индексами 0 и 1, и строка с индексом 2, содержащая пустой элемент в нулевой колонке и текст “ABCD” в первой:

	<i>Столбец 0</i>	<i>Столбец 1</i>
<i>Строка 0</i>		
<i>Строка 1</i>		
<i>Строка 2</i>		ABCD

Если после этого вызвать функцию `rdsCSVSetItem(Csv,1,0,"EFGH")`, текст внутри объекта примет следующий вид:

	<i>Столбец 0</i>	<i>Столбец 1</i>
<i>Строка 0</i>		
<i>Строка 1</i>	EFGH	
<i>Строка 2</i>		ABCD

См. также:

rdsCSVCreate (стр. 556), rdsCSVGetItem (стр. 557), RDS_CSV_LINE (стр. 561), RDS_CSV_TEXT (стр. 568).

А.5.32.4. Команда RDS_CSV_CLEAR – очистка текста

Команда RDS_CSV_CLEAR очищает текст в указанном объекте, удаляя из него все элементы.

Вызов команды:

```
rdsCommandObject (Csv, RDS_CSV_CLEAR) ;  
или  
rdsCommandObjectEx (Csv, RDS_CSV_CLEAR, 0, NULL) ;
```

Параметр:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

См. также:

rdsCSVCreate (стр. 556), rdsCommandObject (стр. 400),
rdsCommandObjectEx (стр. 400).

А.5.32.5. Команда RDS_CSV_CLOSEFILE – закрыть файл

Команда RDS_CSV_CLOSEFILE закрывает файл, ранее открытый командой RDS_CSV_OPENFILEREAD (стр. 564) или RDS_CSV_OPENFILEWRITE (стр. 565).

Вызов команды:

```
rdsCommandObject (Csv, RDS_CSV_CLOSEFILE) ;  
или  
rdsCommandObjectEx (Csv, RDS_CSV_CLOSEFILE, 0, NULL) ;
```

Параметр:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

Примечания:

Открытие файла для чтения (RDS_CSV_OPENFILEREAD) или для записи (RDS_CSV_OPENFILEWRITE) используется при построчном чтении или записи данных командами RDS_CSV_STRFROMFILE (стр. 567) и RDS_CSV_STRTOFILE (стр. 567) соответственно. После того, как все необходимые данные считаны или записаны, файл можно закрыть командой RDS_CSV_CLOSEFILE. При удалении объекта файл закрывается автоматически, поэтому команду RDS_CSV_CLOSEFILE в этом случае можно не давать.

См. также:

rdsCSVCreate (стр. 556), rdsCommandObject (стр. 400),
rdsCommandObjectEx (стр. 400), RDS_CSV_OPENFILEREAD (стр. 564),
RDS_CSV_OPENFILEWRITE (стр. 565), RDS_CSV_STRFROMFILE (стр. 567),
RDS_CSV_STRTOFILE (стр. 567).

A.5.32.6. Команда RDS_CSV_DELIMITERCHAR – символ-разделитель

Команда RDS_CSV_DELIMITERCHAR позволяет считывать и устанавливать код символа-разделителя, используемого для отделения значений в строке друг от друга. По умолчанию для этого используется запятая.

Вызов команды для установки:

```
int iCharCode=... // ASCII-код символа-разделителя  
rdsSetObjectInt (Csv,RDS_CSV_DELIMITERCHAR,0,iCharCode);
```

Вызов команды для чтения:

```
int iCharCode=rdsGetObjectInt (Csv,RDS_CSV_DELIMITERCHAR,0);
```

Параметры и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

iCharCode

Код символа, используемого для разделения значений (внутри объекта это целое число преобразуется в char).

См. также:

```
rdsCSVCreate (стр. 556), rdsSetObjectInt (стр. 407),  
rdsGetObjectInt (стр. 404), RDS_CSV_QUOTECHAR (стр. 566).
```

A.5.32.7. Команда RDS_CSV_FILEERROR – проверка ошибки в последней операции с файлом

Команда RDS_CSV_FILEERROR позволяет проверить, успешно ли завершилась последняя операция объекта с файлом.

Вызов команды:

```
int iError=rdsGetObjectInt (Csv,RDS_CSV_FILEERROR,0);
```

Параметр и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

iError

Нулевое значение – ошибок не было, отличное от нуля значение – последняя операция выполнена с ошибкой.

Примечания:

Эта команда используется для проверки результата выполнения команд работы с файлом: RDS_CSV_STRFROMFILE (стр. 567), RDS_CSV_STRTOFILE (стр. 567), RDS_CSV_OPENFILEREAD (стр. 564), RDS_CSV_OPENFILEWRITE (стр. 565), RDS_CSV_LOADFROMFILE (стр. 563) и RDS_CSV_SAVETOFILE (стр. 566). Не во всех этих командах предусмотрен возврат результата, поэтому для того, чтобы узнать, успешно ли выполнена такая команда, после нее следует вызвать RDS_CSV_FILEERROR.

См. также:

```
rdsCSVCreate (стр. 556), rdsSetObjectInt (стр. 407),  
rdsGetObjectInt (стр. 404), RDS_CSV_FILEISOPEN (стр. 561).
```


А.5.32.8. Команда RDS_CSV_FILEISOPEN – проверка успешности открытия файла для чтения или записи

Команда RDS_CSV_FILEISOPEN позволяет проверить, успешно ли открыт файл командами RDS_CSV_OPENFILEREAD (стр. 564) и RDS_CSV_OPENFILEWRITE (стр. 565).

Вызов команды:

```
int iOpen=rdsGetObjectInt(Csv,RDS_CSV_FILEISOPEN,0);
```

Параметр и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

iOpen

Нулевое значение – файл не открыт (при открытии произошла ошибка – например, файл с указанным именем отсутствует), отличное от нуля значение – файл был успешно открыт.

Примечания:

Эта команда используется для проверки результата выполнения команд открытия файла RDS_CSV_OPENFILEREAD и RDS_CSV_OPENFILEWRITE. В этих командах не предусмотрен возврат результата, поэтому для того, чтобы узнать, успешно ли открылся файл, после них следует вызвать RDS_CSV_FILEISOPEN.

Пример использования этой команды приведен в §2.16.2.

См. также:

```
rdsCSVCreate (стр. 556), rdsSetObjectInt (стр. 407),  
rdsGetObjectInt (стр. 404), RDS_CSV_OPENFILEREAD (стр. 564),  
RDS_CSV_OPENFILEWRITE (стр. 565), RDS_CSV_FILEERROR (стр. 560).
```

А.5.32.9. Команда RDS_CSV_LINE – одна строка текста

Команда RDS_CSV_LINE устанавливает или возвращает строку текста с указанным номером в указанном объекте (в этой строке содержатся элементы текста, разделенные запятыми).

Вызов команды для установки:

```
int iStrNum=... // Номер строки  
char *sStrText=... // Текст строки  
rdsSetObjectStr(Csv,RDS_CSV_LINE,iStrNum,sStrText);
```

Вызов команды для чтения:

```
int iStrNum=... // Номер строки  
char *sStrText=rdsGetObjectStr(Csv,RDS_CSV_LINE,iStrNum);
```

Параметры и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

iStrNum

Номер строки текста, которую нужно установить или получить. Строки нумеруются начиная с нуля.

sStrText

Указатель на строку, содержащую текст строки (набор элементов, разделенных запятыми). При получении текста строки вызовом `rdsGetObjectStr` (стр. 405) этот текст формируется динамически, поэтому после использования он **обязательно** должен быть освобожден вызовом `rdsFree` (стр. 187). Если в строке не содержится ни одного элемента, функция вернет `NULL`.

Примечания:

Эта команда позволяет работать с текстом в формате CSV построчно, не разбивая каждую строку на отдельные элементы.

Следует помнить, что, поскольку объект хранит текст уже разобранным, для того, чтобы вернуть строку с заданным номером, ему приходится заново собирать ее из отдельных значений. По этой причине при выполнении команды `RDS_CSV_LINE` для получения строки объект формирует ее в динамической памяти, и обязанность освобождения этой памяти вызовом `rdsFree` лежит на вызывавшей программе.

См. также:

`rdsCSVCreate` (стр. 556), `rdsSetObjectStr` (стр. 407),
`rdsGetObjectStr` (стр. 405), `rdsFree` (стр. 187), `RDS_CSV_TEXT` (стр. 568).

A.5.32.10. Команда `RDS_CSV_LINECOLUMNS` – число элементов в строке

Команда `RDS_CSV_LINECOLUMNS` возвращает общее число значений в строке с указанным номером.

Вызов команды:

```
int iStrNum=... // Номер строки
int iCount=rdsGetObjectInt(Csv,RDS_CSV_LINECOLUMNS,iStrNum);
```

Параметры и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией `rdsCSVCreate` (стр. 556).

iStrNum

Номер строки текста (строки нумеруются начиная с нуля).

iCount

Число элементов в строке `iStrNum`.

Примечания:

Эта команда возвращает общее число элементов в строке `iStrNum`, включая пустые (фактически, возвращается число, на единицу большее индекса последнего элемента строки). Число элементов в разных строках может отличаться, поскольку вспомогательный объект РДС для работы с текстом в формате CSV не требует одинаковости числа элементов во всех строках текста – вместо этого функция `rdsCSVGetItem` (стр. 557) автоматически возвращает пустые строки при попытке получить несуществующий элемент строки. Если вызывающей программе требуется узнать число элементов в самой длинной строке текста, следует использовать команду `RDS_CSV_MAXCOLUMNS` (стр. 564).

См. также:

`rdsCSVCreate` (стр. 556), `rdsGetObjectInt` (стр. 404),
`rdsCSVGetItem` (стр. 557), `RDS_CSV_MAXCOLUMNS` (стр. 564),
`RDS_CSV_LINECOUNT` (стр. 563).

A.5.32.11. Команда RDS_CSV_LINECOUNT – число строк в тексте

Команда RDS_CSV_LINECOUNT возвращает общее число строк в тексте CSV.

Вызов команды:

```
int iCount=rdsGetObjectInt (Csv,RDS_CSV_LINECOUNT,0);
```

Параметр и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

iCount

Число строк в тексте, включая пустые.

См. также:

rdsCSVCreate (стр. 556), rdsGetObjectInt (стр. 404),
rdsCSVGetItem (стр. 557), RDS_CSV_MAXCOLUMNS (стр. 564),
RDS_CSV_LINECOLUMNS (стр. 562).

A.5.32.12. Команда RDS_CSV_LOADFROMFILE – загрузить текст из файла

Команда RDS_CSV_LOADFROMFILE загружает в указанный объект текст из файла с указанным именем.

Вызов команды:

```
char *sFileName=... // Имя файла  
rdsSetObjectStr (Csv,RDS_CSV_LOADFROMFILE,0,sFileName);
```

Параметры:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

sFileName

Указатель на строку с именем файла, из которого нужно загрузить текст. Имя файла может содержать символические обозначения стандартных папок РДС (стр. 189). Если в имени файла нет пути, он будет считаться находящимся в одной папке с загруженной схемой.

Примечания:

Эта команда загружает в объект Csv весь текст из файла sFileName и разбирает его на отдельные строки и элементы. После этого к элементам текста можно обращаться при помощи функции rdsCSVGetItem (стр. 557). Для получения общего числа строк в загруженном тексте следует использовать команду RDS_CSV_LINECOUNT (стр. 563), для получения общего (максимального) числа элементов в строках – команду RDS_CSV_MAXCOLUMNS (стр. 564).

В команде RDS_CSV_LOADFROMFILE не предусмотрен возврат результата операции, поэтому для того, чтобы узнать, удалось ли загрузить текст из файла, следует выполнить команду RDS_CSV_FILEERROR (стр. 560).

См. также:

rdsCSVCreate (стр. 556), rdsSetObjectStr (стр. 407),
rdsCSVGetItem (стр. 557), RDS_CSV_LINECOUNT (стр. 563),

RDS_CSV_MAXCOLUMNS (стр. 564), RDS_CSV_FILEERROR (стр. 560),
RDS_CSV_TEXT (стр. 568).

А.5.32.13. Команда RDS_CSV_MAXCOLUMNS – число элементов в самой длинной строке

Команда RDS_CSV_MAXCOLUMNS возвращает самое большее из чисел элементов в строках текста в указанном объекте.

Вызов команды:

```
int iMaxCount=rdsGetObjectInt(Csv,RDS_CSV_MAXCOLUMNS,0);
```

Параметр и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

iMaxCount

Самое большее число элементов в строке.

Примечания:

Эта команда ищет в объекте строку с наибольшим числом элементов и возвращает это число. Число элементов в разных строках может отличаться, поскольку вспомогательный объект РДС для работы с текстом в формате CSV не требует одинаковости числа элементов во всех строках текста – вместо этого функция rdsCSVGetItem (стр. 557) автоматически возвращает пустые строки при попытке получить несуществующий элемент строки. Если рассматривать текст в формате CSV как матрицу, то число, возвращенное командой RDS_CSV_MAXCOLUMNS, можно считать числом ее столбцов.

См. также:

rdsCSVCreate (стр. 556), rdsGetObjectInt (стр. 404),
rdsCSVGetItem (стр. 557), RDS_CSV_LINECOLUMNS (стр. 562),
RDS_CSV_LINECOUNT (стр. 563).

А.5.32.14. Команда RDS_CSV_OPENFILEREAD – открыть файл для чтения

Команда RDS_CSV_OPENFILEREAD открывает указанный файл для чтения из него строк в объект при помощи команды RDS_CSV_STRFROMFILE (стр. 567).

Вызов команды:

```
char *sFileName=... // Имя файла  
rdsSetObjectStr(Csv,RDS_CSV_OPENFILEREAD,0,sFileName);
```

Параметры:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

sFileName

Указатель на строку с именем файла. Имя файла может содержать символические обозначения стандартных папок РДС (стр. 189). Если в имени файла нет пути, он будет считаться находящимся в одной папке с загруженной схемой.

Примечания:

По этой команде объект Csv открывает файл sFileName для построчного чтения командой RDS_CSV_STRFROMFILE. При обработке файлов большого размера такой режим может оказаться удобнее, чем загрузка всего файла в память командой RDS_CSV_LOADFROMFILE (стр. 563).

В команде RDS_CSV_OPENFILEREAD не предусмотрен возврат результата операции, поэтому для того, чтобы узнать, удалось ли открыть файл, следует выполнить команду RDS_CSV_FILEISOPEN (стр. 561). Для закрытия файла следует использовать команду RDS_CSV_CLOSEFILE (стр. 559).

Пример использования этой команды приведен в §2.16.2.

См. также:

rdsCSVCreate (стр. 556), rdsSetObjectStr (стр. 407),
RDS_CSV_CLOSEFILE (стр. 559), RDS_CSV_STRFROMFILE (стр. 567),
RDS_CSV_FILEISOPEN (стр. 561), RDS_CSV_FILEERROR (стр. 560),
RDS_CSV_LOADFROMFILE (стр. 563), RDS_CSV_OPENFILEWRITE (стр. 565).

А.5.32.15. Команда RDS_CSV_OPENFILEWRITE – открыть файл для записи

Команда RDS_CSV_OPENFILEWRITE открывает указанный файл для записи в него строк при помощи команды RDS_CSV_STRTOFILE (стр. 567).

Вызов команды:

```
char *sFileName=... // Имя файла  
rdsSetObjectStr(Csv, RDS_CSV_OPENFILEWRITE, 0, sFileName);
```

Параметры:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

sFileName

Указатель на строку с именем файла. Имя файла может содержать символические обозначения стандартных папок РДС (стр. 189). Если в имени файла нет пути, он будет создан в одной папке с загруженной схемой.

Примечания:

По этой команде объект Csv открывает файл sFileName для построчной записи командой RDS_CSV_STRTOFILE. При обработке файлов большого размера такой режим может оказаться удобнее, чем формирование всего текста в памяти объекта и сохранение его в файл командой RDS_CSV_SAVETOFILE (стр. 566).

В команде RDS_CSV_OPENFILEWRITE не предусмотрен возврат результата операции, поэтому для того, чтобы узнать, удалось ли открыть файл, следует выполнить команду RDS_CSV_FILEISOPEN (стр. 561). Для закрытия файла следует использовать команду RDS_CSV_CLOSEFILE (стр. 559).

См. также:

rdsCSVCreate (стр. 556), rdsSetObjectStr (стр. 407),
RDS_CSV_CLOSEFILE (стр. 559), RDS_CSV_STRTOFILE (стр. 567),
RDS_CSV_FILEISOPEN (стр. 561), RDS_CSV_FILEERROR (стр. 560),
RDS_CSV_SAVETOFILE (стр. 566), RDS_CSV_OPENFILEREAD (стр. 564).

А.5.32.16. Команда RDS_CSV_QUOTECHAR – символ-ограничитель строки

Команда RDS_CSV_QUOTECHAR позволяет считывать и устанавливать код символа-ограничителя, используемого для указания границ строк с пробелами. По умолчанию для этого используется двойная кавычка.

Вызов команды для установки:

```
int iCharCode=... // ASCII-код символа-ограничителя
rdsSetObjectInt (Csv, RDS_CSV_QUOTECHAR, 0, iCharCode);
```

Вызов команды для чтения:

```
int iCharCode=rdsGetObjectInt (Csv, RDS_CSV_QUOTECHAR, 0);
```

Параметры и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

iCharCode

Код символа, используемого для ограничения строк (внутри объекта это целое число преобразуется в char).

См. также:

rdsCSVCreate (стр. 556), rdsSetObjectInt (стр. 407),
rdsGetObjectInt (стр. 404), RDS_CSV_DELIMITERCHAR (стр. 560).

А.5.32.17. Команда RDS_CSV_SAVETOFILE – записать текст в файл

Команда RDS_CSV_SAVETOFILE записывает в файл с указанным именем текст из памяти указанного объекта.

Вызов команды:

```
char *sFileName=... // Имя файла
rdsSetObjectStr (Csv, RDS_CSV_SAVETOFILE, 0, sFileName);
```

Параметры:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

sFileName

Указатель на строку с именем файла, в который нужно записать текст. Имя файла может содержать символические обозначения стандартных папок РДС (стр. 189). Если в имени файла нет пути, он будет создан в одной папке с загруженной схемой.

Примечания:

Эта команда записывает в файл sFileName весь текст, находящийся в памяти объекта Csv. В команде не предусмотрен возврат результата операции, поэтому для того, чтобы узнать, удалось ли записать текст в файл, следует выполнить команду RDS_CSV_FILEERROR (стр. 560).

См. также:

rdsCSVCreate (стр. 556), rdsSetObjectStr (стр. 407),
RDS_CSV_FILEERROR (стр. 560), RDS_CSV_TEXT (стр. 568).

A.5.32.18. Команда RDS_CSV_STRFROMFILE – считать строку из файла

Команда RDS_CSV_STRFROMFILE считывает из файла, открытого для построчного чтения командой RDS_CSV_OPENFILEREAD (стр. 564), очередную строку.

Вызов команды:

```
int iStrNum=... // Номер строки-получателя в объекте
BOOL bOk=rdsCommandObjectEx(Csv,RDS_CSV_STRFROMFILE,
                             iStrNum,NULL);
```

Параметры и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

iStrNum

Номер, под которым считанная из файла строка будет записана в объект (строки нумеруются начиная с нуля).

bOk

Результат чтения: TRUE – строка считана, FALSE – произошла ошибка.

Примечания:

Эта команда считывает из ранее открытого командой RDS_CSV_OPENFILEREAD файла очередную строку (то есть последовательность символов до кода перевода строки или, если его нет, до конца файла), разбирает эту строку на элементы и записывает ее в объект Csv под номером iStrNum. Если строка с этим номером уже есть в объекте, она заменяется на считанную. Если строки с этим номером нет, она добавляется. Чтобы узнать, удалось ли загрузить строку из файла, следует проверить результат возврата команды или выполнить команду RDS_CSV_FILEERROR (стр. 560).

Если на момент чтения строки общее число строк в объекте меньше iStrNum, текст в объекте автоматически дополняется пустыми строками. Например, если в объекте три строки (0, 1 и 2), и добавляется строка с номером 5, в объект дополнительно будут добавлены пустые строки с номерами 3 и 4.

Пример использования команды RDS_CSV_STRFROMFILE приведен в §2.16.2.

См. также:

rdsCSVCreate (стр. 556), rdsCommandObjectEx (стр. 400),
RDS_CSV_OPENFILEREAD (стр. 564), RDS_CSV_CLOSEFILE (стр. 559),
RDS_CSV_FILEERROR (стр. 560), RDS_CSV_STRTOFILE (стр. 567).

A.5.32.19. Команда RDS_CSV_STRTOFILE – записать строку в файл

Команда RDS_CSV_STRTOFILE записывает в файл, открытый для построчной записи командой RDS_CSV_OPENFILEWRITE (стр. 565), очередную строку.

Вызов команды:

```
int iStrNum=... // Номер строки в объекте
BOOL bOk=rdsCommandObjectEx(Csv,RDS_CSV_STRTOFILE,
                             iStrNum,NULL);
```

Параметры и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией rdsCSVCreate (стр. 556).

iStrNum

Номер строки в памяти объекта, которая будет записана в файл. Строки нумеруются начиная с нуля.

boK

Результат записи: TRUE – строка записана, FALSE – произошла ошибка (ошибка записи или в объекте нет строки iStrNum).

Примечания:

Эта команда записывает в файл, ранее открытый командой RDS_CSV_OPENFILEWRITE, строку с номером iStrNum из внутренней памяти объекта Csv. Чтобы узнать, удалось ли записать строку в файл, следует проверить результат возврата команды или выполнить команду RDS_CSV_FILEERROR (стр. 560).

Строки в файл пишутся последовательно, одна за другой. Последовательность строк в файле не обязательно будет совпадать с последовательностью строк в памяти объекта, она будет зависеть от параметров iStrNum, с которыми будут вызываться команды RDS_CSV_STRTOFILE. Например, для записи в файл строк в порядке, обратном их следованию в памяти объекта, можно использовать следующий цикл:

```
RDS_НОВЕЖТ Csv=rdsCSVCreate();  
    ... формирование текста в объекте ...  
// Число строк в объекте  
int LineCount=rdsGetObjectInt(Csv,RDS_CSV_LINECOUNT,0);  
// Открытие файла  
rdsSetObjectStr(Csv,RDS_CSV_OPENFILEWRITE,0,"file.csv");  
for(int i=LineCount-1;i>=0;i--)  
    rdsCommandObjectEx(Csv,RDS_CSV_STRTOFILE,i,NULL);  
// Закрытие файла  
rdsCommandObject(Csv,RDS_CSV_CLOSEFILE);  
// Удаление объекта  
rdsDeleteObject(Csv);
```

Можно также, например, все время работать с нулевой строкой объекта: устанавливать в ней элементы вызовом rdsCSVSetItem (стр. 558), а затем сбрасывать ее в файл.

См. также:

rdsCSVCreate (стр. 556), rdsCommandObjectEx (стр. 400),
RDS_CSV_LINECOUNT (стр. 563), RDS_CSV_OPENFILEWRITE (стр. 565),
RDS_CSV_CLOSEFILE (стр. 559), RDS_CSV_FILEERROR (стр. 560),
RDS_CSV_STRFROMFILE (стр. 567), rdsCSVSetItem (стр. 558).

A.5.32.20. Команда RDS_CSV_TEXT – весь текст объекта

Команда RDS_CSV_TEXT устанавливает или возвращает весь текст в формате CSV, содержащийся в объекте.

Вызов команды для установки:

```
char *sText=... // Текст в формате CSV  
rdsSetObjectStr(Csv,RDS_CSV_TEXT,0,sText);
```

Вызов команды для чтения:

```
char *sText=rdsGetObjectStr(Csv,RDS_CSV_TEXT,0);
```


Параметры и результат:

Csv

Идентификатор вспомогательного объекта для работы с CSV, ранее созданного функцией `rdsCSVCreate` (стр. 556).

sText

Указатель на строку, содержащую текст в формате CSV (последовательности текстовых строк с элементами, разделенными запятыми). При получении текста вызовом `rdsGetObjectStr` (стр. 405) этот текст формируется динамически, поэтому после использования он **обязательно** должен быть освобожден вызовом `rdsFree` (стр. 187). Если в объекте не содержится ни одного элемента, функция вернет NULL.

Примечания:

Эта команда позволяет работать с текстом в формате CSV как с единым целым, не разбивая его на строки, а каждую строку – на отдельные элементы.

Следует помнить, что, поскольку объект хранит текст уже разобранным, для того, чтобы вернуть весь текст, ему приходится заново собирать его из отдельных значений. По этой причине при выполнении команды `RDS_CSV_TEXT` объект формирует его в динамической памяти, и обязанность освобождения этой памяти вызовом `rdsFree` лежит на вызвавшей программе.

См. также:

`rdsCSVCreate` (стр. 556), `rdsSetObjectStr` (стр. 407),
`rdsGetObjectStr` (стр. 405), `rdsFree` (стр. 187), `RDS_CSV_LINE` (стр. 561).

A.5.33. Отладочные функции

Описываются функции, облегчающие вывод информации при отладке моделей блоков.

A.5.33.1. `rdsBlockMessageBox` – вывести окно сообщения с указанием имени блока

Функция `rdsBlockMessageBox` выводит стандартное окно сообщения Windows, добавляя к переданному в параметрах тексту сообщения полное имя указанного блока.

```
int RDSCALL rdsBlockMessageBox(  
    RDS_BHANDLE Block,    // Идентификатор блока  
    LPSTR Text,           // Текст сообщения  
    LPSTR Caption,        // Заголовок окна сообщения  
    int Flags              // флаги Windows API (MB_*)  
);
```

Тип указателя на эту функцию:

`RDS_IBhSSI`

Параметры:

Block

Идентификатор (`RDS_BHANDLE`, см. стр. 23) блока, полное имя которого нужно добавить к тексту сообщения. Может иметь значение NULL, в этом случае имя блока к сообщению добавлено не будет.

Text

Указатель на строку с текстом выводимого сообщения. Сообщение может состоять из нескольких строк, разделенных символом перевода строки “\n” (код 10).

Caption

Указатель на строку с заголовком окна выводимого сообщения.

Flags

Битовые флаги сообщения, определяющие набор кнопок в окне, иконку сообщения и т.п. Эти флаги совпадают с флагами стандартной функции Windows API MessageBox.

Возвращаемое значение:

Стандартная константа Windows API, указывающая на нажатую пользователем кнопку окна сообщения. Если сообщение выведено в режиме расчета, всегда возвращается константа IDCANCEL.

Примечания:

Для вывода сообщения функция rdsBlockMessageBox неявно вызывает сервисную функцию rdsMessageBox (стр. 201), добавляя первой строкой к тексту сообщения Text слово “Блок:”, за которым следует полное имя блока Block с указанием всей иерархии подсистем, в которой он находится. Если в параметре Block передан идентификатор корневой подсистемы, первой строкой выводится текст “Система”.

Эта функция, в основном, применяется при отладке моделей для вывода различных диагностических сообщений. Ее параметры и возвращаемое значение более подробно рассмотрены в описании более общей функции rdsMessageBox.

См. также:

rdsMessageBox (стр. 201).

A.5.33.2. rdsdebugBlockInfo – информация о блоке

Функция rdsdebugBlockInfo выводит в файл или сообщает пользователю тип, имя и координаты указанного в ее параметрах блока.

```
void RDSCALL rdsdebugBlockInfo(  
    RDS_BHANDLE Block,      // Идентификатор блока  
    LPSTR LogFile,          // Имя файла или NULL  
    LPSTR Caption           // Заголовок  
);
```

Тип указателя на эту функцию:

RDS_VBhSS

Параметры:

Block

Идентификатор блока, информация о котором запрашивается.

LogFile

Указатель на строку с именем текстового файла, в конец которого нужно добавить информацию о блоке (см. rdsdebugLogString, стр. 571), или NULL, если нужно вывести эту информацию в окне сообщения (см. rdsMessageBox, стр. 201). Имя файла может содержать символические обозначения стандартных папок РДС (стр. 189). Если в имени файла нет пути, он будет считаться находящимся в одной папке с загруженной схемой.

Caption

Указатель на дополнительный текст, который выводится в файл перед информацией о блоке (при LogFile!=NULL) или служит заголовком окна сообщения (при

LogFile==NULL). В этом параметре можно передать NULL, если дополнительный текст или заголовок сообщения не нужны.

Примечания:

Эта функция может быть полезна при отладке моделей блоков: если в схеме несколько блоков с одной и той же проблемной моделью, в ней можно вызывать rdsdebugBlockInfo при обнаружении ошибки или при наступлении какого-либо события – это даст понять, модель какого именно блока вызвана.

См. также:

rdsdebugLogString (стр. 571), rdsBlockMessageBox (стр. 569),
rdsMessageBox (стр. 201).

A.5.33.3. rdsdebugLogString – добавить строку в текстовый файл

Функция rdsdebugLogString добавляет произвольную строку в конец текстового файла с указанным именем.

```
BOOL RDSCALL rdsdebugLogString(  
    LPSTR LogFile,    // Имя файла  
    LPSTR String,     // Строка  
    BOOL ClearFile    // Очистить файл  
);
```

Тип указателя на эту функцию:

RDS_BSSB

Параметры:

LogFile

Указатель на строку с именем текстового файла, в конец которого нужно добавить строку. Имя файла может содержать символические обозначения стандартных папок РДС (стр. 189). Если в имени файла нет пути, он будет считаться находящимся в одной папке с загруженной схемой.

String

Указатель на строку, которую нужно добавить в файл (после нее автоматически будет записан возврат каретки (“\r”, код 13) и перевод строки (“\n”, код 10).

ClearFile

TRUE – очистить файл перед добавлением строки, FALSE – добавить строку в конец существующего файла.

Возвращаемое значение:

TRUE – строка записана, FALSE – ошибка.

Примечания:

Эта функция открывает указанный текстовый файл, записывает в него строку и снова закрывает его. Чаще всего она используется при отладке моделей блоков для ведения журнала выполняемых моделью действий. Поскольку сразу после записи строки файл снова закрывается, критические ошибки в модели, приводящие к аварийному завершению РДС, не приведут к потере журнала, и по последней записи в текстовом файле можно будет судить о том, после какой операции возникла ошибка.

Параметр ClearFile управляет очисткой файла: перед началом ведения журнала можно передать в нем TRUE, чтобы очистить прошлый журнал. Момент начала ведения журнала можно определять, например, по событию завершения загрузки схем

RDS_BFM_AFTERLOAD (стр. 50) или по значению какой-либо глобальной переменной в модели, которая будет служить флагом первого запуска.

См. также:

rdsdebugBlockInfo (стр. 570).

A.5.33.4. rdsSetDebugText – установить отладочный текст

Функция rdsSetDebugText устанавливает или очищает дополнительный текст, выводимый в стандартном сообщении РДС о неустранимой ошибке.

```
void RDSCALL rdsSetDebugText(  
    LPSTR String,    // Текст  
    BOOL Clear      // Установить (TRUE) или добавить (FALSE)  
);
```

Тип указателя на эту функцию:

RDS_VSB

Параметры:

String

Указатель на строку с текстом, или NULL, если нужно очистить текст.

Clear

TRUE – текстом String нужно заменить прежний отладочный текст, FALSE – текст String нужно добавить в конец уже имеющегося.

Примечания:

Эта функция обычно используется при отладке моделей блоков для конкретизации выводимого РДС сообщения о неустранимой ошибке. Перед подозрительной операцией можно установить поясняющий текст, а после нее – сбросить, например:

```
// Установка текста  
rdsSetDebugText("Вызов SomeFunction", TRUE);  
// Выполнение подозрительной операции  
SomeFunction();  
// Сброс текста  
rdsSetDebugText(NULL, TRUE);
```

Если в этом примере неустранимая ошибка возникнет в функции SomeFunction, в сообщении об этой ошибке будет присутствовать текст “Вызов SomeFunction”.

A.5.34. Функции поддержки автоматической компиляции моделей

Описываются функции, обеспечивающие взаимодействие РДС с модулями автоматической компиляции моделей блоков (см. главу 4).

A.5.34.1. rdscompAttachDifferentModel – замена имени подключаемой модели

Функция rdscompAttachDifferentModel позволяет изменить имя автокомпилируемой модели блока в момент ее подключения или в момент сохранения данных блока.

```
void RDSCALL rdscompAttachDifferentModel(  
    LPSTR NewModelName,    // Новое имя модели  
    LPSTR NewAltModelName // Новое альтернативное имя модели  
);
```

Тип указателя на эту функцию:

RDS_VSS

Параметры:

NewModelName

Указатель на строку с новым именем модели.

NewAltModelName

Указатель на строку с новым альтернативным именем модели (см. стр. 94).

Примечания:

Эта функция может вызываться только из функции модуля автокомпиляции (стр. 91) при реакции на события проверки возможности подключения модели к блоку RDS_COMPM_CANATTACHBLK (стр. 96) и сохранения блока RDS_COMPM_SAVEBLOCK (стр. 108), во всех остальных случаях ее вызов игнорируется. При проверке возможности подключения модели замена ее имени может потребоваться если, например, файл модели отсутствует, но модуль автокомпиляции в состоянии найти его без помощи пользователя. При сохранении данных блока замена имени модели может, например, использоваться для выбрасывания из этого имени пути к файлу, если модель хранится в файле и этот файл находится в одной папке с файлом схемы.

Замена имени модели рассматривается в описаниях событий RDS_COMPM_CANATTACHBLK и RDS_COMPM_SAVEBLOCK и в §4.1.

См. также:

RDS_COMPM_CANATTACHBLK (стр. 96), RDS_COMPM_SAVEBLOCK (стр. 108).

A.5.34.2. rdscompCompileModel – компилировать модель

Функция rdscompCompileModel компилирует модель с указанным идентификатором.

```
BOOL RDSCALL rdscompCompileModel(  
    RDS_MODELHANDLE Model,          // Модель  
    BOOL Rebuild                     // Принудительно компилировать  
);
```

Тип указателя на эту функцию:

RDS_BMhB

Параметры:

Model

Идентификатор автокомпилируемой модели (RDS_MODELHANDLE), которую нужно компилировать.

Rebuild

FALSE – не компилировать модель, если в этом нет необходимости. TRUE – компилировать модель в любом случае.

Возвращаемое значение:

TRUE – компиляция выполнена, FALSE – компиляция не проводилась.

Примечания:

Эта функция вызывает модуль автокомпиляции, обслуживающий модель с идентификатором Model, для компиляции этой модели. В параметре Rebuild передается TRUE, если модуль должен скомпилировать модель в любом случае, и FALSE, если

необходимость компиляции модели должна определяться самим модулем. Модуль автокомпиляции сначала вызывается для реакции на событие RDS_COMPM_PREPARE (стр. 107), а затем, если поле Valid структуры RDS_COMPMODELDATA (стр. 93) имеет значение FALSE, для реакции на событие RDS_COMPM_COMPILE (стр. 100). Действия по фактической компиляции модели выполняются модулем в реакции на RDS_COMPM_COMPILE.

См. также:

RDS_COMPM_PREPARE (стр. 107), RDS_COMPM_COMPILE (стр. 100).

A.5.34.3. rdscompGetBlockModelData – получить данные модели блока

Функция rdscompGetBlockModelData возвращает указатель на структуру данных автокомпилируемой модели RDS_COMPMODELDATA (стр. 93) указанного блока.

```
RDS_PCOMPMODELDATA RDSCALL rdscompGetBlockModelData(  
    RDS_BHANDLE Block      // Блок  
);
```

Тип указателя на эту функцию:

RDS_MdBh

Параметр:

Block

Идентификатор блока, для которого нужно найти структуру данных модели.

Возвращаемое значение:

Указатель на структуру данных автоматически компилируемой модели блока Block, или NULL, если у этого блока нет автокомпилируемой модели.

Примечания:

Структура данных модели создается при подключении этой модели к самому первому блоку и существует до тех пор, пока хотя бы один блок схемы связан с этой моделью. Возвращенный функцией указатель можно использовать на протяжении всего времени жизни модели.

См. также:

RDS_COMPMODELDATA (стр. 93), rdscompGetModelData (стр. 575),
rdscompGetModelDataByName (стр. 576).

A.5.34.4. rdscompGetModelBlock – обслуживаемый моделью блок по номеру

Функция rdscompGetModelBlock возвращает идентификатор одного из блоков, с которым связана указанная автокомпилируемая модель. Блок выбирается из списка блоков модели по указанному условному номеру.

```
RDS_BHANDLE RDSCALL rdscompGetModelBlock(  
    RDS_MODELHANDLE Model,      // Модель  
    int BlockNum,              // Номер блока  
    RDS_PBLOCKDESCRIPTION pDescr // Описание блока  
);
```

Тип указателя на эту функцию:

RDS_BhMhIBd

Параметры:

Model

Идентификатор автокомпилируемой модели (RDS_MODELHANDLE), для которой нужно найти обслуживаемый блок.

BlockNum

Номер блока во внутреннем списке блоков, обслуживаемых моделью Model. Блоки в списке нумеруются начиная с нуля.

pDescr

Указатель на структуру описания RDS_BLOCKDESCRIPTION (стр. 113), которую функция должна заполнить параметрами найденного блока. Если вызывающей программе не нужно описание блока, в этом параметре можно передать NULL.

Возвращаемое значение:

Идентификатор найденного блока или NULL, если модель в данный момент обслуживает меньше BlockNum+1 блоков (то есть номер BlockNum выходит за пределы размера списка блоков модели).

Примечания:

Эта функция обычно используется для перебора всех блоков, обслуживаемых конкретной моделью, и выполнения с ними каких-либо действий (например, присвоения им описанной в модели структуры статических переменных). Параметр BlockNum при этом изменяется от нуля до значения, на единицу меньшего поля NBlocks структуры данных модели RDS_COMPMODELDATA (стр. 93).

Следует помнить, что номер во внутреннем списке модели, который получит блок, определяется внутренней логикой РДС и не может быть определен заранее, при этом этот номер может, к тому же, изменяться со временем.

Пример использования функции rdscompGetModelBlock приведен в §4.4.

См. также:

RDS_COMPMODELDATA (стр. 93), RDS_BLOCKDESCRIPTION (стр. 113).

A.5.34.5. rdscompGetModelData – обслуживаемая модулем модель по номеру

Функция rdscompGetModelData возвращает указатель на структуру данных модели RDS_COMPMODELDATA (стр. 93), обслуживаемой указанным модулем автокомпиляции. Модель выбирается из списка модуля по указанному условному номеру.

```
RDS_PCOMPMODELDATA RDSCALL rdscompGetModelData (  
    RDS_COMPHANDLE Module,           // Модуль автокомпиляции  
    int ModelNum                     // Номер модели  
);
```

Тип указателя на эту функцию:

RDS_MdChI

Параметры:

Module

Идентификатор модуля автоматической компиляции (RDS_COMPHANDLE), для которого нужно найти обслуживаемую модель.

ModelNum

Номер модели во внутреннем списке моделей, обслуживаемых модулем Module. Модели в списке нумеруются начиная с нуля.

Возвращаемое значение:

Указатель на структуру данных найденной модели или NULL, если модуль в данный момент обслуживает меньше ModelNum+1 моделей (то есть номер ModelNum выходит за пределы размера списка моделей модуля).

Примечания:

Эта функция обычно используется для перебора всех моделей, обслуживаемых конкретным модулем автокомпиляции, и выполнения с ними каких-либо действий. Параметр ModelNum при этом изменяется от нуля до значения, на единицу меньшего поля NModels структуры данных модуля RDS_COMPMODULEDATA (стр. 92).

Структура данных модели создается при подключении этой модели к самому первому блоку и существует до тех пор, пока хотя бы один блок схемы связан с этой моделью. Возвращенный функцией указатель можно использовать на протяжении всего времени жизни модели.

Следует помнить, что номер во внутреннем списке модуля, который получит модель, определяется внутренней логикой РДС и не может быть известен заранее, при этом этот номер может, к тому же, изменяться со временем. Для получения указателя на структуру данных модели с указанным именем следует использовать функцию rdscompGetModelDataByName (стр. 576).

См. также:

RDS_COMPMODELDATA (стр. 93), RDS_COMPMODULEDATA (стр. 92),
rdscompGetModelDataByName (стр. 576).

A.5.34.6. rdscompGetModelDataByName – модель по имени

Функция rdscompGetModelDataByName возвращает указатель на структуру данных RDS_COMPMODELDATA (стр. 93) для модели с указанным именем, обслуживаемой указанным модулем автокомпиляции.

```
RDS_PCOMPMODELDATA RDSCALL rdscompGetModelDataByName (  
    RDS_COMPHANDLE Module,      // Модуль автокомпиляции  
    LPSTR ModelName             // Имя модели  
);
```

Тип указателя на эту функцию:

RDS_MdChS

Параметры:

Module

Идентификатор модуля автоматической компиляции (RDS_COMPHANDLE), для которого нужно найти обслуживаемую модель.

ModelName

Указатель на строку с именем модели, которую нужно найти.

Возвращаемое значение:

Указатель на структуру данных найденной модели или NULL, если модель с именем ModelName не обслуживается модулем Module.

Примечания:

Имена моделей сравниваются без учета регистра – для поиска модели с именем “Model” можно указывать строки “Model”, “model”, “MODEL” и т.п.

Структура данных модели создается при подключении этой модели к самому первому блоку и существует до тех пор, пока хотя бы один блок схемы связан с этой моделью. Возвращенный функцией указатель можно использовать на протяжении всего времени жизни модели.

См. также:

RDS_COMPMODELDATA (стр. 93).

A.5.34.7. rdscompOpenBlockModelEditor – вызвать редактор модели блока

Функция rdscompOpenBlockModelEditor передает модулю автокомпиляции команду вызвать редактор модели указанного блока.

```
int RDSCALL rdscompOpenBlockModelEditor(  
    RDS_BHANDLE Block    // Идентификатор блока  
);
```

Тип указателя на эту функцию:

RDS_IBh

Параметр:

Block

Идентификатор блока, редактор модели которого нужно вызвать.

Возвращаемое значение:

Одна из констант RDS_FRESULT_*, указывающая на результат выполнения функции:

RDS_FRESULT_OK	Редактор модели вызван.
RDS_FRESULT_ERROR	Ошибка (у блока нет автокомпилируемой модели).
RDS_FRESULT_DELAYED	Выполнение функции отложено.

Примечания:

Эта функция информирует РДС о необходимости вызвать редактор автокомпилируемой модели блока Block. РДС при этом вызовет модель автокомпиляции, обслуживающий модель этого блока, для реакции на событие RDS_COMPM_OPENEDITOR (стр. 106), все остальные действия должен выполнить сам модуль.

Если в момент вызова функции РДС находится в режиме расчета, функция вызвана из какой-либо реакции блока Block или модуля, обслуживающего именно эту модель, вызов редактора будет отложен до завершения очередного такта расчета или соответствующей реакции, при этом функция вернет константу RDS_FRESULT_DELAYED.

См. также:

RDS_COMPM_OPENEDITOR (стр. 106).

A.5.34.8. rdscompRenameModel – переименовать модель

Функция rdscompRenameModel изменяет имя указанной модели в указанном модуле автокомпиляции.

```
BOOL RDSCALL rdscompRenameModel(  
    RDS_COMPHANDLE Module,    // Модуль автокомпиляции
```

```

    LPSTR OldModelName,           // Старое имя
    LPSTR NewModelName,          // Новое имя
    BOOL AllowReplace             // Разрешить замену модели
);

```

Тип указателя на эту функцию:

RDS_BChSSB

Параметры:

Module

Идентификатор модуля автоматической компиляции (RDS_COMPHANDLE), который обслуживает переименовываемую модель.

OldModelName

Указатель на строку с текущим именем модели.

NewModelName

Указатель на строку с новым именем модели.

AllowReplace

TRUE – если модель с именем NewModelName уже есть в модуле Module, она будет отключена от всех блоков, и к ним будет подключена переименованная модель.
FALSE – если модель с именем NewModelName уже есть в модуле Module, переименование будет отменено и функция вернет FALSE.

Возвращаемое значение:

TRUE – модель переименована, FALSE – в модуле Module нет модели OldModelName или, при AllowReplace==FALSE, модель NewModelName уже используется.

Примечания:

Эта функция используется в тех случаях, когда необходимо изменить имя уже используемой модели (например, если в редакторе модели предусмотрена команда “Сохранить как...”). При ее вызове РДС последовательно производит следующие действия:

1. В модуле Module ищется модель с именем OldModelName. Если ее нет, функция возвращает FALSE.
2. В модуле Module ищется модель с именем NewModelName. Если она есть, и в параметре AllowReplace передано FALSE, функция возвращает FALSE.
3. Функция модуля Module вызывается для реакции на событие RDS_COMPM_CANRENAME (стр. 98). Если функция модуля запретит переименование, rdscompRenameModel вернет FALSE.
4. Из памяти РДС выгружается скомпилированная DLL модели OldModelName.
5. Если модель NewModelName существует, из памяти РДС выгружается ее скомпилированная DLL, модель отключается от всех блоков и ее данные в памяти уничтожаются. Блоки, к которым она была подключена, подключаются к модели OldModelName.
6. Имя модели OldModelName меняется на NewModelName, и модуль автокомпиляции Module вызывается для реакции на событие RDS_COMPM_MODELRENAMED (стр. 105).
7. Производится компиляция модели, если это необходимо, скомпилированная DLL модели загружается в память и начинает обслуживать блоки.

Кроме упомянутых событий RDS_COMPM_CANRENAME и RDS_COMPM_MODELRENAMED в процессе переименования модели могут возникнуть и другие: RDS_COMPM_DETACHBLOCK (стр. 101) и RDS_COMPM_ATTACHBLOCK (стр. 95) при передаче блоков от одной модели к

другой, RDS_COMPM_MODELCLEANUP (стр. 104) при уничтожении данных модели NewModelName и т.п.

Изменение альтернативного имени модели (см. стр. 94) функцией rdscompRenameModel не предусмотрено, для этого следует использовать функцию rdscompSetAltModelName (стр. 580).

См. также:

RDS_COMPM_CANRENMODEL (стр. 98), RDS_COMPM_MODELRENAMED (стр. 105), rdscompSetAltModelName (стр. 580).

A.5.34.9. rdscompReturnModelName – возврат имени модели из функции пользовательского интерфейса

Функция rdscompReturnModelName передает в РДС имя модели, указанное пользователем в функции модуля автокомпиляции, вызванной из окна настроек параметров блока.

```
void RDSCALL rdscompReturnModelName(  
    LPSTR NewModelName    // Имя модели  
);
```

Тип указателя на эту функцию:

RDS_VS

Параметр:

NewModelName

Указатель на строку с именем модели, которое нужно занести в соответствующее поле окна параметров блока.

Примечания:

Эта функция может вызываться только из функции модуля автокомпиляции (стр. 91) при реакции на событие RDS_COMPM_EXECFUNCTION (стр. 101). Это событие происходит в моменты различных действий пользователя на вкладке “Компиляция” окна параметров блока (при нажатии кнопок вкладки, после ввода нового текста в поле ввода имени модели и т.п.). Открытие диалога выбора файла модели (если модели хранятся в файлах), проверку допустимости введенного пользователем имени и другие действия при этом должен выполнять модуль автокомпиляции. Функция rdscompReturnModelName используется для возврата в РДС выбранного пользователем или сформированного модулем имени модели.

Пример использования этой функции рассматривается в §4.3.

См. также:

RDS_COMPM_EXECFUNCTION (стр. 101).

A.5.34.10. rdscompReturnModelNameLabel – заголовок поля ввода имени модели

Функция rdscompReturnModelNameLabel передает в РДС заголовок поля ввода имени автокомпилируемой модели, указываемый в окне параметров блока.

```
void RDSCALL rdscompReturnModelNameLabel(  
    LPSTR ModelNameLabel // Имя поля ввода  
);
```

Тип указателя на эту функцию:

RDS_VS

Параметр:

ModelNameLabel

Указатель на строку с заголовком поля ввода имени модели.

Примечания:

Эта функция может вызываться только из функции модуля автокомпиляции (стр. 91) при реакции на события реакции на действия пользователя RDS_COMPM_EXECFUNCTION (стр. 101) и запроса описания возможностей модуля RDS_COMPM_GETOPTIONS (стр. 103). В этих реакциях модуль сообщает РДС название, которое указывается на вкладке “Компиляция” окна параметров блока перед полем ввода имени модели. Например, если модели хранятся в файлах, модуль может установить в качестве заголовка поля строку “Файл модели:”.

Пример использования этой функции рассматривается в §4.3.

См. также:

RDS_COMPM_EXECFUNCTION (стр. 101), RDS_COMPM_GETOPTIONS (стр. 103).

A.5.34.11. rdscompSetAltModelName – установить альтернативное имя модели

Функция rdscompSetAltModelName присваивает указанной автокомпилируемой модели альтернативное имя (см. стр. 94).

```
void RDSCALL rdscompSetAltModelName (  
    RDS_MODELHANDLE Model,           // Модель  
    LPSTR NewAltModelName           // Альтернативное имя  
);
```

Тип указателя на эту функцию:

RDS_VMhS

Параметры:

Model

Идентификатор автокомпилируемой модели (RDS_MODELHANDLE), для которой устанавливается альтернативное имя.

NewAltModelName

Указатель на строку с новым альтернативным именем модели, или NULL, если альтернативное имя нужно сделать пустым.

См. также:

RDS_COMPMODELDATA (стр. 93).

A.5.34.12. rdscompSetBlockModel – подключить модель к блоку

Функция rdscompSetBlockModel подключает к указанному блоку автоматически компилируемую модель с указанными параметрами.

```
int RDSCALL rdscompSetBlockModel (  
    RDS_BHANDLE Block           // Идентификатор блока  
    LPSTR CompModuleName,       // DLL модуля автокомпиляции  
    LPSTR CompModuleFunc,       // Функция модуля автокомпиляции
```

```

        LPSTR ModelName,          // Имя модели
        LPSTR AltModelName       // Альтернативное имя модели
    );

```

Тип указателя на эту функцию:

RDS_IBhSSSS

Параметры:

Block

Идентификатор блока, к которому подключается модель.

CompModuleName

Указатель на строку с именем файла DLL, в которой находится модуль автокомпиляции, который будет обслуживать подключаемую модель. Имя файла может содержать символические обозначения стандартных папок РДС (стр. 189). Если в имени файла нет пути, он будет считаться находящимся в одной папке с загруженной схемой. Для отключения автокомпилируемой модели от блока в этом параметре можно передать NULL (остальные параметры при этом игнорируются).

CompModuleFunc

Указатель на строку с именем функции модуля автокомпиляции, экспортированной из файла CompModuleName. РДС будет использовать эту строку для поиска этой функции в DLL, передавая ее в параметре функции Windows API GetProcAddress.

ModelName

Указатель на строку с именем подключаемой модели.

AltModelName

Указатель на строку с альтернативным именем подключаемой модели, или NULL, если у модели нет альтернативного имени.

Возвращаемое значение:

Одна из констант RDS_FRESULT_*, указывающая на результат выполнения функции:

RDS_FRESULT_OK Модель подключена.

RDS_FRESULT_ERROR Ошибка (нет указанного файла DLL, не экспортирована указанная функция и т.п.).

RDS_FRESULT_DELAYED Выполнение функции отложено.

Примечания:

Эта функция информирует РДС о необходимости подключить к блоку Block новую автоматически компилируемую модель или отключить ее совсем (для этого в CompModuleName передается NULL).

Если в момент вызова функции РДС находится в режиме расчета, функция вызвана из какой-либо реакции блока Block или модуля, обслуживающего именно эту модель, подключение новой модели будет отложено до завершения очередного такта расчета или соответствующей реакции, при этом функция вернет константу RDS_FRESULT_DELAYED.

Наиболее часто встречающееся применение этой функции – добавление в редактор модели возможности выделения одного из блоков и подключения к нему альтернативной версии модели, изменения в которой не будут влиять на остальные блоки.

A.5.34.13. `rdscompSetModelFunction` – установить имена DLL и функции скомпилированной модели

Функция `rdscompSetModelFunction` запоминает во внутренних параметрах автокомпилируемой модели имя файла DLL и имя экспортированной из нее функции, которые получатся после компиляции этой модели.

```
void RDSCALL rdscompSetModelFunction(  
    RDS_MODELHANDLE Model,           // Модель  
    LPSTR LibraryFile,               // Имя DLL  
    LPSTR FunctionName               // Имя функции  
);
```

Тип указателя на эту функцию:

`RDS_VMhSS`

Параметры:

`Model`

Идентификатор автокомпилируемой модели (`RDS_MODELHANDLE`), для которой сообщаются имена скомпилированной DLL и экспортированной из нее функции.

`LibraryFile`

Указатель на строку с именем файла DLL, который будет создан в результате компиляции модели `Model`. Имя файла может содержать символические обозначения стандартных папок РДС (стр. 189). Если в имени файла нет пути, он будет считаться находящимся в одной папке с загруженной схемой.

`FunctionName`

Указатель на строку с именем функции блока, экспортированной из файла DLL `LibraryFile`. РДС будет использовать эту строку для поиска функции в DLL, передавая ее в параметре функции Windows API `GetProcAddress`.

Примечания:

Эта функция указывает РДС, как будет называться файл DLL, полученный в результате компиляции модели `Model`, и каким будет экспортированное из этой DLL имя функции блока. Без вызова `rdscompSetModelFunction` РДС не сможет подключить к блокам скомпилированную модель. Чаще всего она вызывается в реакции модуля автокомпиляции на событие `RDS_COMPM_PREPARE` (стр. 107).

Пример использования этой функции рассматривается в §4.4.

См. также:

`RDS_COMPM_PREPARE` (стр. 107), `RDS_COMPM_COMPILE` (стр. 100).

Приложение Б. Функции, константы и структуры библиотеки RdsCtrl.dll

Описываются функции, экспортированные из библиотеки RdsCtrl.dll, используемые в них структуры и константы. Применение этих функций для управления РДС из внешнего приложения описывается в главе 3.

Б.1. Структуры библиотеки RdsCtrl.dll

Описываются структуры, используемые в различных функциях библиотеки RdsCtrl.dll.

Б.1.1. RDSCTRL_BLOCKMSGDATA – сообщение от блока

В структуре RDSCTRL_BLOCKMSGDATA передаются параметры сообщения, посланного блоком схемы, загруженной в РДС, внешней управляющей программе (для отправки такого сообщения используется сервисная функция РДС rdsRemoteControllerCall, см. стр. 397).

```
typedef struct {  
    DWORD servSize;           // Размер этой структуры  
    LPSTR BlockFullName;      // Полное имя блока  
    int IntMsg;               // Переданное целое число  
    LPSTR StrMsg;             // Переданная строка  
} RDSCTRL_BLOCKMSGDATA;  
typedef RDSCTRL_BLOCKMSGDATA *RDSCTRL_PBLOCKMSGDATA;
```

Поля структуры:

servSize

Размер этой структуры в байтах. Перед передачей сообщения блоку управляющей программе в это поле автоматически заносится значение `sizeof(RDSCTRL_BLOCKMSGDATA)`. Управляющая программа может использовать это поле для проверки соответствия размера переданной структуры собственным ожиданиям.

BlockFullName

Указатель на строку с полным именем блока, отправившего сообщение. Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя `“:Sys1:Sys100:Block1”` говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

IntMsg

Целое число, переданное блоком.

StrMsg

Указатель на строку, переданную блоком. Если блок не передал в сообщении никакой строки, в этом поле записан указатель на пустую строку.

Примечания:

Указатель на эту структуру передается управляющей программе двумя способами:

- Если для реакции на сообщение от блока используется функция обратного вызова, зарегистрированная при помощи функции `rdscrtlRegisterEventStdCallback` (стр. 668), то указатель на RDSCTRL_BLOCKMSGDATA передается в третьем параметре этой функции обратного вызова.

- Если для реакции на сообщение от блока используется сообщение окну управляющей программы, зарегистрированному функцией `rdscrtlRegisterEventMessage` (стр. 667), то указатель на `RDSCTRL_BLOCKMSGDATA` передается в поле `Data` структуры `RDSCTRL_MSGEVENTDATA` (стр. 586), указатель на которую, в свою очередь, передается в параметре `LPARAM` оконного сообщения.

Структура `RDSCTRL_BLOCKMSGDATA` и все строки, на которые указывают ее поля, существуют только во время выполнения реакции управляющего приложения на событие, то есть до завершения функции обратного вызова или реакции на оконное сообщение. Если управляющему приложению нужно сохранить переданные параметры, их нужно скопировать во внутреннюю память приложения до завершения реакции.

Для реакции на сообщения от блоков можно также использовать специализированную функцию обратного вызова, регистрируемую при помощи функции `rdscrtlRegisterBlockMsgCallback` (стр. 665). В этом случае структура `RDSCTRL_BLOCKMSGDATA` не используется.

См. также:

`rdscrtlRegisterEventStdCallback` (стр. 668),
`rdscrtlRegisterEventMessage` (стр. 667),
`RDSCTRL_EVENT_BLOCKMSG` (стр. 593), `RDSCTRL_MSGEVENTDATA` (стр. 586),
`rdscrtlRegisterBlockMsgCallback` (стр. 665),
`rdsRemoteControllerCall` (стр. 397).

Б.1.2. `RDSCTRL_MENUITEM` – описание пункта меню РДС

В структуре `RDSCTRL_MENUITEM` передается описание дополнительного пункта контекстного или главного меню РДС, созданного каким-либо блоком схемы.

```
typedef struct {
    DWORD servSize;           // Размер этой структуры
    LPSTR Text;               // Текст пункта меню
    BOOL Enabled;             // Разрешен
    BOOL Visible;             // Видим
    LPSTR BlockFullName;     // Полное имя блока
    int MenuFunc, MenuData;   // Параметры пункта меню
    BOOL Checked;             // Помечен
    BOOL Divider;            // Разделитель
    BOOL HasKey;              // Есть "горячая клавиша"
    int Key;                  // Код клавиши
    DWORD KeyFlags;          // Флаги клавиши (RDS_M* и RDS_K*)
} RDSCTRL_MENUITEM;
typedef RDSCTRL_MENUITEM *RDSCTRL_PMENUITEM;
```

Поля структуры:

`servSize`

Размер этой структуры в байтах. В это поле автоматически заносится значение `sizeof(RDSCTRL_MENUITEM)`. Управляющая программа может использовать это поле для проверки соответствия размера переданной структуры собственным ожиданиям.

`Text`

Указатель на строку с текстом пункта меню.

Enabled

TRUE – пункт меню разрешен и может быть выбран пользователем. FALSE – пункт меню запрещен, пользователь его видит, но не может выбрать.

Visible

TRUE – пункт видим. FALSE – пункт меню скрыт от пользователя.

BlockFullName

Указатель на строку с полным именем блока, создавшего этот пункт меню. Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

MenuFunc, MenuData

Пара целых чисел, которые передаются модели блока в сообщении RDS_BFM_MENUFUNCTION (стр. 63) при выборе этого пункта меню.

Checked

Пункт меню помечен (TRUE) или нет (FALSE).

Divider

TRUE – это не настоящий пункт меню, а горизонтальный разделитель, пользователь не может его выбрать. FALSE – это обычный пункт меню.

HasKey

TRUE – у пункта меню есть “горячая клавиша”, при нажатии которой пункт автоматически выбирается (только для пунктов главного меню РДС). FALSE – пункт меню не связан с клавишами.

Key

Код “горячей клавиши” пункта меню – только при HasKey==TRUE.

KeyFlags

Битовые флаги, описывающие состояние специальных клавиш клавиатуры в момент нажатия “горячей клавиши” (см. стр. 61) – только при HasKey==TRUE.

Примечания:

Для того, чтобы получить описание пункта главного или контекстного меню РДС, созданного блоком, нужно сначала считать все пункты меню этого блока во внутреннюю память библиотеки RdsCtrl.dll вызовом `rdscrtlReadBlockMenuItems` (стр. 635), а затем получить указатель на структуру `RDSCTRL_MENUITEM`, находящуюся в этой внутренней памяти и описывающую пункт с заданным номером при помощи вызова `rdscrtlGetMenuItemData` (стр. 630). Данные пунктов меню будут находиться во внутренней памяти библиотеки до следующего вызова функции `rdscrtlReadBlockMenuItems`.

Пример использования структуры `RDSCTRL_MENUITEM` приведен в §3.6.5.

См. также:

`rdscrtlReadBlockMenuItems` (стр. 635),
`rdscrtlGetMenuItemData` (стр. 630), `RDS_BFM_MENUFUNCTION` (стр. 63).

Б.1.3. RDSCTRL_MSGEVENTDATA – сообщение от РДС

В структуре RDSCTRL_MSGEVENTDATA передаются параметры сообщения, посланного РДС внешней управляющей программе, если для реакции на него используется сообщение окну управляющей программы, зарегистрированному функцией `rdscrtlRegisterEventMessage` (стр. 667).

```
typedef struct {  
    DWORD servSize;      // Размер этой структуры  
    int Link;            // Идентификатор связи с РДС  
    int Event;           // Событие (RDSCTRLEVENT_*)  
    LPVOID Data;         // Дополнительные параметры (если есть)  
} RDSCTRL_MSGEVENTDATA;  
typedef RDSCTRL_MSGEVENTDATA *RDSCTRL_PMSGEVENTDATA;
```

Поля структуры:

`servSize`

Размер этой структуры в байтах. Перед передачей сообщения управляющей программе в это поле автоматически заносится значение `sizeof(RDSCTRL_MSGEVENTDATA)`. Управляющая программа может использовать это поле для проверки соответствия размера переданной структуры собственным ожиданиям.

`Link`

Идентификатор связи с РДС, через которую пришла информация о событии. Этот идентификатор возвращается при ее установке функцией `rdscrtlCreateLink` (стр. 600).

`Event`

Идентификатор произошедшего события – одна из констант `RDSCTRLEVENT_*`. События РДС, на которые может реагировать внешняя управляющая программа, описаны в Б.2 (стр. 592).

`Data`

Указатель на дополнительные параметры события, если они есть. Если у события есть дополнительные параметры, в этом поле передается указатель на структуру, в которой они содержатся (параметры событий указаны в их описаниях в Б.2). Для параметров разных событий используются разные структуры, поэтому поле `Data` имеет тип `void*`, то есть “универсальный указатель”, и перед использованием его нужно приводить к типу параметров конкретного события, в зависимости от значения поля `Event`.

Примечания:

Эта структура используется только в том случае, если для реакции на события РДС во внешней управляющей программе используются сообщения какому-либо окну этой программы. При этом, получив из РДС информацию о наступлении какого либо события, `RdsCtrl.dll` вызывает функцию Windows API `SendMessage`, в первых трех параметрах которой передаются дескриптор окна, идентификатор сообщения и дополнительное целое число (`WPARAM`), указанные при регистрации отклика на событие в функции `rdscrtlRegisterEventMessage`, а в четвертом (`LPARAM`) – указатель на заполненную структуру `RDSCTRL_BLOCKMSGDATA`.

Структура существует только во время выполнения реакции управляющего приложения на событие, то есть до завершения реакции на оконное сообщение.

См. также:

`rdscrtlRegisterEventMessage` (стр. 667), `rdscrtlCreateLink` (стр. 600).

Б.1.4. RDSCTRL_NEWFILEDATA – описание события смены загруженной схемы

В структуре `RDSCTRL_NEWFILEDATA` передаются параметры сообщения РДС о смене текущей рабочей схемы `RDSCTRLEVENT_NEWFILE` (стр. 597).

```
typedef struct {
    DWORD servSize;      // Размер этой структуры
    int Reason;          // Причина события (RDSCTRLEVENT_NEWFILE_*)
    LPSTR FileName;      // Имя файла
} RDSCTRL_NEWFILEDATA;
typedef RDSCTRL_NEWFILEDATA *RDSCTRL_PNEWFILEDATA;
```

Поля структуры:

`servSize`

Размер этой структуры в байтах. В это поле автоматически заносится значение `sizeof(RDSCTRL_NEWFILEDATA)`. Управляющая программа может использовать это поле для проверки соответствия размера переданной структуры собственным ожиданиям.

`Reason`

Одна из констант `RDSCTRLEVENT_NEWFILE_*`, указывающая на причину смены текущего файла схемы, с которым работает управляемая копия РДС:

<code>RDSCTRLEVENT_NEWFILE_NEW</code>	Создана новая пустая схема.
<code>RDSCTRLEVENT_NEWFILE_TEMPLATE</code>	Новая схема создана по файлу шаблона.
<code>RDSCTRLEVENT_NEWFILE_LOAD</code>	Загружен файл схемы.

`FileName`

Указатель на строку с именем загруженного файла схемы или файла шаблона, по которому создана новая схема. Если событие произошло из-за создания новой пустой схемы, в этом поле записан указатель на пустую строку.

Примечания:

Указатель на эту структуру передается управляющей программе двумя способами:

- Если для реакции используется функция обратного вызова, зарегистрированная при помощи функции `rdscrtlRegisterEventStdCallback` (стр. 668), то указатель на `RDSCTRL_NEWFILEDATA` передается в третьем параметре функции обратного вызова.
- Если для реакции используется сообщение окну управляющей программы, зарегистрированному функцией `rdscrtlRegisterEventMessage` (стр. 667), то указатель на `RDSCTRL_NEWFILEDATA` передается в поле `Data` структуры `RDSCTRL_MSGEVENTDATA` (стр. 586), указатель на которую, в свою очередь, передается в параметре `LPARAM` оконного сообщения.

Структура `RDSCTRL_NEWFILEDATA` и строка, на которую указывает ее поле `FileName`, существуют только во время выполнения реакции управляющего приложения на событие, то есть до завершения функции обратного вызова или реакции на оконное сообщение.

См. также:

`RDSCTRLEVENT_NEWFILE` (стр. 597),
`rdscrtlRegisterEventStdCallback` (стр. 668),

rdctrlRegisterEventMessage (стр. 667),
RDCTRL_MSGEVENTDATA (стр. 586).

Б.1.5. RDCTRL_PROGRESSDATA – описание события хода загрузки или сохранения схемы

В структуре RDCTRL_PROGRESSDATA передаются параметры сообщения РДС о ходе загрузки или сохранения схемы RDCTRLEVENT_PROGRESS (стр. 597). Это событие наступает при любой загрузке и записи схемы через интервалы времени, заданные функцией rdctrlSetProgressDelay (стр. 662).

```
typedef struct {  
    DWORD servSize;      // Размер этой структуры  
    DWORD Current;       // Уже выполнено  
    DWORD Maximum;      // Всего должно быть выполнено  
} RDCTRL_PROGRESSDATA;  
typedef RDCTRL_PROGRESSDATA *RDCTRL_PPROGRESSDATA;
```

Поля структуры:

servSize

Размер этой структуры в байтах. В это поле автоматически заносится значение sizeof(RDCTRL_PROGRESSDATA). Управляющая программа может использовать это поле для проверки соответствия размера переданной структуры собственным ожиданиям.

Current

Число уже выполненных на данный момент элементарных действий по загрузке или сохранению схемы.

Maximum

Общее число элементарных действий по загрузке или сохранению схемы, которые предстоит выполнить.

Примечания:

Событие RDCTRLEVENT_PROGRESS сообщает управляющей программе, что на данный момент выполнено Current операций из Maximum. Это могут быть уже обработанный и общий объемы файла, уже загруженное/сохраненное число объектов и общее число этих объектов и т.п. Обычно это событие используют для вывода какой-либо индикации пользователю в процессе длительной загрузки и сохранения.

Указатель на структуру RDCTRL_PROGRESSDATA передается управляющей программе двумя способами:

- Если для реакции используется функция обратного вызова, зарегистрированная при помощи функции rdctrlRegisterEventStdCallback (стр. 668), то указатель на RDCTRL_PROGRESSDATA передается в третьем параметре этой функции обратного вызова.
- Если для реакции используется сообщение окну управляющей программы, зарегистрированному функцией rdctrlRegisterEventMessage (стр. 667), то указатель на RDCTRL_PROGRESSDATA передается в поле Data структуры RDCTRL_MSGEVENTDATA (стр. 586), указатель на которую, в свою очередь, передается в параметре LPARAM оконного сообщения.

Структура RDCTRL_PROGRESSDATA существуют только во время выполнения реакции управляющего приложения на событие, то есть до завершения функции обратного вызова или реакции на оконное сообщение.

См. также:

RDSCTRLEVENT_PROGRESS (стр. 597),
rdscctrlRegisterEventStdCallback (стр. 668),
rdscctrlRegisterEventMessage (стр. 667),
RDSCCTRL_MSGEVENTDATA (стр. 586), rdscctrlSetProgressDelay (стр. 662).

Б.1.6. RDSCCTRL_SAVEFILEDATA – описание события сохранения схемы

В структуре RDSCCTRL_SAVEFILEDATA передаются параметры сообщения РДС о сохранении схемы RDSCTRLEVENT_SAVEFILE (стр. 598).

```
typedef struct {  
    DWORD servSize;      // Размер этой структуры  
    LPSTR FileName;      // Имя файла  
} RDSCCTRL_SAVEFILEDATA;  
typedef RDSCCTRL_SAVEFILEDATA *RDSCCTRL_PSAVEFILEDATA;
```

Поля структуры:

servSize

Размер этой структуры в байтах. В это поле автоматически заносится значение sizeof(RDSCCTRL_SAVEFILEDATA). Управляющая программа может использовать это поле для проверки соответствия размера переданной структуры собственным ожиданиям.

FileName

Указатель на строку с именем сохраненного файла схемы.

Примечания:

Указатель на эту структуру передается управляющей программе двумя способами:

- Если для реакции используется функция обратного вызова, зарегистрированная при помощи функции rdscctrlRegisterEventStdCallback (стр. 668), то указатель на RDSCCTRL_SAVEFILEDATA передается в третьем параметре этой функции обратного вызова.
- Если для реакции используется сообщение окну управляющей программы, зарегистрированному функцией rdscctrlRegisterEventMessage (стр. 667), то указатель на RDSCCTRL_SAVEFILEDATA передается в поле Data структуры RDSCCTRL_MSGEVENTDATA (стр. 586), указатель на которую, в свою очередь, передается в параметре LPARAM оконного сообщения.

Структура RDSCCTRL_SAVEFILEDATA и строка, на которую указывает ее поле FileName, существуют только во время выполнения реакции управляющего приложения на событие, то есть до завершения функции обратного вызова или реакции на оконное сообщение.

См. также:

RDSCTRLEVENT_SAVEFILE (стр. 598),
rdscctrlRegisterEventStdCallback (стр. 668),
rdscctrlRegisterEventMessage (стр. 667),
RDSCCTRL_MSGEVENTDATA (стр. 586).

Б.1.7. RDSCTRL_SETTINGS – общие параметры РДС

Структура RDSCTRL_SETTINGS используется для описания общих параметров РДС при их получении функцией `rdscrtlGetGeneralSettings` (стр. 612) и установке функцией `rdscrtlSetGeneralSettings` (стр. 614).

```
typedef struct {  
    DWORD servSize;           // Размер этой структуры  
    DWORD UIFlags;           // Настройки пользовательского  
                             // интерфейса (RDSCTRL_SET_*)  
    BOOL SystemAutoSave;     // Автосохранение при выходе  
} RDSCTRL_SETTINGS;  
typedef RDSCTRL_SETTINGS *RDSCTRL_PSETTINGS;
```

Поля структуры:

`servSize`

Размер этой структуры в байтах. Перед вызовом любой из функций, работающих со структурой, управляющая программа должна записать в это поле значение `sizeof(RDSCTRL_SETTINGS)`.

`UIFlags`

Набор битовых флагов, указывающих на разрешение или запрещение элементов пользовательского интерфейса РДС (флаги объединяются битовым ИЛИ):

<code>RDSCTRL_SET_ENABLECALC</code>	Разрешено включение режима моделирования, см. также <code>rdscrtlEnableCalcMode</code> (стр. 606).
<code>RDSCTRL_SET_ENABLEEDIT</code>	Разрешено включение режима редактирования, см. также <code>rdscrtlEnableEditMode</code> (стр. 606).
<code>RDSCTRL_SET_ENABLEMAINWIN</code>	Включена видимость главного окна РДС, см. также <code>rdscrtlShowMainWindow</code> (стр. 614).
<code>RDSCTRL_SET_ENABLEOPTIONS</code>	Разрешено изменение настроек РДС, см. также <code>rdscrtlEnableOptions</code> (стр. 607).
<code>RDSCTRL_SET_ENABLEPARAMED</code>	Разрешено изменение основных параметров блоков – управляющей функции, внешнего вида и т.п., см. также <code>rdscrtlEnablePropEdit</code> (стр. 608).
<code>RDSCTRL_SET_ENABLERUN</code>	Разрешено включение режима расчета, см. также <code>rdscrtlEnableRun</code> (стр. 609).
<code>RDSCTRL_SET_ENABLESYSWIN</code>	Разрешено открытие окон подсистем, см. также <code>rdscrtlEnableSubsystemWindows</code> (стр. 610).
<code>RDSCTRL_SET_ENABLEUI</code>	Разрешен пользовательский интерфейс РДС, см. также <code>rdscrtlEnableUI</code> (стр. 611).

Различные флаги управляют различными функциями интерфейса: при установленном флаге функция разрешена, при сброшенном – запрещена. Битовый флаг `RDSCTRL_SET_ENABLEUI` управляет всем интерфейсом в целом: если он сброшен, работа РДС будет невидима для пользователя.

`SystemAutoSave`

`TRUE` – при выходе из РДС или загрузке новой схемы автоматически сохранять текущую, если в ней были изменения. `FALSE` – обычное поведение: автоматическое сохранение не производится. Для управления автоматическим сохранением можно также использовать функцию `rdscrtlSetAutoSave` (стр. 612).

Примечания:

Разрешать и запрещать отдельные элементы пользовательского интерфейса можно специализированными сервисными функциями, описанными в Б.3.3.

См. также:

`rdscrtlGetGeneralSettings` (стр. 612),
`rdscrtlSetGeneralSettings` (стр. 614), `rdscrtlSetAutoSave` (стр. 612).

Б.1.8. RDSCTRL_ZOOMRECT – параметры прямоугольника масштабирования

Структура `RDSCTRL_ZOOMRECT` используется для описания параметров прямоугольной области, которая должна быть полностью видима через порт вывода, при помощи функции `rdscrtlSetViewportZoomRectEx` (стр. 678).

```
typedef struct {  
    DWORD servSize;      // Размер этой структуры  
    int Left,Top;        // Левый верхний угол прямоугольника  
    int Width,Height;    // Размеры прямоугольника  
    double MaxZoom;      // Максимально допустимый масштаб  
    DWORD Flags;         // Флаги (RDSCTRL_ZOOMRECTFLAGS_*)  
    // Возвращаемые параметры  
    double Zoom;         // Получившийся масштаб  
    int ScrollX,ScrollY; // Получившийся сдвиг  
} RDSCTRL_ZOOMRECT;  
typedef RDSCTRL_ZOOMRECT *RDSCTRL_PZOOMRECT;
```

Поля структуры:

`servSize`

Размер этой структуры в байтах. Перед вызовом любой из функций, работающих со структурой, управляющая программа должна записать в это поле значение `sizeof(RDSCTRL_ZOOMRECT)`.

`Left,Top`

Левый верхний угол (`Left` – горизонтальная координата, `Top` – вертикальная) прямоугольной области, которая должна быть полностью видима через порт вывода. Координаты задаются в текущем масштабе порта или в масштабе 100%, в зависимости от флага `RDSCTRL_ZOOMRECTFLAGS_100` в поле `Flags`.

`Width,Height`

Ширина (`Width`) и высота (`Height`) прямоугольной области, которая должна быть полностью видима через порт вывода. Размеры задаются в текущем масштабе порта или в масштабе 100%, в зависимости от флага `RDSCTRL_ZOOMRECTFLAGS_100` в поле `Flags`.

`MaxZoom`

Максимально допустимый масштаб (РДС не будет устанавливать масштаб, больший этого). Масштаб задается в долях единицы (масштабу 100% соответствует значение поля 1.0).

`Flags`

Битовые флаги, управляющие изменением масштаба. На данный момент библиотекой `RdsCtrl.dll` поддерживается единственный флаг:

`RDSCTRL_ZOOMRECTFLAGS_100` Координаты прямоугольника и его размер заданы не в текущем масштабе порта, а в масштабе 100%.

Zoom

Возвращаемое значение: установленный в результате вызова функции масштаб в долях единицы.

ScrollX, ScrollY

Возвращаемые значения: установленные в результате вызова функции координаты левого верхнего угла (ScrollX – горизонтальная координата, ScrollY – вертикальная) части рабочего поля, видимой через порт вывода. Координаты возвращаются в текущем масштабе порта.

Примечания:

Перед вызовом функции `rdscrtlSetViewportZoomRectEx` управляющая программа записывает в поле `servSize` размер этой структуры, в поля `Left`, `Top`, `Width` и `Height` – координаты прямоугольной области, на отображение которой нужно настроить порт вывода, в поле `MaxZoom` – максимально допустимый масштаб в долях единицы, а в поле `Flags` устанавливает или очищает флаг масштаба задания координат. После завершения функции из полей `Zoom`, `ScrollX` и `ScrollY` управляющая программа может считать масштаб и координаты видимой части рабочего поля, которые были установлены в порте вывода (следует иметь в виду, что выполнение функции `rdscrtlSetViewportZoomRectEx` может быть отложено из-за занятости порта вывода, тогда поля `Zoom`, `ScrollX` и `ScrollY` не получат правильных значений).

См. также:

`rdscrtlSetViewportZoomRectEx` (стр. 678),
`rdscrtlSetViewportZoomRect` (стр. 676).

Б.2. События, на которые может реагировать программа

Описываются все события РДС, на которые может реагировать внешняя управляющая программа.

Б.2.1. Способы реакции на события

Внешняя управляющая программа, работающая с библиотекой `RdsCtrl.dll`, может реагировать на события, возникающие в управляемой копии РДС, если реакция на такие события разрешена функцией `rdscrtlEnableEvents` (стр. 665). Сама реакция может осуществляться несколькими способами.

Самый простой способ реакции на событие – использование функции обратного вызова. При помощи функции `rdscrtlRegisterEventStdCallback` (стр. 668) управляющая программа может связать с каким-либо событием собственную функцию вида

```
void RDSCALL имя_функции(  
    int Link,           // Идентификатор связи с РДС  
    int Event,          // Идентификатор события (RDSCTRLEVENT_*)  
    LPVOID pData,      // Данные события  
    LPVOID pAux        // Дополнительные данные  
);
```

Получив от управляемой копии РДС информацию о наступлении указанного события, `RdsCtrl.dll` вызовет эту функцию, передав в параметре `Link` идентификатор связи с управляемой копией РДС, в которой произошло событие, в параметре `Event` – идентификатор наступившего события, в параметре `pData` – указатель на структуру, описывающую событие (эта структура у каждого события своя), и в параметре `pAux` – указатель на какие-либо дополнительные данные, который был задан при регистрации функции обратного вызова.

Для события RDSCTRLEVENT_BLOCKMSG (сообщение от блока, стр. 593) можно указать специализированную функцию обратного вызова, в которой параметры события будут передаваться не в структуре через указатель pData, а в самих параметрах функции. Такая функция регистрируется вызовом rdscrtlRegisterBlockMsgCallback (стр. 665) и имеет вид

```
void RDSCALL имя_функции(
    int Link,           // Идентификатор связи с РДС
    LPSTR BName,       // Имя блока
    int Imsg,          // Переданное блоком число
    LPSTR Smsg,        // Переданная блоком строка
    LPVOID pAux        // Дополнительные данные
);
```

В параметре Link, как и в стандартной функции обратного вызова, передается идентификатор связи с управляемой копией РДС, в параметре BName – строка с полным именем блока, передавшего сообщение, в Imsg – целое число переданное блоком, в Smsg – указатель на строку, переданную блоком, в pAux – указатель на какие-либо дополнительные данные, который был задан при регистрации функции. Поскольку получение сообщений от блоков требуется в управляющих программах достаточно часто, и при этом обычно нужно каким-либо образом разбирать принятые от блока данные, использование специализированной функции для реакции на это событие часто удобнее использования стандартной.

Если разработчику управляющей программы по каким-либо причинам неудобно использовать функции обратного вызова, можно настроить RdsCtrl.dll таким образом, чтобы при наступлении события указанному окну управляющей программы направлялось сообщение. Для этого нужно вызвать функцию rdscrtlRegisterEventMessage (стр. 667), указав в ее параметрах идентификатор события, реакция на которое настраивается, дескриптор окна (HWND), которому должно направляться сообщение, и целый идентификатор самого сообщения. После этого, получив от управляемой копии РДС информацию о наступлении события, RdsCtrl.dll вызовет функцию Windows API SendMessage, которая передаст указанному окну указанное сообщение, и будет ожидать завершения процедуры этого окна. Все сообщения Windows имеют два параметра: WPARAM и LPARAM; в параметре WPARAM сообщения о событии, направляемого окну, передается значение, указанное при регистрации реакции на сообщение функцией rdscrtlRegisterEventMessage, а в параметре LPARAM – указатель на структуру описания события RDSCTRL_MSGEVENTDATA (стр. 586). В этой структуре передается идентификатор связи с управляемой копией РДС, в которой произошло событие (поле Link), идентификатор самого события (поле Event) и указатель на структуру данных события (поле Data).

См. также:

```
rdscrtlEnableEvents (стр. 665),
rdscrtlRegisterEventStdCallback (стр. 668),
rdscrtlRegisterEventMessage (стр. 667),
RDSCTRL_MSGEVENTDATA (стр. 586),
rdscrtlRegisterBlockMsgCallback (стр. 665),
RDSCTRLEVENT_BLOCKMSG (стр. 593).
```

Б.2.2. RDSCTRLEVENT_BLOCKMSG – сообщение от блока

Событие RDSCTRLEVENT_BLOCKMSG наступает при вызове моделью одного из блоков схемы функции rdsRemoteControllerCall (стр. 397) – с ее помощью блок может по собственной инициативе передать внешней управляющей программе

произвольную строку и целое число. Реакция на это событие возможна следующими способами:

- с помощью стандартной функции обратного вызова, зарегистрированной вызовом `rdscrtlRegisterEventStdCallback` (стр. 668), при этом имя передавшего сообщение блока, переданное число и строка передаются в третьем параметре этой функции (см. Б.2.1) через структуру `RDSCTRL_BLOCKMSGDATA` (стр. 583);
- с помощью сообщения окну управляющей программы, зарегистрированному функцией `rdscrtlRegisterEventMessage` (стр. 667), при этом имя передавшего сообщение блока, переданное число и строка передаются в функцию через структуру `RDSCTRL_BLOCKMSGDATA`, указатель на которую находится в поле `Data` структуры `RDSCTRL_MSGEVENTDATA` (стр. 586), указатель на которую, в свою очередь, передается в параметре `LPARAM` оконного сообщения;
- с помощью специализированной функции обратного вызова, зарегистрированной вызовом `rdscrtlRegisterBlockMsgCallback` (стр. 665), при этом имя передавшего сообщение блока, переданное число и строка передаются в параметрах функции.

Пример реакции на событие `RDSCTRLEVENT_BLOCKMSG` приведен в §3.4.

См. также:

Способы реакции на события (стр. 592), `rdsRemoteControllerCall` (стр. 397), `rdscrtlRegisterEventStdCallback` (стр. 668), `RDSCTRL_BLOCKMSGDATA` (стр. 583), `rdscrtlRegisterEventMessage` (стр. 667), `RDSCTRL_MSGEVENTDATA` (стр. 586), `rdscrtlRegisterBlockMsgCallback` (стр. 665).

Б.2.3. `RDSCTRLEVENT_CALCMODE` – режим моделирования

Событие `RDSCTRLEVENT_CALCMODE` наступает при переходе РДС в режим моделирования. У этого события нет параметров и структуры описания. Реакция на него возможна следующими способами:

- с помощью стандартной функции обратного вызова, зарегистрированной вызовом `rdscrtlRegisterEventStdCallback` (стр. 668), при этом в третьем параметре этой функции (см. Б.2.1) вместо указателя на структуру описания передается `NULL`;
- с помощью сообщения окну управляющей программы, зарегистрированному функцией `rdscrtlRegisterEventMessage` (стр. 667), при этом в поле `Data` структуры `RDSCTRL_MSGEVENTDATA` (стр. 586), указатель на которую передается в параметре `LPARAM` оконного сообщения, вместо указателя на структуру описания события записано значение `NULL`.

См. также:

Способы реакции на события (стр. 592), `rdscrtlRegisterEventStdCallback` (стр. 668), `rdscrtlRegisterEventMessage` (стр. 667), `RDSCTRL_MSGEVENTDATA` (стр. 586), `RDSCTRLEVENT_EDITMODE` (стр. 596).

Б.2.4. `RDSCTRLEVENT_CALCSTART` – запуск расчета

Событие `RDSCTRLEVENT_CALCSTART` наступает при запуске расчета в управляемой копии РДС. У этого события нет параметров и структуры описания. Реакция на него возможна следующими способами:

- с помощью стандартной функции обратного вызова, зарегистрированной вызовом `rdscrtlRegisterEventStdCallback` (стр. 668), при этом в третьем параметре этой функции (см. Б.2.1) вместо указателя на структуру описания передается `NULL`;
- с помощью сообщения окну управляющей программы, зарегистрированному функцией `rdscrtlRegisterEventMessage` (стр. 667), при этом в поле `Data` структуры `RDSCTRL_MSGEVENTDATA` (стр. 586), указатель на которую передается в параметре `LPARAM` оконного сообщения, вместо указателя на структуру описания события записано значение `NULL`.

Пример реакции на событие `RDSCTRL_EVENT_CALCSTART` приведен в §3.4.

См. также:

Способы реакции на события (стр. 592),
`rdscrtlRegisterEventStdCallback` (стр. 668),
`rdscrtlRegisterEventMessage` (стр. 667),
`RDSCTRL_MSGEVENTDATA` (стр. 586), `RDSCTRL_EVENT_CALCSTOP` (стр. 595).

Б.2.5. `RDSCTRL_EVENT_CALCSTOP` – остановка расчета

Событие `RDSCTRL_EVENT_CALCSTOP` наступает при остановке расчета в управляемой копии РДС. У этого события нет параметров и структуры описания. Реакция на него возможна следующими способами:

- с помощью стандартной функции обратного вызова, зарегистрированной вызовом `rdscrtlRegisterEventStdCallback` (стр. 668), при этом в третьем параметре этой функции (см. Б.2.1) вместо указателя на структуру описания передается `NULL`;
- с помощью сообщения окну управляющей программы, зарегистрированному функцией `rdscrtlRegisterEventMessage` (стр. 667), при этом в поле `Data` структуры `RDSCTRL_MSGEVENTDATA` (стр. 586), указатель на которую передается в параметре `LPARAM` оконного сообщения, вместо указателя на структуру описания события записано значение `NULL`.

Пример реакции на событие `RDSCTRL_EVENT_CALCSTOP` приведен в §3.4.

См. также:

Способы реакции на события (стр. 592),
`rdscrtlRegisterEventStdCallback` (стр. 668),
`rdscrtlRegisterEventMessage` (стр. 667),
`RDSCTRL_MSGEVENTDATA` (стр. 586), `RDSCTRL_EVENT_CALCSTART` (стр. 594).

Б.2.6. `RDSCTRL_EVENT_CONNCLOSED` – завершение РДС

Событие `RDSCTRL_EVENT_CONNCLOSED` наступает при разрыве связи с управляемой копией РДС, то есть при завершении программы `rds.exe`. У этого события нет параметров и структуры описания. Реакция на него возможна следующими способами:

- с помощью стандартной функции обратного вызова, зарегистрированной вызовом `rdscrtlRegisterEventStdCallback` (стр. 668), при этом в третьем параметре этой функции (см. Б.2.1) вместо указателя на структуру описания передается `NULL`;
- с помощью сообщения окну управляющей программы, зарегистрированному функцией `rdscrtlRegisterEventMessage` (стр. 667), при этом в поле `Data` структуры `RDSCTRL_MSGEVENTDATA` (стр. 586), указатель на которую передается в параметре `LPARAM` оконного сообщения, вместо указателя на структуру описания события записано значение `NULL`.

Пример реакции на событие `RDSCTRL_EVENT_CONNCLOSED` приведен в §3.4.

См. также:

Способы реакции на события (стр. 592),
rdscrtlRegisterEventStdCallback (стр. 668),
rdscrtlRegisterEventMessage (стр. 667),
RDSCRTL_MSGEVENTDATA (стр. 586).

Б.2.7. RDSCTRLEVENT_EDITMODE – режим редактирования

Событие RDSCTRLEVENT_EDITMODE наступает при переходе РДС в режим редактирования. У этого события нет параметров и структуры описания. Реакция на него возможна следующими способами:

- с помощью стандартной функции обратного вызова, зарегистрированной вызовом rdscrtlRegisterEventStdCallback (стр. 668), при этом в третьем параметре этой функции (см. Б.2.1) вместо указателя на структуру описания передается NULL;
- с помощью сообщения окну управляющей программы, зарегистрированному функцией rdscrtlRegisterEventMessage (стр. 667), при этом в поле Data структуры RDSCRTL_MSGEVENTDATA (стр. 586), указатель на которую передается в параметре LPARAM оконного сообщения, вместо указателя на структуру описания события записано значение NULL.

См. также:

Способы реакции на события (стр. 592),
rdscrtlRegisterEventStdCallback (стр. 668),
rdscrtlRegisterEventMessage (стр. 667),
RDSCRTL_MSGEVENTDATA (стр. 586), RDSCTRLEVENT_CALCMODE (стр. 594).

Б.2.8. RDSCTRLEVENT_LOADREQ – запрос загрузки схемы

Событие RDSCTRLEVENT_LOADREQ наступает при выборе пользователем пункта меню РДС “Файл | Загрузить” или нажатии соответствующей этому пункту кнопки или сочетания клавиш, если непосредственная загрузка схемы запрещена вызовом функции rdscrtlNoDirectLoad (стр. 652). Обычно реакция на это событие используется для вмешательства в загрузку схемы: управляющее приложение в этом случае самостоятельно дает пользователю выбрать загружаемую схему и передает ее в РДС, минуя стандартные функции загрузки схемы из файла.

У события RDSCTRLEVENT_LOADREQ нет параметров и структуры описания, реакция на него возможна следующими способами:

- с помощью стандартной функции обратного вызова, зарегистрированной вызовом rdscrtlRegisterEventStdCallback (стр. 668), при этом в третьем параметре этой функции (см. Б.2.1) вместо указателя на структуру описания передается NULL;
- с помощью сообщения окну управляющей программы, зарегистрированному функцией rdscrtlRegisterEventMessage (стр. 667), при этом в поле Data структуры RDSCRTL_MSGEVENTDATA (стр. 586), указатель на которую передается в параметре LPARAM оконного сообщения, вместо указателя на структуру описания события записано значение NULL.

Если загрузка схем не была запрещена функцией rdscrtlNoDirectLoad, вместо события RDSCTRLEVENT_LOADREQ наступает событие RDSCTRLEVENT_NEWFILE (стр. 597).

Пример реакции на событие RDSCTRLEVENT_LOADREQ приведен в §3.5.

См. также:

Способы реакции на события (стр. 592), `rdctrlNoDirectLoad` (стр. 652),
`rdctrlRegisterEventStdCallback` (стр. 668),
`rdctrlRegisterEventMessage` (стр. 667),
`RDSCTRL_MSGEVENTDATA` (стр. 586), `RDSCTRLEVENT_SAVEFILE` (стр. 598),
`RDSCTRLEVENT_NEWFILE` (стр. 597).

Б.2.9. `RDSCTRLEVENT_NEWFILE` – загрузка или создание схемы

Событие `RDSCTRLEVENT_NEWFILE` наступает сразу после загрузки схемы в память РДС или создания новой схемы. Параметры события (загрузка это или создание, при загрузке – имя файла схемы) описываются структурой `RDSCTRL_NEWFILEDATA` (стр. 587). Реакция на это событие возможна следующими способами:

- с помощью стандартной функции обратного вызова, зарегистрированной вызовом `rdctrlRegisterEventStdCallback` (стр. 668), при этом в третьем параметре этой функции (см. Б.2.1) передается указатель на структуру описания события `RDSCTRL_NEWFILEDATA`;
- с помощью сообщения окну управляющей программы, зарегистрированному функцией `rdctrlRegisterEventMessage` (стр. 667), при этом в поле `Data` структуры `RDSCTRL_MSGEVENTDATA` (стр. 586), указатель на которую передается в параметре `LPARAM` оконного сообщения, записан указатель на структуру описания события `RDSCTRL_NEWFILEDATA`.

Если загрузка схем из файлов запрещена функцией `rdctrlNoDirectLoad` (стр. 652), событие `RDSCTRLEVENT_NEWFILE` не наступает.

См. также:

Способы реакции на события (стр. 592), `RDSCTRL_NEWFILEDATA` (стр. 587),
`rdctrlRegisterEventStdCallback` (стр. 668),
`rdctrlRegisterEventMessage` (стр. 667),
`RDSCTRL_MSGEVENTDATA` (стр. 586), `rdctrlNoDirectLoad` (стр. 652).

Б.2.10. `RDSCTRLEVENT_PROGRESS` – ход загрузки или сохранения схемы

Событие `RDSCTRLEVENT_PROGRESS` наступает через заданный функцией `rdctrlSetProgressDelay` (стр. 662) интервал времени при загрузке или сохранении схемы. Параметры события (общий и уже обработанный на данный момент объем данных) описываются структурой `RDSCTRL_PROGRESSDATA` (стр. 588). Обычно это событие используется в пользовательском интерфейсе управляющего приложения для отображения хода загрузки и сохранения схемы. Реакция на него возможна следующими способами:

- с помощью стандартной функции обратного вызова, зарегистрированной вызовом `rdctrlRegisterEventStdCallback` (стр. 668), при этом в третьем параметре этой функции (см. Б.2.1) передается указатель на структуру описания события `RDSCTRL_PROGRESSDATA`;
- с помощью сообщения окну управляющей программы, зарегистрированному функцией `rdctrlRegisterEventMessage` (стр. 667), при этом в поле `Data` структуры `RDSCTRL_MSGEVENTDATA` (стр. 586), указатель на которую передается в параметре `LPARAM` оконного сообщения, записан указатель на структуру описания события `RDSCTRL_PROGRESSDATA`.

См. также:

Способы реакции на события (стр. 592), RDSCtrl_PROGRESSDATA (стр. 588),
rdscCtrlRegisterEventStdCallback (стр. 668),
rdscCtrlRegisterEventMessage (стр. 667),
RDSCtrl_MSGEVENTDATA (стр. 586), rdscCtrlSetProgressDelay (стр. 662).

Б.2.11. RDSCtrlEVENT_SAVEFILE – сохранение схемы

Событие RDSCtrlEVENT_SAVEFILE может наступать в двух случаях:

- если сохранение схемы в файл не было запрещено вызовом rdscCtrlNoDirectSave (стр. 653), событие наступает сразу после записи схемы в файл;
- если сохранение схемы запрещено вызовом rdscCtrlNoDirectSave, событие наступает при выборе пользователем пункта меню “Файл | Сохранить” или соответствующей ему кнопки или комбинации клавиш.

Имя сохраненного файла передается в структуре RDSCtrl_SAVEFILEDATA (стр. 589).

Реакция на это событие возможна следующими способами:

- с помощью стандартной функции обратного вызова, зарегистрированной вызовом rdscCtrlRegisterEventStdCallback (стр. 668), при этом в третьем параметре этой функции (см. Б.2.1) передается указатель на структуру описания события RDSCtrl_SAVEFILEDATA;
- с помощью сообщения окну управляющей программы, зарегистрированному функцией rdscCtrlRegisterEventMessage (стр. 667), при этом в поле Data структуры RDSCtrl_MSGEVENTDATA (стр. 586), указатель на которую передается в параметре LPARAM оконного сообщения, записан указатель на структуру описания события RDSCtrl_SAVEFILEDATA.

Пример реакции на событие RDSCtrlEVENT_SAVEFILE приведен в §3.5.

См. также:

Способы реакции на события (стр. 592), RDSCtrl_SAVEFILEDATA (стр. 589),
rdscCtrlRegisterEventStdCallback (стр. 668),
rdscCtrlRegisterEventMessage (стр. 667),
RDSCtrl_MSGEVENTDATA (стр. 586), rdscCtrlNoDirectSave (стр. 653).

Б.3. Функции библиотеки RdsCtrl.dll

Описываются функции, экспортированные из библиотеки RdsCtrl.dll, с помощью которых внешняя управляющая программа организует взаимодействие с РДС.

Б.3.1. Доступ к функциям RdsCtrl.dll

Все экспортированные из библиотеки RdsCtrl.dll функции имеют тип RDSCALL (см. стр. 25): аргументы функции передаются в стеке справа налево, стек освобождается вызванной функцией. Для доступа к функции следует получить указатель на нее при помощи функции Windows API GetProcAddress, привести этот указатель к правильному типу, учитывая типы параметров и возвращаемого значения этой функции (тип указателя на каждую функцию приводится в описании этой функции в данном приложении), после чего можно будет вызывать эту функцию непосредственно по указателю на нее. Обычно указатели на все необходимые функции получают сразу после загрузки RdsCtrl.dll: до выгрузки библиотеки из памяти они не изменяются.

Чтобы не получать вручную все указатели на все функции библиотеки, перед включением в текст программы файла “RdsCtrl.h” можно вставить следующее описание:

```
#define RDSCTRL_SERV_FUNC_BODY имя_функции
#include <RdsCtrl.h>
```

В результате в это место текста будет вставлен полный набор глобальных переменных-указателей на все функции библиотеки, одноименных этим функциям, а также дополнительная функция с именем *имя_функции* для заполнения всех этих переменных. Останется только вызвать эту дополнительную функцию, после чего все сервисные функции можно будет вызывать непосредственно по их именам. Такой же механизм используется в моделях блоков для доступа к сервисным функциям РДС (см. стр. 141). Имена глобальных переменных совпадают с именами сервисных функций, поэтому запись вида

```
rdscrtlSetPath("c:\\rds\\rds.exe");
```

будет означать, что функция, указатель на которую находится в глобальной переменной `rdscrtlSetPath`, вызывается с параметром “c:\rds\rds.exe”. Поскольку переменная с именем `rdscrtlSetPath` имеет правильный тип указателя на функцию с нужным типом параметра, ее можно использовать вместо имени функции без всякого приведения типов.

Если перед включением файла “RdsCtrl.h” описать `define`-константу `RDSCTRL_SERV_FUNC_EXTERNAL` (константа может не иметь значения, важен сам факт описания), все описания глобальных переменных будут включены с ключевым словом `extern`, то есть они будут описаны как внешние, находящиеся в другом модуле. Таким образом, если проект DLL состоит из нескольких модулей, в каждом из них можно включить файл “RdsCtrl.h”, при этом в одном из модулей перед включением файла будет описана константа `RDSCTRL_SERV_FUNC_BODY` с именем функции в качестве значения, а во всех остальных – константа `RDSCTRL_SERV_FUNC_EXTERNAL`. В результате в одном модуле будут описаны глобальные переменные-указатели на функции и функция их заполнения, а во всех остальных эти же переменные будут описаны как внешние. Это даст возможность вызывать функции библиотеки по имени в любом из модулей проекта. В отличие от описаний в файле “RdsFunc.h”, если перед включением “RdsCtrl.h” не будет описана ни одна из этих констант, никакие описания, относящиеся к глобальным переменным-указателям на функции, включены в текст программы не будут.

Примеры доступа к функциям `RdsCtrl.dll` приведены в §3.2.

Б.3.2. Функции управления связью с РДС

Описываются функции, управляющие связью внешнего приложения с конкретной управляемой копией РДС. С их помощью можно создать такую связь, настроить ее параметры, запустить и завершить РДС, разорвать связь и т.п.

Б.3.2.1. `rdscrtlClose` – завершить РДС

Функция `rdscrtlClose` завершает копию РДС, управляемую через указанную связь. Управление возвращается вызвавшей программе только после завершения `rds.exe`.

```
void RDSCALL rdscrtlClose(
    int Link          // Идентификатор связи
);
```

Тип указателя на эту функцию:

```
RDSCTRL_VI
```

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

Эта функция передает управляемой через связь Link копии РДС команду завершения и возвращает управление вызвавшей программе только тогда, когда процесс rds.exe завершится. Связь Link при этом не уничтожается и установленные для нее параметры РДС (разрешенные и запрещенные элементы интерфейса, глобальные параметры и т.п.) не теряются. РДС можно снова запустить через эту же связь функцией rdscrtlConnect (стр. 600).

Если управляющей программе не нужно ждать завершения rds.exe, вместо rdscrtlClose следует использовать функцию rdscrtlDisconnect (стр. 602).

Пример использования функции rdscrtlClose приведен в §3.2.

См. также:

rdscrtlCreateLink (стр. 600), rdscrtlConnect (стр. 600),
rdscrtlDisconnect (стр. 602).

Б.3.2.2. rdscrtlConnect – запустить РДС

Функция rdscrtlConnect запускает главную программу РДС и настраивает ее для работы через указанную связь.

```
BOOL RDSCALL rdscrtlConnect(  
    int Link          // Идентификатор связи  
);
```

Тип указателя на эту функцию:

RDSCRTL_BI

Параметр:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

Возвращаемое значение:

TRUE – процесс rds.exe запущен успешно, FALSE – ошибка (не найден файл rds.exe, недостаточно ресурсов и т.п.).

Примечания:

Эта функция запускает главную программу РДС, после чего ей можно управлять через связь Link. Если файл rds.exe и библиотека RdsCtrl.dll находятся в разных папках, путь к rds.exe должен быть предварительно установлен функцией rdscrtlSetPath (стр. 604).

Пример использования функции rdscrtlConnect приведен в §3.2.

См. также:

rdscrtlCreateLink (стр. 600), rdscrtlSetPath (стр. 604).

Б.3.2.3. rdscrtlCreateLink – создать связь с РДС

Функция rdscrtlCreateLink создает связь, через которую можно управлять РДС: запускать rds.exe, загружать схему, переключать режимы и т.п.

```
int RDSCALL rdscrtlCreateLink(void);
```

Тип указателя на эту функцию:

RDSCRTL_IV

Возвращаемое значение:

Уникальный идентификатор созданной связи или `-1`, если связь создать не удалось (например, при нехватке ресурсов).

Примечания:

Прежде чем управлять РДС, необходимо при помощи `rdscrtlCreateLink` создать связь для этого управления. Одна связь управляет одной копией РДС, но управляющее приложение может создать произвольное количество таких связей и независимо управлять несколькими копиями РДС одновременно. Создание связи не приводит к немедленному запуску РДС, при этом только отводится память и подготавливаются все необходимые для работы РДС внутренние структуры. Для фактического запуска РДС через уже созданную связь следует использовать функцию `rdscrtlConnect` (стр. 600).

Во внутренней памяти `RdsCtrl.dll` для каждой связи хранятся свои установки внешнего вида, свои глобальные параметры т.д., при завершении РДС эти установки не теряются. Пока существует связь, можно произвольное число раз запускать через нее РДС с этими запомненными установками. Для окончательного удаления связи используется функция `rdscrtlDeleteLink` (стр. 601), после ее вызова все данные связи уничтожаются и ее идентификатор освобождается (в дальнейшем он может быть присвоен другой связи при очередном вызове `rdscrtlCreateLink`).

Пример использования функции `rdscrtlCreateLink` приведен в §3.2.

См. также:

`rdscrtlDeleteLink` (стр. 601), `rdscrtlConnect` (стр. 600).

Б.3.2.4. `rdscrtlDeleteLink` – уничтожить связь с РДС

Функция `rdscrtlDeleteLink` уничтожает связь, через которую происходит управление запущенной копией РДС.

```
void RDSCALL rdscrtlDeleteLink(  
    int Link           // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

Эта функция уничтожает связь с идентификатором `Link`, больше эту связь нельзя будет использовать для запуска РДС и обмена данными с блоками схемы. Если на момент уничтожения связи через нее управлялась копия РДС, эта копия автоматически завершается.

Идентификаторы связей могут использоваться повторно, поэтому при очередном вызове `rdscrtlCreateLink` библиотека `RdsCtrl.dll` может присвоить новой созданной связи идентификатор одной из уничтоженных ранее.

Для завершения РДС не обязательно уничтожать связь – вместо этого можно вызвать функции `rdscrtlDisconnect` (стр. 602) или `rdscrtlClose` (стр. 599) и использовать связь с идентификатором `Link` позднее для запуска новой копии РДС с теми же параметрами и настройками.

Пример использования функции `rdscrtlDeleteLink` приведен в §3.2.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlDisconnect` (стр. 602),
`rdscrtlClose` (стр. 599).

Б.3.2.5. `rdscrtlDisconnect` – завершить РДС

Функция `rdscrtlDisconnect` завершает копию РДС, управляемую через указанную связь. Управление возвращается вызвавшей программе немедленно, не дожидаясь фактического завершения РДС.

```
void RDSCALL rdscrtlDisconnect(  
    int Link          // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCTRL_VI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

Эта функция передает управляемой через связь `Link` копии РДС команду завершения и немедленно возвращает управление вызвавшей программе. Связь `Link` при этом не уничтожается и установленные для нее параметры РДС (разрешенные и запрещенные элементы интерфейса, глобальные параметры и т.п.) не теряются. РДС можно снова запустить через эту же связь функцией `rdscrtlConnect` (стр. 600).

Если управляющей программе нужно дождаться завершения `rds.exe`, вместо `rdscrtlDisconnect` следует использовать функцию `rdscrtlClose` (стр. 599).

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlConnect` (стр. 600),
`rdscrtlClose` (стр. 599).

Б.3.2.6. `rdscrtlIsConnected` – проверить связь с РДС

Функция `rdscrtlIsConnected` проверяет наличие связи управляющей программы с управляемой копией РДС.

```
BOOL RDSCALL rdscrtlIsConnected(  
    int Link          // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCTRL_BI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Возвращаемое значение:

TRUE – есть связь с процессом rds.exe, FALSE – связи нет (процесс завершился или еще не запускался).

Примечания:

Эта функция проверяет наличие запущенного через связь Link процесса rds.exe. При создании связи автоматического запуска rds.exe не происходит, для запуска РДС обычно используется функция `rdscrtlConnect` (стр. 600), а для завершения – `rdscrtlClose` (стр. 599) или `rdscrtlDisconnect` (стр. 602).

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlRestoreConnection` (стр. 603),
`rdscrtlConnect` (стр. 600), `rdscrtlClose` (стр. 599),
`rdscrtlDisconnect` (стр. 602).

Б.3.2.7. `rdscrtlLeave` – прекратить управление РДС

Функция `rdscrtlLeave` разрывает соединение между РДС и управляющей программой.

```
void RDSCALL rdscrtlLeave(  
    int Link           // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VI`

Параметр:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

Эта функция отсоединяет копию РДС, управляемую через связь Link, от управляющей программы. РДС продолжает работать под полным управлением пользователя. Связь Link не уничтожается, через нее можно запустить новую копию РДС при помощи функции `rdscrtlConnect` (стр. 600).

Снова восстановить внешнее управление процессом rds.exe, отсоединенным при помощи функции `rdscrtlLeave`, невозможно.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlConnect` (стр. 600).

Б.3.2.8. `rdscrtlRestoreConnection` – перезапустить РДС при необходимости

Функция `rdscrtlRestoreConnection` проверяет наличие запущенной копии РДС и перезапускает rds.exe, если такой копии нет.

```
BOOL RDSCALL rdscrtlRestoreConnection(  
    int Link           // Идентификатор связи  
);
```

Тип указателя на эту функцию:

RDSCtrl_BI

Параметр:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Возвращаемое значение:

TRUE – после вызова функции есть связь с процессом `rds.exe`, (процесс не завершился или успешно запущен заново), FALSE – нет связи с РДС.

Примечания:

Эта функция запускает РДС, если в данный момент для связи Link нет работающего процесса `rds.exe`. Если процесс есть, функция не выполняет никаких действий и немедленно возвращает TRUE.

Пример использования функции `rdscrtlRestoreConnection` приведен в §3.2.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlConnect` (стр. 600).

Б.3.2.9. `rdscrtlSetPath` – установить путь к РДС

Функция `rdscrtlSetPath` передает в библиотеку `RdsCtrl.dll` путь к исполняемому файлу РДС.

```
void RDSCALL rdscrtlSetPath(  
    LPSTR Path          // Путь к rds.exe  
);
```

Тип указателя на эту функцию:

RDSCtrl_VS

Параметр:

Path

Указатель на строку с полным путем к исполняемому файлу РДС (`rds.exe`).

Примечания:

Эта функция вызывается в тех случаях, когда `rds.exe` и библиотека `RdsCtrl.dll` находятся в разных папках. Функция не привязана к какой-либо конкретной связи с РДС, ее достаточно вызвать один раз сразу после загрузки библиотеки в память. Если `RdsCtrl.dll` и `rds.exe` находятся в одной папке, вызывать `rdscrtlSetPath` не обязательно.

Б.3.2.10. `rdscrtlSetStringCallback` – регистрация функции возврата строки

Функция `rdscrtlSetStringCallback` регистрирует в `RdsCtrl.dll` функцию возврата строк произвольной длины.

```
void RDSCALL rdscrtlSetStringCallback(  
    RDSCtrl_RETURNSTRING Func // Функция  
);
```

Тип указателя на эту функцию:

RDSCtrl_VRs

Параметр:

Func

Указатель на функцию, которая записывает строку символов в объект (экземпляр класса, массив и т.п.), используемый в управляющей программе для хранения строк произвольной длины.

Примечания:

При помощи этой функции в библиотеку RdsCtrl.dll передается указатель на функцию управляющего приложения вида

```
void RDSCALL имя_функции(LPVOID objptr,LPSTR str);
```

Задача этой функции – записать строку *str* в объект, на который указывает *objptr*, чем бы этот объект ни являлся. Использование функции обратного вызова для передачи строк в управляющую программу позволяет вынести отведение памяти под эти строки за пределы библиотеки RdsCtrl.dll, в результате чего управляющее приложение может хранить эти строки в памяти так, как это удобно его разработчику. Во всех случаях, когда библиотека должна вернуть управляющей программе строку произвольной длины, в одну из функций библиотеки передается указатель на некий объект, и этот указатель используется в качестве первого параметра функции обратного вызова.

Пример использования функции `rdscrtlSetStringCallback` приведен в §3.2, различные варианты функции обратного вызова описываются в §3.1.

Пример:

Допустим, управляющая программа использует для хранения строк произвольной длины класс `String`, для которого определена операция присваивания строки вида

```
String s="строка";
```

Тогда функцию обратного вызова можно написать следующим образом:

```
void RDSCALL MyReturnString(LPVOID ptr,LPSTR str)
{ // Приводим указатель ptr к типу "указатель на String"
  String *pS=(String*)ptr;
  // Если указатель не передан, не делаем ничего
  if(pS==NULL) return;
  // Присваиваем строку объекту
  *pS=str;
}
```

Теперь, зарегистрировав эту функцию в библиотеке вызовом

```
rdscrtlSetStringCallback(MyReturnString);
```

можно загрузить в память управляемой копии РДС какую-либо схему и вызвать один из ее блоков функцией `rdscrtlCallBlockFunctionEx` (стр. 621):

```
String str; // Переменная для возвращаемой строки
rdscrtlCallBlockFunctionEx(
  Link, // Идентификатор связи с РДС
  ":Sys1:Block1", // Полное имя вызываемого блока
  99, // Передаваемое блоку число
  "Строка 1", // Передаваемая блоку строка
  &str); // Указатель для возвращаемой строки
```

В результате, получив от блока с именем “:Sys1:Block1” какую либо строку в ответ, RdsCtrl.dll вызовет функцию `MyReturnString`, передав в ее первом параметре указатель на *str* (то есть последний параметр вызова `rdscrtlCallBlockFunctionEx`), а во втором –

указатель на эту, полученную от блока, строку. Таким образом, строка запишется в объект `str`, и управляющая программа сможет далее работать с ней без учета какой-либо специфики `RdsCtrl.dll`.

Б.3.3. Функции управления интерфейсом пользователя РДС

Описываются функции, разрешающие и запрещающие различные элементы интерфейса пользователя в управляемой копии РДС.

Б.3.3.1. `rdscrtlEnableCalcMode` – разрешение режима моделирования

Функция `rdscrtlEnableCalcMode` разрешает или запрещает пользователю переключать РДС в режим моделирования.

```
void RDSCALL rdscrtlEnableCalcMode(  
    int Link,           // Идентификатор связи  
    BOOL Enable        // Разрешение  
);
```

Тип указателя на эту функцию:

`RDSCTRL_VIB`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Enable`

`TRUE` – режим моделирования разрешен, `FALSE` – запрещен.

Примечания:

Эта функция разрешает (при `Enable==TRUE`) или запрещает (при `Enable==FALSE`) пользователю включать режим моделирования в копии РДС, управляемой через связь `Link`. По умолчанию режим моделирования разрешен, при его запрещении вместе с ним запрещается и режим расчета (нельзя разрешить режим расчета, запретив при этом режим моделирования). Если в момент запрещения режима моделирования РДС находится в нем или в режиме расчета, будет автоматически включен режим редактирования.

Разрешение режима моделирования запоминается в параметрах связи `Link`, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlEnableCalcMode(Link, FALSE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с запрещенным режимом моделирования.

См. также:

```
rdscrtlCreateLink (стр. 600), rdscrtlEnableEditMode (стр. 606),  
rdscrtlEnableRun (стр. 609), rdscrtlSetGeneralSettings (стр. 614),  
rdscrtlConnect (стр. 600).
```

Б.3.3.2. `rdscrtlEnableEditMode` – разрешение режима редактирования

Функция `rdscrtlEnableEditMode` разрешает или запрещает пользователю переключать РДС в режим редактирования.

```
void RDSCALL rdscrtlEnableEditMode(  
    int Link,           // Идентификатор связи
```

```

        BOOL Enable      // Разрешение
    );

```

Тип указателя на эту функцию:

```
RDCTRL_VIB
```

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Enable

TRUE – режим редактирования разрешен, FALSE – запрещен.

Примечания:

Эта функция разрешает (при `Enable==TRUE`) или запрещает (при `Enable==FALSE`) пользователю включать режим редактирования в копии РДС, управляемой через связь *Link* (по умолчанию режим редактирования разрешен). Если в момент запрещения режима редактирования РДС находится в нем, будет автоматически включен режим моделирования.

Разрешение режима редактирования запоминается в параметрах связи *Link*, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlEnableEditMode(Link, FALSE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с запрещенным режимом редактирования и автоматически переключится в режим моделирования после запуска.

См. также:

```

rdscrtlCreateLink (стр. 600), rdscrtlEnableCalcMode (стр. 606),
rdscrtlEnableRun (стр. 609), rdscrtlSetGeneralSettings (стр. 614),
rdscrtlConnect (стр. 600).

```

Б.3.3.3. `rdscrtlEnableOptions` – разрешение настройки РДС

Функция `rdscrtlEnableOptions` разрешает или запрещает пользователю изменять общие настройки РДС.

```

void RDSCALL rdscrtlEnableOptions(
    int Link,          // Идентификатор связи
    BOOL Enable       // Разрешение
);

```

Тип указателя на эту функцию:

```
RDCTRL_VIB
```

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Enable

TRUE – изменение настроек разрешено, FALSE – запрещено.

Примечания:

Эта функция разрешает (при `Enable==TRUE`) или запрещает (при `Enable==FALSE`) пользователю изменять общие настройки РДС при помощи пункта меню “Сервис | Настройки RDS”. По умолчанию изменение настроек разрешено.

Разрешение настройки запоминается в параметрах связи `Link`, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlEnableOptions(Link, FALSE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с заблокированными настройками.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetGeneralSettings` (стр. 614),
`rdscrtlConnect` (стр. 600).

Б.3.3.4. `rdscrtlEnablePropEdit` – разрешение изменения параметров блоков РДС

Функция `rdscrtlEnablePropEdit` разрешает или запрещает пользователю изменять общие параметры блоков схемы.

```
void RDCALL rdscrtlEnablePropEdit(  
    int Link,           // Идентификатор связи  
    BOOL Enable        // Разрешение  
);
```

Тип указателя на эту функцию:

`RDSCTRL_VIB`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Enable`

`TRUE` – изменение параметров блоков разрешено, `FALSE` – запрещено.

Примечания:

Эта функция разрешает (при `Enable==TRUE`) или запрещает (при `Enable==FALSE`) пользователю изменять общие параметры блоков схемы РДС: внешний вид, набор статических переменных, DLL модели и т.п. На разрешение вызова функции настройки блока, за которую отвечает его модель (см. `RDS_BFM_SETUP`, стр. 71) действие функции `rdscrtlEnablePropEdit` не распространяется, она всего лишь не дает пользователю открывать стандартное окно параметров блока. По умолчанию изменение параметров блоков разрешено.

Разрешение изменения параметров запоминается в параметрах связи `Link`, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlEnablePropEdit(Link, FALSE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с заблокированными окнами параметров блоков.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetGeneralSettings` (стр. 614),
`rdscrtlConnect` (стр. 600).

Б.3.3.5. `rdscrtlEnableRun` – разрешение запуска расчета

Функция `rdscrtlEnableRun` разрешает или запрещает пользователю переключать РДС в режим расчета.

```
void RDSCALL rdscrtlEnableRun(  
    int Link,           // Идентификатор связи  
    BOOL Enable        // Разрешение  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIB`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Enable`

`TRUE` – режим расчета разрешен, `FALSE` – запрещен.

Примечания:

Эта функция разрешает (при `Enable==TRUE`) или запрещает (при `Enable==FALSE`) пользователю включать режим расчета в копии РДС, управляемой через связь `Link` (по умолчанию режим расчета разрешен). Если функцией `rdscrtlEnableCalcMode` (стр. 606) был запрещен режим моделирования, режим расчета будет запрещен независимо от параметра функции `rdscrtlEnableRun`. Запрещение режима расчета при работающем расчете не приведет к остановке этого расчета, просто в следующий раз пользователь не сможет его запустить.

Разрешение режима расчета запоминается в параметрах связи `Link`, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlEnableRun(Link, FALSE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с запрещенным режимом расчета.

См. также:

```
rdscrtlCreateLink (стр. 600), rdscrtlEnableEditMode (стр. 606),  
rdscrtlEnableCalcMode (стр. 606), rdscrtlSetGeneralSettings (стр. 614),  
rdscrtlConnect (стр. 600).
```

Б.3.3.6. `rdscrtlEnableRunInterface` – разрешение переключения режимов

Функция `rdscrtlEnableRunInterface` разрешает или запрещает пользователю переключать режимы РДС.

```
void RDSCALL rdscrtlEnableRunInterface(  
    int Link,           // Идентификатор связи  
    BOOL Enable        // Разрешение  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIB`

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Enable

TRUE – переключение режимов разрешено, FALSE – запрещено.

Примечания:

Эта функция разрешает (при `Enable==TRUE`) или запрещает (при `Enable==FALSE`) пользователю переключать режимы в копии РДС, управляемой через связь Link (по умолчанию переключение режимов разрешено). При этом она скрывает от пользователя все кнопки и пункты меню, которые переключают режимы. На программное переключение режимов вызов `rdscrtlEnableRunInterface` не влияет.

Разрешение переключения режимов запоминается в параметрах связи Link, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlEnableRunInterface(Link, FALSE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с запрещенным переключением режимов.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlEnableEditMode` (стр. 606),
`rdscrtlEnableCalcMode` (стр. 606), `rdscrtlEnableRun` (стр. 609),
`rdscrtlSetGeneralSettings` (стр. 614), `rdscrtlConnect` (стр. 600).

Б.3.3.7. `rdscrtlEnableSubsystemWindows` – разрешение открытия окон подсистем

Функция `rdscrtlEnableSubsystemWindows` разрешает или запрещает открытие окон подсистем, включая окно корневой подсистемы.

```
void RDSCALL rdscrtlEnableSubsystemWindows(  
    int Link,           // Идентификатор связи  
    BOOL Enable        // Разрешение  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIB`

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Enable

TRUE – открытие окон подсистем разрешено, FALSE – запрещено.

Примечания:

Эта функция разрешает (при `Enable==TRUE`) или запрещает (при `Enable==FALSE`) как ручное, так и программное открытие окон любых подсистем в копии РДС, управляемой через связь Link (по умолчанию открытие окон разрешено). Чаще всего она используется при работе с портами вывода (см. §3.6), когда содержимое подсистем отображается не в отдельных окнах, принадлежащих РДС, а в собственном окне управляющего приложения.

Разрешение открытия окон запоминается в параметрах связи `Link`, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlEnableSubsystemWindows(Link, FALSE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с заблокированным открытием окон подсистем.

Пример использования функции `rdscrtlEnableSubsystemWindows` приведен в §3.6.1.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetGeneralSettings` (стр. 614),
`rdscrtlConnect` (стр. 600).

Б.3.3.8. `rdscrtlEnableUI` – разрешение интерфейса пользователя РДС

Функция `rdscrtlEnableUI` полностью разрешает или запрещает весь интерфейс пользователя РДС.

```
void RDSCALL rdscrtlEnableUI(  
    int Link,           // Идентификатор связи  
    BOOL Enable        // Разрешение  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIB`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Enable`

`TRUE` – интерфейс пользователя разрешен, `FALSE` – запрещен.

Примечания:

Эта функция разрешает (при `Enable==TRUE`) или запрещает (при `Enable==FALSE`) интерфейс пользователя в копии РДС, управляемой через связь `Link` (по умолчанию интерфейс разрешен). Чаще всего она используется при работе с портами вывода (см. §3.6), когда пользователь работает не непосредственно с РДС, а с окном управляющего приложения, которое передает в РДС его команды.

При запрещенном интерфейсе пользователь не увидит ни главного окна РДС, ни кнопки РДС на панели задач Windows. Разрешение интерфейса запоминается в параметрах связи `Link`, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlEnableUI(Link, FALSE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже без интерфейса пользователя.

Пример использования функции `rdscrtlEnableUI` приведен в §3.6.1.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetGeneralSettings` (стр. 614),
`rdscrtlConnect` (стр. 600).

Б.3.3.9. `rdscrtlGetGeneralSettings` – получить настройки интерфейса пользователя РДС

Функция `rdscrtlGetGeneralSettings` возвращает общие настройки интерфейса пользователя РДС.

```
void RDSCALL rdscrtlGetGeneralSettings(  
    int Link,                // Идентификатор связи  
    RDSCRTL_PSETTINGS pSettings // Настройки  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIOpt`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`pSettings`

Указатель на структуру описания общих параметров РДС `RDSCRTL_SETTINGS` (стр. 590), в которую функция запишет параметры, запомненные для связи `Link`.

Примечания:

Эта функция заполняет структуру, указатель на которую передан в параметре `pSettings`, запомненными для связи `Link` настройками интерфейса пользователя РДС. Управляемая через эту связь копия РДС не обязательно должна быть запущена, эти параметры хранятся во внутренней памяти библиотеки `RdsCtrl.dll` для каждой созданной связи.

См. также:

`rdscrtlCreateLink` (стр. 600), `RDSCRTL_SETTINGS` (стр. 590),
`rdscrtlSetGeneralSettings` (стр. 614).

Б.3.3.10. `rdscrtlSetAutoSave` – установить автосохранение при выходе

Функция `rdscrtlSetAutoSave` включает или выключает автоматическое сохранение измененной схемы при выходе из РДС или при загрузке новой схемы.

```
void RDSCALL rdscrtlSetAutoSave(  
    int Link,                // Идентификатор связи  
    BOOL Enable             // Разрешение  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIB`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Enable`

`TRUE` – автосохранение разрешено, `FALSE` – запрещено.

Примечания:

Эта функция разрешает (при `Enable==TRUE`) или запрещает (при `Enable==FALSE`) автоматическое сохранение схем, измененных в копии РДС, управляемой через связь `Link`, перед выгрузкой их из памяти. По умолчанию автосохранение выключено. При включенном автосохранении сохраняются только измененные схемы.

Разрешение автосохранения запоминается в параметрах связи `Link`, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlSetAutoSave(Link, TRUE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с включенным автосохранением.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetGeneralSettings` (стр. 614),
`rdscrtlConnect` (стр. 600).

Б.3.3.11. `rdscrtlSetExitMode` – установить предупреждение о выходе из РДС

Функция `rdscrtlSetExitMode` устанавливает одно из двух возможных предупреждений пользователю о завершении РДС, либо отключает такое предупреждение.

```
void RDSCALL rdscrtlSetExitMode(  
    int Link, // Идентификатор связи  
    int Mode  // Вид предупреждения  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VII`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Mode`

Целое число, указывающее вид предупреждения пользователю:

- 0 Выход из РДС без предупреждения (по умолчанию).
- 1 Пользователю выводится предупреждение о том, что РДС в данный момент управляется через внешнее приложение, и он должен подтвердить завершение РДС или отказаться от него.
- 2 Пользователю выводится сообщение о том, что РДС управляется через внешнее приложение, выход из РДС запрещен.

Примечания:

Эта функция управляет реакцией на попытку пользователя завершить РДС при внешнем управлении. По умолчанию РДС завершается без каких-либо сообщений пользователю. Чаще всего эту функцию используют для информирования пользователя о том, что завершение РДС в данный момент может повлиять на работу управляющей программы. Если пользовательский интерфейс РДС запрещен, использование этой функции не имеет смысла, поскольку пользователь не сможет завершить РДС по собственной инициативе.

Установленный тип сообщения пользователю запоминается в параметрах связи `Link`, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно,

например, вызвать `rdscrtlSetExitMode` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с запомненным типом предупреждения о выходе.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlConnect` (стр. 600).

Б.3.3.12. `rdscrtlSetGeneralSettings` – установить настройки интерфейса пользователя РДС

Функция `rdscrtlSetGeneralSettings` устанавливает общие настройки интерфейса пользователя РДС.

```
void RDSCALL rdscrtlSetGeneralSettings(  
    int Link,                // Идентификатор связи  
    RDSCRTL_PSETTINGS pSettings // Настройки  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIOpt`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`pSettings`

Указатель на структуру описания общих параметров РДС `RDSCRTL_SETTINGS` (стр. 590), из которой функция считывает параметры для связи `Link`.

Примечания:

Эта функция сразу устанавливает все параметры интерфейса пользователя копии РДС, управляемой через связь `Link`, согласно полям структуры `pSettings`.

Установленные параметры запоминается в связи `Link`, их не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlSetGeneralSettings` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с этими параметрами.

См. также:

`rdscrtlCreateLink` (стр. 600), `RDSCRTL_SETTINGS` (стр. 590),
`rdscrtlGetGeneralSettings` (стр. 612).

Б.3.3.13. `rdscrtlShowMainWindow` – видимость главного окна РДС

Функция `rdscrtlShowMainWindow` устанавливает видимость главного окна РДС.

```
void RDSCALL rdscrtlShowMainWindow(  
    int Link, // Идентификатор связи  
    BOOL Show // Разрешение показывать окно  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIB`

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Show

TRUE – показать главное окно, FALSE – скрыть.

Примечания:

Эта функция разрешает (при `Show==TRUE`) или запрещает (при `Show==FALSE`) показывать пользователю главное окно РДС (окно с главным меню, кнопками и панелью блоков). По умолчанию главное окно отображается. При скрытом главном окне закрытие окна корневой подсистемы приводит к завершению РДС.

Разрешение видимости главного окна запоминается в параметрах связи `Link`, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlShowMainWindow(Link, FALSE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится без главного окна.

Пример использования функции `rdscrtlShowMainWindow` приведен в §3.2.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlConnect` (стр. 600).

Б.3.4. Функции общего назначения

Описываются различные общие функции управления РДС, получения информации о загруженной схеме, переключения режимов и т.п.

Б.3.4.1. `rdscrtlBlockExtIdByName` – внешний идентификатор по имени блока

Функция `rdscrtlBlockExtIdByName` возвращает уникальный целый внешний идентификатор блока с указанным полным именем.

```
DWORD RDSCALL rdscrtlBlockExtIdByName(  
    int Link,           // Идентификатор связи  
    LPSTR BlockName // Полное имя блока  
);
```

Тип указателя на эту функцию:

`RDSCRTL_DwIS`

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

BlockName

Указатель на строку с полным именем блока. Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя `“:Sys1:Sys100:Block1”` говорит о том, что блок с именем `Block1` находится в подсистеме `Sys100`, которая, в свою очередь, находится в подсистеме `Sys1` корневой подсистемы.

Возвращаемое значение:

Уникальный внешний идентификатор блока с именем BlockName, или 0, если такого блока нет в загруженной в данный момент схеме.

Примечания:

Эта функция возвращает уникальный целый идентификатор блока с заданным именем в загруженной в данный момент схеме. Такие идентификаторы присваиваются блокам автоматически и не изменяются при загрузке и сохранении схемы (см. стр. 207).

См. также:

rdscrtlCreateLink (стр. 600), rdsBlockOrConnByExtId (стр. 206),
rdscrtlBlockNameByExtId (стр. 618).

Б.3.4.2. rdscrtlBlockMenuClick – имитация выбора пункта меню блока

Функция rdscrtlBlockMenuClick имитирует выбор пользователем одного из пунктов меню, добавленных в контекстное или главное меню РДС указанным блоком.

```
void RDSCALL rdscrtlBlockMenuClick(  
    int Link,                // Идентификатор связи  
    LPSTR BlockName,        // Полное имя блока  
    int MenuFunc,           // Номер функции пункта меню  
    int MenuData            // Данные пункта меню  
);
```

Тип указателя на эту функцию:

RDSCTRL_VISII

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

BlockName

Указатель на строку с полным именем блока (именем корневой подсистемы считается пустая строка). Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

MenuFunc

Номер функции пункта меню – при вызове модели блока РДС копирует это число в поле Function структуры RDS_MENUFUNCDATA (стр. 63) без какой-либо обработки.

MenuData

Данные пункта меню – при вызове модели блока РДС копирует это число в поле MenuData структуры RDS_MENUFUNCDATA без какой-либо обработки.

Примечания:

Эта функция вызывает в копии РДС, управляемой через связь Link, модель блока с полным именем BlockName для реакции на событие RDS_BFM_MENUFUNCTION (стр. 63), как будто пользователь выбрал пункт меню, созданный этим блоком. С пунктом меню, выбор которого имитируется, связана пара целых чисел MenuFunc / MenuData. Модель

вызванного блока не может отличить вызов из-за того, что пользователь действительно выбрал этот пункт меню, от вызова из-за данной функции. Чаще всего функция `rdscrtlBlockMenuClick` используется при работе РДС через порт вывода (см. §3.6) для предоставления пользователю возможности работы с контекстными меню блоков. Список пунктов меню блока и описания этих пунктов можно получить при помощи функций `rdscrtlGetMenuItemData` (стр. 630) и `rdscrtlReadBlockMenuItems` (стр. 635).

Для успешной работы функции `rdscrtlBlockMenuClick` не обязательно, чтобы пункт меню, с которым связана пара чисел `MenuFunc/MenuData`, существовал в действительности – модель блока будет вызвана с этими числами в любом случае. После вызова этой функции окно подсистемы или порт вывода, в которых отображается вызванный блок, автоматически обновляются.

Пример использования функции приведен в §3.6.5.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdExecMenuItem` (стр. 359),
`RDS_BFM_MENUFUNCTION` (стр. 63), `rdscrtlGetMenuItemData` (стр. 630),
`rdscrtlReadBlockMenuItems` (стр. 635),
`rdscrtlBlockMenuClickEx` (стр. 617).

Б.3.4.3. `rdscrtlBlockMenuClickEx` – имитация выбора пункта меню блока

Функция `rdscrtlBlockMenuClickEx` имитирует выбор пользователем одного из пунктов меню, добавленных в контекстное или главное меню РДС указанным блоком. Обновление окна подсистемы или порта вывода производится в зависимости от параметра функции.

```
void RDSCALL rdscrtlBlockMenuClickEx(  
    int Link,                // Идентификатор связи  
    LPSTR BlockName,        // Полное имя блока  
    int MenuFunc,           // Номер функции пункта меню  
    int MenuData,           // Данные пункта меню  
    BYTE Refresh           // Обновить окно  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VISIIBt`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`BlockName`

Указатель на строку с полным именем блока (именем корневой подсистемы считается пустая строка). Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя `“:Sys1:Sys100:Block1”` говорит о том, что блок с именем `Block1` находится в подсистеме `Sys100`, которая, в свою очередь, находится в подсистеме `Sys1` корневой подсистемы.

`MenuFunc`

Номер функции пункта меню – при вызове модели блока РДС копирует это число в поле `Function` структуры `RDS_MENUFUNCTIONDATA` (стр. 63) без какой-либо обработки.

MenuData

Данные пункта меню – при вызове модели блока РДС копирует это число в поле MenuData структуры RDS_MENUFUNCDATA без какой-либо обработки.

Refresh

1 – после вызова функции обновить окно подсистемы или порт вывода, в котором находится данный блок, 0 – обновлять не нужно.

Примечания:

Эта функция является полным аналогом функции rdscrtlBlockMenuClick (стр. 616), отличаясь от нее только тем, что обновление окна подсистемы или порта вывода после ее вызова можно запретить, передав в параметре Refresh нулевое значение.

См. также:

rdscrtlCreateLink (стр. 600), rdsExecMenuItem (стр. 359),
RDS_BFM_MENUFUNCTION (стр. 63), rdscrtlGetMenuItemData (стр. 630),
rdscrtlReadBlockMenuItems (стр. 635), rdscrtlBlockMenuClick (стр. 616).

Б.3.4.4. rdscrtlBlockNameByExtId – имя блока по внешнему идентификатору

Функция rdscrtlBlockNameByExtId возвращает полное имя блока с указанным целым идентификатором.

```
BOOL RDSCALL rdscrtlBlockNameByExtId(  
    int Link,                // Идентификатор связи  
    DWORD ExtId,            // Идентификатор блока  
    LPVOID ReturnName       // Возвращаемое имя блока  
);
```

Тип указателя на эту функцию:

RDSCRTL_BIDwpV

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

ExtId

Уникальный целый идентификатор блока в схеме (см. стр. 207).

ReturnName

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи rdscrtlSetStringCallback (стр. 604), запишет строку с полным именем блока.

Возвращаемое значение:

TRUE – блок с идентификатором ExtId есть в схеме, FALSE – блок отсутствует.

Примечания:

Эта функция записывает в объект, указатель на который передан через параметр ReturnName, полное имя блока с идентификатором ExtId из схемы, загруженной в данный момент в копию РДС, управляемую через связь Link. Для возврата строки используется функция обратного вызова, ранее зарегистрированная через rdscrtlSetStringCallback.

Возвращаемое полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

См. также:

```
rdscrtlCreateLink (стр. 600), rdsBlockOrConnByExtId (стр. 206),
rdscrtlBlockExtIdByName (стр. 615),
rdscrtlSetStringCallback (стр. 604).
```

Б.3.4.5. rdscrtlBringAppToFront – переместить РДС на передний план

Функция rdscrtlBringAppToFront перемещает на передний план копию РДС, управляемую через указанную связь.

```
void RDSCALL rdscrtlBringAppToFront(
    int Link          // Идентификатор связи
);
```

Тип указателя на эту функцию:

```
RDSCRTL_VI
```

Параметр:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

Примечания:

Эта функция располагает окна управляемой через связь Link копии РДС на переднем плане, поверх других приложений. Если по команде от внешнего приложения РДС будет открывать какое-либо окно с запросом пользователю, перед этим приложение РДС (rds.exe) необходимо переместить на передний план, чтобы пользователь увидел этот запрос.

См. также:

```
rdscrtlCreateLink (стр. 600), rdscrtlMinimizeApp (стр. 634),
rdscrtlRestoreApp (стр. 637), rdscrtlShowMainWindow (стр. 614).
```

Б.3.4.6. rdscrtlCallBlockFunction – передача блоку числа и строки (устаревшая)

Устаревшая функция rdscrtlCallBlockFunction передает блоку целое число и строку и получает от него число и строку в ответ. Сейчас вместо нее чаще используется rdscrtlCallBlockFunctionEx (стр. 621).

```
int RDSCALL rdscrtlCallBlockFunction(
    int Link,                // Идентификатор связи
    LPSTR BlockName,        // Полное имя блока
    int MessageVal,         // Число блоку
    LPSTR MessageStr,       // Строка блоку
    LPSTR ReturnStr,        // Буфер возвращаемой строки
    int ReturnMaxLen,       // Макс длина строки в буфере
    int *pRealReturnLen     // Длина полученной строки
);
```

Тип указателя на эту функцию:

RDSCTRL_IISISSIpI

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdsctrlCreateLink` (стр. 600).

BlockName

Указатель на строку с полным именем блока (именем корневой подсистемы считается пустая строка). Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

MessageVal

Целое число, передаваемое блоку. Оно будет записано в поле Value структуры `RDS_REMOTEMSGDATA` при вызове модели блока для реакции на событие `RDS_BFM_REMOTEMSG` (стр. 42).

MessageStr

Указатель на строку, передаваемую блоку, или NULL, если строку передавать не нужно. Указатель на эту строку будет записан в поле String структуры `RDS_REMOTEMSGDATA` при вызове модели блока для реакции на событие `RDS_BFM_REMOTEMSG`.

ReturnStr

Указатель на буфер (массив символов), в который функция должна записать строку, полученную от блока в ответ.

ReturnMaxLen

Максимальная длина строки, которая уместится в буфер, указатель на который передан в параметре ReturnStr. Поскольку строки в С завершаются нулевым байтом, это число должно быть на единицу меньше размера буфера.

pRealReturnLen

Указатель на целую переменную, в которую функция должна записать длину строки, полученную от блока. Если вызывающей программе не нужна длина этой строки, в этом параметре можно передать NULL.

Возвращаемое значение:

Целое число, возвращенное функцией модели блока при реакции на событие `RDS_BFM_REMOTEMSG`.

Примечания:

Вызов этой функции приводит к тому, что модель блока с полным именем BlockName в схеме, загруженной в копию РДС, управляемую через связь Link, вызовется для реакции на событие `RDS_BFM_REMOTEMSG`. При этом в структуре описания события `RDS_REMOTEMSGDATA` в поле Value будет находиться число из параметра MessageVal, а в поле String – строка из параметра MessageStr. Если в реакции на это событие функция модели блока вызовет `rdsRemoteReply` (стр. 398), переданная в параметре `rdsRemoteReply` строка будет записана в буфер ReturnStr, если ее длина будет не больше ReturnMaxLen. Если строка окажется длиннее, в ReturnStr будет записана

пустая строка, при этом фактическая длина принятой от блока строки будет возвращена через указатель pRealReturnLen. Результат возврата функции модели блока будет результатом функции rdscrtlCallBlockFunction.

В связи со слишком сложным механизмом возврата строки, полученной от блока, в настоящее время вместо функции rdscrtlCallBlockFunction используется rdscrtlCallBlockFunctionEx, которая возвращает строку произвольной длины через функцию обратного вызова. В остальном эти две функции работают одинаково.

См. также:

```
rdscrtlCreateLink (стр. 600),  
RDS_BFM_REMOTEMSG, RDS_REMOTEMSGDATA (стр. 42),  
rdsRemoteReply (стр. 398), rdscrtlCallBlockFunctionEx (стр. 621).
```

Б.3.4.7. rdscrtlCallBlockFunctionEx – передача блоку числа и строки

Функция rdscrtlCallBlockFunctionEx передает блоку целое число и строку и получает от него число и строку в ответ.

```
int RDSCALL rdscrtlCallBlockFunctionEx(  
    int Link,                // Идентификатор связи  
    LPSTR BlockName,         // Полное имя блока  
    int MessageVal,          // Число блоку  
    LPSTR MessageStr,        // Строка блоку  
    LPVOID ReturnStr         // Строка от блока  
);
```

Тип указателя на эту функцию:

```
RDSCRTL_IISISpV
```

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

BlockName

Указатель на строку с полным именем блока (именем корневой подсистемы считается пустая строка). Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

MessageVal

Целое число, передаваемое блоку. Оно будет записано в поле Value структуры RDS_REMOTEMSGDATA при вызове модели блока для реакции на событие RDS_BFM_REMOTEMSG (стр. 42).

MessageStr

Указатель на строку, передаваемую блоку, или NULL, если строку передавать не нужно. Указатель на эту строку будет записан в поле String структуры RDS_REMOTEMSGDATA при вызове модели блока для реакции на событие RDS_BFM_REMOTEMSG.

ReturnStr

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи `rdscrtlSetStringCallback` (стр. 604), запишет строку, полученную от блока в ответ.

Возвращаемое значение:

Целое число, возвращенное функцией модели блока при реакции на событие `RDS_BFM_REMOTEMSG`.

Примечания:

Вызов этой функции приводит к тому, что модель блока с полным именем `BlockName` в схеме, загруженной в копию РДС, управляемую через связь `Link`, вызовется для реакции на событие `RDS_BFM_REMOTEMSG`. При этом в структуре описания события `RDS_REMOTEMSGDATA` в поле `Value` будет находиться число из параметра `MessageVal`, а в поле `String` – строка из параметра `MessageStr`. Если в реакции на это событие функция модели блока вызовет `rdsRemoteReply` (стр. 398), переданная в параметре `rdsRemoteReply` строка будет записана в объект `ReturnStr` при помощи функции обратного вызова, зарегистрированной вызовом `rdscrtlSetStringCallback`. Результат возврата функции модели блока будет результатом функции `rdscrtlCallBlockFunctionEx`.

Пример использования функции `rdscrtlCallBlockFunctionEx` приведен в §3.3.

См. также:

`rdscrtlCreateLink` (стр. 600),
`RDS_BFM_REMOTEMSG`, `RDS_REMOTEMSGDATA` (стр. 42),
`rdsRemoteReply` (стр. 398), `rdscrtlSetStringCallback` (стр. 604).

Б.3.4.8. `rdscrtlCheckModalWindows` – проверить наличие модальных окон

Функция `rdscrtlCheckModalWindows` проверяет наличие открытых модальных окон в управляемой копии РДС.

```
BOOL RDSCALL rdscrtlCheckModalWindows(  
    int Link          // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCTRL_BI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Возвращаемое значение:

`TRUE` – есть открытые модальные окна, `FALSE` – таких окон нет.

Примечания:

Эта функция возвращает `TRUE`, если в копии РДС, управляемой через связь `Link`, в данный момент есть открытые модальные окна. Модальные окна могут помешать некоторым действиям (при открытом модальном окне нежелательно пытаться открыть немодальное – например, окно подсистемы), поэтому управляющей программе может понадобиться эта

информация. Для принудительного закрытия всех открытых в данный момент модальных окон можно использовать функцию `rdscrtlCloseModalWindows` (стр. 624).

РДС автоматически отслеживает собственные модальные окна и окна, открываемые с помощью сервисных функций (см. А.5.28). Если модель блока или модуль автоматической компиляции открывает модальные окна при помощи функций Windows API или сторонних библиотек, необходимо информировать об этом РДС, вызывая перед открытием такого модального окна функцию `rdsBlockModalWinOpen` (стр. 147), а после его закрытия – `rdsBlockModalWinClose` (стр. 146).

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlCloseModalWindows` (стр. 624),
`rdsBlockModalWinOpen` (стр. 147), `rdsBlockModalWinClose` (стр. 146).

Б.3.4.9. `rdscrtlClearSystem` – очистить схему

Функция `rdscrtlClearSystem` удаляет из памяти управляемой копии РДС загруженную схему.

```
void RDSCALL rdscrtlClearSystem(  
    int Link           // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

После выполнения этой функции РДС переходит в состояние, аналогичное состоянию непосредственно после запуска: в памяти нет загруженной схемы. В интерфейсе пользователя РДС нет аналога этой функции.

См. также:

`rdscrtlCreateLink` (стр. 600).

Б.3.4.10. `rdscrtlCloseAllSysExceptRoot` – закрыть окна всех подсистем, открыть окно корневой

Функция `rdscrtlCloseAllSysExceptRoot` оставляет открытым только окно корневой подсистемы.

```
void RDSCALL rdscrtlCloseAllSysExceptRoot(  
    int Link           // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

Эта функция закрывает окна всех подсистем кроме корневой. Если окно корневой подсистемы закрыто, оно открывается автоматически.

См. также:

`rdctrlCreateLink` (стр. 600), `rdctrlCloseAllWindows` (стр. 624),
`rdctrlCloseSysWindow` (стр. 625).

Б.3.4.11. `rdctrlCloseAllWindows` – закрыть все окна

Функция `rdctrlCloseAllWindows` закрывает все окна в управляемой копии РДС.

```
void RDSCALL rdctrlCloseAllWindows(  
    int Link          // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDCTRL_VI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdctrlCreateLink` (стр. 600).

Примечания:

Эта функция закрывает окна всех подсистем (включая корневую), окно редактора слоев и все открытые модальные окна.

См. также:

`rdctrlCreateLink` (стр. 600), `rdctrlCloseAllSysExceptRoot` (стр. 623),
`rdctrlCloseModalWindows` (стр. 624).

Б.3.4.12. `rdctrlCloseModalWindows` – закрыть все модальные окна

Функция `rdctrlCloseModalWindows` закрывает все открытые модальные окна в управляемой копии РДС.

```
void RDSCALL rdctrlCloseModalWindows(  
    int Link          // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDCTRL_VI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdctrlCreateLink` (стр. 600).

Примечания:

Эта функция закрывает все модальные окна, открытые в данный момент в копии РДС, управляемой через связь `Link`. Модальные окна могут помешать некоторым действиям (при открытом модальном окне нежелательно пытаться открыть немодальное – например, окно подсистемы).

РДС автоматически отслеживает собственные модальные окна и окна, открываемые с помощью сервисных функций (см. А.5.28), и может закрыть их при вызове этой функции. Если модель блока или модуль автоматической компиляции открывает модальные окна при помощи функций Windows API или сторонних библиотек, необходимо информировать об этом РДС, вызывая перед открытием такого модального окна функцию `rdsBlockModalWinOpen` (стр. 147), а после его закрытия – `rdsBlockModalWinClose` (стр. 146).

См. также:

`rdsctrlCreateLink` (стр. 600), `rdsctrlCheckModalWindows` (стр. 622),
`rdsBlockModalWinOpen` (стр. 147), `rdsBlockModalWinClose` (стр. 146).

Б.3.4.13. `rdsctrlCloseSysWindow` – закрыть окно подсистемы

Функция `rdsctrlCloseSysWindow` закрывает окно указанной подсистемы.

```
void RDSCALL rdsctrlCloseSysWindow(  
    int Link,           // Идентификатор связи  
    LPSTR SysName      // Полное имя подсистемы  
);
```

Тип указателя на эту функцию:

`RDSCTRL_VIS`

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdsctrlCreateLink` (стр. 600).

SysName

Указатель на строку с полным именем подсистемы (именем корневой подсистемы считается пустая строка). Полное имя подсистемы, как и любого блока, начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этой, которое завершается именем самой подсистемы. Например, полное имя “:Sys1:Sys100:MySys” говорит о том, что подсистема с именем `MySys` находится в подсистеме `Sys100`, которая, в свою очередь, находится в подсистеме `Sys1` корневой подсистемы.

Примечания:

Эта функция закрывает в копии РДС, управляемой через связь *Link*, окно подсистемы с полным именем *SysName*.

См. также:

`rdsctrlCreateLink` (стр. 600), `rdsctrlCloseAllSysExceptRoot` (стр. 623),
`rdsctrlCloseAllWindows` (стр. 624).

Б.3.4.14. `rdsctrlEnableWinRefresh` – разрешение/запрет обновления окон

Функция `rdsctrlEnableWinRefresh` разрешает или запрещает обновление окна указанной подсистемы или немодальных окон указанного блока в режимах моделирования и расчета.

```
void RDSCALL rdsctrlEnableWinRefresh(  
    int Link,           // Идентификатор связи  
    LPSTR BlockName,   // Полное имя блока
```

```

        BOOL Enable,      // Разрешить/запретить обновление
        BOOL Recurse    // Включать вложенные блоки
    );

```

Тип указателя на эту функцию:

```
RDSCtrl_VisBB
```

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

BlockName

Указатель на строку с полным именем блока (именем корневой подсистемы считается пустая строка). Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

Enable

TRUE, если обновление окон нужно разрешить, или FALSE, если его необходимо временно запретить.

Recurse

Если Block – подсистема, значение TRUE в этом параметре также разрешит или запретит (в зависимости от значения Enable) обновление немодальных окон всех блоков этой подсистемы на всех уровнях иерархии (в частности, всех окон вложенных подсистем).

Примечания:

Эта функция обычно используется для временного запрета обновления окон и портов вывода в управляемой через связь Link копии РДС, когда она находится в режиме расчета. Это бывает нужно в тех случаях, когда какие-либо вычисления занимают несколько тактов, и в середине этих вычислений, пока их результаты не готовы, отображать что-либо в окнах нежелательно. В режиме редактирования обновление окон всегда разрешено, и вызов этой функции игнорируется.

Чаще всего функция используется для запрещения обновления окон подсистем, поскольку блоки других типов редко открывают немодальные окна. Принципы управления обновлением окон рассматриваются в описании сервисной функции РДС `rdcEnableWindowRefresh` (стр. 255), которая работает точно так же, как и эта.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdcEnableWindowRefresh` (стр. 255).

Б.3.4.15. `rdscrtlFindBlock` – параметры блока по имени

Функция `rdscrtlFindBlock` возвращает родительскую подсистему, координаты и размеры блока с указанным полным именем.

```

BOOL RDSCALL rdscrtlFindBlock(
    int Link,          // Идентификатор связи
    LPSTR BlockName,  // Полное имя блока
    LPVOID ParentName, // Возвращаемое имя родителя

```

```

    int *pLeft, int *pTop, // Возвращаемые координаты
    int *pWidth, int *pHeight // Возвращаемый размер
);

```

Тип указателя на эту функцию:

RDCTRL_BISpVpIpIpIpI

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

BlockName

Указатель на строку с полным именем блока (именем корневой подсистемы считается пустая строка). Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

ParentName

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи `rdscrtlSetStringCallback` (стр. 604), запишет строку с полным именем родительской подсистемы найденного блока. Именем корневой подсистемы считается пустая строка. Если имя родительской подсистемы не нужно управляющей программе, в этом параметре можно передать NULL.

pLeft, pTop

Указатели на целые переменные, в которые функция запишет горизонтальную (в `pLeft`) и вертикальную (в `pTop`) координаты левого верхнего угла изображения блока на рабочем поле в масштабе 100% (горизонтальная ось координат направлена вправо, вертикальная – вниз, начало координат – левый верхний угол рабочего поля). Если какая-либо координата блока не нужна управляющей программе, в соответствующем ей параметре можно передать NULL.

pWidth, pHeight

Указатели на целые переменные, в которые функция запишет ширину (в `pWidth`) и высоту (в `pHeight`) изображения блока в точках экрана в масштабе 100%. Если какой-либо из размеров блока не нужен управляющей программе, в соответствующем ему параметре можно передать NULL.

Возвращаемое значение:

TRUE – блок с полным именем BlockName есть в схеме, FALSE – блок отсутствует.

Примечания:

Эта функция ищет в схеме, загруженной в копию РДС, управляемую через связь Link, блок с именем BlockName, и возвращает некоторые его параметры.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetStringCallback` (стр. 604).

Б.3.4.16. `rdscrtlFindOpSetProviders` – поиск блоков, выполняющих операцию

Функция `rdscrtlFindOpSetProviders` возвращает список блоков, заявивших о поддержке операции внешнего управления с указанным именем.

```
int RDSCALL rdscrtlFindOpSetProviders(  
    int Link,                // Идентификатор связи  
    LPSTR BlockName,        // Полное имя блока (подсистемы)  
    LPSTR OpSetName,        // Имя операции  
    DWORD Flags,            // Флаги (RDSCRTL_FOSP_*)  
    LPVOID ReturnStr       // Возвращаемый список блоков  
);
```

Тип указателя на эту функцию:

`RDSCRTL_IISDwpV`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`BlockName`

Указатель на строку с полным именем блока (именем корневой подсистемы считается пустая строка), в котором будет проводиться поиск поддерживающих операцию блоков. Практически всегда в этом параметре передается имя какой-либо подсистемы, а чаще всего – пустая строка, то есть имя корневой подсистемы. Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем `Block1` находится в подсистеме `Sys100`, которая, в свою очередь, находится в подсистеме `Sys1` корневой подсистемы.

`OpSetName`

Указатель на строку с именем операции, для выполнения которой ищутся блоки.

`Flags`

Битовые флаги, влияющие на поиск:

`RDSCRTL_FOSP_RECURSIVE` Искать блоки, выполняющие операцию `OpSetName`, не только в подсистеме `BlockName`, но и во всех ее внутренних подсистемах на всех уровнях иерархии.

`RDSCRTL_FOSP_SELF` Проверять поддержку операции `OpSetName` не только во внутренних блоках подсистемы `BlockName`, но и в ней самой.

`ReturnStr`

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи `rdscrtlSetStringCallback` (стр. 604), запишет строку-список полных имен блоков, поддерживающих операцию `OpSetName`. Имена блоков в списке разделены кодом перевода строки “\n” (10).

Возвращаемое значение:

Число найденных блоков (то есть число имен в списке `ReturnStr`).

Примечания:

Эта функция ищет в подсистеме BlockName схемы, загруженной в копию РДС, управляемую через связь Link, все блоки, модели которых заявили о поддержке операции с именем OpSetName. Модели заявляют о поддержке операции вызовом сервисной функции rdsExecutesRemoteOpsSet (стр. 395). Список найденных блоков записывается в объект, указатель на который передан в ReturnStr, при помощи стандартной функции возврата строк, зарегистрированной вызовом rdscrtlSetStringCallback.

Технически функцию rdscrtlFindOpSetProviders можно вызвать не только для подсистемы, но и для блока другого типа, передав его имя в параметре BlockName. Однако, чаще всего управляющему приложению неизвестно, какой именно блок схемы выполняет нужную операцию, поэтому функция обычно вызывается для поиска блоков в какой-либо подсистеме или во всей схеме (для этого в BlockName передается пустая строка).

Пример использования функции rdscrtlFindOpSetProviders приведен в §3.3.

См. также:

rdscrtlCreateLink (стр. 600), rdscrtlSetStringCallback (стр. 604),
rdsExecutesRemoteOpsSet (стр. 395).

Б.3.4.17. rdscrtlGetBlockVarValue – получить значение переменной блока

Функция rdscrtlGetBlockVarValue возвращает строку с текущим значением переменной с указанным именем в указанном блоке.

```
void RDSCALL rdscrtlGetBlockVarValue(  
    int Link,                // Идентификатор связи  
    LPSTR BlockName,         // Полное имя блока  
    LPSTR VarName,           // Имя переменной  
    LPVOID ReturnStr         // Возвращаемое значение  
);
```

Тип указателя на эту функцию:

RDCTRL_VISSpV

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

BlockName

Указатель на строку с полным именем блока (именем корневой подсистемы считается пустая строка). Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

VarName

Указатель на строку с именем переменной, значение которой нужно получить. Для сложных переменных имя может содержать имена полей структур и индексы массивов в синтаксисе, принятом в РДС.

ReturnStr

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи `rdscrtlSetStringCallback` (стр. 604), запишет строку с текущим значением запрошенной переменной. Если указанный блок или указанная переменная отсутствуют, будет возвращена пустая строка.

Примечания:

Эта функция ищет в схеме, загруженной в копию РДС, управляемую через связь `Link`, блок с именем `BlockName`, в нем – переменную с именем `VarName`, и записывает в объект, указатель на который передан в `ReturnStr`, строку с текущим значением этой переменной при помощи функции обратного вызова, зарегистрированной `rdscrtlSetStringCallback`. Независимо от типа переменной, ее значение преобразуется в строку. Имя переменной `VarName` может содержать поля структур и индексы массивов и матриц в синтаксисе РДС, например “Y.Re”, “A[1]”, “M[3,8]” или “Out.Array[3]”.

При получении значения структуры ее поля перечисляются через запятую внутри фигурных скобок: например, структура с двумя вещественными полями может вернуть строку значения “{1.0,2.0}”. При получении значений матриц и массивов все значения элементов перечисляются через запятую внутри фигурных скобок: для массива – последовательно (например, “{1,2,3,4}”), для матрицы – по строкам, причем каждая строка тоже заключена в фигурные скобки (например, “{ {1,2,3,4},{5,6,7,8} }”).

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetStringCallback` (стр. 604),
`rdscrtlSetBlockVarValue` (стр. 637).

Б.3.4.18. `rdscrtlGetMenuItemData` – получить данные пункта меню

Функция `rdscrtlGetMenuItemData` возвращает указатель на структуру данных пункта меню с заданным номером, считанную из управляемой копии РДС последним вызовом `rdscrtlReadBlockMenuItems` (стр. 635).

```
RDSCTRL_PMENUIITEM RDSCALL rdscrtlGetMenuItemData(  
    int Link,           // Идентификатор связи  
    int ItemNum        // Номер пункта меню  
);
```

Тип указателя на эту функцию:

`RDSCTRL_MiII`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`ItemNum`

Порядковый номер пункта меню, считанного в память.

Возвращаемое значение:

Указатель на структуру данных пункта меню `RDSCTRL_MENUITEM` (стр. 584) во внутренней памяти `RdsCtrl.dll`, или `NULL`, если в памяти нет пункта меню с номером `ItemNum`.

Примечания:

Для того, чтобы узнать, какие пункты тот или иной блок добавил в контекстное или главное меню РДС, управляющее приложение сначала считывает полный список этих пунктов в память RdsCtrl.dll вызовом `rdscrtlReadBlockMenuItems`, а затем перебирает их по одному вызовами `rdscrtlGetMenuItemData`. Данные пунктов меню хранятся в памяти RdsCtrl.dll независимо для каждой связи, поэтому вызов `rdscrtlGetMenuItemData(Link, n)` возвращает указатель на пункт меню номер `n` из набора, считанного вызовом `rdscrtlReadBlockMenuItems` именно для связи `Link`. Пункты контекстного меню нумеруются начиная с нуля в порядке своего расположения в меню, порядок пунктов главного меню определяется внутренней логикой РДС. Каждый вызов `rdscrtlReadBlockMenuItems` очищает список загруженных пунктов и считывает из РДС новые.

Пример использования функции `rdscrtlGetMenuItemData` приведен в §3.6.5.

См. также:

`rdscrtlCreateLink` (стр. 600), `RDSCRTL_MENUITEM` (стр. 584),
`rdscrtlReadBlockMenuItems` (стр. 635).

Б.3.4.19. `rdscrtlGetMode` – получить режим работы РДС

Функция `rdscrtlGetMode` возвращает режим, в котором в данный момент находится управляемая копия РДС.

```
int RDSCALL rdscrtlGetMode(  
    int Link // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCRTL_II`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Возвращаемое значение:

Одна из констант `RDSCRTL_MODE_*`:

<code>RDSCRTL_MODE_UNKNOWN</code>	Нет управляемой копии РДС.
<code>RDSCRTL_MODE_EDIT</code>	РДС в режиме редактирования.
<code>RDSCRTL_MODE_CALC</code>	РДС в режиме моделирования.
<code>RDSCRTL_MODE_RUNNING</code>	РДС в режиме расчета.

Примечания:

Эта функция запрашивает текущий режим у копии РДС, управляемой через связь `Link`. Если приложение `rds.exe` для этой связи не запущено, возвращается константа `RDSCRTL_MODE_UNKNOWN`.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetEditMode` (стр. 640),
`rdscrtlSetCalcMode` (стр. 638), `rdscrtlStartCalc` (стр. 642).

Б.3.4.20. `rdscrtlGetModFlag` – проверить наличие изменений в схеме

Функция `rdscrtlGetModFlag` возвращает признак наличия изменений в схеме, загруженной в управляемую копию РДС.

```
BOOL RDSCALL rdscrtlGetModFlag(  
    int Link           // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCTRL_BI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Возвращаемое значение:

`TRUE` – схема изменена, `FALSE` – изменений не было или схема не загружена.

Примечания:

Если с момента загрузки или записи схемы, находящейся в памяти копии РДС, управляемой через связь `Link`, в эту схему были внесены какие-либо изменения, функция вернет `TRUE`. При необходимости, флагом наличия изменений в схеме можно управлять вручную функцией `rdscrtlSetModFlag` (стр. 640).

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetModFlag` (стр. 640).

Б.3.4.21. `rdscrtlListBlocks` – получить список блоков

Функция `rdscrtlListBlocks` возвращает список имен блоков в указанной подсистеме схемы, загруженной в управляемую копию РДС.

```
int RDSCALL rdscrtlListBlocks(  
    int Link,           // Идентификатор связи  
    LPSTR SysName,     // Полное имя подсистемы  
    BOOL Recurse,      // Включать блоки подсистем  
    BOOL TypeChar,     // Добавлять букву типа  
    DWORD Types,       // Типы блоков  
    LPVOID ReturnList // Возвращаемый список  
);
```

Тип указателя на эту функцию:

`RDSCTRL_IISBBDwpV`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`SysName`

Указатель на строку с полным именем подсистемы, список блоков которой нужно получить (именем корневой подсистемы считается пустая строка). Полное имя подсистемы, как и любого другого блока, начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от

корневой подсистемы до этой. Например, полное имя “:Sys1:Sys100:MySys” говорит о том, что подсистема с именем MySys находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

Recurse

TRUE – включать в список блоки, находящиеся во внутренних подсистемах на всех уровнях иерархии, FALSE – включать в список только блоки, непосредственно находящиеся в SysName.

TypeChar

Добавлять (TRUE) или нет (FALSE) перед каждым именем блока букву, обозначающую его тип, и символ двоеточия. Для обозначения типов используются следующие буквы:

B	простой блок;
S	подсистема;
I	внешний вход;
O	внешний выход;
P	ввод шины.

Types

Маска типов блоков, которые нужно включить в список, или 0, если в список нужно включать все блоки независимо от их типа. Маска типов представляет собой стандартные константы типов блоков RDS_BT* (см. стр. 113), объединенные битовым ИЛИ:

RDS_BTSYSTEM	подсистема (включая корневую);
RDS_BTSIMPLEBLOCK	простой блок;
RDS_BTINPUTBLOCK	внешний вход;
RDS_BTOUTPUTBLOCK	внешний выход;
RDS_BTBUSPORT	ввод шины.

ReturnList

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи rdscrtlSetStringCallback (стр. 604), запишет текст списка имен блоков. Имена блоков в списке разделены кодом перевода строки “\n” (10). Список содержит не полные имена блоков, а относительные, в подсистеме SysName (см. примечание).

Возвращаемое значение:

Число блоков в списке.

Примечания:

Эта функция формирует список имен блоков, находящихся в подсистеме с полным именем SysName в схеме, загруженной в копию РДС, управляемую через связь Link. В список попадут блоки, типы которых соответствуют маске Types, и находящиеся непосредственно в подсистеме SysName (при Recurse==FALSE) или в подсистеме SysName и всех ее внутренних подсистемах (при Recurse==TRUE). Список записывается в объект, указатель на который передан в ReturnList, при помощи стандартной функции возврата строк, зарегистрированной вызовом rdscrtlSetStringCallback.

Вид списка, возвращаемого этой функцией, зависит от параметров Recurse и TypeChar. Предположим, что в корневой подсистеме схемы находится подсистема с именем “Sys1”, в которой содержатся простые блоки “Block1”, “Block2” и “Block3”, а также подсистема “Sys2”. В подсистеме “Sys2” при этом находятся блоки “Block1” и “Block5”. Теперь, если управляющая программа сделает вызов

```
rdscrtlListBlocks(Link, ":Sys1", FALSE, FALSE, 0, List)
```

(в обоих параметрах Recurse и TypeChar передано FALSE), то в объект List будет записан список следующего вида:

```
Block1  
Block2  
Block3  
Sys2
```

Список содержит не полные имена блоков, а имена, приведенные относительно подсистемы, для которой запрошен список. Для получения полного имени нужно добавить к именам в списке полное имя подсистемы и двоеточие: в данном случае, например, полное имя первого блока в списке будет “:Sys1:Block1”.

Если управляющая программа сделает вызов

```
rdscrtlListBlocks(Link, ":Sys1", TRUE, FALSE, 0, List)
```

(в параметре Recurse передано TRUE, в TypeChar – FALSE), то в объект List будет записан список, в который будут добавлены внутренние блоки подсистемы “Sys2”:

```
Block1  
Block2  
Block3  
Sys2  
Sys2:Block1  
Sys2:Block5
```

При вызове

```
rdscrtlListBlocks(Link, ":Sys1", FALSE, TRUE, 0, List)
```

(в параметре Recurse передано FALSE, в TypeChar – TRUE) перед каждым именем в списке будет добавлена буква типа (“B” для простых блоков, “S” для подсистем):

```
B:Block1  
B:Block2  
B:Block3  
S:Sys2
```

Наконец, при вызове

```
rdscrtlListBlocks(Link, ":Sys1", TRUE, TRUE, 0, List)
```

(в обоих параметрах Recurse и TypeChar передано TRUE) в объект List будет записан список с внутренними блоками подсистемы “Sys2” и буквами типов:

```
B:Block1  
B:Block2  
B:Block3  
S:Sys2  
B:Sys2:Block1  
B:Sys2:Block5
```

См. также:

rdscrtlCreateLink (стр. 600), rdscrtlFindOpSetProviders (стр. 628).

Б.3.4.22. rdscrtlMinimizeApp – свернуть приложение РДС

Функция rdscrtlMinimizeApp сворачивает управляемую копию РДС.

```
void RDSCALL rdscrtlMinimizeApp(
    int Link          // Идентификатор связи
);
```

Тип указателя на эту функцию:

RDSCtrl_VI

Параметр:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

Эта функция сворачивает копию РДС, управляемую через связь Link. Ее вызов не запоминается в параметрах связи, поэтому после завершения и повторного запуска главная программа РДС не будет автоматически свернута.

Для восстановления свернутого приложения следует использовать функцию `rdscrtlRestoreApp` (стр. 637).

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlRestoreApp` (стр. 637),
`rdscrtlBringAppToFront` (стр. 619).

Б.3.4.23. `rdscrtlReadBlockMenuItems` – считать пункты меню блока в память

Функция `rdscrtlReadBlockMenuItems` считывает во внутреннюю память библиотеки `RdsCtrl.dll` пункты, добавленные указанным блоком в контекстное или главное меню РДС.

```
int RDSCALL rdscrtlReadBlockMenuItems(
    int Link,          // Идентификатор связи
    LPSTR BlockName,   // Полное имя блока
    int Type           // Тип меню (RDSCtrl_MENUTYPE_*)
);
```

Тип указателя на эту функцию:

RDSCtrl_IISI

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

BlockName

Указатель на строку с полным именем блока (именем корневой подсистемы считается пустая строка). Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

Type

Одна из констант RDSCtrl_MENUTYPE_*, указывающая, пункты какого именно меню считываются:

RDSCtrl_MENUTYPE_BLK	пункты контекстного меню блока, вызываемого на его изображении;
RDSCtrl_MENUTYPE_SYS	пункты контекстного меню подсистемы, вызываемого на свободном месте ее окна;
RDSCtrl_MENUTYPE_MAIN	пункты, добавленные блоком в главное меню РДС “Система Дополнительно”.

Возвращаемое значение:

Число считанных в память пунктов меню.

Примечания:

Эта функция считывает во внутреннюю память библиотеки RdsCtrl.dll набор пунктов меню, добавленных блоком BlockName в контекстное или главное (в зависимости от Type) меню копии РДС, управляемой через связь Link. Для каждой связи выделяется своя, независимая от других связей, область памяти для хранения считанных пунктов меню, поэтому вызовы rdscctrlReadBlockMenuItems, сделанные для разных связей, не мешают друг другу. Считанные пункты меню хранятся в памяти до следующего вызова rdscctrlReadBlockMenuItems, их данные можно получать при помощи функции rdscctrlGetMenuItemData (стр. 630).

При запросе пунктов главного меню РДС (в параметре Type передана константа RDSCtrl_MENUTYPE_MAIN) имя блока, переданное в BlockName игнорируется, и возвращается полный список пунктов, добавленных в меню всеми блоками схемы.

Пример использования функции rdscctrlReadBlockMenuItems приведен в §3.6.5.

См. также:

rdscctrlCreateLink (стр. 600), rdscctrlGetMenuItemData (стр. 630),
RDSCtrl_MENUITEM (стр. 584).

Б.3.4.24. rdscctrlResetCalc – сбросить расчет

Функция rdscctrlResetCalc сбрасывает расчет в копии РДС, управляемой через указанную связь.

```
void RDSCALL rdscctrlResetCalc(  
    int Link          // Идентификатор связи  
);
```

Тип указателя на эту функцию:

RDSCtrl_VI

Параметр:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscctrlCreateLink (стр. 600). Расчет сбрасывается в копии РДС, управляемой через эту связь.

Примечания:

Эта функция сбрасывает расчет в копии РДС, управляемой через связь Link, то есть переводит всю схему в исходное состояние.

См. также:

rdctrlCreateLink (стр. 600), rdctrlStartCalc (стр. 642),
rdctrlStopCalc (стр. 642).

Б.3.4.25. rdctrlRestoreApp – восстановить свернутое приложение РДС

Функция rdctrlRestoreApp разворачивает ранее свернутую управляемую копию РДС.

```
void RDSCALL rdctrlRestoreApp(  
    int Link          // Идентификатор связи  
);
```

Тип указателя на эту функцию:

RDCTRL_VI

Параметр:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdctrlCreateLink (стр. 600).

Примечания:

Эта функция восстанавливает свернутую копию РДС, управляемую через связь Link. Для сворачивания приложения следует использовать функцию rdctrlMinimizeApp (стр. 634).

См. также:

rdctrlCreateLink (стр. 600), rdctrlMinimizeApp (стр. 634),
rdctrlBringAppToFront (стр. 619).

Б.3.4.26. rdctrlSetBlockVarValue – установить значение переменной блока

Функция rdctrlSetBlockVarValue изменяет текущее значение переменной с указанным именем в указанном блоке согласно переданной строке.

```
void RDSCALL rdctrlSetBlockVarValue(  
    int Link,          // Идентификатор связи  
    LPSTR BlockName,   // Полное имя блока  
    LPSTR VarName,     // Имя переменной  
    LPSTR ValueStr     // Строка значения  
);
```

Тип указателя на эту функцию:

RDCTRL_VISSS

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdctrlCreateLink (стр. 600).

BlockName

Указатель на строку с полным именем блока. Полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этого блока, которое завершается

именем самого блока. Например, полное имя “:Sys1:Sys100:Block1” говорит о том, что блок с именем Block1 находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

VarName

Указатель на строку с именем переменной, значение которой нужно установить. Для сложных переменных имя может содержать поля структур и индексы массивов в синтаксисе, принятом в РДС.

ValueStr

Указатель на строку, содержащую новое значение переменной.

Примечания:

Эта функция ищет в схеме, загруженной в копию РДС, управляемую через связь Link, блок с именем BlockName, в нем – переменную с именем VarName, и присваивает ей значение из строки ValueStr. Строка преобразуется в значение, соответствующее типу переменной. Имя переменной VarName может содержать поля структур и индексы массивов и матриц в синтаксисе РДС, например “Y.Re”, “A[1]”, “M[3,8]” или “Out.Array[3]”.

При установке значения структуры ее поля перечисляются через запятую внутри фигурных скобок: например, структуре с двумя вещественными полями можно передать строку значения “{1.0,2.0}”. Для установки значений матриц и массивов можно использовать один из двух возможных вариантов синтаксиса строки. Если все элементы матрицы/массива должны получить одно значение, в строке в квадратных скобках указывается размерность, а за ней – значение элемента. Например, получив строку “[10]3.4” массив вещественных чисел заполнится десятью одинаковыми значениями 3.4, а строка “[2,3]8”, переданная матрице, установит ее размер в 2x3 (две строки, три столбца), и заполнит ее всю значением 8. Если же значения элементов массива или матрицы необходимо установить индивидуально, они перечисляются через запятую внутри фигурных скобок: для массива – последовательно (например, “{1,2,3,4}”), для матрицы – по строкам, причем каждая строка тоже заключена в фигурные скобки (например, “{ {1,2,3,4},{5,6,7,8} }”). Размер матрицы или массива при этом будет автоматически определен по количеству переданных значений.

См. также:

rdscrtlCreateLink (стр. 600), rdscrtlGetBlockVarValue (стр. 629).

Б.3.4.27. rdscrtlSetCalcMode – включить режим моделирования

Функция rdscrtlSetCalcMode переводит управляемую копию РДС из режима редактирования в режим моделирования.

```
void RDSCALL rdscrtlSetCalcMode(  
    int Link          // Идентификатор связи  
);
```

Тип указателя на эту функцию:

RDSCRTL_VI

Параметр:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

Примечания:

Если копия РДС, управляемая через связь `Link`, находится в режиме редактирования, вызов этой функции переводит ее в режим моделирования. Для перехода в режим моделирования из режима расчета следует остановить расчет вызовом `rdscrtlStopCalc` (стр. 642).

Пример использования функции `rdscrtlSetCalcMode` приведен в §3.2.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetEditMode` (стр. 640),
`rdscrtlStopCalc` (стр. 642), `rdscrtlStartCalc` (стр. 642).

Б.3.4.28. `rdscrtlSetControllerName` – установить имя управляющей программы

Функция `rdscrtlSetControllerName` устанавливает строку имени внешней управляющей программы.

```
void RDSCALL rdscrtlSetControllerName(  
    int Link,           // Идентификатор связи  
    LPSTR Name         // Имя управляющей программы  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIS`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Name`

Указатель на строку с именем внешней управляющей программы.

Примечания:

Эта функция устанавливает для копии РДС, управляемой через связь `Link`, строку `Name` в качестве имени управляющей программы. Эта строка передается в РДС без обработки, модели блоков могут обращаться к ней через поле `ControllerName` структуры `RDS_REMOTEMSGDATA` (стр. 42) в момент реакции на событие `RDS_BFM_REMOTEMSG`, а также при помощи функции `rdscrtlGetRemoteControllerName` (стр. 396). Модели могут использовать имя управляющего приложения для того чтобы, например, выполнять разные действия в зависимости от того, какое именно приложение ими управляет. Имя приложения никак не связано с именем его исполняемого файла, это просто некоторая строка, о которой разработчик приложения должен договориться с разработчиками моделей блоков.

Управляющее приложение не обязано устанавливать свое имя: если оно не вызовет `rdscrtlSetControllerName`, в поле `ControllerName` структуры `RDS_REMOTEMSGDATA` будет записан указатель на пустую строку.

Строка с именем управляющего приложения запоминается в параметрах связи `Link`, ее не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlSetControllerName` один раз сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) эта строка будет передана в РДС автоматически.

См. также:

rdscrtlCreateLink (стр. 600),
RDS_BFM_REMOTEMSG, RDS_REMOTEMSGDATA (стр. 42),
rdsGetRemoteControllerName (стр. 396).

Б.3.4.29. rdscrtlSetEditMode – включить режим редактирования

Функция rdscrtlSetEditMode переводит управляемую копию РДС в режим редактирования.

```
void RDSCALL rdscrtlSetEditMode(  
    int Link          // Идентификатор связи  
);
```

Тип указателя на эту функцию:

RDSCTRL_VI

Параметр:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

Примечания:

Если копия РДС, управляемая через связь Link, находится в режимах моделирования или расчета, вызов этой функции переводит ее в режим редактирования. Включение режима редактирования обычно требует некоторого времени, поэтому на момент завершения функции rdscrtlSetEditMode режим РДС может еще не переключиться.

См. также:

rdscrtlCreateLink (стр. 600), rdscrtlSetCalcMode (стр. 638),
rdscrtlStartCalc (стр. 642), rdscrtlStopCalc (стр. 642).

Б.3.4.30. rdscrtlSetModFlag – управление флагом изменений в схеме

Функция rdscrtlSetModFlag устанавливает или сбрасывает флаг наличия изменений в схеме, загруженной в управляемую копию РДС.

```
void RDSCALL rdscrtlSetModFlag(  
    int Link,          // Идентификатор связи  
    BOOL Modified     // Наличие изменений  
);
```

Тип указателя на эту функцию:

RDSCTRL_VIB

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

Modified

TRUE – взвести флаг (РДС будет считать схему измененной), FALSE – сбросить флаг (РДС будет считать, что в схеме не было изменений).

Примечания:

Флаг наличия изменений в схеме автоматически взводится РДС при любом изменении схемы (добавлении или удалении блоков, настройки их параметров и т.п.) и автоматически сбрасывается при сохранении схемы. От состояния этого флага зависит, будет ли выведено предупреждение пользователю перед выгрузкой измененной схемы и будет ли выполнено автосохранение схемы, если оно включено (см. функцию `rdscrtlSetAutoSave`, стр. 612). Функция `rdscrtlSetModFlag` позволяет вручную управлять этим флагом в схеме, загруженной в копию РДС, управляемую через связь `Link`.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlGetModFlag` (стр. 632),
`rdscrtlSetAutoSave` (стр. 612).

Б.3.4.31. `rdscrtlSetString` – установить глобальную строку

Функция `rdscrtlSetString` устанавливает строку с идентификатором, по которому модели блоков могут обращаться к этой строке.

```
void RDSCALL rdscrtlSetString(  
    int Link,           // Идентификатор связи  
    int Id,             // Идентификатор строки  
    LPSTR String        // Строка  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIIS`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Id`

Целый идентификатор устанавливаемой строки.

`String`

Указатель на текст устанавливаемой строки.

Примечания:

Эта функция устанавливает в копии РДС, управляемой через связь `Link`, строку с идентификатором `Id` и текстом `String`. Модель любого блока схемы, загруженной в эту копию РДС, может обратиться к строке по идентификатору при помощи функции `rdsGetRemoteControllerString` (стр. 396). Например, если управляющее приложение сделает вызов `rdscrtlSetString(Link, 23, "abcd")`, модель блока, сделав вызов `rdsGetRemoteControllerString(23)`, получит указатель на строку "abcd".

Набор строк с идентификаторами запоминается в параметрах связи `Link`, эти строки не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, сделать все вызовы `rdscrtlSetString` один раз сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) все эти строки будут переданы в РДС автоматически.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdsGetRemoteControllerString` (стр. 396),
`rdscrtlConnect` (стр. 600).

Б.3.4.32. `rdscrtlStartCalc` – запустить расчет

Функция `rdscrtlStartCalc` запускает расчет в управляемой копии РДС.

```
void RDSCALL rdscrtlStartCalc(  
    int Link           // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

Если копия РДС, управляемая через связь `Link`, находится в режимах моделирования или редактирования, вызов этой функции переводит ее в режим расчета. Включение режима расчета требует некоторого времени, поэтому на момент завершения функции `rdscrtlStartCalc` расчет может еще не запуститься.

Вызов `rdscrtlStartCalc` не сбрасывает расчет перед запуском, фактически, эта функция продолжает остановленный расчет. Для сброса расчета следует использовать вызов `rdscrtlResetCalc` (стр. 636).

Пример использования функции `rdscrtlStartCalc` приведен в §3.2.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetEditMode` (стр. 640),
`rdscrtlSetCalcMode` (стр. 638), `rdscrtlResetCalc` (стр. 636),
`rdscrtlStopCalc` (стр. 642).

Б.3.4.33. `rdscrtlStopCalc` – остановить расчет

Функция `rdscrtlStopCalc` останавливает работающий расчет в управляемой копии РДС.

```
void RDSCALL rdscrtlStopCalc(  
    int Link           // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

Если копия РДС, управляемая через связь `Link`, находится в режиме расчета, вызов этой функции останавливает расчет и переводит ее в режим моделирования. В режимах моделирования и редактирования вызов функции игнорируется. Остановка расчета требует некоторого времени, поэтому на момент завершения функции `rdscrtlStopCalc` расчет может еще работать.

Пример использования функции `rdscrtlStopCalc` приведен в §3.2.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetEditMode` (стр. 640),
`rdscrtlSetCalcMode` (стр. 638), `rdscrtlResetCalc` (стр. 636),
`rdscrtlStartCalc` (стр. 642).

Б.3.5. Функции загрузки и сохранения схемы

Описываются функции, позволяющие загружать и сохранять схемы в управляемой копии РДС, а также передавать содержимое схемы из управляющего приложения в РДС и обратно.

Б.3.5.1. `rdscrtlDeleteExchangeMemory` – удалить разделяемую память

Функция `rdscrtlDeleteExchangeMemory` освобождает разделяемую память, использовавшуюся для передачи схемы между РДС и управляющим приложением функциями `rdscrtlSaveSystemTaggedMem` (стр. 658) и, в одном из случаев, `rdscrtlSaveSystemTaggedEx` (стр. 656).

```
void RDSCALL rdscrtlDeleteExchangeMemory(  
    int Link           // Идентификатор связи  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VI`

Параметр:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

Эта функция дает копии РДС, управляемой через связь `Link`, команду освободить разделяемую область памяти, созданную для передачи загруженной в данный момент схемы управляющему приложению. Такая разделяемая область создается в двух случаях:

- при вызове функции `rdscrtlSaveSystemTaggedMem`;
- при вызове функции `rdscrtlSaveSystemTaggedEx` с указанием флага `RDSCRTL_TAGGED_SHAREDMMEM`.

В обоих случаях РДС создает разделяемую область памяти, записывает туда все данные схемы и оставляет эту область открытой, чтобы она не уничтожилась. Разделяемые области памяти в Windows автоматически уничтожаются, как только будет закрыт последний дескриптор (`HANDLE`), указывающий на нее. Когда управляющее приложение откроет разделяемую область, считывает из нее данные и закроет ее, оно должно вызвать функцию `rdscrtlDeleteExchangeMemory`, чтобы на стороне РДС эта область тоже закрылась и освободилась, поскольку на нее больше никто не ссылается.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSaveSystemTaggedMem` (стр. 658),
`rdscrtlSaveSystemTaggedEx` (стр. 656).

Б.3.5.2. `rdscrtlEndBlockByBlockLoad` – завершить поблочную загрузку схемы

Функция `rdscrtlEndBlockByBlockLoad` завершает или отменяет поблочную загрузку схемы, начатую вызовом `rdscrtlStartBlockByBlockLoad` (стр. 663).

```

BOOL RDSCALL rdscrtlEndBlockByBlockLoad(
    int Link,           // Идентификатор связи
    BOOL Apply         // Применить или отменить
);

```

Тип указателя на эту функцию:

RDSCTRL_BIB

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Apply

TRUE – собрать схему из переданных в РДС данных блоков и связей, FALSE – отменить сборку схемы и удалить переданные данные.

Возвращаемое значение:

TRUE – схема успешно собрана из переданных объектов, FALSE – при сборке схемы возникла ошибка.

Примечания:

Эта функция завершает поблочную передачу схемы из управляющего приложения в копию РДС, управляемую через связь Link. Поблочная загрузка начинается вызовом `rdscrtlStartBlockByBlockLoad`, за которым следуют вызовы `rdscrtlSetBlockByBlockLoadPiece` (стр. 661), каждый из которых передает из управляющего приложения в РДС блок, связь, шину или какие-либо вспомогательные данные. После того, как данные всех блоков и связей переданы в РДС, управляющее приложение вызывает `rdscrtlEndBlockByBlockLoad` с `Apply==TRUE` – в этот момент из переданных данных в РДС формируется схема. Для отмены загрузки и уничтожения уже переданных данных можно в любой момент вызвать `rdscrtlEndBlockByBlockLoad` с `Apply==FALSE`, при этом схема, загруженная в данный момент в РДС, останется в памяти неизменной.

Пример использования функции `rdscrtlEndBlockByBlockLoad` приведен в §3.5.

См. также:

`rdscrtlCreateLink` (стр. 600),
`rdscrtlSetBlockByBlockLoadPiece` (стр. 661),
`rdscrtlStartBlockByBlockLoad` (стр. 663).

Б.3.5.3. `rdscrtlEndBlockByBlockSave` – завершить поблочное сохранение схемы

Функция `rdscrtlEndBlockByBlockSave` завершает или отменяет поблочное сохранение схемы, начатое вызовом `rdscrtlStartBlockByBlockSave` (стр. 664).

```

void RDSCALL rdscrtlEndBlockByBlockSave(
    int Link           // Идентификатор связи
);

```

Тип указателя на эту функцию:

RDSCTRL_VI

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Примечания:

Эта функция завершает поблочную передачу схемы из копии РДС, управляемой через связь Link, в управляющее приложение. Поблочное сохранение начинается вызовом `rdscrtlStartBlockByBlockSave`, за которым следуют вызовы `rdscrtlGetBlockByBlockSavePiece` (стр. 645), каждый из которых получает из РДС блок, связь, шину или какие-либо вспомогательные данные. После того, как данные всех блоков и связей получены из РДС, управляющее приложение вызывает `rdscrtlEndBlockByBlockSave` для освобождения вспомогательной памяти, отведенной `RdsCtrl.dll` для временного хранения передаваемых данных.

Пример использования функции `rdscrtlEndBlockByBlockSave` приведен в §3.5.

См. также:

`rdscrtlCreateLink` (стр. 600),
`rdscrtlGetBlockByBlockSavePiece` (стр. 645),
`rdscrtlStartBlockByBlockSave` (стр. 664).

Б.3.5.4. `rdscrtlGetBlockByBlockSavePiece` – получить очередной объект при поблочном сохранении

Функция `rdscrtlGetBlockByBlockSavePiece` считывает из управляемой копии РДС данные очередного объекта в процессе поблочного сохранения схемы.

```
int RDSCALL rdscrtlGetBlockByBlockSavePiece(  
    int Link,                // Идентификатор связи  
    DWORD *pExtId,           // Идентификатор объекта  
    DWORD *pParentId,        // Идентификатор родителя  
    LPVOID ReturnTxt         // Текст описания объекта  
);
```

Тип указателя на эту функцию:

`RDSCRTL_IIPDwpDwpV`

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

pExtId

Указатель на целую (DWORD) переменную, в которую функция запишет уникальный внешний идентификатор считанного блока, связи или шины (см. стр. 207). Корневая подсистема схемы всегда имеет нулевой идентификатор. Для вспомогательных объектов, не имеющих идентификаторов, в эту переменную записывается ноль. Если вызывающей программе не нужен этот идентификатор, в параметре `pExtId` можно передать NULL.

pParentId

Указатель на целую (DWORD) переменную, в которую функция запишет уникальный внешний идентификатор родительской подсистемы считанного блока, связи или шины. Для вспомогательных объектов и корневой подсистемы, не имеющих

родительской подсистемы в принципе, в эту переменную записывается ноль. Если вызывающей программе не нужен этот идентификатор, в параметре `pParentId` можно передать `NULL`.

`ReturnTxt`

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи `rdscrtlSetStringCallback` (стр. 604), запишет текстовое описание полученного из РДС объекта.

Возвращаемое значение:

Тип полученного из РДС объекта – одна из констант `RDS_SFTAG_*`:

<code>RDS_SFTAG_ROOT</code>	корневая подсистема;
<code>RDS_SFTAG_SIMPLEBLOCK</code>	простой блок;
<code>RDS_SFTAG_SYSTEM</code>	подсистема (кроме корневой);
<code>RDS_SFTAG_INPUTBLOCK</code>	внешний вход;
<code>RDS_SFTAG_OUTPUTBLOCK</code>	внешний выход;
<code>RDS_SFTAG_BUSPORT</code>	ввод шины;
<code>RDS_SFTAG_CONNECTION</code>	связь;
<code>RDS_SFTAG_BUS</code>	шина;
<code>RDS_SFTAG_CONNSTYLES</code>	стили связей (дополнительный блок данных);
<code>RDS_SFTAG_TYPES</code>	описания структур (дополнительный блок данных);
<code>RDS_SFTAG_EOF</code>	объекты кончились, больше нет данных.

Примечания:

Эта функция получает из копии РДС, управляемой через связь `Link`, описание очередного объекта схемы. После вызова функции `rdscrtlStartBlockByBlockSave` (стр. 664) управляющее приложение может считывать загруженную в память РДС схему объект за объектом при помощи `rdscrtlGetBlockByBlockSavePiece`. Каждый вызов передает из РДС очередной объект, при этом его уникальный идентификатор, если он есть, записывается по указателю `pExtId`, идентификатор его родительской подсистемы (опять же, если она есть) – по указателю `pParentId`, а текстовое описание самого объекта записывается в объект управляющего приложения `ReturnTxt` стандартной функцией обратного вызова, зарегистрированной при помощи `rdscrtlSetStringCallback`.

Объектами, получаемыми из РДС, в данном случае являются блоки, связи и шины (только у них есть идентификаторы), а также набор стилей связей и набор описаний структур, хранящиеся в схеме. Два последних объекта не являются обязательными для сохранения – вся необходимая для работы схемы информация о внешнем виде связей находится в описаниях самих связей, а описания структур – в описаниях блоков, которые их используют. Следует помнить, что в описание подсистемы, возвращаемое через параметр `ReturnTxt`, не входят внутренние блоки этой подсистемы – они будут переданы отдельно, при следующих вызовах `rdscrtlGetBlockByBlockSavePiece`.

Результатом функции является число, соответствующее типу объекта, данные которого записаны в `pExtId`, `pParentId` и `ReturnTxt`. Возврат функцией константы `RDS_SFTAG_EOF` свидетельствует о том, что объекты в схеме закончились. Получив эту константу, управляющее приложение должно вызвать функцию `rdscrtlEndBlockByBlockSave` (стр. 644) для завершения поблочного сохранения схемы.

Чаще всего поблочное сохранение схемы используется в тех случаях, когда управляющее приложение самостоятельно реализует хранение схем РДС (например, в какой-либо базе данных) и вмешивается в процесс загрузки и сохранения схемы. Данные, полученные при поблочном сохранении схемы функцией

rdscrtlGetBlockByBlockSavePiece, можно передать в РДС при ее поблочной загрузке функцией rdscrtlSetBlockByBlockLoadPiece (стр. 661). Следует отметить, что, хотя идентификаторы объектов и их родительских подсистем возвращаются этой функцией вместе с типом и описанием объекта, сохранять эти идентификаторы отдельно не обязательно. Они входят, вместе с другими параметрами, в описание самого объекта и при поблочной загрузке не используются. Управляющая программа может использовать эти идентификаторы, например, для группировки блоков по подсистемам при хранении в базе данных.

Пример использования функции rdscrtlGetBlockByBlockSavePiece приведен в §3.5.

См. также:

rdscrtlCreateLink (стр. 600), rdscrtlStartBlockByBlockSave (стр. 664),
rdscrtlEndBlockByBlockSave (стр. 644),
rdscrtlSetBlockByBlockLoadPiece (стр. 661),
rdscrtlSetStringCallback (стр. 604).

Б.3.5.5. rdscrtlGetSystemContent – получить полный текст схемы

Функция rdscrtlGetSystemContent возвращает схему, загруженную в данный момент в управляемую копию РДС, в текстовом формате.

```
BOOL RDSCALL rdscrtlGetSystemContent(  
    int Link,                // Идентификатор связи  
    LPVOID ReturnText,      // Возвращаемый текст  
    int Mode,               // Способ передачи (RDSCTRL_GSC_*)  
    RDSCTRL_RETURNSTRING Func // Функция возврата  
);
```

Тип указателя на эту функцию:

RDSCTRL_BIPVIRs

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

ReturnText

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи rdscrtlSetStringCallback (стр. 604) или переданная в параметре Func, запишет текст схемы.

Mode

Способ передачи текста схемы между РДС и управляющим приложением:

RDSCTRL_GSC_TRANSFILE текст схемы будет записан во временный файл, а затем
загружен библиотекой RdsCtrl.dll в память;
RDSCTRL_GSC_TRANSPIPE текст схемы будет передан в RdsCtrl.dll через “трубу”
(одно из средств взаимодействия процессов в
Windows).

Func

Указатель на функцию обратного вызова, которую нужно использовать для записи текста в ReturnText, или NULL, если следует использовать стандартную функцию, зарегистрированную при помощи rdscrtlSetStringCallback.

Возвращаемое значение:

TRUE – текст схемы успешно передан, FALSE – при передаче возникла ошибка.

Примечания:

Эта функция записывает в объект, указатель на который передан через параметр `ReturnText`, всю схему, загруженную на данный момент в копию РДС, управляемую через связь `Link`. Схема преобразуется в текстовый формат, который используется в РДС для хранения блоков и схем в файлах. Таким образом, если записать текст, полученный объектом `ReturnText`, в отдельный файл, схему из этого файла можно будет загрузить средствами РДС.

Параметр `Mode` управляет способом передачи текста схемы из РДС в управляющее приложение. Разработчик может использовать любое из двух его значений в зависимости от специфики работы управляющего приложения (передача через файл может быть медленнее, но экономнее по памяти). В большинстве случаев целесообразно использовать способ `RDSCtrl_GSC_TRANSPIPE`.

Данные схемы, переданные в управляющее приложение функцией `rdscctrlGetSystemContent`, могут быть потом переданы обратно в РДС парной к ней функцией `rdscctrlLoadSystemFromMem` (стр. 649). Следует помнить, что текст схемы с большим количеством блоков и связей занимает значительный объем памяти.

См. также:

`rdscctrlCreateLink` (стр. 600), `rdscctrlSetStringCallback` (стр. 604),
`rdscctrlLoadSystemFromMem` (стр. 649).

Б.3.5.6. `rdscctrlLoadSystemFromFile` – загрузить схему из файла

Функция `rdscctrlLoadSystemFromFile` загружает в управляемую копию РДС схему из файла с указанным именем.

```
BOOL RDSCALL rdscctrlLoadSystemFromFile(  
    int Link,           // Идентификатор связи  
    LPSTR FileName,    // Имя файла  
    BOOL SavePrompt    // Предложить сохранение  
);
```

Тип указателя на эту функцию:

`RDSCtrl_BISB`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscctrlCreateLink` (стр. 600).

`FileName`

Указатель на строку с полным путем к загружаемому файлу.

`SavePrompt`

TRUE – предлагать пользователю сохранить загруженную в данный момент схему, если в ней были изменения, перед загрузкой новой схемы из файла `FileName`.
FALSE – выгрузить старую схему из памяти без запросов пользователю, не сохраняя ее.

Возвращаемое значение:

TRUE – загрузка выполнена успешно, FALSE – при загрузке возникли ошибки.

Примечания:

Эта функция загружает новую схему в копию РДС, управляемую через связь `Link`, из файла `FileName`. Пример ее использования приведен в §3.2.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSaveSystemToFile` (стр. 660).

Б.3.5.7. `rdscrtlLoadSystemFromMem` – загрузить схему из области памяти

Функция `rdscrtlLoadSystemFromMem` загружает в управляемую копию РДС схему в текстовом формате из указанной области памяти.

```
BOOL RDSCALL rdscrtlLoadSystemFromMem(  
    int Link,           // Идентификатор связи  
    LPVOID Start,      // Указатель на начало области  
    DWORD Size,        // Размер области  
    DWORD Flags        // Флаги (RDSCRTL_LSFM_*)  
);
```

Тип указателя на эту функцию:

`RDSCRTL_BIPVDwDw`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Start`

Указатель на начало области памяти, в которой находится схема в текстовом формате.

`Size`

Размер области памяти `Start` в байтах (длина текста схемы).

`Flags`

Битовые флаги `RDSCRTL_LSFM_*`, управляющие поведением функции и способом передачи данных:

`RDSCRTL_LSFM_SAVEPROMPT` Если этот флаг взведен, РДС перед загрузкой схемы предложит пользователю сохранить текущую схему, если в ней были сделаны какие-либо изменения.

`RDSCRTL_LSFM_TRANSMAP` Текст схемы из управляющего приложения в РДС будет передаваться через разделяемую область памяти.

`RDSCRTL_LSFM_TRANSFILE` Текст схемы из управляющего приложения в РДС будет передаваться через временный файл.

Флаги `RDSCRTL_LSFM_TRANSMAP` и `RDSCRTL_LSFM_TRANSFILE` являются взаимоисключающими, может быть указан только один из них.

Возвращаемое значение:

`TRUE` – загрузка выполнена успешно, `FALSE` – при загрузке возникли ошибки.

Примечания:

Эта функция загружает новую схему в копию РДС, управляемую через связь `Link`, из области памяти управляющего приложения, начинающейся с адреса `Start` и занимающей `Size` байтов. Схема в памяти должна быть записана в текстовом формате, используемом в РДС для хранения блоков и схем в файлах. При помощи флагов

RDSCTRL_LSFM_TRANSMAP и RDSCTRL_LSFM_TRANSFILE можно указать способ передачи текста схемы из управляющего приложения в РДС – передача через файл может быть медленнее, но экономнее по памяти. В большинстве случаев целесообразно использовать способ RDSCTRL_LSFM_TRANSMAP.

Функция `rdscrtlLoadSystemFromMem` совместима по формату данных с функцией `rdscrtlGetSystemContent` (стр. 647): данные, полученные из РДС при помощи `rdscrtlGetSystemContent`, могут быть потом переданы обратно функцией `rdscrtlLoadSystemFromMem`.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlGetSystemContent` (стр. 647).

Б.3.5.8. `rdscrtlLoadSystemTagged` – загрузить схему из файла в специальном двоичном формате

Функция `rdscrtlLoadSystemTagged` загружает в управляемую копию РДС схему в специальном двоичном формате из файла с указанным именем.

```
BOOL RDSCALL rdscrtlLoadSystemTagged(  
    int Link,           // Идентификатор связи  
    LPSTR FileName,    // Имя файла  
    DWORD Flags        // Флаги (не используются)  
);
```

Тип указателя на эту функцию:

RDSCTRL_BISDw

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`FileName`

Указатель на строку с полным путем к загружаемому файлу.

`Flags`

Флаги функции (в настоящее время этот параметр никак не используется).

Возвращаемое значение:

TRUE – загрузка выполнена успешно, FALSE – при загрузке возникли ошибки.

Примечания:

Эта функция загружает новую схему в копию РДС, управляемую через связь `Link`, из файла `FileName`. Схема, находившаяся в памяти РДС до вызова функции, безвозвратно теряется, изменения в ней не сохраняются. Файл `FileName` должен содержать схему в специальном двоичном формате, который формируется функцией сохранения схемы `rdscrtlSaveSystemTagged` (стр. 654).

Следует помнить, что, несмотря на присутствие в этом формате (см. стр. 654) идентификаторов блоков, связей и шин и их родительских подсистем в виде двух отдельных тридцатидвухбитных чисел перед текстовым описанием каждого объекта, при загрузке схемы эти два числа не используются – вместо них идентификаторы считываются из текстового описания самого объекта. По этой причине, формируя данные для загрузки схемы функцией `rdscrtlLoadSystemTagged`, нельзя изменить идентификаторы объектов

схемы, заменив пару чисел в блоке описания объекта. Поскольку эти два числа в блоке описания игнорируются при загрузке, они могут быть любыми.

В настоящее время вместо функции `rdctrlLoadSystemTagged` чаще используется поблочная загрузка схемы (см. §3.5), которая работает с близким форматом данных.

См. также:

`rdctrlCreateLink` (стр. 600), `rdctrlSaveSystemTagged` (стр. 654).

Б.3.5.9. `rdctrlLoadSystemTaggedEx` – загрузить схему из файла или памяти в специальном двоичном формате

Функция `rdctrlLoadSystemTaggedEx` загружает в управляемую копию РДС схему в специальном двоичном формате из файла или разделяемой области памяти с указанным именем.

```
BOOL RDSCALL rdctrlLoadSystemTaggedEx(  
    int Link,           // Идентификатор связи  
    LPSTR FileName,    // Имя файла или области памяти  
    DWORD Flags,        // флаги (RDCTRL_TAGGED_*)  
    DWORD Size         // Размер области памяти  
);
```

Тип указателя на эту функцию:

`RDCTRL_BISDwDw`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdctrlCreateLink` (стр. 600).

`FileName`

Указатель на строку с полным путем к загружаемому файлу или с именем разделяемой области памяти.

`Flags`

Возможна передача единственного флага `RDCTRL_TAGGED_SHAREDMMEM`, указывающего на размещение загружаемой схемы: если он сброшен, схема будет загружаться из файла с именем `FileName`, если взведен – из разделяемой области памяти с именем `FileName`.

`Size`

Размер разделяемой области `FileName` в байтах при взведенном флаге `RDCTRL_TAGGED_SHAREDMMEM`. При сброшенном флаге параметр игнорируется.

Возвращаемое значение:

`TRUE` – загрузка выполнена успешно, `FALSE` – при загрузке возникли ошибки.

Примечания:

Эта функция загружает новую схему в копию РДС, управляемую через связь `Link`, из файла или области памяти с именем `FileName`. Схема, находившаяся в памяти РДС до вызова функции, безвозвратно теряется, изменения в ней не сохраняются. Файл или область памяти `FileName` должны содержать схему в специальном двоичном формате, который формируется функцией сохранения схемы `rdctrlSaveSystemTagged` (стр. 654).

При взведенном флаге `RDSCTRL_TAGGED_SHAREDMMEM` схема загружается не из файла с именем `FileName`, а из разделяемой области памяти с этим именем, созданной функцией Windows API `CreateFileMapping`. В этом случае процедура загрузки схемы выглядит примерно следующим образом:

```
char Name[]="MyGlobalMemory"; // Имя разделяемой области памяти
DWORD Size=... // Размер схемы в специальном формате
// Создание разделяемой области
HANDLE FileMapping=CreateFileMapping(
    (HANDLE)0xFFFFFFFF, // Нет связанного файла
    NULL,                // Атрибуты безопасности
    PAGE_READWRITE,      // Чтение/запись
    0,                   // Старшее 32хбитное слово размера
    Size,                // Младшее 32хбитное слово размера
    Name);               // Имя создаваемой области
if(FileMapping==NULL)
{
    ...Ошибка...
}
// Подключение области к адресному пространству процесса и
// получение указателя на нее
void *FileMappingPtr=MapViewOfFile(
    FileMapping,          // Разделяемая область
    FILE_MAP_WRITE,       // Доступ для записи
    0,0,                  // Смещение от начала (нет)
    Size);               // Размер запрашиваемой части
if(FileMappingPtr==NULL)
{
    ...Ошибка...
}

... запись данных схемы по адресу FileMappingPtr ...

// Загрузка схемы в РДС
BOOL result=rdscrtLoadSystemTaggedEx(
    Link,Name,RDSCTRL_TAGGED_SHAREDMMEM,Size);
// Уничтожение разделяемой области
UnMapViewOfFile(FileMappingPtr);
CloseHandle(FileMapping);
```

Следует помнить, что, несмотря на присутствие в этом формате (см. стр. 654) идентификаторов блоков, связей и шин и их родительских подсистем в виде двух отдельных тридцатидвухбитных чисел перед текстовым описанием каждого объекта, при загрузке схемы эти два числа не используются – вместо них идентификаторы считываются из текстового описания самого объекта. По этой причине, формируя данные для загрузки схемы функцией `rdscrtLoadSystemTaggedEx`, нельзя изменить идентификаторы объектов схемы, заменив пару чисел в блоке описания объекта. Поскольку эти два числа в блоке описания игнорируются при загрузке, они могут быть любыми.

В настоящее время вместо функции `rdscrtLoadSystemTaggedEx` чаще используется поблочная загрузка схемы (см. §3.5), которая работает с близким форматом данных.

См. также:

`rdscrtCreateLink` (стр. 600), `rdscrtSaveSystemTagged` (стр. 654).

Б.3.5.10. `rdscrtNoDirectLoad` – запрет загрузки схемы пользователем

Функция `rdscrtNoDirectLoad` разрешает или запрещает пользователю загружать схемы в РДС.

```

void RDSCALL rdsctrlNoDirectLoad(
    int Link,           // Идентификатор связи
    BOOL NoLoad        // Запрет загрузки
);

```

Тип указателя на эту функцию:

RDSCTRL_VIB

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdsctrlCreateLink (стр. 600).

NoLoad

TRUE – загрузка схемы пользователем заблокирована, FALSE – загрузка разрешена.

Примечания:

Эта функция запрещает (при NoLoad==TRUE) или разрешает (при NoLoad==FALSE) пользователю самостоятельно загружать схемы в копию РДС, управляемую через связь Link. По умолчанию загрузка схем разрешена, и пользователь может самостоятельно выбирать загружаемые файлы через меню РДС. Если загрузка схем запрещена, при выборе пользователем пункта меню РДС “Файл | Загрузить” или нажатии соответствующей этому пункту кнопки вместо того, чтобы открыть стандартный диалог выбора файла, РДС пошлет управляющему приложению событие RDSCTRLEVENT_LOADREQ (стр. 596). Реагируя на него, приложение должно самостоятельно предоставить пользователю возможность выбрать загружаемую схему и передать ее в РДС при помощи одной из описанных в данном разделе функций. Это дает управляющему приложению возможность перехватывать попытки пользователя загрузить схему и самостоятельно организовать хранение схем, например, в базе данных.

Разрешение загрузки схем запоминается в параметрах связи Link, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать rdsctrlNoDirectLoad(Link, TRUE) сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова rdsctrlConnect (стр. 600) РДС запустится уже с заблокированной загрузкой схем.

Пример использования функции rdsctrlNoDirectLoad приведен в §3.5.

См. также:

rdsctrlCreateLink (стр. 600), RDSCTRLEVENT_LOADREQ (стр. 596),
rdsctrlNoDirectSave (стр. 653), rdsctrlConnect (стр. 600).

Б.3.5.11. rdsctrlNoDirectSave – запрет сохранения схемы пользователем

Функция rdsctrlNoDirectSave разрешает или запрещает пользователю сохранять схемы.

```

void RDSCALL rdsctrlNoDirectSave(
    int Link,           // Идентификатор связи
    BOOL NoSave        // Запрет сохранения
);

```

Тип указателя на эту функцию:

RDSCTRL_VIB

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

NoSave

TRUE – сохранение схемы пользователем заблокировано, FALSE – сохранение разрешено.

Примечания:

Эта функция запрещает (при `NoSave==TRUE`) или разрешает (при `NoSave==FALSE`) пользователю самостоятельно сохранять схемы из копии РДС, управляемой через связь Link. По умолчанию сохранение схем разрешено, и пользователь может сохранять схемы в файлы через меню РДС. Если сохранение схем запрещено, при выборе пользователем пункта меню РДС “Файл | Сохранить” или нажатии соответствующей этому пункту кнопки РДС сразу пошлет управляющему приложению событие `RDSCTRLEVENT_SAVEFILE` (стр. 598), не сохраняя при этом схему. Реагируя на это событие, приложение должно самостоятельно сохранить схему при помощи одной из описанных в данном разделе функций. Это дает управляющему приложению возможность перехватывать попытки пользователя сохранить схему и самостоятельно организовать ее хранение, например, в базе данных.

Разрешение сохранения схемы запоминается в параметрах связи Link, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать `rdscrtlNoDirectSave(Link, TRUE)` сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова `rdscrtlConnect` (стр. 600) РДС запустится уже с заблокированным сохранением схем.

Пример использования функции `rdscrtlNoDirectSave` приведен в §3.5.

См. также:

`rdscrtlCreateLink` (стр. 600), `RDSCTRLEVENT_SAVEFILE` (стр. 598),
`rdscrtlNoDirectLoad` (стр. 652), `rdscrtlConnect` (стр. 600).

Б.3.5.12. `rdscrtlSaveSystemTagged` – записать схему в файл в специальном двоичном формате

Функция `rdscrtlSaveSystemTagged` записывает схему, загруженную в данный момент в управляемую копию РДС, в файл с указанным именем в специальном двоичном формате.

```
BOOL RDSCALL rdscrtlSaveSystemTagged(  
    int Link,           // Идентификатор связи  
    LPSTR FileName,    // Имя файла  
    DWORD Flags        // Флаги (не используются)  
);
```

Тип указателя на эту функцию:

`RDSCTRL_BISDw`

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

FileName

Указатель на строку с полным путем к записываемому файлу.

Flags

Флаги функции (в настоящее время этот параметр никак не используется).

Возвращаемое значение:

TRUE – запись выполнена успешно, FALSE – при записи возникли ошибки.

Примечания:

Эта функция записывает схему из копии РДС, управляемой через связь Link, в файл FileName с использованием специального двоичного формата, близкого к параметрам функций поблочного сохранения и загрузки схем rdscrtlGetBlockByBlockSavePiece (стр. 645) и rdscrtlSetBlockByBlockLoadPiece (стр. 661). В этом формате каждый блок данных начинается с однобайтовой константы из набора RDS_SFTAG_*, указывающей на тип этого блока. Все эти константы описаны в файле “RdsDef.h”

Первый блок данных всегда начинается с константы RDS_SFTAG_DATATYPE, за которым следует четырехбайтовое беззнаковое число номера версии формата – на данный момент это число всегда равно нулю. Таким образом, первый блок данных двоичного файла всегда занимает пять байтов и в текущей версии РДС выглядит так:

RDS_SFTAG_DATATYPE (0xff)	1 байт
—	—
— 0 —	4 байта
—	—

За этим блоком данных в произвольном порядке следуют другие, описывающие блоки, связи и шины схемы, а также наборы стилей связей и типов структур. Типы структур, хранящиеся в схеме, сохраняются следующим образом:

RDS_SFTAG_TYPES (0x02)	1 байт
—	—
— длина текстового описания (N) —	4 байта
—	—
текстовое описание структур	N байтов

Описание стилей связей устроено похожим образом:

RDS_SFTAG_CONNSTYLES (0x01)	1 байт
—	—
— длина текстового описания (N) —	4 байта
—	—
текстовое описание стилей	N байтов

И описание типов структур, и описание стилей связей не являются необходимыми блоками данных – все параметры внешнего вида связей и описания структур, используемых блоками, всегда сохраняются внутри описаний этих связей и блоков.

Описание любого блока, связи или шины выглядит следующим образом:

тип объекта	1 байт
внешний идентификатор объекта	4 байта
внешний идентификатор родительской подсистемы	4 байта
длина текстового описания (N)	4 байта
текстовое описание объекта	N байтов

При этом, как и при поблочном сохранении схемы (стр. 645), для типов объектов используются следующие константы:

RDS_SFTAG_ROOT	(0x15)	– корневая подсистема;
RDS_SFTAG_SIMPLEBLOCK	(0x10)	– простой блок;
RDS_SFTAG_SYSTEM	(0x11)	– подсистема (кроме корневой);
RDS_SFTAG_INPUTBLOCK	(0x12)	– внешний вход;
RDS_SFTAG_OUTPUTBLOCK	(0x13)	– внешний выход;
RDS_SFTAG_BUSPORT	(0x14)	– ввод шины;
RDS_SFTAG_CONNECTION	(0x20)	– связь;
RDS_SFTAG_BUS	(0x21)	– шина.

Файл завершается однобайтовым блоком данных с константой RDS_SFTAG_EOF (0x00).

В настоящее время вместо функции `rdscrtlSaveSystemTagged` чаще используется поблочное сохранение схемы (см. §3.5).

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlLoadSystemTagged` (стр. 650),
`rdscrtlGetBlockByBlockSavePiece` (стр. 645).

Б.3.5.13. `rdscrtlSaveSystemTaggedEx` – записать схему в файл или разделяемую память в специальном двоичном формате

Функция `rdscrtlSaveSystemTaggedEx` записывает схему, загруженную в данный момент в управляемую копию РДС, в файл или разделяемую область памяти с указанным именем в специальном двоичном формате.

```
BOOL RDSCALL rdscrtlSaveSystemTaggedEx(
    int Link,           // Идентификатор связи
    LPSTR FileName,    // Имя файла/области
    DWORD Flags,       // Флаги (RDSCRTL_TAGGED_*)
    DWORD *pSize      // Возвращаемый размер области памяти
);
```

Тип указателя на эту функцию:

`RDSCRTL_BISDwpDw`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

FileName

Указатель на строку с полным путем к записываемому файлу (при сброшенном флаге RDSCTRL_TAGGED_SHAREDMMEM) или с именем разделяемой области памяти (при взведенном флаге).

Flags

Флаги функции – одна из констант RDSCTRL_TAGGED_*:

RDSCTRL_TAGGED_SHAREDMMEM Если этот флаг сброшен, схема будет записана в файл FileName, если взведен – в область памяти с этим именем.

RDSCTRL_TAGGED_DBLBUF Использовать промежуточный буфер (см. примечания, только при взведенном флаге RDSCTRL_TAGGED_SHAREDMMEM).

pSize

Указатель на целую (DWORD) переменную, в которую функция запишет размер созданной разделяемой области памяти или файла.

Возвращаемое значение:

TRUE – запись выполнена успешно, FALSE – при записи возникли ошибки.

Примечания:

Эта функция записывает схему из копии РДС, управляемой через связь Link, в файл с именем FileName или в разделяемую область памяти с именем FileName, в зависимости от переданных флагов. Формат файла или области памяти совпадает с форматом функции rdscrtlSaveSystemTagged (стр. 654). Если флаг RDSCTRL_TAGGED_SHAREDMMEM в параметре Flags не установлен, действие функции полностью совпадает с действием rdscrtlSaveSystemTagged, за исключением того, что эта функция возвращает размер созданного файла.

Если же флаг RDSCTRL_TAGGED_SHAREDMMEM установлен, РДС создаст разделяемую область памяти с именем FileName, запишет в нее схему, и оставит область открытой, чтобы она не была автоматически уничтожена Windows. Управляющая программа должна считать данные из этой разделяемой области, а затем дать РДС команду закрыть ее при помощи функции rdscrtlDeleteExchangeMemory (стр. 643). В целом, использование функции rdscrtlSaveSystemTaggedEx при работе через разделяемую память выглядит следующим образом:

```
DWORD Size;          // Размер области
char Name[]="MyGlobalMemory"; // Имя разделяемой области
// Сохраняем схему в область памяти
if(!rdscrtlSaveSystemTaggedEx(
    Link,Name,RDSCTRL_TAGGED_SHAREDMMEM,&Size))
{ ...Ошибка... }
// Открываем созданную РДС область (она не закрыта на
// стороне РДС, поэтому не уничтожена)
HANDLE FileMapping;
FileMapping=OpenFileMapping(
    FILE_MAP_WRITE, // Доступ для чтения и записи
    FALSE,          // Не наследовать HANDLE
    Name);          // Имя области памяти
// Подключаем область к адресному пространству процесса
// и получаем указатель на нее
void *FileMappingPtr=NULL;
```

```

if(FileMapping)
    FileMappingPtr=MapViewOfFile(
        FileMapping,    // Область
        FILE_MAP_WRITE, // Доступ для чтения и записи
        0,0,             // Смещение от начала (нет)
        Size);          // Размер запрашиваемой части
if(FileMappingPtr)
    {
        ... обработка данных схемы по адресу FileMappingPtr ...
        // Отключаем область от памяти процесса
        UnmapViewOfFile(FileMappingPtr);
    }
// Закрываем область на своей стороне
if(FileMapping) CloseHandle(FileMapping);
// Даем команду РДС тоже закрыть эту область
rdscrtlDeleteExchangeMemory(Link);
// Теперь не осталось ни одного дескриптора, ссылающегося
// на область, и она будет уничтожена Windows

```

Флаг `RDSCTRL_TAGGED_DBLBUF` управляет способом вычисления размера создаваемой разделяемой области памяти на стороне РДС. Если он сброшен, РДС сначала вычислит общий объем, необходимый для сохранения схемы, а потом отведет область памяти этого размера и сохранит схему в нее. Если же этот флаг взведен, РДС сохранит схему в промежуточную, динамически отводимую область памяти, увеличивая ее размер в процессе сохранения, а затем создаст разделяемую память такого же размера и перепишет сохраненные данные в нее. Первый способ не требует дополнительной памяти, второй способ может оказаться несколько быстрее. В большинстве случаев первый способ, со сброшенным флагом `RDSCTRL_TAGGED_DBLBUF`, предпочтительнее.

В настоящее время вместо функции `rdscrtlSaveSystemTaggedEx` чаще используется поблочное сохранение схемы (см. §3.5).

См. также:

```

rdscrtlCreateLink (стр. 600), rdscrtlSaveSystemTagged (стр. 654),
rdscrtlGetBlockByBlockSavePiece (стр. 645),
rdscrtlLoadSystemTaggedEx (стр. 651).

```

Б.3.5.14. `rdscrtlSaveSystemTaggedMem` – записать схему в разделяемую память в специальном двоичном формате

Функция `rdscrtlSaveSystemTaggedMem` записывает схему, загруженную в данный момент в управляемую копию РДС, в разделяемую область памяти, подбирая этой области уникальное имя.

```

BOOL RDSCALL rdscrtlSaveSystemTaggedMem(
    int Link,           // Идентификатор связи
    DWORD Flags,       // Флаги (RDSCTRL_TAGGED_*)
    LPVOID MemName,    // Возвращаемое имя области
    DWORD *pSize       // Возвращаемый размер области памяти
);

```

Тип указателя на эту функцию:

```

RDSCTRL_BIDwpVpDw

```

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Flags

Флаги функции – на данный момент поддерживается единственный флаг:

`RDSCTRL_TAGGED_DBLBUF` Использовать промежуточный буфер (см. примечания).

MemName

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи `rdscrtlSetStringCallback` (стр. 604), запишет имя созданной разделяемой области памяти.

pSize

Указатель на целую (DWORD) переменную, в которую функция запишет размер созданной разделяемой области памяти.

Возвращаемое значение:

TRUE – запись выполнена успешно, FALSE – при записи возникли ошибки.

Примечания:

Эта функция записывает схему из копии РДС, управляемой через связь Link, в разделяемую область памяти с именем, которое будет сформировано автоматически и возвращено через параметр MemName при помощи функции, ранее зарегистрированной вызовом `rdscrtlSetStringCallback`. Формат области памяти совпадает с форматом, записываемым функцией `rdscrtlSaveSystemTagged` (стр. 654).

При вызове этой функции РДС создаст разделяемую область памяти с автоматически выбранным уникальным именем, запишет в нее схему, и оставит область открытой, чтобы она не была уничтожена Windows. Управляющая программа должна считать данные из этой разделяемой области, а затем дать РДС команду закрыть ее при помощи функции `rdscrtlDeleteExchangeMemory` (стр. 643). Если, допустим, управляющее приложение хранит строки произвольной длины в объектах некоторого класса String, для которого определена функция-член `c_str(void)`, возвращающая указатель на хранимую внутри объекта строку, и для работы с таким классом зарегистрирована функция обратного вызова (стр. 604), то использование функции `rdscrtlSaveSystemTaggedMem` может выглядеть следующим образом:

```
DWORD Size;          // Размер области
String Name;         // Имя разделяемой области
// Сохраняем схему в область памяти (имя запишется в Name
// функцией обратного вызова
if(!rdscrtlSaveSystemTaggedMem(
    Link, 0, &Name, &Size))
{ ...Ошибка... }
// Открываем созданную РДС область (она не закрыта на
// стороне РДС, поэтому не уничтожена
HANDLE FileMapping;
FileMapping=OpenFileMapping(
    FILE_MAP_WRITE, // Доступ для чтения и записи
    FALSE,          // Не наследовать HANDLE
    Name.c_str());  // Имя области памяти
```

```

// Подключаем область к адресному пространству процесса
// и получаем указатель на нее
void *FileMappingPtr=NULL;
if(FileMapping)
    FileMappingPtr=MapViewOfFile(
        FileMapping,    // Область
        FILE_MAP_WRITE, // Доступ для чтения и записи
        0,0,             // Смещение от начала (нет)
        Size);          // Размер запрашиваемой части
if(FileMappingPtr)
{
    ... обработка данных схемы по адресу FileMappingPtr ...
    // Отключаем область от памяти процесса
    UnmapViewOfFile(FileMappingPtr);
}
// Закрываем область на своей стороне
if(FileMapping) CloseHandle(FileMapping);
// Даем команду РДС тоже закрыть эту область
rdscrtlDeleteExchangeMemory(Link);
// Теперь не осталось ни одного дескриптора, ссылающегося
// на область, и она будет уничтожена Windows

```

Флаг `RDSCTRL_TAGGED_DBLBUF`, как и в `rdscrtlSaveSystemTaggedEx` (стр. 656), управляет способом вычисления размера создаваемой разделяемой области памяти на стороне РДС. Если он сброшен, РДС сначала вычислит общий объем, необходимый для сохранения схемы, а потом отведет область памяти этого размера и сохранит схему в нее. Если же этот флаг взведен, РДС сохранит схему в промежуточную, динамически отводимую область памяти, увеличивая ее размер в процессе сохранения, а затем создаст разделяемую память такого же размера и переписет сохраненные данные в нее. Первый способ не требует дополнительной памяти, второй способ может оказаться несколько быстрее. В большинстве случаев первый способ, со сброшенным флагом `RDSCTRL_TAGGED_DBLBUF`, предпочтительнее.

В настоящее время вместо функции `rdscrtlSaveSystemTaggedMem` чаще используется поблочное сохранение схемы (см. §3.5).

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSaveSystemTaggedEx` (стр. 656),
`rdscrtlSetStringCallback` (стр. 604),
`rdscrtlGetBlockByBlockSavePiece` (стр. 645),
`rdscrtlLoadSystemTaggedEx` (стр. 651).

Б.3.5.15. `rdscrtlSaveSystemToFile` – сохранить схему в файл

Функция `rdscrtlSaveSystemToFile` записывает схему, загруженную в данный момент в управляемую копию РДС, в указанный файл в обычном для РДС формате.

```

BOOL RDSCALL rdscrtlSaveSystemToFile(
    int Link,           // Идентификатор связи
    LPSTR FileName     // Имя файла
);

```

Тип указателя на эту функцию:

`RDSCTRL_BIS`

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

FileName

Указатель на строку с полным именем файла, в который нужно сохранить схему. Если в этом параметре передать `NULL` или указатель на пустую строку, схема будет сохранена в тот файл, из которого она была загружена или в который в последний раз сохранялась.

Возвращаемое значение:

`TRUE` – запись выполнена успешно, `FALSE` – при записи возникли ошибки.

Примечания:

Эта функция записывает схему из копии РДС, управляемой через связь `Link`, в файл с именем `FileName` или в последний использовавшийся для загрузки или записи файл, если в `FileName` передана пустая строка или `NULL`.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlLoadSystemFromFile` (стр. 648).

Б.3.5.16. `rdscrtlSetBlockByBlockLoadPiece` – передать очередной объект при поблочной загрузке

Функция `rdscrtlSetBlockByBlockLoadPiece` передает управляемой копии РДС данные очередного объекта в процессе поблочной загрузки схемы.

```
void RDSCALL rdscrtlSetBlockByBlockLoadPiece(  
    int Link,           // Идентификатор связи  
    int Tag,            // Тип объекта  
    LPSTR Text          // Текст описания объекта  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIIS`

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Tag

Тип передаваемого объекта – одна из констант `RDS_SFTAG_*`:

<code>RDS_SFTAG_ROOT</code>	корневая подсистема;
<code>RDS_SFTAG_SIMPLEBLOCK</code>	простой блок;
<code>RDS_SFTAG_SYSTEM</code>	подсистема (кроме корневой);
<code>RDS_SFTAG_INPUTBLOCK</code>	внешний вход;
<code>RDS_SFTAG_OUTPUTBLOCK</code>	внешний выход;
<code>RDS_SFTAG_BUSPORT</code>	ввод шины;
<code>RDS_SFTAG_CONNECTION</code>	связь;
<code>RDS_SFTAG_BUS</code>	шина;
<code>RDS_SFTAG_CONNSTYLES</code>	стили связей (дополнительный блок данных);
<code>RDS_SFTAG_TYPES</code>	описания структур (дополнительный блок данных);

RDS_SFTAG_EOF

объекты кончились, больше нет данных.

Text

Указатель на строку с текстовым описанием передаваемого объекта.

Примечания:

Эта функция передает в копию РДС, управляемую через связь Link, описание очередного объекта схемы. После вызова функции `rdscrtlStartBlockByBlockLoad` (стр. 663) управляющее приложение может отправлять в память РДС какую-либо схему объект за объектом при помощи `rdscrtlSetBlockByBlockLoadPiece`. Каждый вызов передает в РДС очередной объект: в параметре Tag передается его тип, в параметре Text – текстовое описание, такое же, какое возвращается функцией `rdscrtlGetBlockByBlockSavePiece` (стр. 645).

При помощи этой функции в РДС в произвольном порядке передаются блоки, связи и шины, а также набор стилей связей и набор описаний структур, хранящиеся в схеме. Два последних объекта передавать не обязательно – вся необходимая для работы схемы информация о внешнем виде связей находится в описаниях самих связей, а описания структур – в описаниях блоков, которые их используют. Описания стилей связей и типов структур могут пригодиться, если пользователю разрешено редактировать схему – в этом случае он сможет выбирать стили связей из общего набора, а также использовать в блоках структуры, которые присутствуют в общем наборе структур схемы, но не используются ни в одном из ее блоков на данный момент.

Чаще всего поблочная загрузка схемы используется в тех случаях, когда эта схема формируется из каких-либо хранящихся отдельно (например, в базе данных) объектов.

Пример использования функции `rdscrtlSetBlockByBlockLoadPiece` приведен в §3.5.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlStartBlockByBlockLoad` (стр. 663),
`rdscrtlEndBlockByBlockLoad` (стр. 643),
`rdscrtlGetBlockByBlockSavePiece` (стр. 645).

Б.3.5.17. `rdscrtlSetProgressDelay` – установка интервала между сообщениями о ходе загрузки/сохранения

Функция `rdscrtlSetProgressDelay` устанавливает интервал в миллисекундах, через который при загрузке или сохранении схемы в управляемой копии РДС наступает событие `RDSCTRLEVENT_PROGRESS` (стр. 597).

```
void RDSCALL rdscrtlSetProgressDelay(  
    int Link,           // Идентификатор связи  
    DWORD Delay         // Интервал, мс  
);
```

Тип указателя на эту функцию:

`RDSCTRL_VIDw`

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

Delay

Интервал между событиями в миллисекундах. Нулевое значение отключает событие RDSCTRLEVENT_PROGRESS.

Примечания:

Эта функция устанавливает интервал, с которым управляемая через связь Link копия РДС будет посылать управляющему приложению информацию о ходе сохранения или загрузки схемы (событие RDSCTRLEVENT_PROGRESS). Обычно эта информация используется для вывода пользователю какого-либо индикатора, показывающего, какой объем информации уже загружен или сохранен, и сколько еще осталось.

Интервал, установленный функцией rdscrtlSetProgressDelay, запоминается в параметрах связи Link, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно, например, вызвать эту функцию сразу после создания связи, когда еще нет работающей копии РДС. При этом после вызова rdscrtlConnect (стр. 600) в РДС будет передано установленное значение интервала.

Если реакция на событие RDSCTRLEVENT_PROGRESS не зарегистрирована в управляющем приложении, установленное значение интервала между событиями все равно запоминается, но событие при этом происходить не будет.

См. также:

rdscrtlCreateLink (стр. 600), RDSCTRLEVENT_PROGRESS (стр. 597),
rdscrtlConnect (стр. 600).

Б.3.5.18. rdscrtlStartBlockByBlockLoad – начать поблочную загрузку схемы

Функция rdscrtlStartBlockByBlockLoad начинает поблочную загрузку схемы в управляемую копию РДС, при которой каждый блок, связь или шина схемы передаются в РДС отдельным вызовом.

```
void RDSCALL rdscrtlStartBlockByBlockLoad(  
    int Link,           // Идентификатор связи  
    DWORD Flags        // Флаги (не используются)  
);
```

Тип указателя на эту функцию:

RDSCTRL_VIDw

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

Flags

Флаги функции (в настоящее время этот параметр никак не используется).

Примечания:

Эта функция начинает поблочную передачу схемы в копию РДС, управляемую через связь Link. После ее вызова можно в произвольном порядке передавать в РДС описания различных объектов схемы функцией rdscrtlSetBlockByBlockLoadPiece (стр. 661), для завершения загрузки и сборки схемы из набора переданных объектов следует вызвать rdscrtlEndBlockByBlockLoad (стр. 643).

Пример использования функции rdscrtlStartBlockByBlockLoad приведен в §3.5.

См. также:

```
rdscrtlCreateLink (стр. 600),  
rdscrtlSetBlockByBlockLoadPiece (стр. 661),  
rdscrtlEndBlockByBlockLoad (стр. 643),  
rdscrtlStartBlockByBlockSave (стр. 664).
```

Б.3.5.19. rdscrtlStartBlockByBlockSave – начать поблочное сохранение схемы

Функция `rdscrtlStartBlockByBlockSave` начинает поблочную передачу схемы из управляемой копии РДС в управляющую программу, при которой каждый блок, связь или шина схемы считываются из РДС отдельным вызовом.

```
int RDSCALL rdscrtlStartBlockByBlockSave(  
    int Link,          // Идентификатор связи  
    DWORD Flags       // Флаги (не используются)  
);
```

Тип указателя на эту функцию:

```
RDSCRTL_IIDw
```

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Flags`

Флаги функции (в настоящее время этот параметр никак не используется).

Возвращаемое значение:

Общее число объектов, доступных для считывания.

Примечания:

Эта функция начинает поблочную передачу схемы из копии РДС, управляемой через связь `Link`. После ее вызова можно последовательно получать из РДС описания различных объектов схемы функцией `rdscrtlGetBlockByBlockSavePiece` (стр. 645), для завершения передачи схемы следует вызвать `rdscrtlEndBlockByBlockSave` (стр. 644). Функция возвращает число объектов (блоков, связей, шин, дополнительных описаний), которые можно считать из РДС. Для получения всех объектов схемы можно либо вызвать `rdscrtlGetBlockByBlockSavePiece` указанное число раз, либо вызывать ее до тех пор, пока она не вернет константу `RDS_SFTAG_EOF` (признак конца объектов).

Функция `rdscrtlStartBlockByBlockSave` создает в памяти большой объем дополнительных структур, используемых при чтении объектов, поэтому крайне важно вызвать `rdscrtlEndBlockByBlockSave` как можно быстрее после завершения считывания схемы, чтобы освободить эту память.

Пример использования функции `rdscrtlStartBlockByBlockSave` приведен в §3.5.

См. также:

```
rdscrtlCreateLink (стр. 600),  
rdscrtlGetBlockByBlockSavePiece (стр. 645),  
rdscrtlEndBlockByBlockSave (стр. 644),  
rdscrtlStartBlockByBlockLoad (стр. 663).
```


Б.3.6. Функции реакции на события

Описываются функции, позволяющие управляющему приложению программировать свою реакцию на события, происходящие в РДС (см. Б.2).

Б.3.6.1. `rdscrtlEnableEvents` – разрешение реакции на события

Функция `rdscrtlEnableEvents` разрешает или запрещает реакцию управляющего приложения на события РДС.

```
void RDSCALL rdscrtlEnableEvents(  
    int Link,           // Идентификатор связи  
    BOOL Enable        // Разрешение  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIB`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Enable`

`TRUE` – реакция на события разрешена, `FALSE` – запрещена (по умолчанию).

Примечания:

Эта функция разрешает (при `Enable==TRUE`) или запрещает (при `Enable==FALSE`) управляющему приложению реагировать на различные события, происходящие в РДС (см. Б.2, стр. 592). По умолчанию реакция на события запрещена. Разрешение реакции на события запоминается в параметрах связи `Link`, его не нужно устанавливать заново при повторном запуске РДС через эту связь. Можно вызвать `rdscrtlEnableCalcMode` сразу после создания связи, до фактического запуска РДС функцией `rdscrtlConnect` (стр. 600).

Разумеется, разрешение реакции на события влияет только на те события, реакции для которых зарегистрированы управляющей программой. В `RdsCtrl.dll` не предусмотрено никакой “реакции по умолчанию” – если реакция на событие не зарегистрирована, управляющее приложение никак не сможет узнать о наступлении этого события.

Пример использования функции `rdscrtlEnableEvents` приведен в §3.4.

См. также:

События в управляющем приложении (стр. 592), `rdscrtlCreateLink` (стр. 600), `rdscrtlConnect` (стр. 600).

Б.3.6.2. `rdscrtlRegisterBlockMsgCallback` – регистрация функции для реакции на сообщение от блока

Функция `rdscrtlRegisterBlockMsgCallback` регистрирует в `RdsCtrl.dll` функцию управляющего приложения, которая будет вызываться при получении сообщения `RDSCTRLEVENT_BLOCKMSG` (стр. 593) от одного из блоков схемы.

```
void RDSCALL rdscrtlRegisterBlockMsgCallback(  
    int Link,           // Идентификатор связи  
    RDSCRTL_BMCALLBACK CallBack, // Функция  
    LPVOID pAuxData    // Доп. данные  
);
```

Тип указателя на эту функцию:

RDSCTRL_VICb1pV

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdsctrlCreateLink` (стр. 600).

CallBack

Указатель на функцию, которая будет вызываться при каждом получении сообщения от блока.

pAuxData

Указатель на дополнительные данные, который будут передаваться в функцию `CallBack` при каждом вызове.

Примечания:

Эта функция запоминает для связи `Link` указатель на функцию, которая будет вызываться каждый раз, когда модель одного из блоков схемы, загруженной в управляемую через связь `Link` копию РДС, вызовет функцию `rdsRemoteControllerCall` (стр. 397) для передачи сообщения управляющей программе. Функция, указатель на которую передается в параметре `CallBack`, должна иметь следующий вид:

```
void RDSCALL имя_функции(  
    int Link,           // Идентификатор связи с РДС  
    LPSTR BName,       // Имя блока  
    int Imsg,          // Переданное блоком число  
    LPSTR Smsg,        // Переданная блоком строка  
    LPVOID pAux        // Дополнительные данные  
);
```

В параметре `Link` этой функции передается идентификатор связи с управляемой копией РДС, в которой произошло событие (можно зарегистрировать одну и ту же функцию для нескольких связей), в параметре `BName` – строка с полным именем блока, передавшего сообщение, в `Imsg` – целое число, переданное блоком, в `Smsg` – указатель на строку, переданную блоком, в `pAux` – указатель, переданный в параметре `pAuxData` функции `rdsctrlRegisterBlockMsgCallback` при регистрации реакции (это единственный способ передать в функцию реакции какие-либо дополнительные параметры).

Регистрация функции реакции на сообщение от блока отменяет прежнюю регистрацию реакции на сообщение `RDSCTRLEVENT_BLOCKMSG` для связи `Link`, независимо от того, как именно эта реакция была зарегистрирована.

Пример использования функции `rdsctrlRegisterBlockMsgCallback` приведен в §3.4.

См. также:

Способы реакции на события (стр. 592), `RDSCTRLEVENT_BLOCKMSG` (стр. 593), `rdsRemoteControllerCall` (стр. 397), `rdsctrlUnregisterEvent` (стр. 669), `rdsctrlRegisterEventStdCallback` (стр. 668), `rdsctrlRegisterEventMessage` (стр. 667), `rdsctrlCreateLink` (стр. 600).

Б.3.6.3. `rdscrtlRegisterEventMessage` – регистрация оконного сообщения для реакции на событие

Функция `rdscrtlRegisterEventMessage` регистрирует в `RdsCtrl.dll` дескриптор окна и параметры сообщения, которое нужно направить этому окну при наступлении в РДС указанного события.

```
void RDSCALL rdscrtlRegisterEventMessage(  
    int Link,           // Идентификатор связи  
    int Event,          // Идентификатор события (RDSTRLEVENT_*)  
    HWND Window,        // Дескриптор окна  
    UINT Message,       // Идентификатор сообщения  
    WPARAM WParam       // Первый параметр сообщения  
);
```

Тип указателя на эту функцию:

`RDSTRCTRL_VIIHwUWp`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`Event`

Целый идентификатор события, реакция на которое регистрируется (см. Б.2, стр. 592), то есть одна из констант `RDSTRLEVENT_*`.

`Window`

Дескриптор окна управляющего приложения, которому нужно направлять сообщение при наступлении события `Event`.

`Message`

Целый идентификатор оконного сообщения, которое будет направлено окну `Window` при наступлении события `Event`.

`WParam`

Первый параметр оконного сообщения (во втором будет передан указатель на структуру `RDSTRCTRL_MSGEVENTDATA`, стр. 586).

Примечания:

Эта функция запоминает для связи `Link` дескриптор (`HWND`) окна управляющей программы `Window`, и параметры сообщения (идентификатор `Message` и первый параметр `WParam`), которое будет направлено этому окну при наступлении в управляемой через связь `Link` копии РДС события `Event`. Параметры события будут находиться в структуре `RDSTRCTRL_MSGEVENTDATA`, указатель на которую передается в втором (`LPARAM`) параметре сообщения (приведение типа `LPARAM` к типу `RDSTRCTRL_MSGEVENTDATA*` необходимо будет произвести в процедуре окна, которая вызовется для реакции на это сообщение). Идентификатор оконного сообщения `Message` обычно выбирается так, чтобы он не совпадал с одним из системных сообщений `Windows` (обычно для этого используют значения между константой `Windows API WM_USER` и числом `0x7FFF`), параметр `WParam` выбирают произвольно – он передается в сообщении без какой-либо обработки внутри `RdsCtrl.dll`. Для отправки сообщения окну `Window` используется функция `Windows API SendMessage`, ожидающая завершения процедуры окна.

Таким образом, при наступлении в управляемой копии РДС события Event, библиотекой RdsCtrl.dll будет вызвана процедура окна Window (допустим, она называется WindowProc) со следующими параметрами:

```
LRESULT CALLBACK WindowProc(  
    HWND hwnd,           // Window  
    UINT uMsg,           // Message  
    WPARAM wParam,      // WParam  
    LPARAM lParam        // указатель на RDSCtrl_MSGEVENTDATA  
);
```

Возвращаемое процедурой окна значение игнорируется.

Регистрация реакции на событие через оконное сообщение отменяет прежнюю регистрацию реакции на это же событие для связи Link, независимо от того, как именно эта реакция была зарегистрирована.

См. также:

Способы реакции на события (стр. 592), RDSCtrl_MSGEVENTDATA (стр. 586),
rdscctrlRegisterEventStdCallback (стр. 668),
rdscctrlRegisterBlockMsgCallback (стр. 665),
rdscctrlCreateLink (стр. 600), rdscctrlUnregisterEvent (стр. 669).

Б.3.6.4. rdscctrlRegisterEventStdCallback – регистрация функции для реакции на событие

Функция rdscctrlRegisterEventStdCallback регистрирует в RdsCtrl.dll функцию управляющего приложения, которая будет вызываться при наступлении в РДС указанного события.

```
void RDSCALL rdscctrlRegisterEventStdCallback(  
    int Link,           // Идентификатор связи  
    int Event,         // Идентификатор события (RDSCtrlEVENT_*)  
    RDSCtrl_CALLBACK CallBack, // Функция  
    LPVOID pAuxData   // Доп. данные  
);
```

Тип указателя на эту функцию:

RDSCtrl_VIICbpV

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscctrlCreateLink (стр. 600).

Event

Целый идентификатор события, реакция на которое регистрируется (см. Б.2, стр. 592), то есть одна из констант RDSCtrlEVENT_*.

CallBack

Указатель на функцию, которая будет вызываться при каждом событии Event.

pAuxData

Указатель на дополнительные данные, который будут передаваться в функцию CallBack при каждом вызове.

Примечания:

Эта функция запоминает для связи Link указатель на функцию, которая будет вызываться каждый раз при наступлении события Event в копии РДС, управляемой через связь Link. Функция, указатель на которую передается в параметре Callback, должна иметь следующий вид:

```
void RDSCALL имя_функции(  
    int Link,          // Идентификатор связи с РДС  
    int Event,         // Идентификатор события  
    LPVOID pData,      // Данные события  
    LPVOID pAux        // Дополнительные данные  
);
```

В параметре Link этой функции передается идентификатор связи с управляемой копией РДС, в которой произошло событие (можно зарегистрировать одну и ту же функцию для нескольких связей), в параметре Event – идентификатор наступившего события (можно зарегистрировать одну и ту же функцию для разных событий), в параметре pData – указатель на структуру, описывающую событие (эта структура у каждого события своя), и в параметре pAux – указатель, переданный в параметре pAuxData функции rdscrtlRegisterEventStdCallback при регистрации реакции (это единственный способ передать в функцию реакции какие-либо дополнительные параметры).

Регистрация функции реакции на событие отменяет прежнюю регистрацию реакции на это же событие для связи Link, независимо от того, как именно эта реакция была зарегистрирована.

Пример использования функции rdscrtlRegisterEventStdCallback приведен в §3.4.

См. также:

Способы реакции на события (стр. 592),
rdscrtlRegisterEventMessage (стр. 667),
rdscrtlRegisterBlockMsgCallback (стр. 665),
rdscrtlCreateLink (стр. 600), rdscrtlUnregisterEvent (стр. 669).

Б.3.6.5. rdscrtlUnregisterEvent – отмена регистрации реакции на событие

Функция rdscrtlUnregisterEvent прекращает реакцию управляющего приложения на указанное событие РДС.

```
void RDSCALL rdscrtlUnregisterEvent(  
    int Link,  // Идентификатор связи  
    int Event  // Событие  
);
```

Тип указателя на эту функцию:

RDSCRTL_VII

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

Event

Целый идентификатор события, реакция на которое отменяется (см. Б.2, стр. 592), то есть одна из констант `RDSCTRL_EVENT_*`.

Примечания:

После вызова этой функции управляющее приложение перестает получать из копии РДС, управляемой через связь `Link`, информацию о наступлении события `Event`, если ранее реакция на это событие была зарегистрирована. Регистрация реакции на событие отменяется независимо от того, как именно эта реакция осуществлялась: через функцию обратного вызова или через оконное сообщение.

См. также:

Способы реакции на события (стр. 592),
`rdscrtlRegisterEventMessage` (стр. 667),
`rdscrtlRegisterBlockMsgCallback` (стр. 665),
`rdscrtlRegisterEventStdCallback` (стр. 668),
`rdscrtlCreateLink` (стр. 600).

Б.3.7. Функции для работы с портами вывода

Описываются функции, управляющие работой портов вывода (см. §3.6), то есть рисованием подсистем схемы внутри окна управляющего приложения.

Б.3.7.1. `rdscrtlGetViewportParams` – получить масштаб и сдвиг подсистемы в порте вывода

Функция `rdscrtlGetViewportParams` возвращает масштаб и положение изображения подсистемы в указанном порте вывода.

```
void RDSCALL rdscrtlGetViewportParams(  
    int Link,           // Идентификатор связи  
    int VpId,          // Идентификатор порта вывода  
    double *pZoom,     // Возвращаемый масштаб  
    int *pScrollX,     // Возвращаемый гориз.сдвиг  
    int *pScrollY      // Возвращаемый верт. сдвиг  
);
```

Тип указателя на эту функцию:

`RDSCTRL_VIIPDpIpI`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`VpId`

Идентификатор порта вывода, присвоенный ему при создании функцией `rdscrtlSetViewport` (стр. 674).

`pZoom`

Указатель на вещественную переменную, в которую функция запишет текущий масштаб порта вывода в долях единицы (1.0 – 100%, 0.5 – 50% и т.д.). Если управляющей программе не нужен масштаб, в этом параметре можно передать `NULL`.

pScrollX, pScrollY

Указатели на целые переменные, в которые функция запишет горизонтальную (pScrollX) и вертикальную (pScrollY) координаты левого верхнего угла области рабочего поля подсистемы, видимой в порте вывода. Координаты возвращаются в точках экрана в текущем масштабе подсистемы. Если управляющей программе не нужны эти координаты, в этих параметрах можно передать NULL.

Примечания:

Эта функция возвращает параметры отображения подсистемы, установленные для порта вывода VpId в копии РДС, управляемой через связь Link. Координаты левого верхнего угла видимой в порте области возвращаются в точках экрана в текущем установленном для порта масштабе, поэтому их можно использовать для отображения полос прокрутки.

Пример использования функции rdsctrlGetViewportParams приведен в §3.6.2.

См. также:

rdsctrlCreateLink (стр. 600), rdsctrlSetViewport (стр. 674),
rdsctrlSetViewportParams (стр. 675).

Б.3.7.2. rdsctrlGetViewportSysArea – размеры рабочего поля подсистемы в порте вывода

Функция rdsctrlGetViewportSysArea возвращает размеры рабочего поля подсистемы, отображаемой в данный момент в указанном порте вывода, в указанном масштабе.

```
BOOL RDSCALL rdsctrlGetViewportSysArea(  
    int Link,           // Идентификатор связи  
    int VpId,           // Идентификатор порта вывода  
    double Zoom,        // Интересующий масштаб  
    int *pWidth,        // Возвращаемая ширина поля  
    int *pHeight       // Возвращаемый высота поля  
);
```

Тип указателя на эту функцию:

RDCTRL_BIIDpIpI

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdsctrlCreateLink (стр. 600).

VpId

Идентификатор порта вывода, присвоенный ему при создании функцией rdsctrlSetViewport (стр. 674).

Zoom

Масштаб в долях единицы (1.0 – 100%, 2.0 – 200%), в котором нужно вернуть размеры рабочего поля. Если размеры нужно вернуть в текущем установленном для порта VpId масштабе, в параметре Zoom передается отрицательное значение (например, -1.0).

pWidth, pHeight

Указатели на целые переменные, в которые функция запишет ширину (pWidth) и высоту (pHeight) рабочей области подсистемы в указанном параметром Zoom масштабе.

Возвращаемое значение:

TRUE – значения получены, FALSE – ошибка (нет порта вывода с идентификатором VpId, нет связи с РДС и т.п.)

Примечания:

Эта функция возвращает размеры рабочего поля подсистемы, привязанной в данный момент к порту вывода VpId в копии РДС, управляемой через связь Link. Размеры возвращаются в точках экрана в масштабе Zoom, или в текущем масштабе, если в Zoom передано отрицательное значение. Размеры рабочего поля в текущем масштабе можно использовать для задания диапазонов горизонтальной и вертикальной полос прокрутки.

Пример использования функции rdsctrlGetViewportSysArea приведен в §3.6.2.

См. также:

rdsctrlCreateLink (стр. 600), rdsctrlSetViewport (стр. 674).

Б.3.7.3. rdsctrlGetVPMouseLevel – тип реакции на мышшь у подсистемы в порте вывода

Функция rdsctrlGetVPMouseLevel возвращает тип реакции подсистемы, привязанной в данный момент к указанному порту вывода, на действия пользователя мышью.

```
int RDSCALL rdsctrlGetVPMouseLevel(  
    int Link,           // Идентификатор связи  
    int VpId           // Идентификатор порта вывода  
);
```

Тип указателя на эту функцию:

RDCTRL_III

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdsctrlCreateLink (стр. 600).

VpId

Идентификатор порта вывода, присвоенный ему при создании функцией rdsctrlSetViewport (стр. 674).

Возвращаемое значение:

Тип реакции подсистемы на мышшь:

- 0 ни окно подсистемы, ни ее блоки не реагируют ни на какие действия мышью;
- 1 окно подсистемы или хотя бы один из ее блоков реагирует на нажатия кнопок и перемещение курсора с нажатыми кнопками, но никто в подсистеме не реагирует на перемещение курсора без нажатия кнопок;
- 2 окно подсистемы или один из ее блоков реагирует на перемещения курсора мыши без нажатия кнопок.

Примечания:

Эта функция возвращает особенности реакции подсистемы, привязанной в данный момент к порту вывода `VpId` в копии РДС, управляемой через связь `Link`, на действия пользователя мышью. Обычно она используется для того, чтобы определить, какие именно события, связанные с мышью, управляющее приложение должно передавать в РДС (постоянная передача всех действий пользователя в РДС приводит к ненужной нагрузке на связь между управляющим приложением и РДС). Если функция вернула нулевое значение, никакую информацию о действиях мышью передавать в РДС не нужно: в подсистеме некому ее обрабатывать. Если функция вернула значение 1, в РДС следует передавать только нажатия и отпускания кнопок мыши и перемещения курсора при нажатых кнопках. Если же функция вернет значение 2, в РДС нужно передавать все действия пользователя.

Для передачи в РДС информации о движениях курсора мыши и нажатии и отпуске ее кнопок используется функция `rdscrtlViewportMouse` (стр. 682). Пример использования функции `rdscrtlGetVPMouseLevel` приведен в §3.6.2.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetViewport` (стр. 674),
`rdscrtlViewportMouse` (стр. 682).

Б.3.7.4. `rdscrtlReleaseViewport` – уничтожить порт вывода

Функция `rdscrtlReleaseViewport` уничтожает ранее созданный порт вывода с указанным идентификатором.

```
void RDSCALL rdscrtlReleaseViewport(  
    int Link,           // Идентификатор связи  
    int VpId           // Идентификатор порта вывода  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VII`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`VpId`

Идентификатор уничтожаемого порта вывода, присвоенный ему при создании функцией `rdscrtlSetViewport` (стр. 674).

Примечания:

Эта функция уничтожает порт вывода `VpId` в копии РДС, управляемой через связь `Link`. Обычно она вызывается перед уничтожением окна управляющего приложения, внутри которого открыт порт вывода. Для того, чтобы сменить подсистему, привязанную к порту вывода, следует вызвать функцию `rdscrtlSetViewport` для уже существующего порта, уничтожать порт для этого не нужно.

Пример использования функции `rdscrtlReleaseViewport` приведен в §3.6.1.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetViewport` (стр. 674).

Б.3.7.5. `rdscrtlSetViewport` – создать порт вывода

Функция `rdscrtlSetViewport` создает порт вывода в указанном окне управляющего приложения.

```
int RDSCALL rdscrtlSetViewport(  
    int Link,           // Идентификатор связи  
    int VpId,           // Идентификатор порта вывода  
    HWND Window,        // Окно, в котором создается порт  
    LPSTR FullSysName,  // Имя привязанной подсистемы  
    DWORD Flags         // Флаги (не используются)  
);
```

Тип указателя на эту функцию:

`RDSCRTL_IIIHwSDw`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`VpId`

–1, если создается новый порт вывода, или идентификатор существующего порта вывода (результат прошлого вызова `rdscrtlSetViewport`), если нужно изменить его параметры.

`Window`

Дескриптор окна управляющего приложения, внутри которого создается порт вывода.

`FullSysName`

Полное имя подсистемы загруженной схемы, которая привязывается к порту вывода. Полное имя подсистемы, как и любого блока, начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этой, которое завершается именем самой подсистемы. Например, полное имя “:Sys1:Sys100:MySys” говорит о том, что подсистема с именем `MySys` находится в подсистеме `Sys100`, которая, в свою очередь, находится в подсистеме `Sys1` корневой подсистемы. Для привязки корневой подсистемы передается пустая строка.

`Flags`

Флаги функции (в настоящее время этот параметр никак не используется).

Возвращаемое значение:

Идентификатор созданного (или измененного, если в `VpId` передан идентификатор, а не –1) порта вывода.

Примечания:

Эта функция создает внутри окна управляющего приложения `Window` порт вывода, в котором будет отображаться подсистема `FullSysName`, находящаяся в схеме, загруженной в данный момент в копию РДС, управляемую через связь `Link`. Если `VpId==–1`, создается новый порт, если в `VpId` передан идентификатор существующего порта, его параметры изменяются и он настраивается на отображение подсистемы `FullSysName` в окне `Window`. Функция возвращает уникальный идентификатор порта вывода, который используется во всех функциях для работы с этим портом.

Для отображения какой-либо подсистемы в порте вывода необходимо после его создания задать координаты и размеры порта в окне при помощи функции `rdctrlSetViewportRect` (стр. 676).

Пример использования функции `rdctrlSetViewport` приведен в §3.6.2.

См. также:

`rdctrlCreateLink` (стр. 600), `rdctrlSetViewportRect` (стр. 676),
`rdctrlReleaseViewport` (стр. 673).

Б.3.7.6. `rdctrlSetViewportParams` – установить масштаб и сдвиг подсистемы в порте вывода

Функция `rdctrlSetViewportParams` устанавливает масштаб и положение изображения подсистемы в указанном порте вывода.

```
void RDSCALL rdctrlSetViewportParams(  
    int Link,           // Идентификатор связи  
    int VpId,           // Идентификатор порта вывода  
    double Zoom,        // Масштаб  
    int ScrollX,        // Горизонтальный сдвиг  
    int ScrollY,        // Вертикальный сдвиг  
    DWORD Flags        // Флаги (не используются)  
);
```

Тип указателя на эту функцию:

`RDCTRL_VIIDIIDw`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdctrlCreateLink` (стр. 600).

`VpId`

Идентификатор порта вывода, присвоенный ему при создании функцией `rdctrlSetViewport` (стр. 674).

`Zoom`

Устанавливаемый масштаб подсистемы в долях единицы (1.0 – 100%, 0.5 – 50% и т.д.), или –1, если масштаб изменять не нужно.

`ScrollX, ScrollY`

Горизонтальная (`ScrollX`) и вертикальная (`ScrollY`) координаты левого верхнего угла области рабочего поля подсистемы, видимой в порте вывода, в текущем масштабе.

`Flags`

Флаги функции (в настоящее время этот параметр никак не используется).

Примечания:

Эта функция устанавливает параметры отображения подсистемы для порта вывода `VpId` в копии РДС, управляемой через связь `Link`. Координаты левого верхнего угла видимой в порте области задаются в точках экрана в текущем для порта масштабе, поэтому их можно использовать для взаимодействия с полосами прокрутки.

Пример использования функции `rdctrlSetViewportParams` приведен в §3.6.2.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetViewport` (стр. 674),
`rdscrtlGetViewportParams` (стр. 670).

Б.3.7.7. `rdscrtlSetViewportRect` – задать положение порта вывода в окне

Функция `rdscrtlSetViewportRect` устанавливает координаты прямоугольной области, занимаемой портом вывода в окне управляющего приложения.

```
void RDSCALL rdscrtlSetViewportRect (  
    int Link,                // Идентификатор связи  
    int VpId,               // Идентификатор порта вывода  
    int Left,int Top,       // Левый верхний угол  
    int Width,int Height   // Размеры  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIIIIII`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`VpId`

Идентификатор порта вывода, присвоенный ему при создании функцией `rdscrtlSetViewport` (стр. 674).

`Left, Top`

Горизонтальная (`Left`) и вертикальная (`Top`) координаты левого верхнего угла прямоугольной области, занимаемой портом вывода, в окне управляющего приложения.

`Width, Height`

Ширина (`Width`) и высота (`Height`) порта вывода в окне управляющего приложения.

Примечания:

Эта функция устанавливает координаты и размеры порта вывода `VpId` в копии РДС, управляемой через связь `Link`. Пример ее использования приведен в §3.6.2.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetViewport` (стр. 674).

Б.3.7.8. `rdscrtlSetViewportZoomRect` – настроить масштаб и сдвиг подсистемы по прямоугольнику

Функция `rdscrtlSetViewportZoomRect` так настраивает масштаб и сдвиг изображения подсистемы в порте вывода, чтобы указанный в параметрах функции прямоугольник был полностью видим в порте.

```
void RDSCALL rdscrtlSetViewportZoomRect (  
    int Link,                // Идентификатор связи  
    int VpId,               // Идентификатор порта вывода  
    int Left,int Top,       // Левый верхний угол  
    int Width,int Height,   // Размеры  
    double MaxZoom,        // Max масштаб
```

```
        DWORD Flags           // флаги
    );
```

Тип указателя на эту функцию:

RDCTRL_VIIIIIIIDDw

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdctrlCreateLink` (стр. 600).

VpId

Идентификатор порта вывода, присвоенный ему при создании функцией `rdctrlSetViewport` (стр. 674).

Left, Top

Горизонтальная (Left) и вертикальная (Top) координаты левого верхнего угла области рабочего поля подсистемы, которая должна быть полностью видима в порте вывода. Координаты задаются в текущем масштабе или в масштабе 100% в зависимости от наличия флага `RDCTRL_ZOOMRECTFLAGS_100`.

Width, Height

Ширина (Width) и высота (Height) области рабочего поля подсистемы, которая должна быть полностью видима в порте вывода. Размеры задаются в текущем масштабе или в масштабе 100% в зависимости от наличия флага `RDCTRL_ZOOMRECTFLAGS_100`.

MaxZoom

Максимальный (самый крупный) масштаб в долях единицы, который разрешено установить функции.

Flags

Флаги функции – на данный момент поддерживается единственный флаг:
`RDCTRL_ZOOMRECTFLAGS_100` Координаты и размеры прямоугольной области заданы не в текущем масштабе, а в масштабе 100%.

Примечания:

Эта функция настраивает параметры порта вывода `VpId` в копии РДС, управляемой через связь `Link` так, чтобы прямоугольная область, заданная параметрами `Left`, `Top`, `Width`, и `Height`, была полностью видима в порте вывода и заняла максимальную площадь. Функция попытается настроить масштаб так, чтобы указанный прямоугольник занял всю площадь порта вывода, однако, если необходимый для этого масштаб окажется больше `MaxZoom`, функция установит масштаб `MaxZoom` и разместит прямоугольник в центре порта вывода. Координаты прямоугольника задаются в текущем масштабе порта (при сброшенном флаге `RDCTRL_ZOOMRECTFLAGS_100`) или в масштабе 100% (при взведенном флаге).

Если порт вывода чем-то занят (например, обновляется в данный момент), выполнение функции может быть отложено, поэтому если сразу после ее вызова запросить параметры порта функцией `rdctrlGetViewportParams` (стр. 670), она может вернуть еще не изменившийся масштаб и координаты видимой области. Чтобы узнать, выполнено изменение масштаба или отложено, следует использовать функцию `rdctrlSetViewportZoomRectEx` (стр. 678).

См. также:

`rdctrlCreateLink` (стр. 600), `rdctrlSetViewport` (стр. 674),
`rdctrlSetViewportZoomRectEx` (стр. 678),
`rdctrlSetViewportParams` (стр. 675), `rdctrlViewportFit` (стр. 680).

Б.3.7.9. `rdctrlSetViewportZoomRectEx` – настроить масштаб и сдвиг подсистемы по прямоугольнику (расширенная)

Функция `rdctrlSetViewportZoomRectEx` так настраивает масштаб и сдвиг изображения подсистемы в порте вывода, чтобы указанный в параметрах функции прямоугольник был полностью видим в порте.

```
BOOL RDSCALL rdctrlSetViewportZoomRectEx(  
    int Link,           // Идентификатор связи  
    int VpId,          // Идентификатор порта вывода  
    RDCTRL_PZOOMRECT pRectData // Описание прямоугольника  
);
```

Тип указателя на эту функцию:

`RDCTRL_BIIZr`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdctrlCreateLink` (стр. 600).

`VpId`

Идентификатор порта вывода, присвоенный ему при создании функцией `rdctrlSetViewport` (стр. 674).

`pRectData`

Указатель на структуру `RDCTRL_ZOOMRECT` (стр. 591), описывающую параметры прямоугольника, который должен уместиться в порте вывода.

Возвращаемое значение:

`TRUE` – функция выполнена, `FALSE` – возникла ошибка (например, пользователь вышел из РДС) или выполнение функции отложено до освобождения порта.

Примечания:

Эта функция настраивает параметры порта вывода `VpId` в копии РДС, управляемой через связь `Link` так, чтобы прямоугольная область, описываемая параметром `pRectData`, была полностью видима в порте вывода и заняла максимальную площадь.

См. также:

`rdctrlCreateLink` (стр. 600), `rdctrlSetViewport` (стр. 674),
`RDCTRL_ZOOMRECT` (стр. 591), `rdctrlSetViewportZoomRect` (стр. 676),
`rdctrlViewportFit` (стр. 680).

Б.3.7.10. `rdctrlUpdateViewport` – обновить порт вывода

Функция `rdctrlUpdateViewport` перерисовывает изображение указанного порта вывода.

```
void RDSCALL rdctrlUpdateViewport(  
    int Link,           // Идентификатор связи
```

```

        int VpId          // Идентификатор порта вывода
    );

```

Тип указателя на эту функцию:

RDCTRL_VII

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdctrlCreateLink` (стр. 600).

VpId

Идентификатор уничтожаемого порта вывода, присвоенный ему при создании функцией `rdctrlSetViewport` (стр. 674).

Примечания:

Эта функция дает команду обновить изображение подсистемы в порте вывода VpId в копии РДС, управляемой через связь Link, и ожидает окончания этого обновления.

Пример использования функции `rdctrlUpdateViewport` приведен в §3.6.2.

См. также:

`rdctrlCreateLink` (стр. 600), `rdctrlSetViewport` (стр. 674).

Б.3.7.11. `rdctrlViewportBlockAtPos` – получить имя блока в указанной точке порта вывода

Функция `rdctrlViewportBlockAtPos` возвращает полное имя блока, изображению которого принадлежит указанная точка порта вывода.

```

BOOL RDSCALL rdctrlViewportBlockAtPos(
    int Link,          // Идентификатор связи
    int VpId,          // Идентификатор порта вывода
    int x,int y,        // Координаты точки
    LPVOID BlkName     // Возвращаемое имя блока
);

```

Тип указателя на эту функцию:

RDCTRL_BIIIPV

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdctrlCreateLink` (стр. 600).

VpId

Идентификатор порта вывода, присвоенный ему при создании функцией `rdctrlSetViewport` (стр. 674).

x, y

Горизонтальная (x) и вертикальная (y) координаты точки, в которой ищется блок. Координаты указываются в системе координат окна управляющего приложения, в котором размещается порт вывода VpId.

BlkName

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи `rdscrtlSetStringCallback` (стр. 604), запишет строку с полным именем блока.

Возвращаемое значение:

TRUE – в указанной точке есть блок, FALSE – блок не найден.

Примечания:

Эта функция записывает в объект, указатель на который передан через параметр `BlkName`, полное имя блока, изображению которого принадлежит точка (x,y) в окне порта вывода `VpId` в копии РДС, управляемой через связь `Link`. Для возврата строки используется функция обратного вызова, ранее зарегистрированная через `rdscrtlSetStringCallback`.

Возвращаемое полное имя блока начинается с двоеточия, за которым следует последовательное перечисление через двоеточие имен всех подсистем на пути от корневой подсистемы до этого блока, которое завершается именем самого блока. Например, полное имя “.:Sys1:Sys100:Block1” говорит о том, что блок с именем `Block1` находится в подсистеме `Sys100`, которая, в свою очередь, находится в подсистеме `Sys1` корневой подсистемы.

Пример использования функции `rdscrtlViewportBlockAtPos` приведен в §3.6.3.

См. также:

`rdscrtlCreateLink` (стр. 600), `rdscrtlSetViewport` (стр. 674),
`rdscrtlSetStringCallback` (стр. 604).

Б.3.7.12. `rdscrtlViewportFit` – настроить порт вывода на изображение всей подсистемы

Функция `rdscrtlViewportFit` настраивает масштаб и положение изображения подсистемы в указанном порте вывода так, чтобы в него уместилась вся подсистема.

```
void RDSCALL rdscrtlViewportFit(  
    int Link,           // Идентификатор связи  
    int VpId,           // Идентификатор порта вывода  
    DWORD Flags,        // Флаги (не используются)  
    double *pZoom,      // Возвращаемый масштаб  
    int *pScrollX,      // Возвращаемый гориз. сдвиг  
    int *pScrollY       // Возвращаемый верт. сдвиг  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VIIDwpDpIpI`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`VpId`

Идентификатор порта вывода, присвоенный ему при создании функцией `rdscrtlSetViewport` (стр. 674).

`Flags`

Флаги функции (в настоящее время этот параметр никак не используется).

pZoom

Указатель на вещественную переменную, в которую функция запишет текущий масштаб порта вывода в долях единицы (1.0 – 100%, 0.5 – 50% и т.д.). Если управляющей программе не нужен масштаб, в этом параметре можно передать NULL.

pScrollX, pScrollY

Указатели на целые переменные, в которые функция запишет горизонтальную (pScrollX) и вертикальную (pScrollY) координаты левого верхнего угла области рабочего поля подсистемы, видимой в порте вывода. Координаты возвращаются в точках экрана в текущем масштабе подсистемы. Если управляющей программе не нужны эти координаты, в этих параметрах можно передать NULL.

Примечания:

Эта функция настраивает параметры отображения подсистемы, привязанной в данный момент к порту вывода VpId в копии РДС, управляемой через связь Link, таким образом, чтобы все рабочее поле подсистемы уместилось в этот порт.

См. также:

rdscrtlCreateLink (стр. 600), rdscrtlSetViewport (стр. 674),
rdscrtlSetViewportParams (стр. 675),
rdscrtlSetViewportZoomRect (стр. 676).

Б.3.7.13. rdscrtlViewportKeyboard – вызвать в РДС реакцию на клавиатуру

Функция rdscrtlViewportKeyboard передает в подсистему, привязанную к указанному порту вывода, информацию о нажатии или отпускании клавиши и вызывает реакцию этой подсистемы на клавиатуру.

```
BOOL RDSCALL rdscrtlViewportKeyboard(  
    int Link,           // Идентификатор связи  
    int VpId,           // Идентификатор порта вывода  
    int Operation,      // Нажатие/отпускание (RDSCTRL_KEYOP_*)  
    int KeyCode,        // Код клавиши  
    int RepeatCount,    // Число повторов нажатия  
    DWORD ButShift      // Флаги Ctrl,Alt,Shift (RDSCTRL_KEYF_*)  
);
```

Тип указателя на эту функцию:

RDSCTRL_BIIIIIDw

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией rdscrtlCreateLink (стр. 600).

VpId

Идентификатор порта вывода, присвоенный ему при создании функцией rdscrtlSetViewport (стр. 674).

Operation

Целая константа, указывающая на нажатие или отпускание клавиши:

RDSCTRL_KEYOP_DOWN клавиша нажата;
RDSCTRL_KEYOP_UP клавиша отпущена.

KeyCode

Виртуальный код клавиши (VK_*) согласно описаниям Windows API.

RepeatCount

Только при автоповторе нажатия клавиши – число повторов с момента прошлого вызова реакции подсистемы на клавиатуру.

ButShift

Битовые флаги, описывающие состояние специальных клавиш клавиатуры и кнопок мыши в момент нажатия данной клавиши:

RDSCtrl_KEYF_CTRL	нажата клавиша Ctrl;
RDSCtrl_KEYF_ALT	нажата клавиша Alt;
RDSCtrl_KEYF_SHIFT	нажата клавиша Shift;
RDSCtrl_KEYF_LEFT	нажата левая кнопка мыши;
RDSCtrl_KEYF_RIGHT	нажата правая кнопка мыши;
RDSCtrl_KEYF_MIDDLE	нажата средняя кнопка мыши.

Возвращаемое значение:

TRUE – подсистема или один из ее блоков среагировали на нажатие или отпускание этой клавиши, FALSE – на клавишу никто не среагировал.

Примечания:

Эта функция вызывает в подсистеме, привязанной к порту вывода VpId в копии РДС, управляемой через связь Link, реакцию на нажатие или отпускание клавиши, указанной в параметрах функции. Чаще всего она вызывается управляющим приложением в ответ на нажатие или отпускание клавиши в окне, в котором размещается порт вывода – таким образом действия пользователя транслируются в РДС, и блоки схемы получают возможность реагировать на клавиатуру. В результате вызова функции модели блоков подсистемы вызываются для реакции на событие RDS_BFM_KEYDOWN (стр. 60) или RDS_BFM_KEYUP (стр. 62), а если они не обработали его, вызывается модель самой подсистемы для реакции на событие RDS_BFM_WINDOWKEYDOWN (стр. 72) или RDS_BFM_WINDOWKEYUP (стр. 73).

Пример использования функции rdscrtlViewportKeyboard приведен в §3.6.4.

См. также:

rdscrtlCreateLink (стр. 600), rdscrtlSetViewport (стр. 674),
RDS_BFM_KEYDOWN (стр. 60), RDS_BFM_KEYUP (стр. 62),
RDS_BFM_WINDOWKEYDOWN (стр. 72), RDS_BFM_WINDOWKEYUP (стр. 73),
rdscrtlViewportMouse (стр. 682).

Б.3.7.14. rdscrtlViewportMouse – вызвать в РДС реакцию на мышшь

Функция rdscrtlViewportMouse передает в подсистему, привязанную к указанному порту вывода, информацию о перемещении курсора мыши и нажатии ее кнопок, вызывая в этой подсистеме соответствующие реакции.

```
BOOL RDSCALL rdscrtlViewportMouse(  
    int Link,           // Идентификатор связи  
    int VpId,          // Идентификатор порта вывода  
    int x, int y,       // Координаты курсора  
    int Operation,      // Операция (RDSCtrl_MOUSEOP_*)  
    DWORD ButShift      // Флаги (RDSCtrl_MOUSEF_*)  
);
```

Тип указателя на эту функцию:

RDSCtrl_BIIIIIDw

Параметры:

Link

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

VpId

Идентификатор порта вывода, присвоенный ему при создании функцией `rdscrtlSetViewport` (стр. 674).

x, y

Горизонтальная (x) и вертикальная (y) координаты курсора мыши. Координаты указываются в системе координат окна управляющего приложения, в котором размещается порт вывода VpId.

Operation

Целая константа, указывающая на произошедшее событие:

RDSCtrl_MOUSEOP_DOWN	нажата кнопка мыши;
RDSCtrl_MOUSEOP_UP	отпущена кнопка мыши;
RDSCtrl_MOUSEOP_MOVE	курсор мыши перемещен;
RDSCtrl_MOUSEOP_DBL	двойной щелчок левой кнопкой мыши.

ButShift

Битовые флаги, описывающие нажатую/отпущенную кнопку мыши и состояние специальных клавиш клавиатуры в момент события Operation:

RDSCtrl_MOUSEF_LEFT	левая кнопка мыши;
RDSCtrl_MOUSEF_RIGHT	правая кнопка мыши;
RDSCtrl_MOUSEF_MIDDLE	средняя кнопка мыши;
RDSCtrl_MOUSEF_CTRL	нажата клавиша Ctrl;
RDSCtrl_MOUSEF_ALT	нажата клавиша Alt;
RDSCtrl_MOUSEF_SHIFT	нажата клавиша Shift.

При отработке нажатия и отпускания кнопок мыши в параметре ButShift следует передавать только один из флагов RDSCtrl_MOUSEF_LEFT, RDSCtrl_MOUSEF_RIGHT и RDSCtrl_MOUSEF_MIDDLE – тот, который соответствует нажатой или отпущенной кнопке.

Возвращаемое значение:

TRUE – подсистема или один из ее блоков среагировали на мышшь, FALSE – на мышшь никто не среагировал.

Примечания:

Эта функция вызывает в подсистеме, привязанной к порту вывода VpId в копии РДС, управляемой через связь Link, реакцию на действия пользователя мышью, указанные в параметрах функции. Чаще всего она вызывается управляющим приложением в ответ на действия пользователя в окне, в котором размещается порт вывода – таким образом они транслируются в РДС, и блоки схемы получают возможность реагировать на мышшь. В результате вызова функции в зависимости от значения Operation модели блоков подсистемы вызываются для реакции на событие из набора RDS_BFM_MOUSE*, а если они не обработали его, вызывается модель самой подсистемы для реакции на сходное событие из набора RDS_BFM_WINDOWMOUSE*.

В некоторых случаях действия пользователя можно не передавать в РДС: например, если ни окно подсистемы, ни один из блоков этой подсистемы не реагируют на перемещения

мышь без нажатых кнопок, функцию `rdscrtlViewportMouse` можно вызывать, только если хотя бы одна кнопка мыши нажата, разгрузив тем самым канал передачи данных между управляющим приложением и РДС. Чтобы узнать, какие именно действия мышью требуются подсистеме, привязанной в данный момент к порту вывода, следует использовать функцию `rdscrtlGetVPMouseLevel` (стр. 672).

Пример использования функции `rdscrtlViewportMouse` приведен в §3.6.4.

См. также:

```
rdscrtlCreateLink (стр. 600), rdscrtlSetViewport (стр. 674),  
rdscrtlGetVPMouseLevel (стр. 672), RDS_BFM_MOUSEDOWN (стр. 67),  
RDS_BFM_MOUSEUP (стр. 69), RDS_BFM_MOUSEMOVE (стр. 68),  
RDS_BFM_MOUSEDBLCLICK (стр. 64), RDS_BFM_WINDOWMOUSEDOWN (стр. 74),  
RDS_BFM_WINDOWMOUSEUP (стр. 76), RDS_BFM_WINDOWMOUSEMOVE (стр. 75),  
RDS_BFM_WINDOWMOUSEDBLCLICK (стр. 73),  
rdscrtlViewportKeyboard (стр. 681).
```

Б.3.7.15. `rdscrtlViewportSystem` – получить имя подсистемы в порте вывода

Функция `rdscrtlViewportSystem` возвращает полное имя подсистемы, привязанной в данный момент к указанному порту вывода.

```
BOOL RDSCALL rdscrtlViewportSystem(  
    int Link,           // Идентификатор связи  
    int VpId,          // Идентификатор порта вывода  
    LPVOID SysName     // Возвращаемое имя подсистемы  
);
```

Тип указателя на эту функцию:

`RDSCTRL_BIIpV`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdscrtlCreateLink` (стр. 600).

`VpId`

Идентификатор порта вывода, присвоенный ему при создании функцией `rdscrtlSetViewport` (стр. 674).

`SysName`

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи `rdscrtlSetStringCallback` (стр. 604), запишет строку с полным именем подсистемы. Для корневой подсистемы будет возвращена пустая строка.

Возвращаемое значение:

`TRUE` – функция выполнена успешно, `FALSE` – ошибка (нет такого порта и т.п.).

Примечания:

Эта функция записывает в объект, указатель на который передан через параметр `SysName`, полное имя подсистемы привязанной к порту вывода `VpId` в копии РДС, управляемую через связь `Link`. Для возврата строки используется функция обратного вызова, ранее зарегистрированная через `rdscrtlSetStringCallback`.

Возвращаемое полное имя подсистемы начинается с двоеточия, за которым следует последовательное перечисление через двоеточие всех имен подсистем на пути от корневой подсистемы до этой. Например, полное имя “:Sys1:Sys100:MySys” говорит о том, что подсистема с именем MySys находится в подсистеме Sys100, которая, в свою очередь, находится в подсистеме Sys1 корневой подсистемы.

Пример использования функции `rdctrlViewportSystem` приведен в §3.6.3.

См. также:

`rdctrlCreateLink` (стр. 600), `rdctrlSetViewport` (стр. 674),
`rdctrlSetStringCallback` (стр. 604).

Б.3.7.16. `rdctrlVPPopupHint` – получить текст всплывающей подсказки

Функция `rdctrlVPPopupHint` возвращает текст и параметры всплывающей подсказки, выдаваемой блоком, изображению которого принадлежит указанная точка порта вывода.

```
DWORD RDSCALL rdctrlVPPopupHint(  
    int Link,           // Идентификатор связи  
    int VpId,          // Идентификатор порта вывода  
    int x,int y,        // Координаты точки  
    LPVOID HintText,   // Возвращаемый текст подсказки  
    int *pLeft,int *pTop, // Левый верхний угол зоны  
    int *pRight,int *pBottom, // Правый нижний угол зоны  
    int *pReshowTimeout, // Интервал повторного вывода, мс  
    int *pHideTimeout   // Интервал скрытия, мс  
);
```

Тип указателя на эту функцию:

`RDCTRL_DwIIIIpVpIpIpIpIpIpI`

Параметры:

`Link`

Идентификатор связи с РДС, присвоенный ей при создании функцией `rdctrlCreateLink` (стр. 600).

`VpId`

Идентификатор порта вывода, присвоенный ему при создании функцией `rdctrlSetViewport` (стр. 674).

`x, y`

Горизонтальная (x) и вертикальная (y) координаты точки, к которой запрашивается подсказка (в этой точке будет искаться блок). Координаты указываются в системе координат окна управляющего приложения, в котором размещается порт вывода `VpId`.

`HintText`

Указатель на объект управляющего приложения, в который функция обратного вызова, зарегистрированная при помощи `rdctrlSetStringCallback` (стр. 604), запишет текст всплывающей подсказки, полученный от блока по координатам (x,y). Текст может содержать несколько строк, разделенных кодом перевода строки “\n” (10).

`pLeft, pTop, pRight, pBottom`

Указатели на целые переменные, в которые функция запишет координаты левого верхнего (`pLeft` – горизонтальная координата, `pTop` – вертикальная) и правого

нижнего (pRight – горизонтальная координата, pBottom – вертикальная) углов области действия подсказки. При выходе курсора из этой прямоугольной области необходимо запросить новую подсказку. Координаты указываются в системе координат окна управляющего приложения, в котором размещается порт вывода VpId. Если вызывающей программе не нужны эти координаты, в этих параметрах можно передать NULL.

pReshowTimeout

Указатель на целую переменную, в которую функция запишет возвращенный моделью блока интервал времени в миллисекундах после гашения подсказки, по прошествии которого подсказку необходимо вывести снова. Если вызывающей программе не нужен этот интервал, в параметре pReshowTimeout можно передать NULL.

pHideTimeout

Указатель на целую переменную, в которую функция запишет возвращенный моделью блока интервал времени в миллисекундах после вывода подсказки, по прошествии которого ее необходимо убрать с экрана. Если вызывающей программе не нужен этот интервал, в параметре pHideTimeout можно передать NULL.

Возвращаемое значение:

Уникальный внешний идентификатор блока, выдавшего текст подсказки, или 0, если по координатам (x,y) нет изображения блока или если блок не выдает подсказку.

Примечания:

Эта функция ищет в подсистеме, привязанной в данный момент к порту вывода VpId в копии РДС, управляемой через связь Link, блок, изображению которого принадлежит точка с оконными координатами (x,y). Модель этого блока вызывается для реакции на событие RDS_BFM_POPUPHINT (стр. 70), после чего полученный от нее текст и параметры всплывающей подсказки передаются в управляющую программу через параметры функции. Функция обычно используется для организации вывода всплывающих подсказок блоков в управляющем приложении.

Пример использования функции rdscrtlVPPopupHint приведен в §3.6.6.

См. также:

rdscrtlCreateLink (стр. 600), rdscrtlSetViewport (стр. 674),
RDS_BFM_POPUPHINT (стр. 70), rdscrtlViewportBlockAtPos (стр. 679).

Б.3.8. Отладочные функции

Описываются функции, облегчающие отладку приложения, управляющего РДС.

Б.3.8.1. rdscrtlClearLog – очистить журнал

Функция rdscrtlClearLog удаляет файл журнала, в который библиотека RdsCtrl.dll записывает служебную информацию о связи с РДС и обнаруженных ошибках.

void RDSCALL rdscrtlClearLog(void);

Тип указателя на эту функцию:

RDSCRTL_VV

Примечания:

Имя файла журнала задается функцией rdscrtlSetLogFile (стр. 688).

См. также:

`rdscrtlSetLogFile` (стр. 688).

Б.3.8.2. `rdscrtlEnableLog` – включить/выключить журнал

Функция `rdscrtlEnableLog` разрешает или запрещает ведение файла журнала, в которой библиотека `RdsCtrl.dll` записывает служебную информацию о связи с РДС и обнаруженных ошибках.

```
void RDSCALL rdscrtlEnableLog(  
    BOOL Enable,      // Разрешить/запретить  
    BOOL Clear       // Очистить журнал  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VBB`

Параметры:

`Enable`

`TRUE` – разрешить ведение журнала, `FALSE` – запретить.

`Clear`

`TRUE` – очистить файл журнала, `FALSE` – не очищать.

Примечания:

При включенном журнале в текстовый файл, имя которого задано функцией `rdscrtlSetLogFile` (стр. 688), будут записываться служебные сообщения о ходе установки связи управляющего приложения с РДС и о возникающих ошибках. В журнал можно также добавлять пользовательские сообщения функцией `rdscrtlLogString` (стр. 687).

По умолчанию ведение журнала запрещено.

См. также:

`rdscrtlSetLogFile` (стр. 688), `rdscrtlLogString` (стр. 687).

Б.3.8.3. `rdscrtlLogString` – записать текст в журнал

Функция `rdscrtlLogString` записывает указанный текст в файл журнала библиотеки `RdsCtrl.dll`.

```
void RDSCALL rdscrtlLogString(  
    LPSTR String,    // Текст  
    BOOL CrLf       // Перевести строку  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VSB`

Параметры:

`String`

Указатель на текст, который нужно записать в журнал.

`CrLf`

`TRUE` – перевести строку после записи текста, `FALSE` – не переводить.

Примечания:

Имя файла журнала задается функцией `rdscrtlSetLogFile` (стр. 688). В него записывается не только текст, переданный функцией `rdscrtlLogString`, но и служебные сообщения `RdsCtrl.dll`. Для записи текста ведение журнала должно быть разрешено функцией `rdscrtlEnableLog` (стр. 687).

См. также:

`rdscrtlSetLogFile` (стр. 688), `rdscrtlEnableLog` (стр. 687).

Б.3.8.4. `rdscrtlSetLogFile` – задать имя файла журнала

Функция `rdscrtlSetLogFile` задает имя для файла журнала `RdsCtrl.dll`.

```
void RDSCALL rdscrtlSetLogFile(  
    LPSTR FileName, // Полное имя файла  
    BOOL Clear      // Очистить журнал  
);
```

Тип указателя на эту функцию:

`RDSCRTL_VSB`

Параметры:

`FileName`

Указатель на строку с полным именем файла журнала.

`Clear`

`TRUE` – очистить файл журнала, `FALSE` – не очищать.

Примечания:

По умолчанию ведение журнала запрещено, для его разрешения после `rdscrtlSetLogFile` следует вызвать функцию `rdscrtlEnableLog` (стр. 687), в первом параметре которой передано `TRUE`.

См. также:

`rdscrtlEnableLog` (стр. 687).

Приложение В. Параметры командной строки РДС

Описываются параметры командной строки главной программы РДС (rds.exe), при помощи которых можно сразу после запуска РДС загрузить схему или изменить режим работы.

В.1. Передача параметров в командной строке и указание имени файла схемы

Как и для любого приложения Windows, для РДС в командной строке запуска можно указывать различные параметры. Параметры отделяются друг от друга пробелами, если в тексте одного из параметров есть пробелы (например, если это имя файла), его следует заключать в двойные кавычки. Параметры не чувствительны к регистру символов – слова “/run” и “/RUN” будут считаться одним и тем же параметром. Сами параметры и их назначение описываются далее в этом приложении.

Первый неопознанный РДС параметр будет считаться именем файла схемы, которую нужно загрузить. Имя файла необходимо указывать с полным путем, стандартные обозначения путей РДС (см. стр. 189) использовать нельзя – на момент разбора параметров командной строки стандартные пути РДС еще не установлены, и могут измениться в процессе этого разбора.

Примеры:

```
rds.exe "c:\work\rds\sample.rds"
```

После запуска РДС загрузить схему “sample.rds” из папки “c:\work\rds”.

```
rds.exe "c:\work\rds\sample.rds" /run /hide
```

После запуска РДС загрузить схему “sample.rds” из папки “c:\work\rds” и запустить расчет, главное окно РДС сделать невидимым.

```
rds.exe /calcmode "c:\work\rds\sample.rds"
```

После запуска РДС загрузить схему “sample.rds” из папки “c:\work\rds” и перейти в режим моделирования.

В.2. “/calcmode” – перейти в режим моделирования

Если в командной строке РДС указан параметр “/calcmode”, после загрузки схемы РДС перейдет в режим моделирования (по умолчанию РДС запускается в режиме редактирования). Без указания имени файла схемы в командной строке этот параметр не имеет смысла, поскольку переключать режимы РДС можно только при загруженной схеме.

Пример:

```
rds.exe "c:\work\rds\sample.rds" /calcmode
```

После запуска РДС загрузить схему “sample.rds” из папки “c:\work\rds” и перейти в режим моделирования.

В.3. “/end” – прекратить разбор параметров

Если в командной строке указан параметр “/end”, все следующие за ним параметры не будут восприняты РДС. Вместо этого они будут запомнены, и модели блоков смогут обращаться к ним при помощи сервисных функций `rdsFindCmdParam` (стр. 153), `rdsGetCmdParam` (стр. 155) и `rdsGetCmdParamCount` (стр. 156). Так можно передать параметры непосредственно моделям блоков без обработки РДС.

Пример:

```
rds.exe "c:\work\rds\sample.rds" /end param1 param2 "param 3"
```

После запуска РДС загрузить схему “sample.rds” из папки “c:\work\rds” и запомнить параметры “param1”, “param2” и “param 3” для выдачи их моделям блоков (функция `rdsGetCmdParamCount` в данном случае вернет число 3).

В.4. “/hide” – скрыть главное окно РДС

Если в командной строке указан параметр “/hide”, главное окно РДС будет скрыто от пользователя. Роль главного окна будет выполнять окно корневой подсистемы загруженной схемы – при его закрытии РДС завершится.

При внешнем управлении через библиотеку `RdsCtrl.dll` (см. главу 3) для скрытия главного окна РДС используется функция `rdscrtlShowMainWindow` (стр. 614).

Пример:

```
rds.exe "c:\work\rds\sample.rds" /hide
```

После запуска РДС загрузить схему “sample.rds” из папки “c:\work\rds” и скрыть главное окно.

В.5. “/ini” – задать путь к INI-файлам РДС

Если в командной строке указан параметр “/ini”, следующий за ним параметр трактуется как путь к папке, в которой размещаются все файлы параметров РДС. Этот путь будет подставляться вместо обозначения “\$INIS” (стр. 189) в сервисных функциях.

Пример:

```
rds.exe /ini "d:\appdata\rds" "c:\work\rds\sample.rds"
```

После запуска РДС установить в качестве папки INI-файлов “d:\appdata\rds” и загрузить схему “sample.rds” из папки “c:\work\rds”.

В.6. “/nosplash” – не выводить заставку РДС

Если в командной строке указан параметр “/nosplash”, в процессе запуска РДС не будет выводиться стандартная заставка. Обычно этот параметр используют при включении РДС в состав каких-либо программных комплексов, имеющих собственную заставку.

Пример:

```
rds.exe /nosplash "c:\work\rds\sample.rds"
```

Запустить РДС, не выводя заставку, и загрузить схему “sample.rds” из папки “c:\work\rds”.

В.7. “/run” – перейти в режим расчета

Если в командной строке РДС указан параметр “/run”, после загрузки схемы РДС запустит расчет загруженной схемы (по умолчанию РДС запускается в режиме редактирования). Без указания имени файла схемы в командной строке этот параметр не имеет смысла, поскольку переключать режимы РДС можно только при загруженной схеме.

Пример:

```
rds.exe "c:\work\rds\sample.rds" /run
```

После запуска РДС загрузить схему “sample.rds” из папки “c:\work\rds” и перейти в режим расчета.

В.8. “/server” – запустить выделенный сервер РДС

Если в командной строке указан параметр “/server”, РДС запустится в режиме выделенного сервера (см. §2.15.1). Следующий за ним параметр трактуется как номер порта сервера. Если необходимо запустить сервер с номером порта по умолчанию, заданным в настройках РДС, вместо номера порта следует указать слово “default”.

При работе РДС в режиме выделенного сервера загрузка схемы невозможна, поэтому указанное в параметрах имя файла схемы будет проигнорировано.

Примеры:

```
rds.exe /server 2345
```

Запустить РДС в режиме выделенного сервера, работающего через порт 2345.

```
rds.exe /server default
```

Запустить РДС в режиме выделенного сервера, работающего через порт, указанный в настройках.

В.9. “/skip” – не разбирать следующий параметр

Если в командной строке указан параметр “/skip”, следующий за ним параметр не будет воспринят РДС, вместо этого он будет запомнен в списке параметров, доступных моделям блоков при помощи сервисных функций `rdsFindCmdParam` (стр. 153), `rdsGetCmdParam` (стр. 155) и `rdsGetCmdParamCount` (стр. 156). Таким образом можно передать параметры непосредственно моделям блоков без обработки РДС.

Начиная со следующего после пропущенного параметра РДС возобновит разбор командной строки, поэтому если нужно передать моделям блоков загружаемой схемы сразу несколько параметров, лучше сгруппировать их в конце строки после параметра “/end” (стр. 689).

Пример:

```
rds.exe /skip param1 "c:\work\rds\sample.rds" /skip "param 2" /hide
```

После запуска РДС загрузить схему “sample.rds” из папки “c:\work\rds” и запомнить параметры “param1” и “param 2” для выдачи их моделям блоков (функция `rdsGetCmdParamCount` в данном случае вернет число 2). Главное окно РДС будет скрыто от пользователя.

В.10. “/temp” – задать путь к папке временных файлов РДС

Если в командной строке указан параметр “/temp”, следующий за ним параметр трактуется как путь к папке, в которой будут размещаться все временные файлы, создаваемые РДС. Этот путь будет подставляться вместо обозначения “\$TEMP\$” (стр. 189) в сервисных функциях.

Пример:

```
rds.exe /temp "d:\appdata\rds\temp" "c:\work\rds\sample.rds"
```

После запуска РДС установить в качестве папки временных файлов “d:\appdata\rds\temp” и загрузить схему “sample.rds” из папки “c:\work\rds”.

В.11. “/wintemp” – использовать папку временных файлов Windows

Если в командной строке указан параметр “/wintemp”, в качестве папки временных файлов РДС будет использоваться стандартная папка временных файлов Windows. Путь к ней будет подставляться в сервисных функциях не только вместо обозначения “\$WINTEMP\$”, но и вместо “\$TEMP\$” (стр. 189).

Пример:

```
rds.exe /wintemp "c:\work\rds\sample.rds"
```

После запуска РДС установить в качестве папки временных файлов РДС стандартную папку временных файлов Windows и загрузить схему “sample.rds” из папки “c:\work\rds”.

Алфавитный указатель

BOOL.....	24
COLORREF.....	24
DLL_PROCESS_ATTACH.....	27
DLL_PROCESS_DETACH.....	28
DWORD.....	24
HBITMAP.....	24, 166
HDC.....	24, 57, 128, 365
HINSTANCE.....	24, 27, 154, 166
HWND.....	24, 131, 155, 158, 166
LOGFONT.....	24, 381
LPSTR.....	24
LPVOID.....	25
POINT.....	25
PS_DASH.....	118, 286, 382
PS_DASHDOT.....	118, 286, 382
PS_DASHDOTDOT.....	118, 286, 382
PS_DOT.....	118, 286, 382
PS_INSIDEFRAME.....	287, 383
PS_NULL.....	118, 286, 383
PS_SOLID.....	118, 286, 383
R2_COPYPEN.....	383
R2_NOT.....	383
RDS_ALTBKNAME_CENTER.....	116, 242
RDS_ALTBKNAME_LEFT.....	116, 242
RDS_ALTBKNAME_RIGHT.....	116, 242
RDS_ARRAYACCESSDATA.....	112
RDS_ARRAYCOLS.....	337
RDS_ARRAYDATA.....	338
RDS_ARRAYEXISTS.....	339
RDS_ARRAYITEM.....	339
RDS_ARRAYITEMADDR.....	340
RDS_ARRAYROWS.....	341
RDS_BCALL_ALLOWSTOP.....	306, 309
RDS_BCALL_CHECKSUPPORT.....	32, 306, 309, 314
RDS_BCALL_FIRST.....	306, 314
RDS_BCALL_LAST.....	306, 314
RDS_BCALL_SUBSYSTEMS.....	306, 309
RDS_BDF_ALLOWRESIZE.....	115
RDS_BDF_FREEMOUSEMOVE.....	115, 125
RDS_BDF_HASPICTURE.....	115, 244
RDS_BDF_KBDEVENTS.....	115, 125
RDS_BDF_LOCKHEIGHT.....	115
RDS_BDF_LOCKWIDTH.....	115
RDS_BDF_MOUSEEVENTS.....	115, 125
RDS_BDF_NAMEOFF.....	115
RDS_BDF_POPUPHINT.....	115
RDS_BDF_RUNEVERYCYCLE.....	115
RDS_BDF_SELFDRAW.....	115
RDS_BDF_SETUPBYDCLICK.....	116

RDS_BDF_SETUPFUNC.....	116, 247
RDS_BDF_SHOWMAINPOINT.....	116
RDS_BDF_TEXTRECT.....	116, 244
RDS_BDNP_ABOVE.....	116
RDS_BDNP_BELOW.....	116
RDS_BDNP_CUSTOM.....	116
RDS_BEN_INPUTS.....	215
RDS_BEN_OUTPUTS.....	215
RDS_BEN_TRACELINKS.....	215, 217
RDS_BEU_STORECHANGED.....	537
RDS_BFM_AFTERLOAD.....	50
RDS_BFM_AFTERSAVE.....	51
RDS_BFM_BEFORESAVE.....	51
RDS_BFM_BLOCKPANEL.....	55
RDS_BFM_CALCMODE.....	31
RDS_BFM_CHECKFUNCSUPPORT.....	32
RDS_BFM_CLEANUP.....	33
RDS_BFM_CONTEXTPOPUP.....	56
RDS_BFM_DRAW.....	57
RDS_BFM_DRAWADDITIONAL.....	59
RDS_BFM_DYNVARCHANGE.....	34
RDS_BFM_EDITMODE.....	35
RDS_BFM_FUNCTIONCALL.....	35
RDS_BFM_INIT.....	38
RDS_BFM_KEYDOWN.....	60
RDS_BFM_KEYUP.....	62
RDS_BFM_LOADBIN.....	52, 277
RDS_BFM_LOADSTATE.....	39, 251, 277
RDS_BFM_LOADTXT.....	52, 278
RDS_BFM_MANUALDELETE.....	78
RDS_BFM_MANUALINSERT.....	79
RDS_BFM_MENUFUNCTION.....	63
RDS_BFM_MODEL.....	40
RDS_BFM_MOUSEDBLCLICK.....	64
RDS_BFM_MOUSEDOWN.....	67
RDS_BFM_MOUSEMOVE.....	68
RDS_BFM_MOUSESELECT.....	80
RDS_BFM_MOUSEUP.....	69
RDS_BFM_MOVED.....	81, 240, 246
RDS_BFM_NETCONNECT.....	84
RDS_BFM_NETDATAACCEPTED.....	85
RDS_BFM_NETDATARECEIVED.....	87
RDS_BFM_NETDISCONNECT.....	88
RDS_BFM_NETERROR.....	89
RDS_BFM_POPUPHINT.....	70, 249
RDS_BFM_PREMODEL.....	41
RDS_BFM_REMOTEMSG.....	42
RDS_BFM_RENAME.....	82, 241
RDS_BFM_RESETCALC.....	44, 252
RDS_BFM_RESIZE.....	82, 246
RDS_BFM_RESIZING.....	84

RDS_BFM_SAVEBIN.....	53, 280
RDS_BFM_SAVESTATE.....	45, 253, 280
RDS_BFM_SAVETXT.....	54, 281
RDS_BFM_SETUP.....	71, 247
RDS_BFM_STARTCALC.....	46
RDS_BFM_STOPCALC.....	46
RDS_BFM_TIMER.....	47
RDS_BFM_UNLOADSYSTEM.....	47
RDS_BFM_VARCHECK.....	48
RDS_BFM_WINDOWKEYDOWN.....	72
RDS_BFM_WINDOWKEYUP.....	73
RDS_BFM_WINDOWMOUSEDCLICK.....	73
RDS_BFM_WINDOWMOUSEDOWN.....	74
RDS_BFM_WINDOWMOUSEMOVE.....	75
RDS_BFM_WINDOWMOUSEUP.....	76
RDS_BFM_WINDOWOPERATION.....	76
RDS_BFM_WINREFRESH.....	77, 263
RDS_BFR_BADVARMSG.....	48
RDS_BFR_DONE.....	26
RDS_BFR_ERROR.....	48, 52, 54
RDS_BFR_NOTPROCESSED.....	64, 67-70, 74-76, 80
RDS_BFR_SHOWMENU.....	67
RDS_BFR_STOP.....	60, 62, 72, 73, 83, 84
RDS_BHANDLE.....	23, 28, 121
RDS_BLOCKDATA.....	26, 28
RDS_BLOCKDESCRIPTION.....	113, 220
RDS_BLOCKDIMENSIONS.....	117, 221, 222, 226
RDS_BLOCKPANEL_OP_HIDE.....	173
RDS_BLOCKPANEL_OP_HIDEEXCEPT.....	173
RDS_BLOCKPANEL_OP_SELECT.....	174
RDS_BLOCKPANEL_OP_SHOW.....	173
RDS_BLOCKPANEL_OP_SHOWALL.....	174
RDS_BTALLTYPES.....	420
RDS_BTBUSPORT.....	102, 114, 127, 420, 633
RDS_BTDLBLOCK.....	114
RDS_BTINPUTBLOCK.....	102, 114, 127, 420, 633
RDS_BTOUTPUTBLOCK.....	102, 114, 127, 420, 633
RDS_BTSIMPLEBLOCK.....	102, 114, 127, 420, 633
RDS_BTSYSTEM.....	102, 113, 127, 420, 633
RDS_CAOCOUNT.....	205
RDS_CAODELETE.....	205
RDS_CAOGET.....	205
RDS_CAOPREALLOCATE.....	205
RDS_CAORESTORE.....	205
RDS_CAOSET.....	205
RDS_CAOSETCURRENT.....	205
RDS_CFD_ABSPATH.....	199
RDS_CFD_CREATEPROMPT.....	199
RDS_CFD_MUSTEXIST.....	199
RDS_CFD_OPEN.....	199
RDS_CFD_OVERWRITEPROMPT.....	199

RDS_CFD_SAVE.....	199
RDS_CHANDLE.....	23, 119, 415
RDS_COMP_AR_LOADCLIPBRD.....	96
RDS_COMP_AR_LOADFROMFILE.....	96
RDS_COMP_AR_LOADSYSTEM.....	96
RDS_COMP_AR_LOADUNDO.....	96
RDS_COMP_AR_MANUALSET.....	96
RDS_COMP_AR_RENAMEMODEL.....	96
RDS_COMP_AR_UNKNOWN.....	96
RDS_COMPBLOCKOPDATA.....	95, 101
RDS_COMPCANATTACHBLKDATA.....	97
RDS_COMPEXECFUNCDATA.....	102
RDS_COMPFLAG_CANCHANGESTRUCT.....	103
RDS_COMPFLAG_FUNCMODEL BrowSE.....	102, 103
RDS_COMPFLAG_FUNCMODELCREATE.....	102, 103
RDS_COMPFLAG_FUNCMODELSAVEAS.....	102, 103
RDS_COMPFLAG_FUNCMODELUSERINPUT.....	102, 103
RDS_COMPHANDLE.....	24, 92, 93
RDS_COMPILEDATA.....	100
RDS_COMPM_ATTACHBLOCK.....	95
RDS_COMPM_CANATTACHBLK.....	96
RDS_COMPM_CANRENMODEL.....	98
RDS_COMPM_CLEANUP.....	99
RDS_COMPM_CLOSEALLWIN.....	99
RDS_COMPM_COMPILE.....	100
RDS_COMPM_DETACHBLOCK.....	101
RDS_COMPM_EXECFUNCTION.....	101
RDS_COMPM_GETOPTIONS.....	103
RDS_COMPM_INIT.....	103
RDS_COMPM_MODECHANGE.....	104
RDS_COMPM_MODELCLEANUP.....	104
RDS_COMPM_MODELINIT.....	105
RDS_COMPM_MODELRENAMED.....	105
RDS_COMPM_OPENEDITOR.....	106
RDS_COMPM_PREPARE.....	107
RDS_COMPM_SAVEBLOCK.....	108
RDS_COMPM_SAVESYSTEM.....	109
RDS_COMPM_SETUP.....	110
RDS_COMPM_STRUCTCHANGE.....	110
RDS_COMPMODELDATA.....	93
RDS_COMPMODELRENAMEDATA.....	98
RDS_COMPMODULEDATA.....	92
RDS_COMPPREPAREDATA.....	107
RDS_COMPR_DONE.....	96, 100, 101
RDS_COMPR_ERROR.....	96, 100, 101
RDS_COMPR_ERRORNOMSG.....	97
RDS_COMPSAVEBLOCKDATA.....	108
RDS_COMPSAVESYSTEMDATA.....	109
RDS_COMPSTRUCTCHGDATA.....	110
RDS_CONNAPPEARANCE.....	118, 204, 225, 227, 248
RDS_CONNDESCRIPTION.....	119, 226

RDS_CONTEXTPOPUPDATA.....	56
RDS_CSV_CLEAR.....	559
RDS_CSV_CLOSEFILE.....	559
RDS_CSV_DELIMITERCHAR.....	560
RDS_CSV_FILEERROR.....	560
RDS_CSV_FILEISOPEN.....	561
RDS_CSV_LINE.....	561
RDS_CSV_LINECOLUMNS.....	562
RDS_CSV_LINECOUNT.....	563
RDS_CSV_LOADFROMFILE.....	563
RDS_CSV_MAXCOLUMNS.....	564
RDS_CSV_OPENFILEREAD.....	564
RDS_CSV_OPENFILEWRITE.....	565
RDS_CSV_QUOTECHAR.....	566
RDS_CSV_SAVETOFILE.....	566
RDS_CSV_STRFROMFILE.....	567
RDS_CSV_STRTOFILE.....	567
RDS_CSV_TEXT.....	568
RDS_CTALLTYPES.....	420
RDS_CTBUS.....	120, 127, 420
RDS_CTCONNECTION.....	120, 127, 420
RDS_CTRLCALC.....	31
RDS_DISABLED.....	31
RDS_DRAWDATA.....	57
RDS_DVPARENT.....	347, 353
RDS_DVROOT.....	348, 353
RDS_DVSELF.....	347, 353
RDS_DYNVARLINK.....	34, 121, 348, 353
RDS_EDITORPARAMETERS.....	122, 258
RDS_EDITORTOOLBARS.....	126, 259, 265
RDS_EWF_ALLBARS.....	266
RDS_EWF_CALCTOOLBAR.....	259
RDS_EWF_DISPLAYTOOLBAR.....	259
RDS_EWF_LAYERSTOOLBAR.....	259
RDS_EWF_PRINTTOOLBAR.....	260
RDS_EWF_STATUSBAR.....	260
RDS_EWF_ZOOMTOOLBAR.....	260
RDS_FINDBYEXTIDDATA.....	126
RDS_FORM_CLEAR.....	505
RDS_FORM_INVALIDATE.....	505
RDS_FORM_SHOWMODAL.....	506
RDS_FORMCTRL_BUTTON.....	493, 494
RDS_FORMCTRL_CHECKBOX.....	493, 495
RDS_FORMCTRL_COLOR.....	493, 495
RDS_FORMCTRL_COMBOEDIT.....	493, 495
RDS_FORMCTRL_COMBOLIST.....	493, 495
RDS_FORMCTRL_DIRDIALOG.....	493, 496
RDS_FORMCTRL_DISPLAY.....	493, 496
RDS_FORMCTRL_EDIT.....	493, 496
RDS_FORMCTRL_FONTSELECT.....	493, 496
RDS_FORMCTRL_HOTKEY.....	493, 496

RDS_FORMCTRL_LABEL.....	493, 497
RDS_FORMCTRL_LISTANDEDIT.....	493, 497
RDS_FORMCTRL_MULTILINE.....	493, 497
RDS_FORMCTRL_NONVISUAL.....	493, 498
RDS_FORMCTRL_OPENDIALOG.....	493, 498
RDS_FORMCTRL_PAINTBOX.....	493, 498
RDS_FORMCTRL_RADIOBUTTON.....	493, 499
RDS_FORMCTRL_RANGEEDIT.....	493, 499
RDS_FORMCTRL_SAVEDIALOG.....	493, 499
RDS_FORMCTRL_UPDOWN.....	493, 500
RDS_FORMFLAG_CHECK.....	493, 500
RDS_FORMFLAG_CHECKRADIO.....	493, 501
RDS_FORMFLAG_DISABLED.....	493, 501
RDS_FORMFLAG_LCENTER.....	494, 501
RDS_FORMFLAG_LINE.....	494, 501
RDS_FORMFLAG_LRIGHT.....	494, 501
RDS_FORMSERVEVENT_CHANGE.....	128
RDS_FORMSERVEVENT_CLICK.....	128
RDS_FORMSERVEVENT_DRAW.....	128
RDS_FORMSERVFUNCDATA.....	128
RDS_FORMVAL_2NDEDITENABLED.....	506
RDS_FORMVAL_AUXLISTITEM.....	508
RDS_FORMVAL_AUXLISTWIDTH.....	509
RDS_FORMVAL_CHECK.....	510
RDS_FORMVAL_ENABLED.....	511
RDS_FORMVAL_HKSHIFTS.....	512
RDS_FORMVAL_ITEMINDEX.....	513
RDS_FORMVAL_LIST.....	515
RDS_FORMVAL_MLHEIGHT.....	516
RDS_FORMVAL_MLRETURNS.....	517
RDS_FORMVAL_PBBEVEL.....	517
RDS_FORMVAL_PBHEIGHT.....	518
RDS_FORMVAL_RANGEMAX.....	520
RDS_FORMVAL_UPDOWNINC.....	521
RDS_FORMVAL_UPDOWNMAX.....	522
RDS_FORMVAL_UPDOWNMIN.....	524
RDS_FORMVAL_VALUE.....	525
RDS_FRESULT_DELAYED.....	577, 581
RDS_FRESULT_ERROR.....	577, 581
RDS_FRESULT_OK.....	577, 581
RDS_FUNCPARAMCAST.....	302
RDS_FUNCPARAMPVOID.....	304
RDS_FUNCPROVIDERLINK.....	129, 317
RDS_FUNCPROVIDERLINK_SUCCESS.....	305
RDS_FUNCTIONCALldata.....	36
RDS_FVERSION.....	144
RDS_FVERSIONBUILD.....	144
RDS_FVERSIONHIGH.....	144
RDS_FVERSIONLOW.....	144
RDS_GBD_NONE.....	222
RDS_GBD_USEVARS.....	222

RDS_GBD_USEZOOM.....	222
RDS_GEF_BLOCKNAME.....	257
RDS_GEF_VARNAME.....	257
RDS_GETFLAG.....	142
RDS_GFBOLD.....	283, 380
RDS_GFCHARSET.....	283, 380
RDS_GFCOLOR.....	283, 378, 380, 382
RDS_GFESCAPEMENT.....	283, 380
RDS_GFFONTALLHEIGHT.....	283, 380
RDS_GFFONTBASIC.....	283, 380
RDS_GFFONTSTYLES.....	283, 380
RDS_GFHEIGHT.....	283, 380
RDS_GFITALIC.....	283, 380
RDS_GFMODE.....	382
RDS_GFNAME.....	283, 379
RDS_GFS_BDIAGONAL.....	378
RDS_GFS_CROSS.....	378
RDS_GFS_DIAGCROSS.....	378
RDS_GFS_EMPTY.....	378
RDS_GFS_FDIAGONAL.....	378
RDS_GFS_HORIZONTAL.....	378
RDS_GFS_SOLID.....	378
RDS_GFS_VERTICAL.....	378
RDS_GFSIZE.....	283, 380
RDS_GFSTRIKEOUT.....	283, 380
RDS_GFSTYLE.....	378, 382
RDS_GFUNDERLINE.....	283, 380
RDS_GFWIDTH.....	382
RDS_GS_DISABLEAUTOCOMP.....	423
RDS_GS_DISABLEBLOCKAPPEARANCE.....	423
RDS_GS_DISABLEBUSPACK.....	423
RDS_GS_DISABLECONNAPPEARANCE.....	423
RDS_GS_DISABLECONNSTATE.....	423
RDS_GS_DISABLEDCLICK.....	423
RDS_GS_DISABLEDLLFUN.....	423
RDS_GS_DISABLEDLLOPTIONS.....	423
RDS_GS_DISABLEDRAWTYPE.....	423
RDS_GS_DISABLEEDITORPARAMS.....	424
RDS_GS_DISABLELAYERCHANGE.....	424
RDS_GS_DISABLEPICTURE.....	424
RDS_GS_DISABLEREMARKS.....	424
RDS_GS_DISABLESIZING.....	424
RDS_GS_DISABLEVARCHANGE.....	424
RDS_GS_DISABLEVARVALUES.....	424
RDS_GSIBAKFILESCount.....	158
RDS_GSICMDPARAMCOUNT.....	158
RDS_GSIDEFAULTPORT.....	158
RDS_GSIINSTSTRUCTCOUNT.....	158, 332
RDS_GSIMODIFIED.....	159
RDS_GSISAVELOADACTION.....	159
RDS_GSISTOPPING.....	159

RDS_GSITICKPARITY.....	159
RDS_GSIUIENABLED.....	159
RDS_GSIUNDOSIZE.....	159
RDS_GSPAPPEXE.....	160
RDS_GSPAPPPATH.....	160
RDS_GSPBAKFILEEXT.....	160
RDS_GSPBLOCKLIBPATH.....	160
RDS_GSPBLOCKPANELPATH.....	160
RDS_GSPDEFAULTHOST.....	160
RDS_GSPDLLPATH.....	160
RDS_GSPINCLUDEPATH.....	160
RDS_GSPINIPATH.....	160
RDS_GSPMODELSPATH.....	161
RDS_GSPSYSTEMFILE.....	161
RDS_GSPSYSTEMFULLPATH.....	161
RDS_GSPTEMPLATEPATH.....	161
RDS_GSPTEMPPPATH.....	161
RDS_HBCL_AUTODELETE.....	424
RDS_HBCL_BLOCKARRAY.....	425
RDS_HBCL_BLOCKCOUNT.....	426
RDS_HBCL_CLEAR.....	426
RDS_HBCL_CONNARRAY.....	427
RDS_HBCL_CONNCOUNT.....	428
RDS_HCE_ERR_ALLOC.....	416, 417
RDS_HCE_ERR_BADLINE.....	415, 417
RDS_HCE_ERR_BADOBJECT.....	415, 417
RDS_HCE_ERR_INVBLKBUS.....	415, 417
RDS_HCE_ERR_OK.....	415, 416
RDS_HCE_RESET.....	417
RDS_HINI_CREATESECTION.....	480
RDS_HINI_DELETEKEYLAST.....	481
RDS_HINI_DELETESECTION.....	481
RDS_HINI_GETLASTERROR.....	482
RDS_HINI_LOADFILE.....	482
RDS_HINI_RESET.....	483
RDS_HINI_SAVEBLOCKTEXT.....	483
RDS_HINI_SAVEFILE.....	484
RDS_HINI_SETTEXT.....	485
RDS_HOBJECT.....	23
RDS_HSTR_DEFENDOFFLINE.....	463, 466
RDS_HSTR_DEFENDOFTEXT.....	463, 466
RDS_HSTR_DEFUNKNOWNWORD.....	464, 471
RDS_HSTR_ENDOFFLINEID.....	466
RDS_HSTR_ENDOFTEXTID.....	466
RDS_HSTR_GETLASTWORD.....	467
RDS_HSTR_GETRESTOFTEXT.....	467
RDS_HSTR_IGNORECASE.....	468
RDS_HSTR_READDOUBLE.....	468
RDS_HSTR_READINT.....	469
RDS_HSTR_SETTEXT.....	470
RDS_HSTR_UNKNOWNID.....	471

RDS_HVAR_CLEARENAMES.....	444
RDS_HVAR_CLEARTYPEREN.....	444
RDS_HVAR_CLEARVARREN.....	445
RDS_HVAR_DELAUTO.....	445
RDS_HVAR_DELVAR.....	446
RDS_HVAR_FIINDEX.....	162, 439
RDS_HVAR_FALL.....	163, 439
RDS_HVAR_FALLNS.....	163, 439
RDS_HVAR_FALLPLAIN.....	163, 439
RDS_HVAR_FALLPLAINNS.....	163, 439
RDS_HVAR_FARRAYS.....	162, 439
RDS_HVAR_FARROFSTRUCT.....	439
RDS_HVAR_FCHAR.....	162, 439
RDS_HVAR_FDOUBLE.....	162, 439
RDS_HVAR_FFLOAT.....	162, 439
RDS_HVAR_FINT.....	163, 439
RDS_HVAR_FLOGICAL.....	163, 439
RDS_HVAR_FNOOFFSET.....	439
RDS_HVAR_FNOSTRUCTNAME.....	439
RDS_HVAR_FRUNTIME.....	163, 439
RDS_HVAR_FSHORT.....	163, 439
RDS_HVAR_FSIGNAL.....	163, 439
RDS_HVAR_FSTRING.....	163, 439
RDS_HVAR_FSTRUCT.....	163, 439
RDS_HVAR_GETAUTOCONN.....	447
RDS_HVAR_GETAUTOCOUNT.....	447
RDS_HVAR_GETAUTOMAIN.....	448
RDS_HVAR_GETFIELDCOUNT.....	449
RDS_HVAR_GETTYPENAME.....	449
RDS_HVAR_GETTYPESTRING.....	450
RDS_HVAR_GETVARFLAGS.....	450
RDS_HVAR_GETVARRANK.....	451
RDS_HVAR_RBADBLOCKTYPE.....	436
RDS_HVAR_REMPTYVARSET.....	436, 443
RDS_HVAR_RESET.....	451
RDS_HVAR_RNOBLKSIGNALS.....	437
RDS_HVAR_RNOTYPENAME.....	443
RDS_HVAR_ROK.....	436, 442
RDS_HVAR_ROKRENAMED.....	443
RDS_HVAR_RVARCHHECKERR.....	436
RDS_HVAR_SETTYPENAME.....	452
RDS_HVAR_SETVARFLAGS.....	452
RDS_INITSERVSIZE.....	142
RDS_INTVERSION.....	143
RDS_KALT.....	61
RDS_KBDFLAGS.....	61
RDS_KCTRL.....	61
RDS_KEYDATA.....	61
RDS_KSHIFT.....	61
RDS_LINEDESCRIPTION.....	129, 232
RDS_LNBEZIER.....	130

RDS_LNLINE.....	130
RDS_LS_ERROR.....	159
RDS_LS_LOADAUTOCOMP.....	160
RDS_LS_LOADCLIPBRD.....	80, 159
RDS_LS_LOADCONTENT.....	159
RDS_LS_LOADFROMFILE.....	80, 159
RDS_LS_LOADROOT.....	159
RDS_LS_LOADTAGGED.....	160
RDS_LS_LOADUNDO.....	159
RDS_LS_SAVEAUTOCOMP.....	109, 160
RDS_LS_SAVECLIPBRD.....	108, 159
RDS_LS_SAVECONTENT.....	108, 159
RDS_LS_SAVEROOT.....	108, 159
RDS_LS_SAVETAGGED.....	109, 160
RDS_LS_SAVETOFILE.....	108, 159
RDS_LS_SAVEUNDO.....	108, 159
RDS_MANUALDELETEDATA.....	79
RDS_MANUALINSERTDATA.....	80
RDS_MENU_CHECKED.....	356, 357, 361, 362
RDS_MENU_DISABLED.....	356, 357, 361, 362
RDS_MENU_DIVIDER.....	356, 357, 361
RDS_MENU_HIDDEN.....	357, 361, 362
RDS_MENU_SHORTCUT.....	357, 363
RDS_MENU_UNIQUECAPTION.....	357, 363
RDS_MENUFUNCDATA.....	63
RDS_MENUITEM.....	23, 356
RDS_MLEFTBUTTON.....	61, 66
RDS_MMIDDLEBUTTON.....	61, 66
RDS_MODELHANDLE.....	24, 93, 573
RDS_MOUSECAPTURE.....	30
RDS_MOUSEDATA.....	65
RDS_MOUSEFLAGS.....	61
RDS_MOVEDATA.....	81
RDS_MR_DRAG.....	81
RDS_MR_KEYBOARD.....	81
RDS_MR_SET.....	81, 240, 246
RDS_MR_UNDOREDO.....	81
RDS_MRRIGHTBUTTON.....	61, 66
RDS_NEEDSDLLREDRAW.....	30
RDS_NETACCEPTDATA.....	86
RDS_NETBLOCK.....	24, 87, 89, 393
RDS_NETCONNDATA.....	85
RDS_NETERR_ACCEPT.....	90
RDS_NETERR_CLIENTCONN.....	90
RDS_NETERR_DISCONNECT.....	90
RDS_NETERR_GENERAL.....	90
RDS_NETERR_NOBLOCK.....	90
RDS_NETERR_RECEIVE.....	90
RDS_NETERR_SEND.....	90
RDS_NETERRORDATA.....	89
RDS_NETRECEIVEDDATA.....	87

RDS_NETSEND_NOWAIT.....	390
RDS_NETSEND_SERVREPLY.....	86, 390
RDS_NETSEND_UDP.....	390
RDS_NETSEND_UPDATE.....	390
RDS_NETSTATION.....	23, 87, 89, 393
RDS_NOEXTERNFUNCPTRS.....	141
RDS_NOHOBJMACROS.....	141
RDS_NOVERSIONDEFINES.....	144
RDS_NOWINREFRESH.....	30, 256
RDS_OPENEDITORDATA.....	106
RDS_PAN_CAPTION.....	548
RDS_PAN_CLIENTHEIGHT.....	548
RDS_PAN_CLIENTWIDTH.....	549
RDS_PAN_F_BORDER.....	546
RDS_PAN_F_CAPTION.....	546
RDS_PAN_F_HIDDEN.....	546
RDS_PAN_F_MOVEABLE.....	546
RDS_PAN_F_NOBUTTON.....	546
RDS_PAN_F_PAINTMSG.....	546
RDS_PAN_F_SCALABLE.....	546
RDS_PAN_F_SIZEABLE.....	546
RDS_PAN_FLAGS.....	549
RDS_PAN_HEIGHT.....	550
RDS_PAN_LEFT.....	550
RDS_PAN_MAXCLHEIGHT.....	551
RDS_PAN_MAXCLWIDTH.....	552
RDS_PAN_MINCLHEIGHT.....	552
RDS_PAN_MINCLWIDTH.....	553
RDS_PAN_TOP.....	554
RDS_PAN_VISIBLE.....	554
RDS_PAN_WIDTH.....	555
RDS_PANDESCRIPTION.....	131, 547
RDS_PANOP_CREATE.....	55
RDS_PANOP_DESTROY.....	55
RDS_PANOP_MOVED.....	55
RDS_PANOP_PAINT.....	55
RDS_PANOP_RESIZED.....	55
RDS_PANOPERATION.....	55
RDS_PARRAYACCESSDATA.....	112
RDS_PBAR_ADDTOPOS.....	540
RDS_PBAR_HIDE.....	540
RDS_PBAR_MAX.....	541
RDS_PBAR_POSITION.....	541
RDS_PBAR_RESET.....	542
RDS_PBAR_SETCAPTION.....	543
RDS_PBAR_SHOW.....	543
RDS_PBLOCKDATA.....	29
RDS_PBLOCKDESCRIPTION.....	113
RDS_PBLOCKDIMENSIONS.....	117
RDS_PCOLORREF.....	25, 156, 197
RDS_PCOMPBLOCKOPDATA.....	95

RDS_PCOMPCANATTACHBLKDATA.....	97
RDS_PCOMPEXECFUNCDATA.....	102
RDS_PCOMPILEDATA.....	100
RDS_PCOMPMODELDATA.....	93
RDS_PCOMPMODELRENAMEDATA.....	98
RDS_PCOMPMODULEDATA.....	92
RDS_PCOMPPREPAREDATA.....	107
RDS_PCOMPSAVEBLOCKDATA.....	108
RDS_PCOMPSAVESYSTEMDATA.....	109
RDS_PCOMPSTRUCTCHGDATA.....	111
RDS_PCONNAPPEARANCE.....	118
RDS_PCONNDESCRIPTION.....	119
RDS_PCONTEXTPOPUPDATA.....	56
RDS_PDRAWDATA.....	57
RDS_PDYNVARLINK.....	121
RDS_PEDITORPARAMETERS.....	123
RDS_PEDITORTOOLBARS.....	126
RDS_PFINDBYEXTIDDATA.....	127
RDS_PFORMSERVFUNCDATA.....	128
RDS_PFUNCPROVIDERLINK.....	129
RDS_PFUNCTIONCALLDATA.....	36
RDS_PKEYDATA.....	61
RDS_PLINEDESCRIPTION.....	130
RDS_PMANUALDELETEDATA.....	79
RDS_PMANUALINSERTDATA.....	80
RDS_PMENUFUNCDATA.....	64
RDS_PMOUSEDATA.....	65
RDS_PMOVEDATA.....	81
RDS_PNETACCEPTDATA.....	86
RDS_PNETCONNDATA.....	85
RDS_PNETERRORDATA.....	89
RDS_PNETRECEIVEDDATA.....	87
RDS_POINTDESCRIPTION.....	133
RDS_POPENEDITORDATA.....	106
RDS_POPUPHINTDATA.....	70
RDS_PPANDESCRIPTION.....	131
RDS_PPANOPERATION.....	55
RDS_PPOINTDESCRIPTION.....	133
RDS_PPOPUPHINTDATA.....	70
RDS_PREMOTEMSGDATA.....	42
RDS_PRESIZEDATA.....	83
RDS_PSBADTYPE.....	134
RDS_PSBADVAR.....	134
RDS_PSERVFONTPARAMS.....	135
RDS_PSNORMAL.....	134
RDS_PT_STRINGTOTEXT.....	192
RDS_PT_TEXTTOSTRING.....	192
RDS_PT_VARTYPECHAR.....	163, 192
RDS_PT_VARTYPETEXT.....	192
RDS_PTBLOCK.....	134
RDS_PTBUS.....	134

RDS_PTBUSTOBLOCK.....	134
RDS_PTIMERDESCRIPTION.....	136
RDS_PTINTERNAL.....	134
RDS_PVARDESCRIPTION.....	138
RDS_PWINOPERATIONDATA.....	77
RDS_PWINREFRESHDATA.....	78
RDS_REMOTEMSGDATA.....	42
RDS_RESIZEDATA.....	83
RDS_SERV_FUNC_BODY.....	141
RDS_SERVFONTPARAMS.....	135, 279, 290
RDS_SERVFONTPARAMSNAMESIZE.....	135
RDS_SETFLAG.....	145
RDS_SFTAG_BUS.....	646, 656, 661
RDS_SFTAG_BUSPORT.....	646, 656, 661
RDS_SFTAG_CONNECTION.....	646, 656, 661
RDS_SFTAG_CONNSTYLES.....	646, 655, 661
RDS_SFTAG_DATATYPE.....	655
RDS_SFTAG_EOF.....	646, 662
RDS_SFTAG_INPUTBLOCK.....	646, 656, 661
RDS_SFTAG_OUTPUTBLOCK.....	646, 656, 661
RDS_SFTAG_ROOT.....	646, 656, 661
RDS_SFTAG_SIMPLEBLOCK.....	646, 656, 661
RDS_SFTAG_SYSTEM.....	646, 656, 661
RDS_SFTAG_TYPES.....	646, 655, 661
RDS_SHORT.....	25
RDS_SLP_AFTER.....	277
RDS_SLP_BEFORE.....	276
RDS_SLP_BOTTOM.....	276
RDS_SLP_TOP.....	276
RDS_SPP_REFRESH.....	250
RDS_SPP_RELATIVE.....	250
RDS_SRF_IGNORECASE.....	195
RDS_SRF_STDPATHS.....	195
RDS_SSIFASTTEXTSAVE.....	172
RDS_SSIWAITCURSOR.....	172
RDS_STARTSTOPDATA.....	46
RDS_STDICON_BLOCK.....	368
RDS_STDICON_DISABLEDCONN.....	368
RDS_STDICON_EYE.....	368
RDS_STDICON_GREENSQUARE.....	368
RDS_STDICON_PENCIL.....	368
RDS_STDICON_REDCIRCEXCLAM.....	368
RDS_STDICON_REDGEAR.....	368
RDS_STDICON_REDSQUARE.....	368
RDS_STDICON_REDTRIEXCLAM.....	368
RDS_STDICON_RUN.....	368
RDS_STDICON_STOP.....	368
RDS_STDICON_SYSTEM.....	368
RDS_STDICON_YELCIRCEXCLAM.....	368
RDS_STDICON_YELLOWGEAR.....	368
RDS_STDICON_YELLOWQUESTION.....	368

RDS_STDICON_YELLOWSSQUARE.....	368
RDS_SWO_CLOSE.....	77
RDS_SWO_OPEN.....	77
RDS_SYSCMD_CALCMODE.....	153
RDS_SYSCMD_EDITMODE.....	153
RDS_SYSCMD_LOADFILE.....	152
RDS_SYSCMD_LOADTEMPLATE.....	152
RDS_SYSCMD_MESSAGEBOX.....	153
RDS_SYSCMD_RESETCALC.....	153
RDS_SYSCMD_SAVEFILE.....	152
RDS_SYSCMD_STARTCALC.....	152
RDS_TFN_CHANGEEXT.....	196
RDS_TFN_EXCLUDEPATHBS.....	196
RDS_TFN_GETEXT.....	196
RDS_TFN_GETNAME.....	196
RDS_TFN_GETPATH.....	196
RDS_TFN_GETPATHNOBS.....	196
RDS_TFN_INCLUDEPATHBS.....	196
RDS_THREADAUX.....	178
RDS_THREADMAIN.....	178
RDS_TIMERDESCRIPTION.....	136, 299
RDS_TIMERF_FIXFREQ.....	138, 301
RDS_TIMERID.....	23, 47
RDS_TIMERM_DELETE.....	137, 301
RDS_TIMERM_LOOP.....	137, 301
RDS_TIMERM_STOP.....	137, 301
RDS_TIMERMASK_F.....	138
RDS_TIMERMASK_M.....	138
RDS_TIMERMASK_S.....	138
RDS_TIMERS_SIGNAL.....	137, 301
RDS_TIMERS_SYSTIMER.....	137, 301
RDS_TIMERS_TIMER.....	137, 301
RDS_TIMERS_WINREF.....	137, 301
RDS_TUNKNOWN.....	113, 120, 127
RDS_VAIC_MESSAGE.....	346
RDS_VAIC_SINGLE.....	346
RDS_VAIC_SINGLEMSG.....	346
RDS_VAIC_STOPCALC.....	346
RDS_VARCHECKFAILED.....	29
RDS_VARDESCRIPTION.....	138
RDS_VARFLAG_EXT_CHGNAME.....	432
RDS_VARFLAG_INPUT.....	138
RDS_VARFLAG_MENU.....	138
RDS_VARFLAG_ONEINDEX.....	139
RDS_VARFLAG_OUTPUT.....	138
RDS_VARFLAG_RUN.....	138
RDS_VARFLAG_SHOWNAME.....	139
RDS_VARTYPE_ARRAY.....	49, 139, 194
RDS_VARTYPE_CHAR.....	49, 193
RDS_VARTYPE_DOUBLE.....	49, 194
RDS_VARTYPE_FLOAT.....	49, 194

RDS_VARTYPE_INT.....	49, 194
RDS_VARTYPE_LOGICAL.....	49, 193
RDS_VARTYPE_RUNTIME.....	49, 194
RDS_VARTYPE_SHORT.....	49, 193
RDS_VARTYPE_SIGNAL.....	49, 193
RDS_VARTYPE_STRING.....	49, 194
RDS_VARTYPE_STRUCT.....	49, 194
RDS_VARTYPE_STRUCTEND.....	49
RDS_VHANDLE.....	23, 121, 138
RDS_WINOPERATIONDATA.....	77
RDS_WINREFRESHDATA.....	78
RDS_WINREFRESHWAITING.....	30, 256
rdsActivateOutputConnections.....	203
rdsAdditionalContextMenuItem.....	354
rdsAdditionalContextMenuItemEx.....	355
rdsAddLayer.....	270
rdsAddToDynStr.....	182
rdsAllocate.....	183
rdsAltConnAppearanceOp.....	204
rdsApplicationIsActive.....	145
rdsAtoD.....	184
rdsAtoI.....	184
rdsBadSystemTime.....	146
rdsBCLAddBlock.....	421
rdsBCLAddConn.....	422
rdsBCLCreateList.....	420
rdsBCLExecuteGroupSetDialog.....	423
rdsBCLGetBlockArray.....	428
rdsBCLGetConnArray.....	429
rdsBEUCreate.....	536
rdsBEUStore.....	538
rdsBlockByFullName.....	206
rdsBlockDataSyncCall.....	176
rdsBlockMessageBox.....	569
rdsBlockModalWinClose.....	146
rdsBlockModalWinOpen.....	147
rdsBlockOrConnByExtId.....	206
rdsBlockVarFromMem.....	319
rdsBlockVarToMem.....	320
rdsBringAppToFront.....	148
rdsBroadcastFuncCallsDelayed.....	305
rdsBroadcastFunctionCalls.....	307
rdsBroadcastFunctionCallsEx.....	308
rdsCalcProcessIsRunning.....	148
rdsCalcProcessNeverStarted.....	149
RDSCALL.....	25, 26, 91
rdsCallBlockFunction.....	310
rdsCallBlockFunctionDelayed.....	311
rdsCallColorDialog.....	197
rdsCallDirDialog.....	197
rdsCallerThreadType.....	178

rdsCallFileDialog.....	198
rdsCancelPaste.....	149
rdsCEAddBezier.....	409
rdsCEAddBlockPoint.....	410
rdsCEAddBusPoint.....	411
rdsCEAddChannel.....	412
rdsCEAddInternalPoint.....	413
rdsCEAddLine.....	414
rdsCEClearEditor.....	418
rdsCECreateBus.....	418
rdsCECreateConnBus.....	415
rdsCECreateConnection.....	419
rdsCECreateEditor.....	408
rdsCEEditBus.....	419
rdsCEEditConnBus.....	416
rdsCEEditConnection.....	419
rdsChangeMenuItem.....	356
rdsChangeRegWinTitle.....	150
rdsCheckBlockFunctionSupport.....	313
rdsCheckRectVisibility.....	254
rdsCheckSystemWindow.....	255
rdsClearRuntimeType.....	321
rdsCloseSystemWindow.....	255
rdsCommandObject.....	400
rdsCommandObjectEx.....	400
rdscompAttachDifferentModel.....	572
rdscompCompileModel.....	573
rdscompGetBlockModelData.....	574
rdscompGetModelBlock.....	574
rdscompGetModelData.....	575
rdscompGetModelDataByName.....	576
rdscompOpenBlockModelEditor.....	577
rdscompRenameModel.....	577
rdscompReturnModelName.....	579
rdscompReturnModelNameLabel.....	579
rdscompSetAltModelName.....	580
rdscompSetBlockModel.....	580
rdscompSetModelFunction.....	582
rdsCopyRuntimeType.....	322
rdsCopyVarArray.....	342
rdsCopyVarGeneral.....	323
rdsCountBlocks.....	208
rdsCreateAndSubscribeDV.....	347
rdsCreateBlockFromFile.....	209
rdsCreateDynamicVar.....	348
rdsCreateFullBlockNameString.....	210
rdsCreateVarDescriptionString.....	324
rdsCreateVarTypeText.....	325
rdsCSVCreate.....	556
rdsCSVGetItem.....	557
rdsCSVSetItem.....	558

RDSCTRL_BLOCKMSGDATA.....	583
RDSCTRL_FOSP_RECURSIVE.....	628
RDSCTRL_FOSP_SELF.....	628
RDSCTRL_GSC_TRANSFILE.....	647
RDSCTRL_GSC_TRANSPIPE.....	647
RDSCTRL_KEYF_ALT.....	682
RDSCTRL_KEYF_CTRL.....	682
RDSCTRL_KEYF_LEFT.....	682
RDSCTRL_KEYF_MIDDLE.....	682
RDSCTRL_KEYF_RIGHT.....	682
RDSCTRL_KEYF_SHIFT.....	682
RDSCTRL_KEYOP_DOWN.....	681
RDSCTRL_KEYOP_UP.....	681
RDSCTRL_LSFM_SAVEPROMPT.....	649
RDSCTRL_LSFM_TRANSFILE.....	649
RDSCTRL_LSFM_TRANSMAP.....	649
RDSCTRL_MENUITEM.....	584, 630
RDSCTRL_MENUTYPE_BLK.....	636
RDSCTRL_MENUTYPE_MAIN.....	636
RDSCTRL_MENUTYPE_SYS.....	636
RDSCTRL_MODE_CALC.....	631
RDSCTRL_MODE_EDIT.....	631
RDSCTRL_MODE_RUNNING.....	631
RDSCTRL_MODE_UNKNOWN.....	631
RDSCTRL_MOUSEF_ALT.....	683
RDSCTRL_MOUSEF_CTRL.....	683
RDSCTRL_MOUSEF_LEFT.....	683
RDSCTRL_MOUSEF_MIDDLE.....	683
RDSCTRL_MOUSEF_RIGHT.....	683
RDSCTRL_MOUSEF_SHIFT.....	683
RDSCTRL_MOUSEOP_DBL.....	683
RDSCTRL_MOUSEOP_DOWN.....	683
RDSCTRL_MOUSEOP_MOVE.....	683
RDSCTRL_MOUSEOP_UP.....	683
RDSCTRL_MSGEVENTDATA.....	586
RDSCTRL_NEWFILEDATA.....	587
RDSCTRL_PBLOCKMSGDATA.....	583
RDSCTRL_PMENUITEM.....	584, 630
RDSCTRL_PMSGEVENTDATA.....	586
RDSCTRL_PNEWFILEDATA.....	587
RDSCTRL_PPROGRESSDATA.....	588
RDSCTRL_PROGRESSDATA.....	588
RDSCTRL_PSAVEFILEDATA.....	589
RDSCTRL_PSETTINGS.....	590
RDSCTRL_PZOOMRECT.....	591, 678
RDSCTRL_RETURNSTRING.....	604
RDSCTRL_SAVEFILEDATA.....	589
RDSCTRL_SERV_FUNC_BODY.....	599
RDSCTRL_SERV_FUNC_EXTERNAL.....	599
RDSCTRL_SET_ENABLECALC.....	590
RDSCTRL_SET_ENABLEEDIT.....	590

RDSCTRL_SET_ENABLEMAINWIN.....	590
RDSCTRL_SET_ENABLEOPTIONS.....	590
RDSCTRL_SET_ENABLEPARAMED.....	590
RDSCTRL_SET_ENABLERUN.....	590
RDSCTRL_SET_ENABLESYSWIN.....	590
RDSCTRL_SET_ENABLEUI.....	590
RDSCTRL_SETTINGS.....	590, 612, 614
RDSCTRL_TAGGED_DBLBUF.....	657, 659
RDSCTRL_TAGGED_SHAREDMMEM.....	651, 657
RDSCTRL_ZOOMRECT.....	591
RDSCTRL_ZOOMRECTFLAGS_100.....	591, 677
rdsetrlBlockExtIdByName.....	615
rdsetrlBlockMenuClick.....	616
rdsetrlBlockMenuClickEx.....	617
rdsetrlBlockNameByExtId.....	618
rdsetrlBringAppToFront.....	619
rdsetrlCallBlockFunction.....	619
rdsetrlCallBlockFunctionEx.....	621
rdsetrlCheckModalWindows.....	622
rdsetrlClearLog.....	686
rdsetrlClearSystem.....	623
rdsetrlClose.....	599
rdsetrlCloseAllSysExceptRoot.....	623
rdsetrlCloseAllWindows.....	624
rdsetrlCloseModalWindows.....	624
rdsetrlCloseSysWindow.....	625
rdsetrlConnect.....	600
rdsetrlCreateLink.....	600
rdsetrlDeleteExchangeMemory.....	643
rdsetrlDeleteLink.....	601
rdsetrlDisconnect.....	602
rdsetrlEnableCalcMode.....	606
rdsetrlEnableEditMode.....	606
rdsetrlEnableEvents.....	665
rdsetrlEnableLog.....	687
rdsetrlEnableOptions.....	607
rdsetrlEnablePropEdit.....	608
rdsetrlEnableRun.....	609
rdsetrlEnableRunInterface.....	609
rdsetrlEnableSubsystemWindows.....	610
rdsetrlEnableUI.....	611
rdsetrlEnableWinRefresh.....	625
rdsetrlEndBlockByBlockLoad.....	643
rdsetrlEndBlockByBlockSave.....	644
RDSCTRLEVENT_BLOCKMSG.....	593
RDSCTRLEVENT_CALCMODE.....	594
RDSCTRLEVENT_CALCSTART.....	594
RDSCTRLEVENT_CALCSTOP.....	595
RDSCTRLEVENT_CONNCLOSED.....	595
RDSCTRLEVENT_EDITMODE.....	596
RDSCTRLEVENT_LOADREQ.....	596

RDSCTRLEVENT_NEWFILE.....	597
RDSCTRLEVENT_NEWFILE_LOAD.....	587
RDSCTRLEVENT_NEWFILE_NEW.....	587
RDSCTRLEVENT_NEWFILE_TEMPLATE.....	587
RDSCTRLEVENT_PROGRESS.....	597
RDSCTRLEVENT_SAVEFILE.....	598
rdsetrlFindBlock.....	626
rdsetrlFindOpSetProviders.....	628
rdsetrlGetBlockByBlockSavePiece.....	645
rdsetrlGetBlockVarValue.....	629
rdsetrlGetGeneralSettings.....	612
rdsetrlGetMenuItemData.....	630
rdsetrlGetMode.....	631
rdsetrlGetModFlag.....	632
rdsetrlGetSystemContent.....	647
rdsetrlGetViewportParams.....	670
rdsetrlGetViewportSysArea.....	671
rdsetrlGetVPMouseLevel.....	672
rdsetrlIsConnected.....	602
rdsetrlLeave.....	603
rdsetrlListBlocks.....	632
rdsetrlLoadSystemFromFile.....	648
rdsetrlLoadSystemFromMem.....	649
rdsetrlLoadSystemTagged.....	650
rdsetrlLoadSystemTaggedEx.....	651
rdsetrlLogString.....	687
rdsetrlMinimizeApp.....	634
rdsetrlNoDirectLoad.....	652
rdsetrlNoDirectSave.....	653
rdsetrlReadBlockMenuItems.....	635
rdsetrlRegisterBlockMsgCallback.....	665
rdsetrlRegisterEventMessage.....	667
rdsetrlRegisterEventStdCallback.....	668
rdsetrlReleaseViewport.....	673
rdsetrlResetCalc.....	636
rdsetrlRestoreApp.....	637
rdsetrlRestoreConnection.....	603
rdsetrlSaveSystemTagged.....	654
rdsetrlSaveSystemTaggedEx.....	656
rdsetrlSaveSystemTaggedMem.....	658
rdsetrlSaveSystemToFile.....	660
rdsetrlSetAutoSave.....	612
rdsetrlSetBlockByBlockLoadPiece.....	661
rdsetrlSetBlockVarValue.....	637
rdsetrlSetCalcMode.....	638
rdsetrlSetControllerName.....	639
rdsetrlSetEditMode.....	640
rdsetrlSetExitMode.....	613
rdsetrlSetGeneralSettings.....	614
rdsetrlSetLogFile.....	688
rdsetrlSetModFlag.....	640

rdsetrlSetPath.....	604
rdsetrlSetProgressDelay.....	662
rdsetrlSetString.....	641
rdsetrlSetStringCallback.....	604
rdsetrlSetViewport.....	674
rdsetrlSetViewportParams.....	675
rdsetrlSetViewportRect.....	676
rdsetrlSetViewportZoomRect.....	676
rdsetrlSetViewportZoomRectEx.....	678
rdsetrlShowMainWindow.....	614
rdsetrlStartBlockByBlockLoad.....	663
rdsetrlStartBlockByBlockSave.....	664
rdsetrlStartCalc.....	642
rdsetrlStopCalc.....	642
rdsetrlUnregisterEvent.....	669
rdsetrlUpdateViewport.....	678
rdsetrlViewportBlockAtPos.....	679
rdsetrlViewportFit.....	680
rdsetrlViewportKeyboard.....	681
rdsetrlViewportMouse.....	682
rdsetrlViewportSystem.....	684
rdsetrlVPPopupHint.....	685
rdsdebugBlockInfo.....	570
rdsdebugLogString.....	571
rdsDeleteBlock.....	210
rdsDeleteBlockTimer.....	298
rdsDeleteConnection.....	211
rdsDeleteDVByLink.....	349
rdsDeleteDynamicVar.....	350
rdsDeleteObject.....	401
rdsDeleteSystemState.....	250
rdsDtoA.....	185
rdsDuplicateBlock.....	212
rdsDynStrCat.....	186
rdsDynStrCopy.....	187
rdsEnableCommandQueue.....	151
rdsEnableMenuItem.....	358
rdsEnableWindowRefresh.....	255
rdsEnumBlocks.....	212
rdsEnumConnectedBlocks.....	214
rdsEnumConnectedBlocksByVar.....	216
rdsEnumDynVarSubscribers.....	351
rdsExecMenuItem.....	359
rdsExecuteCommand.....	151
rdsExecutePrintDialog.....	200
rdsExecutesRemoteOpsSet.....	395
rdsFindBlockVar.....	326
rdsFindCmdParam.....	153
rdsFindNextConnectedLine.....	218
rdsFindStructVar.....	327
rdsFontTextToStruct.....	290

rdsForceBlockRedraw.....	219
rdsFORMAddEdit.....	492
rdsFORMAddTab.....	501
rdsFORMClear.....	527
rdsFORMCreate.....	491
rdsFORMEnableControl.....	528
rdsFORMEnableSidePanel.....	502
rdsFORMGetBool.....	528
rdsFORMGetDouble.....	529
rdsFORMGetEnableCheck.....	530
rdsFORMGetInt.....	530
rdsFORMGetString.....	531
rdsFORMSetBool.....	532
rdsFORMSetComboList.....	532
rdsFORMSetDouble.....	533
rdsFORMSetEnableCheck.....	534
rdsFORMSetInt.....	534
rdsFORMSetMultilineHeight.....	535
rdsFORMShowModal.....	536
rdsFORMShowModalEx.....	503
rdsFORMShowModalServ.....	504
rdsFree.....	187
rdsGetAppInstance.....	154
rdsGetAppWindowHandle.....	154
rdsGetBlockDescription.....	220
rdsGetBlockDimensions.....	221
rdsGetBlockDimensionsEx.....	221
rdsGetBlockFlags.....	222
rdsGetBlockLink.....	223
rdsGetBlockTimerDescr.....	299
rdsGetBlockVar.....	328
rdsGetBlockVarBase.....	329
rdsGetBlockVarDefValueStr.....	330
rdsGetChildBlockByName.....	224
rdsGetCmdParam.....	155
rdsGetCmdParamCount.....	156
rdsGetConnAppearance.....	225
rdsGetConnDescription.....	225
rdsGetConnDimensions.....	226
rdsGetConnStyleAppearance.....	227
rdsGetCustomColors.....	156
rdsGetEditorFont.....	257
rdsGetEditorParameters.....	258
rdsGetEditorToolBars.....	259
rdsGetEditorWindowFlags.....	259
rdsGetFirstBlock.....	228
rdsGetFirstConn.....	228
rdsGetFullFilePath.....	188
rdsGetHugeDouble.....	157
rdsGetIOBlockByVarName.....	229
rdsGetIOBlockLink.....	230

rdsGetLayerConfigName.....	270
rdsGetLayerId.....	271
rdsGetLayerIdInConfig.....	271
rdsGetLayerName.....	272
rdsGetLayerParams.....	273
rdsGetLineDescription.....	231
rdsGetMainWindow.....	158
rdsGetMouseObjectId.....	232
rdsGetNextBlock.....	233
rdsGetNextConn.....	234
rdsGetObjectArray.....	402
rdsGetObjectDouble.....	403
rdsGetObjectDoubleP.....	404
rdsGetObjectInt.....	404
rdsGetObjectStr.....	405
rdsGetParentBlock.....	235
rdsGetPictureObjectId.....	236
rdsGetPointDescription.....	237
rdsGetRelFilePath.....	190
rdsGetRemoteControllerName.....	396
rdsGetRemoteControllerString.....	396
rdsGetRootSystem.....	238
rdsGetRuntimeTypeData.....	330
rdsGetScreenCoords.....	260
rdsGetStructVar.....	331
rdsGetSystemInt.....	158
rdsGetSystemPath.....	160
rdsGetTextWord.....	291
rdsGetTextWordDyn.....	293
rdsGetTopWindowBlock.....	261
rdsGetVarArrayAccessData.....	343
rdsGetVarArrayParams.....	344
rdsGetVarField.....	332
rdsHasRemoteController.....	397
rdsHideAllEditorToolBars.....	261
rdsINIClearText.....	485
rdsINICreateSection.....	486
rdsINICreateTextHolder.....	473
rdsINIDeleteSection.....	486
rdsINIDeleteValue.....	487
rdsINILoadFile.....	487
rdsINIOpenSection.....	474
rdsINIReadBool.....	488
rdsINIReadDouble.....	475
rdsINIReadDoubleP.....	476
rdsINIReadInt.....	477
rdsINIReadString.....	477
rdsINISaveBlockText.....	489
rdsINISaveFile.....	489
rdsINISetText.....	490
rdsINIWriteBool.....	490

rdsINIWriteDouble.....	478
rdsINIWriteInt.....	479
rdsINIWriteString.....	480
rdsInputString.....	200
rdsIsRoot.....	238
rdsIsValidVarName.....	161
rdsItoA.....	191
rdsListVarTypes.....	162
rdsLoadSystemState.....	251
rdsLockBlockData.....	178
rdsMainWindowVisible.....	164
rdsMakeUniqueBlockName.....	239
rdsMessageBox.....	201
rdsModalWindowExists.....	164
rdsModalWindowMustClose.....	165
rdsMoveBlock.....	240
rdsNetBroadcastData.....	389
rdsNetCloseConnection.....	391
rdsNetConnect.....	391
rdsNetSendData.....	392
rdsNetServer.....	394
rdsNotifyDynVarSubscribers.....	352
rdsOpenSystemWindow.....	261
rdsOpenSystemWindowEx.....	262
rdsPANCreate.....	545
rdsPANGetDescr.....	547
rdsParentIsRoot.....	240
rdsPBARCreate.....	538
rdsPBARIncrement.....	543
rdsPBARSetPos.....	544
rdsPBARShow.....	544
rdsProcessText.....	192
rdsQueueCallBlockFunction.....	313
rdsReadBlockData.....	277
rdsReadColorText.....	294
rdsReadFontText.....	295
rdsReadHexText.....	297
rdsReadLineStyleText.....	298
rdsRefreshBlockWindows.....	263
rdsRegisterContextMenuItem.....	360
rdsRegisterContextMenuItemEx.....	361
rdsRegisterFuncProvider.....	315
rdsRegisterFunction.....	316
rdsRegisterMenuItem.....	362
rdsRegisterWindow.....	166
rdsRegWinActivateNotify.....	167
rdsRemoteControllerCall.....	397
rdsRemoteReply.....	398
rdsRenameBlock.....	241
rdsReportTextLoadError.....	278
rdsResetSystemState.....	252

rdsResizeVarArray.....	344
rdsRestartBlockTimer.....	299
rdsRunWithoutEvents.....	168
rdsSaveSystemState.....	252
rdsScrollWindowToBlock.....	263
rdsScrollWindowToRect.....	264
rdsSelectBlock.....	241
rdsServiceVersion.....	169
rdsSetBlockAltNameText.....	242
rdsSetBlockComment.....	243
rdsSetBlockFlags.....	243
rdsSetBlockLayer.....	244
rdsSetBlockModel.....	245
rdsSetBlockRect.....	246
rdsSetBlockSetupFuncName.....	247
rdsSetBlockTimer.....	300
rdsSetBlockVarDefValueByCur.....	333
rdsSetBlockVarDefValueStr.....	334
rdsSetBlockVarFlags.....	335
rdsSetConnAppearance.....	247
rdsSetConnLayer.....	248
rdsSetCurLayerConfig.....	274
rdsSetCurLayerConfigByName.....	274
rdsSetDebugText.....	572
rdsSetEditorToolBars.....	265
rdsSetEditorWindowFlags.....	265
rdsSetExclusiveCalc.....	169
rdsSetHintText.....	249
rdsSetLayerParams.....	275
rdsSetLayerPosition.....	276
rdsSetMenuItemOptions.....	363
rdsSetModifiedFlag.....	171
rdsSetObjectDouble.....	406
rdsSetObjectInt.....	407
rdsSetObjectStr.....	407
rdsSetPointPosition.....	249
rdsSetRuntimeType.....	335
rdsSetSystemInt.....	171
rdsSetSystemUpdate.....	172
rdsSetSystemWindowBounds.....	267
rdsSetSystemWindowCaption.....	267
rdsSetSystemWindowRect.....	268
rdsSetZoomPercent.....	269
rdsShowBlockPanelTab.....	173
rdsShowMainWindow.....	174
rdsStartCalc.....	174
rdsStopBlockTimer.....	302
rdsStopCalc.....	175
rdsSTRAddKeyword.....	460
rdsSTRAddKeywordsArray.....	461
rdsSTRCreateTextReader.....	460

rdsSTRGetDoubleWord.....	471
rdsSTRGetIntWord.....	472
rdsSTRGetWord.....	462
rdsStringReplace.....	194
rdsSTRSetTextToRead.....	472
rdsStructToFontText.....	279
rdsSubscribeToDynamicVar.....	353
rdsSubscribeToFuncProvider.....	317
rdsSystemInEditMode.....	175
rdsTMPCreateEmptyFile.....	386
rdsTMPCreateFileSet.....	387
rdsTMPDeleteFile.....	388
rdsTMPDeleteFileSet.....	388
rdsTMPRememberFileName.....	388
rdsTransformFileName.....	196
rdsUnlockAndCall.....	180
rdsUnlockBlockData.....	181
rdsUnregisterFuncProvider.....	317
rdsUnregisterMenuItem.....	364
rdsUnregisterWindow.....	175
rdsUnsubscribeFromDynamicVar.....	354
rdsUnsubscribeFromFuncProvider.....	318
rdsUpdateExtIdsRange.....	176
rdsVarArrayIndexCheck.....	345
rdsVarUsesStructType.....	336
rdsVSAddAutoConn.....	430
rdsVSAddTypeRename.....	431
rdsVSAddVar.....	432
rdsVSAddVarByDescr.....	433
rdsVSAddVarByTypeText.....	434
rdsVSAddVarRename.....	435
rdsVSApplyToBlock.....	436
rdsVSClearEditor.....	453
rdsVSClearTypeRenames.....	454
rdsVSClearVarRenames.....	454
rdsVSCreateByDescr.....	437
rdsVSCreateEditor.....	430
rdsVSCreateFromBlock.....	438
rdsVSDeleteAutoConn.....	455
rdsVSDeleteVar.....	455
rdsVSExecuteEditor.....	438
rdsVSFindAutoConn.....	440
rdsVSGetAutoConn.....	456
rdsVSGetAutoCount.....	456
rdsVSGetAutoMain.....	457
rdsVSGetFieldCount.....	457
rdsVSGetStructName.....	458
rdsVSGetStructRank.....	458
rdsVSGetVarDefValueStr.....	441
rdsVSGetVarDescription.....	441
rdsVSInstallStruct.....	442

rdsVSSetStructName.....	459
rdsVSSetVarFlags.....	443
rdsVSUsesStructType.....	444
rdsWriteBlockData.....	280
rdsWriteBlockDataText.....	280
rdsWriteColorText.....	281
rdsWriteFontText.....	282
rdsWriteHexText.....	285
rdsWriteLineStyleText.....	286
rdsWriteWordDoubleText.....	287
rdsWriteWordStringText.....	288
rdsWriteWordValueText.....	289
rdsXGArc.....	365
rdsXGChord.....	366
rdsXGDrawBlockPicture.....	367
rdsXGDrawStdIcon.....	368
rdsXGEllipse.....	369
rdsXGFillRect.....	369
rdsXGFontSizeToHeight.....	370
rdsXGGetStdIconSize.....	371
rdsXGGetTextSize.....	371
rdsXGGetVisibleRect.....	372
rdsXGInvertRect.....	373
rdsXGLineTo.....	373
rdsXGMoveTo.....	374
rdsXGPie.....	374
rdsXGPolygon.....	375
rdsXGPolyline.....	375
rdsXGRectangle.....	376
rdsXGRoundRect.....	377
rdsXGSetBrushStyle.....	377
rdsXGSetClipRect.....	378
rdsXGSetFont.....	379
rdsXGSetFontByParStr.....	381
rdsXGSetLogFont.....	381
rdsXGSetPenStyle.....	382
rdsXGSetPixel.....	384
rdsXGTextOut.....	384
rdsXGTextRect.....	385
rdsXGTriangle.....	385
RECT.....	25
\$DLL\$.....	189
\$INI\$.....	189, 690
\$LIB\$.....	189
\$MODEL\$\$.....	189
\$PANEL\$.....	189
\$RDSINCLUDE\$.....	189
\$TEMP\$.....	189, 691
\$WINTMP\$.....	189, 691