Руководство программиста

Считыватель бесконтактный
Руководство программиста

Версия 6.3

Москва

Содержание

Часть I Общая информация 12

1 Перечень нормативных документов 12
2 Ссылки на документацию 12
3 Список сокращений 12
4 Предупреждение 13

Часть II Взаимодействие считывателя с управляющим устройством 16

1 Интерфейсы 16
USB 16
RS232 17
RS485 17
2 Протокол обмена данными 17
Кадр запроса 18
Кадр запроса шифрованный 18
Кадр ответа 19
Кадр ответа шифрованный 20
Коды завершения команды 22
Байтстаффинг 25
Контрольная сумма кадра 25
Многобайтовые переменные 26

Часть III Система команд 28

1 Команды управления считывателем 28
0x0E Выдача состояния считывателя 28
0x10 Инициализация микросхемы-считывателя 29
0x04 Включение микросхемы-считывателя 29
0x64 Выдача версии считывателя 29
0x22 Выдача серийного номера микросхемы-считывателя 29
0x20 Сброс/выключение электромагнитного поля (RF) 30
0x51 Переключение режима электромагнитного поля (RF) 30
0x05 Подача звукового сигнала 30
0x07 Управление светодиодом 31
0x0F Изменение скорости обмена по COM-порту 31
0x70 Очистка flash-памяти с ключами 32
0x79 Запись во flash-память блока с ключом 32
0x6E Проверка заполнения блока flash-памяти считывателя 32
0x6F (Сброс) аутентификация считывателя 32
2 Команды управления картами типа A стандарта ISO 14443 34
0x43 Активация карты типа A, находящейся в состоянии Idle 34
0x1D Перевод активной карты типа A в состояние Halt 34
0x44 Активация карты типа A, находящейся в состоянии Halt 34
0x35 Чтение информации ATS из карты 34
0x36 Установка протокола и параметров работы с картой 35
3 Команды управления картами типа B стандарта ISO 14443 35
0x56 Активация карт типа B, находящихся в состоянии Idle 35
0x54 Установка параметров протокола в данном сеансе связи с картой типа B 36

0x55 Перевод активной карты типа B в состояние Halt 36
0x57 Активация карт типа B, находящихся в состоянии Halt 36
4 Команда управления метками стандарта ISO 15693 37
0x30 Инвентаризация меток 15693 37
0x32 Перевод метки 15693 в состояние QUIET 38
5 Команды обмена данными с картой Mifare Classic 38
Вычисление абсолютного номера блока 38
0x16 (обратная совместимость) Кодирование ключа 38
0x18 (обратная совместимость) Аутентификация ключом, заданным в команде 39
0x14 Аутентификация ключом, заданным в команде 39
0x17 Запись ключа в EEPROM считывателя 39
0x15 Аутентификация ключом, находящимся в EEPROM считывателя 40
0x19 Чтение блока 40
0x1A Запись блока 40
0x1B Операция Value 40
0x1C Персонализация UID 41
0x26 Изменение нагрузки антенны карт 41
6 Команды обмена данными с картой Mifare UltraLight (C) 41
0x25 (устарела) Чтение одной страницы 41
0x19 Чтение четырёх страниц 41
0x1E Запись страницы 42
0x2C Запись ключа аутентификации 42
0x2D Аутентификация карты 42
7 Команды обмена данными с картой Mifare Plus 42
Таблица формирования параметра типа значения 42
Таблица формирования параметра режима защиты передачи данных 43
Таблица допустимых режимов защиты передачи данных 43
0xA8 Запись данных персонализации в карту 44
0xAA Персонализация карты 44
0xA0 Управление аутентификацией 44
0xA6 Чтение нескольких блоков SL2 45
0xA7 Запись нескольких блоков SL2 45
0xA4 Чтение данных 45
0xA5 Запись данных 45
0xA1 Прибавление значения 46
0xA1 Вычитание значения 46
0xA1 Запись данных из буфера переноса в блок 46
0xA1 Прибавление значения с последующей записью данных из буфера переноса в блок 47
0xA1 Вычитание значения с последующей записью данных из буфера переноса в блок 47
0xA1 Запись данных блока значения в буфер переноса 47
0xA2 Начальный и промежуточный запрос поддержки виртуальных карт 48
0xA2 Завершающий запрос поддержки виртуальных карт 48
0xA3 Выбор виртуальной карты 48
0xA3 Снятие выбора виртуальной карты 49
0xA9 Поиск релейной атаки 49
8 Обмен данными с картой Mifare DES Fire 49
Таблица константных значений 49
Формат команд для управления считывателем при помощи микропрограммы внешнего контроллера.
9 Команда непосредственного обмена данными с картой 50
0x48 Непосредственный обмен с картой 50
10 Команды конфигурации устройств на шине RS485 51
0x7A Чтение адреса устройства 51
0x77 Запись адреса устройства 51
11 Команды для работы со считывателем NFC663 51
0x11 Активация устройства 51

0x12 Отмена выбора ... 52
0x13 Обмен ... 52
0x40 Сброс протокола ... 53
0x33 Запрос атрибутов ... 53
0x34 Выбор параметров ... 53
0x1F Проверка присутствия ... 54
0x41 Установка конфигурации ... 54
0x42 Чтение конфигурации ... 54
0x28 Чтение серийного номера (NFCID3) ... 54
0x23 Чтение заданного количества байт из указанного адреса EEPROM ... 54
0x24 Запись одного байта данных в указанный адрес EEPROM ... 55
0x2A Запись нескольких байт данных в указанную страницу EEPROM ... 55

Часть IV Библиотека Clscrfl.dll 58

1 Функции управления интерфейсом ... 58
CLSCRF_Create ... 58
CLSCRF_Destroy ... 58
CLSCRF_Open ... 58
CLSCRF_OpenRS ... 59
CLSCRF_OpenUSB ... 59
CLSCRF_IsOpened ... 60
CLSCRF_Close ... 60
CLSCRF_GetIOTimeout ... 60
CLSCRF_SetIOTimeout ... 61
CLSCRF_GetLastInternalError ... 61
2 Функции управления считывателем ... 61
CLSCRF_GetState ... 62
CLSCRF_Mfrc_On ... 62
CLSCRF_Mfrc_Off ... 63
CLSCRF_Get_Mfrc_Version ... 63
CLSCRF_Get_Mfrc_Serial_Number ... 63
CLSCRF_Mfrc_Rf_Off_On ... 64
CLSCRF_Mfrc_Set_Rf_Mode ... 64
CLSCRF_Sound ... 65
CLSCRF_Led ... 65
CLSCRF_UART_Baudrate ... 65
CLSCRF_EraseFlash ... 66
CLSCRF_WriteFlashValue ... 66
CLSCRF_CheckFlashValueFilled ... 66
3 Функции управления защищенным режимом ... 67
CLSCRF_Crypto_SaveAESKeysToFile ... 67
CLSCRF_Crypto_GenerateAndSaveAESKeysToFile ... 67
CLSCRF_Crypto_LoadAESKeysFromFile ... 68
CLSCRF_Crypto_WriteFlash_AESKeys ... 68
CLSCRF_Crypto_AuthenticateReader ... 68
CLSCRF_Crypto_SetEncryptionMode ... 69
CLSCRF_Crypto_GetEncryptionMode ... 69
4 Функции работы со списком карт ... 69
CLSCRF_Cards_Add ... 69
CLSCRF_Cards_GetFirst ... 70
CLSCRF_Cards_GetLast ... 70
CLSCRF_Cards_Get ... 70
CLSCRF_Cards_GetCount ... 70
CLSCRF_Cards_Clear ... 70
CLSCRF_Cards_SetCurrent ... 71
CLSCRF_Cards_GetCurrent ... 71
CLSCRF_Cards_Search ... 71

CLSCRF_Cards_CreateNew_A ... 71
CLSCRF_Cards_CreateNew_B ... 72
CLSCRF_Cards_CreateNew15693 ... 72
CLSCRF_Cards_CreateNewEPCUID ... 73
CLSCRF_Cards_Delete ... 73
CLSCRF_CARD ... 73
5 Функции управления картами типа A стандарта ISO 14443 ... 75
CLSCRF_Activate_Idle_A ... 75
CLSCRF_Halt_A ... 75
CLSCRF_Activate_Wakeup_A ... 75
CLSCRF_ISO14443A_4_RATS ... 76
CLSCRF_ISO14443A_4_PPS ... 76
6 Функции управления картами типа B стандарта ISO 14443 ... 77
CLSCRF_Activate_Idle_B ... 77
CLSCRF_Attrib_B ... 77
CLSCRF_Halt_B ... 78
CLSCRF_Activate_Wakeup_B ... 78
7 Функции управления метками стандарта ISO 15693 ... 79
CLSCRF_FindAllTags_15693 ... 79
CLSCRF_Inventory_15693 ... 80
CLSCRF_Stay_Quiet_15693 ... 81
CLSCRF_Select_15693 ... 81
CLSCRF_ResetToReady_15693 ... 82
8 Функции обмена данными с картами Mifare Classic ... 82
(обратная совместимость) CLSCRF_MifareStandard_HostCodeKey ... 82
(обратная совместимость) CLSCRF_MifareStandard_AuthKey ... 83
CLSCRF_MifareStandard_AuthKeyDirect ... 83
CLSCRF_MifareStandard_WriteKeyToE2 ... 83
CLSCRF_MifareStandard_AuthE2 ... 84
CLSCRF_MifareStandard_Read ... 84
CLSCRF_MifareStandard_Write ... 85
CLSCRF_MifareStandard_Decrement ... 85
CLSCRF_MifareStandard_Increment ... 86
CLSCRF_MifareStandard_Restore ... 86
CLSCRF_MifareStandard_EV1_PersonalizeUid ... 86
CLSCRF_MifareStandard_EV1_SetLoadModulationType ... 87
9 Функции обмена данными с картами Mifare Ultralight (C) ... 87
CLSCRF_MifareUltralight_Read ... 87
CLSCRF_MifareUltralight_Write ... 88
CLSCRF_MifareUltralightC_WriteKey ... 88
CLSCRF_MifareUltralightC_Authenticate ... 88
10 Функции обмена данными с метками стандарта ISO 15693 ... 88
CLSCRF_ReadSingleBlock_15693 ... 89
CLSCRF_WriteSingleBlock_15693 ... 89
CLSCRF_LockBlock_15693 ... 90
CLSCRF_ReadMultipleBlocks_15693 ... 90
CLSCRF_WriteAFI_15693 ... 91
CLSCRF_LockAFI_15693 ... 92
CLSCRF_WriteDSFID_15693 ... 92
CLSCRF_LockDSFID_15693 ... 93
CLSCRF_GetSystemInfo_15693 ... 93
CLSCRF_GetMultipleBSS_15693 ... 94
CLSCRF_SetEAS_15693 ... 95
CLSCRF_ResetEAS_15693 ... 95
CLSCRF_LockEAS_15693 ... 96
CLSCRF_EASAlarm_15693 ... 96

11 Функции обмена данными с картами Mifare Plus 97

CLSCRF_MifarePlus_WritePersoExplicit 97
CLSCRF_MifarePlus_CommitPerso 97
CLSCRF_MifarePlus_Authenticate 98
CLSCRF_MifarePlus_MultiBlockRead 98
CLSCRF_MifarePlus_MultiBlockWrite 98
CLSCRF_MifarePlus_ReadData 99
CLSCRF_MifarePlus_WriteData 99
CLSCRF_MifarePlus_Increment 100
CLSCRF_MifarePlus_Decrement 100
CLSCRF_MifarePlus_Transfer 101
CLSCRF_MifarePlus_IncrementTransfer 101
CLSCRF_MifarePlus_DecrementTransfer 102
CLSCRF_MifarePlus_Restore 102
CLSCRF_MifarePlus_VirtualCardSupport 102
CLSCRF_MifarePlus_VirtualCardSupportLast 103
CLSCRF_MifarePlus_VirtualCardSelect 103
CLSCRF_MifarePlus_VirtualCardDeselect 104
CLSCRF_MifarePlus_ProximityCheck 104

12 Функции обмена данными с картами Mifare DES Fire (EV1) 104

CLSCRF_MifareDesFire_Authenticate 104
CLSCRF_MifareDesFire_SetTransferType 105
CLSCRF_MifareDesFire_ChangeKeySettings 105
CLSCRF_MifareDesFire_GetKeySettings 105
CLSCRF_MifareDesFire_ChangeKey 106
CLSCRF_MifareDesFire_GetKeyVersion 106
CLSCRF_MifareDesFire_CreateApplication 106
CLSCRF_MifareDesFire_DeleteApplication 107
CLSCRF_MifareDesFire_GetApplicationIDs 107
CLSCRF_MifareDesFire_GetDFNames 108
CLSCRF_MifareDesFire_SelectApplication 108
CLSCRF_MifareDesFire_FormatPICC 108
CLSCRF_MifareDesFire_GetVersion 108
CLSCRF_MifareDesFire_FreeMemory 109
CLSCRF_MifareDesFire_SetConfiguration 109
CLSCRF_MifareDesFire_GetCardUID 109
CLSCRF_MifareDesFire_GetFileIDs 110
CLSCRF_MifareDesFire_GetISOFileIDs 110
CLSCRF_MifareDesFire_GetFileSettings 110
CLSCRF_MifareDesFire_ChangeFileSettings 111
CLSCRF_MifareDesFire_CreateStdDataFile 111
CLSCRF_MifareDesFire_CreateBackupDataFile 111
CLSCRF_MifareDesFire_CreateValueFile 112
CLSCRF_MifareDesFire_CreateLinearRecordFile 113
CLSCRF_MifareDesFire_CreateCyclicRecordFile 113
CLSCRF_MifareDesFire_DeleteFile 114
CLSCRF_MifareDesFire_ReadData 114
CLSCRF_MifareDesFire_WriteData 115
CLSCRF_MifareDesFire_GetValue 115
CLSCRF_MifareDesFire_Credit 115
CLSCRF_MifareDesFire_Debit 116
CLSCRF_MifareDesFire_LimitedCredit 116
CLSCRF_MifareDesFire_WriteRecord 116
CLSCRF_MifareDesFire_ReadRecords 117
CLSCRF_MifareDesFire_ClearRecordFile 117
CLSCRF_MifareDesFire_CommitTransaction 117
CLSCRF_MifareDesFire_AbortTransaction 118
CLSCRF_DESFIRE_HW_SW_INFO 118

CLSCRF_DESFIRE_MORE_INFO ... 118
CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS ... 119
CLSCRF_DESFIRE_FILE_SETTINGS ... 119
CLSCRF_DESFIRE_CONFIGURATION ... 120
CLSCRF_DESFIRE_DFNAME ... 121
CLSCRF_DESFIRE_LIMITED_CREDIT_ENABLED ... 121
CLSCRF_DESFIRE_PICC_MASTER_KEY_SETTINGS ... 122
CLSCRF_DESFIRE_APPLICATION_MASTER_KEY_SETTINGS ... 122
CLSCRF_DESFIRE_NEW_APPLICATION_KEY_SETTINGS ... 123
CLSCRF_DESFIRE_MASTER_KEY_SETTINGS ... 123
CLSCRF_DESFIRE_KEY_DATA ... 123
CLSCRF_DESFIRE_MASTER_KEY_SETTINGS_AND_LENGTH ... 124
13 Функции работы со считывателем NFC663 ... 124
CLSCRF_NFC663_ActivateCard ... 125
CLSCRF_NFC663_Deselect ... 125
CLSCRF_NFC663_Exchange ... 126
CLSCRF_NFC663_ResetProtocol ... 126
CLSCRF_NFC663_AttributeRequest ... 126
CLSCRF_NFC663_ParameterSelect ... 127
CLSCRF_NFC663_PresenceCheck ... 128
CLSCRF_NFC663_SetConfig ... 128
CLSCRF_NFC663_GetConfig ... 128
CLSCRF_NFC663_GetSerialNo ... 129
CLSCRF_NFC663_E2_Read ... 129
CLSCRF_NFC663_E2_Write ... 129
CLSCRF_NFC663_E2_WritePage ... 129
PHPAL_I18092MPI_DATARATE_106 ... 130
PHPAL_I18092MPI_DATARATE_212 ... 130
PHPAL_I18092MPI_DATARATE_424 ... 130
PHPAL_I18092MPI_FRAMESIZE_64 ... 130
PHPAL_I18092MPI_FRAMESIZE_128 ... 130
PHPAL_I18092MPI_FRAMESIZE_192 ... 130
PHPAL_I18092MPI_FRAMESIZE_254 ... 131
PHPAL_I18092MPI_DESELECT_DSL ... 131
PHPAL_I18092MPI_DESELECT_RLS ... 131
PH_EXCHANGE_DEFAULT ... 131
PH_EXCHANGE_TXCHAINING ... 131
PH_EXCHANGE_RXCHAINING ... 131
PH_EXCHANGE_RXCHAINING_BUFSIZE ... 132
PH_EXCHANGE_TX_CRC ... 132
PH_EXCHANGE_RX_CRC ... 132
PH_EXCHANGE_PARITY ... 132
PH_EXCHANGE_LEAVE_BUFFER_BIT ... 132
PH_EXCHANGE_BUFFERED_BIT ... 132
PHPAL_I18092MPI_CONFIG_PACKETNO ... 133
PHPAL_I18092MPI_CONFIG_DID ... 133
PHPAL_I18092MPI_CONFIG_NAD ... 133
PHPAL_I18092MPI_CONFIG_WT ... 133
PHPAL_I18092MPI_CONFIG_FSL ... 133
PHPAL_I18092MPI_CONFIG_MAXRETRYCOUNT ... 133
14 Функции демонстрационные для работы со стандартом NFC ... 133
Функции работы с NDEF сообщениями ... 134
CLSCRF_NFC_NDEF_Message_Create ... 134
CLSCRF_NFC_NDEF_Message_GetRecordsCount ... 134
CLSCRF_NFC_NDEF_Message_GetDataSize ... 134
CLSCRF_NFC_NDEF_Message_GetData ... 135
CLSCRF_NFC_NDEF_Message_AddRecord ... 135
CLSCRF_NFC_NDEF_Message_AddRecordEmpty ... 135

CLSCRF_NFC_NDEF_Message_AddRecordText ... 135
CLSCRF_NFC_NDEF_Message_AddRecordUri ... 136
CLSCRF_NFC_NDEF_Message_AddRecordMimeMedia ... 136
CLSCRF_NFC_NDEF_Message_AddRecordMimeMediaText ... 137
CLSCRF_NFC_NDEF_Message_GetRecord ... 137
CLSCRF_NFC_NDEF_Message_Destroy ... 137
Функции работы с NDEF записями ... 138
CLSCRF_NFC_NDEF_Record_Create ... 138
CLSCRF_NFC_NDEF_Record_GetDataSize ... 138
CLSCRF_NFC_NDEF_Record_GetTnf ... 138
CLSCRF_NFC_NDEF_Record_SetTnf ... 138
CLSCRF_NFC_NDEF_Record_GetTypeLength ... 139
CLSCRF_NFC_NDEF_Record_GetType ... 139
CLSCRF_NFC_NDEF_Record_GetTypeStr ... 139
CLSCRF_NFC_NDEF_Record_SetType ... 140
CLSCRF_NFC_NDEF_Record_SetTypeStr ... 140
CLSCRF_NFC_NDEF_Record_GetPayloadLength ... 140
CLSCRF_NFC_NDEF_Record_GetPayload ... 140
CLSCRF_NFC_NDEF_Record_GetPayloadText ... 141
CLSCRF_NFC_NDEF_Record_GetPayloadUri ... 141
CLSCRF_NFC_NDEF_Record_SetPayload ... 141
CLSCRF_NFC_NDEF_Record_GetIdLength ... 142
CLSCRF_NFC_NDEF_Record_GetId ... 142
CLSCRF_NFC_NDEF_Record_GetIdStr ... 142
CLSCRF_NFC_NDEF_Record_SetId ... 143
CLSCRF_NFC_NDEF_Record_Destroy ... 143
Функции работы с протоколом LLCP ... 143
CLSCRF_NFC_LLCP_Create ... 143
CLSCRF_NFC_LLCP_SetTimeout ... 143
CLSCRF_NFC_LLCP_SetBaudrate ... 144
CLSCRF_NFC_LLCP_SetGeneralInformation ... 144
CLSCRF_NFC_LLCP_SetNFCID3Information ... 144
CLSCRF_NFC_LLCP_ServerOpen ... 145
CLSCRF_NFC_LLCP_ServerReceive ... 145
CLSCRF_NFC_LLCP_ServerClose ... 145
CLSCRF_NFC_LLCP_ClientOpen ... 145
CLSCRF_NFC_LLCP_ClientTransmit ... 145
CLSCRF_NFC_LLCP_ClientClose ... 146
CLSCRF_NFC_LLCP_Destroy ... 146
Функции работы с протоколом SNEP ... 146
CLSCRF_NFC_SNEP_Create ... 146
CLSCRF_NFC_SNEP_Receive ... 146
CLSCRF_NFC_SNEP_Transmit ... 147
CLSCRF_NFC_SNEP_Destroy ... 147
Функции работы с NFC метками ... 147
CLSCRF_NFC_ForumTags_SupposeTypeISO14443A ... 147
CLSCRF_NFC_ForumTags_BeginType4 ... 148
CLSCRF_NFC_ForumTags_Format ... 148
CLSCRF_NFC_ForumTags_Read ... 148
CLSCRF_NFC_ForumTags_Write ... 149
Примеры работы с демонстрационными функциями NFC ... 149
Приём сообщения NDEF с текстовой записью ... 152
Чтение сообщения из меток типа NFC Forum Tag 2, 4 ... 152
Получение сообщения от удалённого устройства в режиме P2P ... 153
Разбор принятого сообщения ... 154
Отправка сообщения NDEF с текстовой записью ... 155
Формирование отправляемого сообщения ... 155
Запись сообщения в метки типа NFC Forum Tag 2, 4 ... 156

Отправка сообщения на удалённое устройство в режиме P2P 157
15 Функции непосредственного обмена данными с картой 158
CLSCRF_DirectIO_Card 158
16 Функции конфигурации устройств на шине RS485 159
CLSCRF_GetIOAddress 159
CLSCRF_SetIOAddress 159
CLSCRF_ReadDeviceAddress 159
CLSCRF_WriteDeviceAddress 160
17 Обработка ошибок при использовании библиотеки Clscrfl.dll 160

Часть V Рекомендации по работе с картами в протоколе T=CL 164

1 Активация карты типа A из состояния IDLE 164
Переключить режим Rf на тип A скорость 106. 164
Активировать карту типа A из состояния IDLE. 164
Запросить параметры протокола карты. 164
Установить текущие параметры протокола обмена с картой типа A. 165
Переключить режим Rf на тип A скорость 106. 165
2 Активация карты типа A из состояния HALT 165
Переключить режим Rf на тип A скорость 106. 166
Активировать карту типа A из состояния HALT. 166
Запросить параметры протокола карты. 166
Установить текущие параметры протокола обмена с картой типа A. 167
Переключить режим Rf на тип A скорость 424. 167
3 Активация карты типа B из состояния IDLE 167
Переключить режим Rf на тип B скорость 106. 167
Активировать карту типа B из состояния IDLE. 167
Установить текущие параметры протокола обмена с картой типа B. 168
Переключить режим Rf на тип B скорость 106. 168
4 Активация карты типа B из состояния HALT 168
Переключить режим Rf на тип B скорость 106. 168
Активировать карту типа B из состояния HALT. 168
Установить текущие параметры протокола обмена с картой типа B. 169
Переключить режим Rf на тип B скорость 212. 169
5 Деактивация карты 169
Деактивировать карту, работающую в протоколе T=CL. 169

МикроЭМ

Руководство программиста

Часть I

1 Общая информация

Настоящее руководство предназначено для ознакомления с протоколом обмена данными между хостом и считывателем RFID-карт UEM и содержит сведения, необходимые для разработки прикладного программного обеспечения.

1.1 Перечень нормативных документов

ISO 14443, части 1-4
ISO 15693, части 1-3
ISO 18000, часть 3

1.2 Ссылки на документацию

Документация по картам NXP (http://www.nxp.com/products/identification_and_security/smart_card_ics/mifare_smart_card_ics/#products)

Докментация по микросхеме-считывателю MFRC531 (http://www.nxp.com/products/identification_and_security/reader_ics/nfc_contactless_reader_ics/series/MFRC531.html)

Докментация по микросхеме-считывателю CLRC632 (http://www.nxp.com/products/identification_and_security/reader_ics/nfc_contactless_reader_ics/series/CLRC632.html)

1.3 Список сокращений

APDU – формат команды протокола T=CL (ISO 7816-4)
ATQ – результат операции REQA (ISO 14443-3)
ATS – ответ на операцию SELECT (допустимые параметры протокола T=CL)
ATTRIB – команда выбора карты типа B (ISO 14443-3)
CLA – первый байт команды APDU
Dr – характеристика скорости потока данных от считывателя к карте
Ds – характеристика скорости потока данных от карты к считывателю
HLTA – команда перевода карты типа A в состояние HALT
HLTB – команда перевода карты типа B в состояние HALT
INS – второй байт команды APDU
Lc – длина передаваемых данных в команде APDU
Le – длина ожидаемых данных в ответе на команду APDU
P1 – третий байт (параметр) команды APDU
P2 – четвертый байт (параметр) команды APDU
PICC – карта с бесконтактным интерфейсом (RFID-карта)
PPS – запрос на установку параметров протокола T=CL
PUPI – псевдоуникальный номер (идентификатор) карты типа B

RATS – запрос на получение ATS
REQA – запрос на активацию карт типа A из состояния IDLE
REQB – запрос на активацию карт типа B из состояния IDLE
Rf – излучаемая считывателем радиочастота
SAK – результат (подтверждение) операции SELECT
T=CL – протокол обмена данными с RFID-картами (ISO 14443-4)
UID – уникальный номер (идентификатор) карты типа A
WUPA – запрос на активацию карт типа A из состояния HALT
WUPB – запрос на активацию карт типа B из состояния HALT
NFC (Near Field Communication) – технология беспроводной передачи данных по принципу индуктивной связи
NDEF (NFC Data Exchange Format) – формат данных для передачи сообщений по технологии NFC.
P2P (Peer To Peer) – обмен данными по принципу "точка-точка" между двумя устройствами
LLCP (Logical Link Control Protocol) – протокол уплавления логическими соединениями между устройствами, работающими по протоколу ISO18092 (NFC P2P)
SNEP (Simple NDEF Exchange Protocol) – протокол обмена сообщениями NFC в формате NDEF

1.4 Предупреждение

Внимание! Перед началом работы с картами, внимательно изучите документацию на эти карты: неосторожное использование команд может повлечь выход карт из строя.

МикроЭМ

Руководство программиста

Часть II

2 Взаимодействие считывателя с управляющим устройством

2.1 Интерфейсы

Считыватель работает под управлением компьютера или какого-либо другого управляющего устройства (мастера). В зависимости от исполнения считыватель подключается к мастеру либо по шине USB, либо к последовательному порту RS232 или RS485. Протокол обмена – единый для всех интерфейсов.

2.1.1 USB

Для работы со считывателем по интерфейсу USB необходимо установить драйвер, входящий в комплект поставки. Чтобы установить UEM USB-драйвер считывателя для операционной системы Windows XP, необходимо выполнить следующее.

1. Вставьте компакт-диск, поставляемый с UEM USB считывателем, в привод CD-ROM, затем вставьте считыватель в гнездо USB компьютера. Автоматически начнет работать мастер установки нового оборудования. В правом нижнем углу экрана вашего компьютера при обнаружении считывателя появится сообщение "Найдено новое оборудование".
2. Если в диалоговом окне "Мастер нового оборудования" появится предложение подключиться к узлу Windows Update для поиска программного обеспечения, выберите "Нет, не в этот раз" и нажмите "Далее".
3. В диалоговом окне "Мастер нового оборудования" выберите "Установка из указанного места", затем нажмите "Далее".
4. Убедитесь, что выбрано "Выполнить поиск наиболее подходящего драйвера в указанных местах". Снимите флаг "Поиск на сменных носителях", установите флаг "Включить следующее место поиска", затем нажмите "Обзор".
5. В диалоговом окне "Обзор папок" найдите и выберите папку Driver на поставляемом со считывателем компакт-диске (например, E:\Driver). Нажмите "OK".
6. В диалоговом окне "Мастер нового оборудования" нажмите "Далее".
7. В диалоговом окне "Установка оборудования" с предостережением о тестировании программного обеспечения нажмите "Все равно продолжить".

8. В диалоговом окне "Мастер нового оборудования" нажмите "Готово".
9. Чтобы убедиться в том, что драйвер установлен успешно, достаточно увидеть UEM USB SmartCard Reader в Диспетчере устройств. Для этого нажмите правой кнопкой мыши на иконку "Мой компьютер" на рабочем столе, в появившемся меню выберите "Свойства" и на вкладке "Оборудование" нажмите "Диспетчер устройств". Затем нажмите значок "+"(плюс) рядом с группой "Устройства чтения смарт-карт", чтобы раскрыть ее список.

2.1.2 RS232

Обмен данными (передача запросов и ответов) по последовательному порту осуществляется с параметрами 8N1 и начальной скоростью 9600 бод. После установления связи скорость обмена можно повысить. Допустимы следующие скорости обмена: 9600, 14400, 19200, 38400, 57600, 115200 бод.

2.1.3 RS485

Шина RS485 – полудуплексная. На ней допускается подключение до 32 считывателей, но в любой момент времени на ней должно работать не более одного передатчика. Устройства на шине RS485 должны иметь разные адреса. Значение адреса устройства находится в диапазоне от 0 до 255. При изготовлении устройство получает адрес 0. На команду по адресу 0 устройство отвечает независимо от своего адреса. Команды по адресу 0 допустимы лишь для интерфейсов USB и RS232, а также в том в случае, когда на шине RS485 присутствует только одно устройство. Для назначения устройству нового адреса допускается подключать его к любому доступному интерфейсу. Параметры обмена по последовательному порту – аналогичные параметрам интерфейса RS232.

2.2 Протокол обмена данными

Обмен данными со считывателем осуществляется кадрами переменной длины в режиме «запрос-ответ». Инициатором обмена может быть только мастер системы (компьютер). На запрос, составленный в соответствии с протоколом, считыватель обязан выдать ответ, если адрес запроса нулевой или совпадает с адресом считывателя.

Помимо открытой передачи данных, в считыватель поддерживает работу с шифрованным каналом обмена.

2.2.1 Кадр запроса

Поле	Длина, байт	Значение
Стартовое условие	1	0xFD
Адрес	1	Для USB и RS232 всегда 0
Идентификатор кадра	1	
Код команды	1	См. далее Набор команд
Параметры и данные команды	XX	Зависят от команды
Контрольная сумма кадра	2	CRC16
Стоповое условие	1	0xFE

1. При обмене данными без ошибок идентификаторы кадра у любой пары следующих друг за другом запросов должны отличаться.
2. После получения запроса считыватель проверяет правильность приема путем анализа контрольной суммы кадра. При неправильном приеме устройство с адресом 0 выдает код завершения команды в кадре ответа, равный MI_CRCERR, команду не выполняет и выходных данных в кадре ответа не выдает. Устройство с адресом, отличным от 0, при неправильном приеме команду не выполняет и ответа не выдает.
3. Если Адрес в запросе отличен от 0 и не совпадает с адресом устройства, команда не выполняется и ответ не выдается.

2.2.2 Кадр запроса шифрованный

Поле	Длина, байт	Значение
Стартовое условие	1	0xFD
Адрес	1	Для USB и RS232 всегда 0
Идентификатор кадра	1	
Признак шифрации	1	0x00
Код операции и данные команды	16*x	Зависят от команды, передаются в виде 128-

		битных шифро-блоков EncData[16*n]
Контрольная сумма кадра	2	CRC16
Стоповое условие	1	0xFE

1. При формировании пакета запроса, формируется вектор, состоящий из:
1) байта кода запроса OpCode[1];
2) данных запроса Data[XX];
3) 32-разрядной контрольной суммы от кода запроса и данных апроса (полином 0x04C11DB7) CRC32({OpCode[1] || Data[XX]});
4) байта 0x80;
5) последовательности нулевых байт {0x00, 0x00, …}, которыми дополняется вектор до длины, кратной 16 байтам.

Этот вектор шифруется выбранными для режима защищенного обмена 128-битными сессионным ключом SessionKey[16] и вектором инициализации IV[16] в режиме AES128-CBC.

Таким образом получается следующая формула вычисления шифроблоков: EncData[16*n] = EncAES128CBC({OpCode[1] || Data[XX] || CRC32({OpCode[1] || Data[XX]}), 0x80, {0x00, 0x00, …}}, SessionKey[16], IV[16])

2. При обмене данными без ошибок идентификаторы кадра у любой пары следующих друг за другом запросов должны отличаться.

3. После получения запроса считыватель проверяет правильность приема путем анализа контрольной суммы CRC16 кадра. При неправильном приеме устройство с адресом 0 выдает код завершения команды в кадре ответа, равный MI_CRCERR, команду не выполняет и выходных данных в кадре ответа не выдает. Устройство с адресом, отличным от 0, при неправильном приеме команду не выполняет и ответа не выдает.

4. Если Адрес в запросе отличен от 0 и не совпадает с адресом устройства, команда не выполняется и ответ не выдается.

5. После дешифрации последовательности блоков, производится проверка их целостности путем вычисления вложенной контрольной суммы CRC32 и ее сверки. Вектор инициализации обновляется для использования в последующих операциях (де-)шифрования.

6. При возникновении каких-либо ошибок приемо-передачи при шифрованном режиме обмена, считыватель остается в этом режиме до его перезагрузки по питанию.

2.2.3 Кадр ответа

1. Для формирования ответа считывателем должны быть приняты как минимум стартовый и стоповый байты, идентификатор кадра, код команды и контрольная сумма. Также должны отсутствовать ошибки байтстаффинга в

пределах кадра. Иначе никакие ответы не формируются и компьютер должен повторить запрос по окончании таймаута, выставив соответствующий признак повторного запроса (прежний идентификатор кадра).

Поле	Длина, байт	Значение
Стартовое условие	1	0xFD
Адрес устройства	1	
Идентификатор кадра	1	Повторение идентификатора из запроса
Код команды	1	Повторение кода команды из запроса
Код завершения команды	1	См. далее Коды завершения команды
Выходные данные	XX	Зависят от команды, могут отсутствовать
Контрольная сумма кадра	2	CRC16
Стоповое условие	1	0xFE

2. Считыватель вправе игнорировать новый запрос если он не успел полностью обработать предыдущий. Гарантируется что считыватель готов к приему новой команды к моменту окончания передачи первого кадра ответа на последний запрос.

3. Считыватель вправе игнорировать запрос, если количество байтов между стартовым и стоповым условиями больше размера его буфера обмена (300 байтов).

4. Если компьютер не получил валидный ответ, запрос передается повторно по окончании таймаута с прежним идентификатором кадра.

5. Если считыватель получил команду, совпадающую с предыдущей по идентификатору кадра и коду команды, и выполнил предыдущую, то повторную он не выполняет, а только повторяет ответ на нее.

2.2.4 Кадр ответа шифрованный

1. Для формирования ответа считывателем должны быть приняты как минимум стартовый и стоповый байты, идентификатор кадра, код команды и контрольная сумма. Также должны отсутствовать ошибки байтстаффинга в пределах кадра. Иначе никакие ответы не формируются и компьютер должен повторить запрос по окончании таймаута, выставив соответствующий признак повторного запроса (прежний идентификатор кадра).

Поле	Длина	Значение

	, байт	
Стартовое условие	1	0xFD
Адрес устройства	1	
Идентификатор кадра	1	Повторение идентификатора из запроса
Признак шифрации	1	0x00
Выходные данные	16*m	Шифрованные блоки EncData[16*m]
Контрольная сумма кадра	2	CRC16
Стоповое условие	1	0xFE

2. Полученный ряд шифрованных блоков расшифровывается при помощи выбранных для защищенного режима, 128-битных, сессионного ключа SessionKey[16] и вектора инициализации IV[16], в режиме AES128-CBC. Затем производится поиск с конца расшифрованного вектора байта 0x80. Стоящие перед ним 4 байта интерпретируются как CRC32 (полином 0x04C11DB7) и проверяются путем вычисления контрольной суммы предыдущей последовательности. В результате расшифрования получается вектор данных, содержащий последовательно код операции, код ответа и набор данных ответа. При этом вектор инициализации обновляется для последующих операций.

Таким образом получается следующая формула расшифровки блоков: {OpCode[1] || Data[XX] || CRC32({OpCode[1] || Data[XX]}), 0x80, {0x00, 0x00, …} = DecAES128CBC(EncData[16*m]}, SessionKey[16], IV[16])

3. Считыватель вправе игнорировать новый запрос если он не успел полностью обработать предыдущий. Гарантируется что считыватель готов к приему новой команды к моменту окончания передачи первого кадра ответа на последний запрос.

4. Считыватель вправе игнорировать запрос, если количество байтов между стартовым и стоповым условиями больше размера его буфера обмена (300 байтов).

5. Если компьютер не получил валидный ответ, запрос передается повторно по окончании таймаута с прежним идентификатором кадра.

6. Если считыватель получил команду, совпадающую с предыдущей по идентификатору кадра и коду команды, и выполнил предыдущую, то повторную он не выполняет, а только повторяет ответ на нее.

7. При возникновении каких-либо ошибок приемо-передачи при шифрованном режиме обмена, считыватель остается в этом режиме до его перезагрузки по

питанию.

2.2.5 Коды завершения команды

Description	Dec	Hex	Описание
MI_OK	(0)	0x00	Операция выполнена успешно
MI_NOTAGERR	(-1)	0xFF	Карта не отвечает
MI_CRCERR	(-2)	0xFE	Контрольная сумма неверна
MI_AUTHERR	(-4)	0xFC	Неверное значение ключа
MI_PARITYERR	(-5)	0xFB	Ошибка четности
MI_CODEERR	(-6)	0xFA	Неверный код завершения
MI_SERNRERR	(-8)	0xF8	Неверный байт целостности UID
MI_KEYERR	(-9)	0xF7	Ошибка в процессе загрузки ключей
MI_NOTAUTHERR	(-10)	0xF6	Сектор не аутентифицирован
MI_BITCOUNTERR	(-11)	0xF5	Неверное количество принятых битов
MI_BYTECOUNTERR	(-12)	0xF4	Неверное количество принятых байтов
MI_WRITEERR	(-15)	0xF1	Ошибка записи данных
MI_OVFLERR	(-19)	0xED	Переполнение буфера обмена
MI_FRAMINGERR	(-21)	0xEB	Неверные старт/стоповые условия
MI_UNKNOWN_COMMAND	(-23)	0xE9	Неизвестная операция
MI_COLLERR	(-24)	0xE8	Коллизия
MI_RESETERR	(-25)	0xE7	Ошибка сброса считывателя
MI_INTERFACEERR	(-26)	0xE6	Ошибка инициализации считывателя

MI_NOBITWISEANTICOLL	(-28)	0xE4	Ошибка побитовой антиколлизии
MI_CODINGERR	(-31)	0xE1	Неверное кодирование подтверждения
UEM_HARD_IS_ABSENT	(-40)	0xD8	Отсутствует необходимый компонент
UEM_UNKNOWN_CMD	(-41)	0xD7	Неизвестная команда
UEM_CMD_NOT_SUPPORTED	(-42)	0xD6	Данной версией команда не поддерживается
UEM_MFRC_WRONG_MODE	(-43)	0xD5	Требуется установить другой режим
UEM_WRONG_CRYPTO_MODE	(-44)	0xD4	Сбой синхронизации режимов
UEM_FLASH_ERASING_REQUIRED	(-45)	0xD3	Требуется очистка flash-памяти считывателя
UEM_KEY_IS_ABSENT	(-46)	0xD2	Ключ отсутствует во flash
UEM_TRANSCEIVER_FAILED	(-47)	0xD1	Приемопередатчик неисправен
UEM_ICODE_STACK_OVERFLOW	(-48)	0xD0	Переполнение стека идентификаторов
UEM_HALTB_ERR	(-49)	0xCF	Карта B не перешла в режим HALT
MI_CASCLEVEX	(-52)	0xCC	Ошибка 10-байтового UID
MI_BAUDRATE_NOT_SUPPORTED	(-54)	0xCA	Данная скорость обмена не поддерживается
UEM_SAM_TIMEOUT	(-55)	0xC9	Превышение времени ожидания ответа
UEM_SAM_APDU_ERR	(-56)	0xC8	Ошибка формата APDU
UEM_SAM_INVALID_CARD_MAC	(-57)	0xC7	Неверный MAC карты
UEM_SAM_AUTHENTICATION_ERR	(-58)	0xC6	Ошибка аутентификации
UEM_SAM_BYTECOUNTERR	(-59)	0xC5	Неверное число принятых байт
MI_WRONG_PARAMETER_VALUE	(-60)	0xC4	Недопустимое значение параметра

I1_TIMEOUT	(-70)	0xBA	Превышение времени ожидания
MI_MFP_GENERAL_MANIPULATE_ERR	(-81)	0xAF	Общая ошибка работы с картой
MI_MFP_INVALID_CARD_MAC	(-82)	0xAE	Неверный MAC в ответе карты
MI_MFP_LENGTH_ERROR	(-84)	0xAC	Неверно задана длина
MI_MFP_NO_STATE_FOR_COMMAND	(-85)	0xAB	Для текущего состояния карты данная команда не допустима
MI_MFP_NOT_EXISTING_BLOCK	(-86)	0xAA	Блок с указанным номером не существует
MI_MFP_INVALID_BLOCK_NUMBER	(-87)	0xA9	Неверно указан номер блока, попытка аутентификации карты в режиме SL0
MI_MFP_INVALID_MAC	(-88)	0xA8	MAC в запросе не верен
MI_MFP_COMMAND_OVERFLOW	(-89)	0xA7	Слишком много операций записи или чтения за текущую сессию или транзакцию
MI_MFP_AUTHENTICATION_ERR	(-90)	0xA6	Не выполнены условия доступа Требуемый блок не существует Запрошена операция над блоком-значением, но указанный блок таковым не является
MI_NY_IMPLEMENTED	(-100)	0x9C	Неизвестная ошибка
MI_RECBUF_OVERFLOW	(-112)	0x90	Переполнение буфера приемника
MI_VALERR	(-124)	0x84	Неверный формат блока VALUE
UEM_UNSUPPORTED_PARAMETER	(-125)	0x85	Параметр не поддерживается
UEM_UNSUCCESS_CHAINING	(-126)	0x86	Цепочка приема не завершена
UEM_TEMPERATURE_ERROR	(-127)	0x87	Перегрев микросхемы считывателя
UEM_UNKNOWN_ERROR	(-128)	0x88	Неизвестная ошибка

2.2.6 Байтстаффинг

Если между стартовым и стоповым условием встречаются специальные символы (0xFD, 0xFE, 0xFF), то они кодируются в соответствии с таблицей байтстаффинга:

Специальный символ	Кодирование
0xFD	0xFF 0x02
0xFE	0xFF 0x01
0xFF	0xFF 0x00

2.2.7 Контрольная сумма кадра

1. Контрольная сумма кадра (CRC) есть средство контроля его целостности. CRC считается над всеми полями кадра кроме стартового и стопового байтов и поля самой CRC.
2. Контрольная сумма кадра в данном протоколе реализована согласно стандартам CCITT X.25 или ISO 3309 или RFC1331 (PPP). Ниже приведен предельно упрощенный алгоритм реализации применительно к одному байту.
3. Для кадра имеем начальное значение CRC = 0xFFFF, байты считаются начиная с первого. По окончании расчета CRC инвертируется.
4. Вычисление CRC при передаче производится ДО проведения байтстаффинга, а при приеме сначала производится байтстаффинг, а потом производится проверка CRC.

```
unsigned short CRC;
void DoCRC( unsigned char D )
{
    unsigned char i;
    unsigned short w;
    w = ( D ^ CRC) & 0xFF;
    i = 8;
    do {
        if ( w & 1 )
        {
            w >>= 1;
            w ^= 0x8408;
```

```
            }
            else
            {
                    w >>= 1;
            }
    } while( --i );
    CRC = w ^ ( CRC >> 8 );
}
```

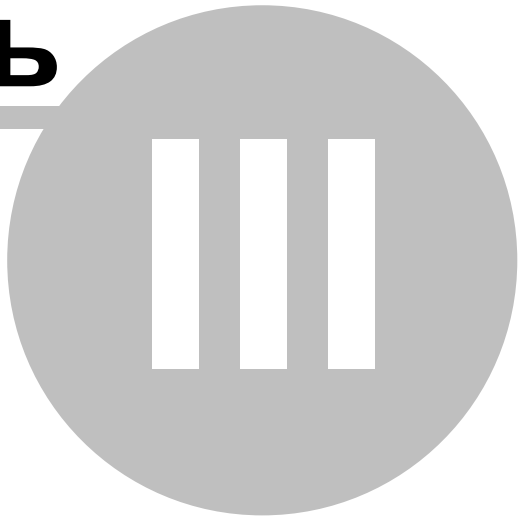
2.2.8 Многобайтовые переменные

1. Целые многобайтовые значения передаются младшим байтом вперед.
2. Строки передаются первым символом вперед. Конец строки произвольной длины отмечается нулевым байтом. Если строка короче отведенного для нее поля, оставшиеся байты игнорируются (заполняются произвольным значением).

МикроЭМ

Руководство программиста

Часть



3 Система команд

Описание команд приведено в следующем формате (пример):
0x05 CLSCRF_ Sound
IN: Count[1]
OUT: ACK[1]
где
0x05 – шестнадцатеричный код команды длиной 1 байт типа unsigned char
CLSCRF_ Sound – имя функции или символическое наименование команды
IN: – перечень параметров, передаваемых считывателю
Count[1] – имя очередного параметра и его длина в байтах
OUT: – перечень полей ответа от считывателя
ACK[1] – имя очередного поля ответа и его длина в байтах

Параметры команды и поля ответа перечисляются в порядке их следования в канале связи.

Везде по тексту ACK[1] – это код завершения операции – 1 байт типа signed char. При успешном выполнении операции должен быть равен 0. Полный список возможных его значений приведен в п.1.2.5.

Описание остальных полей ответа и всех параметров команд приведено для каждой команды индивидуально.

3.1 Команды управления считывателем

3.1.1 0x0E Выдача состояния считывателя

0x0E CLSCRF_GetState
OUT: ACK[1]; State[2]
State[2] – состояние считывателя:
бит 15 - микросхема-считыватель включена,
бит 14 - электромагнитное поле включено,
бит 13 - считыватель поддерживает режимы ICODE,
бит 12 - считыватель поддерживает режимы 14443-B,
бит 11 - считыватель поддерживает NFC,
биты 11-9 - резерв,
бит 8 - признак нажатия кнопки,
бит 7 - режим NFC,
биты 6-4 - текущий режим электромагнитного поля:
000 - режим ISO 14443-A (скорость см. биты 3-0)
001 - режим ISO 14443-B (скорость см. биты 3-0)
100 - режим ICODE SLI ISO 15693

110 - режим ICODE EPC ISO 18000-3
111 - режим ICODE UID ISO 18000-3
биты 3-2 - текущая скорость приема в режимах ISO 14443 (поток данных от карты к считывателю):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод
биты 1-0 - текущая скорость передачи в режимах ISO 14443 (поток данных от считывателя к карте):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод.

3.1.2 0x10 Инициализация микросхемы-считывателя

0x10 CLSCRF_Mfrc_On
OUT: ACK[1]

Поскольку при включении питания считывателя микросхема-считыватель выключена, то перед началом работы со считывателем должна быть выдана команда 0x10 (если ответ отличен от нуля, команду необходимо повторить, иначе дальнейшая работа со считывателем невозможна).

3.1.3 0x04 Включение микросхемы-считывателя

0x04 CLSCRF_Mfrc_Off
IN: 0x80; 0x01
OUT: ACK[1]

3.1.4 0x64 Выдача версии считывателя

0x64 CLSCRF_Get_Mfrc_Version
OUT: ACK[1]; VersionIC[5]; VersionFW[1]
VersionIC[5] – версия микросхемы-считывателя;
VersionFW[1] – версия микропрограммы.

3.1.5 0x22 Выдача серийного номера микросхемы-считывателя

0x22 CLSCRF_Get_Mfrc_Serial_Number
OUT: ACK[1]; SerNum[4]
SerNum[4] – серийный номер микросхемы-считывателя.

3.1.6 0x20 Сброс/выключение электромагнитного поля (RF)

0x20 CLSCRF_Mfrc_Rf_Off_On
IN: TimePeriod[2]
OUT: ACK[1]
TimePeriod[2] – значение задержки (мс).

RF выключается, а затем, если Time period отличен от нуля, выжидается пауза размером (Time period) мс, после чего RF включается снова.

3.1.7 0x51 Переключение режима электромагнитного поля (RF)

0x51 CLSCRF_Mfrc_Set_Rf_Mode
IN: RfMode[1]
OUT: ACK[1]
RfMode[1] – код режима, биты которого означают следующее:
бит 7 - резерв
биты 6-4 - устанавливаемый режим электромагнитного поля:
000 - ISO 14443-A (скорость см. биты 3-0)
001 - ISO 14443-B (скорость см. биты 3-0)
100 - ICODE SLI ISO 15693
110 - ICODE EPC ISO 18000-3
111 - ICODE UID ISO 18000-3
биты 3-2 - устанавливаемая скорость приема в режимах ISO 14443 (поток данных от карты к считывателю):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод
биты 1-0 - устанавливаемая скорость передачи в режимах ISO 14443 (поток данных от считывателя к карте):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод

3.1.8 0x05 Подача звукового сигнала

0x05 CLSCRF_Sound
IN: Count[1]
OUT: ACK[1]
Count[1] – количество звуковых импульсов. Продолжительность каждого импульса - 100 мс. Интервал между импульсами – также 100 мс.

Если считыватель получил команду подачи звукового сигнала, а звуковые импульсы от предыдущей такой же команды еще не закончились, то команда не выполняется, а отправляется только подтверждение об успешном выполнении команды.

3.1.9 0x07 Управление светодиодом

0x07 CLSCRF_Led
IN: BlinkColor[1]; BlinkCount[1]; PostColor[1]
OUT: ACK[1]
BlinkColor[1] – цвет мигания:
0 – не мигать;
1 – мигать красным цветом;
2 – мигать зеленым цветом;
3 – мигать желто-оранжевым цветом;
(Частота миганий – 2 Гц).
BlinkCount[1] – количество миганий, если 0 – не мигать.
PostColor[1] – постоянный режим светодиода по окончании мигания:
0 – погасить;
1 – зажечь красным цветом;
2 – зажечь зеленым цветом;
3 – зажечь желто-оранжевым цветом.

3.1.10 0x0F Изменение скорости обмена по COM-порту

0x0F CLSCRF_UART_Baudrate
IN: divisor[1]
OUT: ACK[1]
divisor[1] = целая часть от (1500000 / BaudRate),
где BaudRate = {9600, 14400, 19200, 38400, 57600, 115200}.

Данная команда используется только для интерфейсов RS232 и RS485.

После успешного выполнения команды необходимо закрыть COM-порт компьютера и снова открыть его на новой скорости.

Если требуется сохранить эту настройку в считывателе, чтобы после отключения питания, он снова запускался на установленной скорости, необходимо выполнить команду 69 00 00.

Например:

Хост: 0F 0D (установка скорости 115200 бод)

Считыватель: 00

Закрытие и открытие COM-порта хоста на новой скорости.

Хост: 69 00 00 (сохранение новой скорости)

Считыватель: 00

3.1.11 0x70 Очистка flash-памяти с ключами

0x70 CLSCRF_EraseFlash
IN: PassPhrase[4]= 0x87654321
OUT: ACK[1]
PassPhrase[4] – константное выражение для защиты от случайного стирания (0x87654321)

3.1.12 0x79 Запись во flash-память блока с ключом

0x79 CLSCRF_WriteFlashValue
IN: Address[2]; Value[16]
OUT: ACK[1]
Address[2] – адрес блока во flash {0..239};
Value[16] – ключ для записи

3.1.13 0x6E Проверка заполнения блока flash-памяти считывателя

0x6E CLSCRF_CheckFlashValueFilled
IN: Address[2]
OUT: ACK[1]
Address[2] – адрес блока во flash {0 ...239};
ACK[1] – код возврата: 0x00 – результат команды успешный и блок заполнен; 0xFF – результат команды успешный и блок пуст, другие значения – ошибка при выполнении команды.

3.1.14 0x6F (Сброс) аутентификация считывателя

0x6F CLSCRF_Crypto_AuthenticateReader

Команда выполняет (сброс) аутентификацию считывателя и установку режима (открытого) шифрованного обмена данными между считывателем и хостом.

Сброс аутентификации:
IN: AuthType[1] = 0x00; KeyNumber[2] = 0x0000
OUT: ACK[1]
Аутентификация:
IN: AuthType[1] = 0xAA; KeyNumber[2]
OUT: EncKeyNrRndB[16]
IN 0x6F
IN: 0xAF; KeyNumber[2]; EncKeyNhRndARndBh'[32]
OUT: EncKeyNrRndAr'[16]

Вычисления:

1. Reader: генерация RndB[16];
2. Reader: EncKeyNrRndB[16] = Enc(IV[16], RndB[16], KeyNr[16]);
3. Host: RndBh[16] = Dec(IV[16], EncKeyNrRndB[16], KeyNh[16]);
4. Host: RndBh'[16] = ЦиклСдвигВлевоНаОдинБайт(RndBh[16]);
5. Host: генерация RndA[16];
6. Host: RndARndBh'[32] = RndA[16] || RndBh'[16];
7. Host: EncKeyNhRndARndBh'[32] = Enc(IV[16], RndARndBh'[32], KeyNh [16]);
8. Reader: RndArRndBhr'[32] = Dec(IV[16], EncKeyNhRndARndBh'[32], KeyNr [16]);
9. Reader: RndAr[16] || RndBhr'[16] = RndArRndBhr'[32];
10. Reader: RndB'[16] = ЦиклСдвигВлевоНаОдинБайт(RndB[16]);
11. Reader: Сравнение(RndB'[16], RndBhr'[16])
12. Reader: RndAr'[16] = ЦиклСдвигВлевоНаОдинБайт(RndAr[16]);
13. Reader: EncKeyNrRndAr'[16] = Enc(IV[16], RndAr'[16], KeyNr[16]);
14. Host: RndArh'[16] = Dec(IV[16], EncKeyNrRndAr'[16], KeyNh[16]);
15. Host: RndA'[16] = ЦиклСдвигВлевоНаОдинБайт(RndA[16]);
16. Host: Сравнение(RndA'[16], RndArh'[16])

Функции:

DataA || DataB – операция присоединения вектора DataA к вектору DataB;

Enc(IV, Data, Key) – функция шифрования AES128 в режиме CBC данных Data ключом Key с начальным вектором IV, меняющая после вычисления значение IV;

Dec(IV, Data, Key) – функция дешифрования AES128 в режиме CBC данных Data ключом Key с начальным вектором IV, меняющая после вычисления значение IV;

ЦиклСдвигВлевоНаОдинБайт(Data) - циклически сдвиг на один байт в сторону младшего байта;

Сравнение(DataA, DataB) – сравнение векторов на предмет их равенства. В случае неравенства, процедура аутентификации считается не прошедшей.

Параметры:

IV[16] – начальный вектор, который при запуске общей процедуры устанавливается в 0x00, затем обновляется при каждой криптографической операции при аутентификации;

RndA[16] – случайный вектор, генерируемый хостом;

RndB[16] – случайный вектор, генерируемый считывателем;

KeyN[16] – ключ из записи под номером KeyNumber;

AuthType[1] – тип аутентификации (0x00 – сброс аутентификации, 0xAA - аутентификация);

KeyNumber[2] – номер блока данных flash-памяти считывателя, используемый в качестве ключа AES при шифрованнном обмене.

Подготовка шифрованного обмена

После удачногоо завершения процедуры аутентификации, считыватель и хост готовят сессионные ключи для дальнейшего обмена по шифрованному каналу:

SessionKey[16] = RndA[0..3] || RndB[0..3] || RndA[12..15] || RndB[12..15]

Вектор инициализации для выполнения последующих функций шифрации обнуляется: IV[16] = {0x00}

Формат шифрованного обмена

Шифрованный обмен данными выполняется при помощи (де-)шифрования AES128 в режиме CBC, ключом SessionKey[16] и начальным вектором инициализации IV[16] = {0x00}. Каждая последующая операция (де-)шифрования берёт за основу вектор инициализации IV[16], полученный от предыдущей операции.

3.2 Команды управления картами типа A стандарта ISO 14443

3.2.1 0x43 Активация карты типа A, находящейся в состоянии Idle

0x43 CLSCRF_Activate_Idle_A
OUT: ACK[1]; atq[2]; sak[1]; uid_len[1]; uid[uid_len]
atq[2] – ATQ;
sak[1] – SAK;
uid_len[1] – длина уникального номера карты (может быть 4, 7 или 10);
uid[uid_len] – уникальный номер карты.

3.2.2 0x1D Перевод активной карты типа A в состояние Halt

0x1D CLSCRF_Halt_A
OUT: ACK[1]

3.2.3 0x44 Активация карты типа A, находящейся в состоянии Halt

0x44 CLSCRF_Activate_Wakeup_A
IN: uid_len[1]; uid[uid_len]
OUT: ACK[1]; atq[2]; sak[1]
uid_len[1] – длина уникального номера карты (м.б. 4, 7 или 10);
uid[uid_len] – уникальный номер карты;
atq[2] – ATQ;
sak[1] – SAK.

3.2.4 0x35 Чтение информации ATS из карты

0x35 CLSCRF_ISO14443A_4_RATS
IN: CID[1]
OUT: ACK[1]; DataLength[2]; ATS[DataLength]

CID[1] - логический идентификатор карты для обмена по протоколу T=CL – любое число в диапазоне 0..14 (см. ISO 14443-3 7.10.6);
DataLength[2] – количество принятых от карты байт данных (ATS);
ATS[DataLength] – структура Answer To Select (см. ISO 14443-4 5.1, 5.2)

3.2.5 0x36 Установка протокола и параметров работы с картой

0x36 CLSCRF_ISO14443A_4_PPS
IN: CID[1] Baudrate[1]
OUT: ACK[1]
CID[1] - логический идентификатор карты для обмена по протоколу T=CL – любое число в диапазоне 0..14 (см. ISO 14443-3 7.10.6);
Baudrate[1] – значение устанавливаемой скорости обмена данными с картой:
Биты 7-4 – установить в 0
биты 3-2 - устанавливаемая скорость приема в режимах ISO 14443 (поток данных от карты к считывателю):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод
биты 1-0 - устанавливаемая скорость передачи в режимах ISO 14443 (поток данных от считывателя к карте):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод

3.3 Команды управления картами типа B стандарта ISO 14443

3.3.1 0x56 Активация карт типа B, находящихся в состоянии Idle

0x56 CLSCRF_Activate_Idle_B
IN: AFI[1]; SNI[1]
OUT: ACK[1]; CARD0[11]; CARD1[11]; CARD2[11]; …
AFI[1] – идентификатор семейства приложений (некий фильтр, если равен 0, то активируются все карты);
SNI[1] – индекс количества временнЫх слотов:
0 => SlotQuantity = 1 слот,
1 => SlotQuantity = 2 слота,
2 => SlotQuantity = 4 слота,
3 => SlotQuantity = 8 слотов,

4 => SlotQuantity = 16 слотов;

CARDi[11] = PUPI[4]; AppData[4]; ProtInfo[3] – информация об i-й активированной карте;

PUPI[4] – псевдоуникальный идентификатор карты;

AppData[4] – данные для идентификации приложений, имеющихся в карте;

ProtInfo[3] – информация о параметрах протокола обмена с картой.

3.3.2 0x54 Установка параметров протокола в данном сеансе связи с картой типа B

0x54 CLSCRF_Attrib_B

IN: ParamLen[1]; Params[ParamLen]

OUT: ACK[1]; AttrLen[1]; Attrib[AttrLen]

ParamLen[1] – длина входных параметров;

Params[ParamLen] – входные параметры (см. ISO 14443-3 п. 7.10.1, кроме первого байта 0x1D и двух заключительных байтов CRC_B);

AttrLen[1] – длина ответа;

Attrib[AttrLen] – ответ (см. ISO 14443-3 п. 7.11, кроме двух заключительных байтов CRC_B).

3.3.3 0x55 Перевод активной карты типа B в состояние Halt

0x55 CLSCRF_Halt_B

IN: PUPI [4]

OUT: ACK[1]

PUPI[4] – псевдоуникальный идентификатор карты.

3.3.4 0x57 Активация карт типа B, находящихся в состоянии Halt

0x57 CLSCRF_Activate_Wakeup_B

IN: AFI[1]; SNI[1]

OUT: ACK[1]; CARD0[11]; CARD1[11]; CARD2[11]; …

AFI[1] – идентификатор семейства приложений (некий фильтр, если равен 0, то активируются все карты);

SNI[1] – индекс количества временнЫх слотов:

0 => SlotQuantity = 1 слот,

1 => SlotQuantity = 2 слота,

2 => SlotQuantity = 4 слота,

3 => SlotQuantity = 8 слотов,

4 => SlotQuantity = 16 слотов;

CARDi[11] = PUPI[4]; AppData[4]; ProtInfo[3] – информация об i-й активированной карте;

PUPI[4] – псевдо-уникальный идентификатор карты;

AppData[4] – данные для идентификации приложений, имеющихся в карте;
ProtInfo[3] – информация о параметрах протокола обмена с картой.

3.4 Команда управления метками стандарта ISO 15693

3.4.1 0x30 Инвентаризация меток 15693

Рекурсивная инвентаризация меток
0x30 CLSCRF_FindAllTags_15693
IN: RFU[2]; AFI[1]
OUT: ACK[1]; RespLen[2]; Resp[RespLen]
RFU[2] – зарезервированные байты (установить в 0x0000);
AFI[1] – идентификатор семейства приложений;
RespLen[2] – количество байтов ответа считывателя;
Resp[RespLen] – ответ от считывателя.

Для каждой найденной метки Resp[RespLen] содержит следующую информацию:
RetCode – код завершения запроса в этом слоте - 1 байт
Count – количество байтов, полученных от метки - 1 байт
Если Count отличен от 0, то далее следует ответ от метки длиной Count (см. ISO 15693 п.10.3.1):
Flags - 1 байт
DSFID - 1 байт
UID - 8 байтов
Если RetCode отличен от 0, то далее может присутствовать
CRC16 - 2 байта.

Единичная инвентаризация меток
0x31 CLSCRF_Inventory_15693
IN: Flags[1]; Inventory[1]; AFI[1]; MaskLen[1]; MaskVal [от 0 до 8]
OUT: ACK[1]; RespLen[2]; Resp[RespLen]
Flags[1] – флаги запроса (см. ISO 15693-3 п.7.3);
Inventory[1] – код команды Inventory (всегда равен 0x01);
AFI[1] – идентификатор семейства приложений;
MaskLen[1] – количество битов маски;
MaskVal[от 0 до 8] – массив байтов, содержащий маску;
RespLen[2] – количество байтов ответа считывателя;
Resp[RespLen] – ответ от считывателя.

Для каждого временнОго слота Resp[RespLen] содержит следующую информацию:
RetCode – код завершения запроса в этом слоте - 1 байт
Count – количество байтов, полученных от метки - 1 байт

Если Count отличен от 0, то далее следует ответ от метки длиной Count (см. ISO 15693 п.10.3.1):

Flags - 1 байт
DSFID - 1 байт
UID - 8 байтов

Если RetCode отличен от 0, то далее может присутствовать
CRC16 - 2 байта.

3.4.2 0x32 Перевод метки 15693 в состояние QUIET

0x32 CLSCRF_Stay_Quiet_15693
IN: Flags[1]; StayQuiet[1]; UID[8]
OUT: ACK[1]
Flags[1] – флаги запроса (см. ISO 15693-3 п.7.3);
StayQuiet[1] – код команды Stay Quiet (всегда равен 0x02);
UID[8] – массив байтов, содержащий уникальный идентификатор метки.

3.5 Команды обмена данными с картой Mifare Classic

3.5.1 Вычисление абсолютного номера блока

Секторы 0..31-й содержат 4 блока (относительные номера 0..3).

Начиная с 32-го, сектор содержит по 16 блоков (относительные номера 0..15).

Абсолютный номер блока вычисляется следующим образом:
1. <абсолютный номер блока> = <номер сектора>*4 + <номер блока в секторе>
2. Если номер сектора >= 32, то прибавляем (<номер сектора> - 32)*12.

3.5.2 0x16 (обратная совместимость) Кодирование ключа

0x16 CLSCRF_MifareStandard_HostCodeKey
IN: uncoded[6]
OUT: ACK[1]; coded[12]
uncoded [6] – некодированный ключ;
coded[12] – кодированный ключ.

3.5.3 0x18 (обратная совместимость) Аутентификация ключом, заданным в команде

0x18 CLSCRF_MifareStandard_AuthKey
IN: key_type[1]; snr[4]; keys[12]; AbsBlockNo[1]
OUT: ACK[1]
key_type[1] – тип ключа:
0x60 - Key A,
0x61 - Key B;
snr[4] – UID карты (для карт с 7-байтовым UID[0..6], поддерживающих протокол Mifare Standard, в качестве snr[0..3] использовать байты UID[3..6]);
keys[12] – кодированный ключ;
AbsBlockNo[1] – абсолютный номер блока.

3.5.4 0x14 Аутентификация ключом, заданным в команде

0x14 CLSCRF_MifareStandard_AuthKeyDirect
IN: key_type[1]; snr[4]; key[6]; AbsBlockNo[1]
OUT: ACK[1]
key_type[1] – тип ключа:
0x60 - Key A,
0x61 - Key B;
snr[4] – UID карты (для карт с 7-байтовым UID[0..6], поддерживающих протокол Mifare Standard, в качестве snr[0..3] использовать байты UID[3..6]);
key[6] – ключ;
AbsBlockNo[1] – абсолютный номер блока.

3.5.5 0x17 Запись ключа в EEPROM считывателя

0x17 CLSCRF_MifareStandard_WriteKeyToE2
IN: key_type[1]; sector[1]; uncoded_keys[6]
OUT: ACK[1]
key_type[1] – тип ключа:
0x60 - Key A,
0x61 - Key B;
sector[1] – номер сектора;
uncoded_keys[6] – некодированный ключ.
Считыватель поддерживает в данном режиме работу с секторами с индексами 0..15, то есть с первым килобайтом памяти карты.

3.5.6 0x15 Аутентификация ключом, находящимся в EEPROM считывателя

0x15 CLSCRF_MifareStandard_AuthE2
IN: key_type[1]; snr[4]; sector[1]; AbsBlockNo[1]
OUT: ACK[1]
key_type[1] – тип ключа:
0x60 - Key A,
0x61 - Key B;
snr[4] – UID карты;
sector[1] – номер сектора;
AbsBlockNo[1] – абсолютный номер блока.
Считыватель поддерживает в данном режиме работу с секторами с индексами 0..15, то есть с первым килобайтом памяти карты.

3.5.7 0x19 Чтение блока

0x19 CLSCRF_MifareStandard_Read
IN: AbsBlockNo[1]
OUT: ACK[1]; Data[16]
AbsBlockNo[1] – абсолютный номер блока.
Data[16] – прочитанные из блока данные.

3.5.8 0x1A Запись блока

0x1A CLSCRF_MifareStandard_Write
IN: AbsBlockNo[1]; Data[16]
OUT: ACK[1]
AbsBlockNo[1] – абсолютный номер блока;
Data[16] – 16 байтов данных, записываемых в блок.

3.5.9 0x1B Операция Value

0x1B (Этот код операции используют три функции:
CLSCRF_MifareStandard_Decrement
CLSCRF_MifareStandard_Increment
CLSCRF_MifareStandard_Restore)
IN: dd_mode[1]; AbsSrcBlockNo[1]; Value[4]; AbsTgtBlockNo[1]
OUT: ACK[1]
dd_mode[1] – код операции: PICC_DECREMENT 0xC0,
PICC_INCREMENT 0xC1,
PICC_RESTORE 0xC2;
AbsSrcBlockNo[1] – абсолютный номер исходного блока;
Value[4] – значение;
AbsTgtBlockNo[1] – абсолютный номер блока-результата.

3.5.10 0x1C Персонализация UID

0x1C CLSCRF_MifareStandard_EV1_PersonalizeUid
IN: UidMode[1]
OUT: ACK[1]
UidMode[1] – режим UID (режим, в который переводится карта)
Допустимо любое из значений:
UIDF0 = 0x00,
UIDF1 = 0x40,
UIDF2 = 0x20,
UIDF3 = 0x60.

3.5.11 0x26 Изменение нагрузки антенны карт

0x26 CLSCRF_MifareStandard_EV1_SetLoadModulationType
IN: UidMode[1]
OUT: ACK[1]
ModType[1] – тип нагрузки: 0x01 - Сильная (по умолчанию), 0x00 - нормальная.

3.6 Команды обмена данными с картой Mifare UltraLight (C)

3.6.1 0x25 (устарела) Чтение одной страницы

0x25
IN: Addr[1]
OUT: ACK[1]; Data[4]
Addr[1] – номер страницы;
Data[4] – прочитанные данные.

3.6.2 0x19 Чтение четырёх страниц

0x19 CLSCRF_MifareUltralight_Read
IN: Addr[1]
OUT: ACK[1]; Data[16]
Addr[1] – номер страницы;
Data[16] – прочитанные данные.

3.6.3 0x1E Запись страницы

0x1E CLSCRF_MifareUltralight_Write
IN: Addr[1]; Data[4]
OUT: ACK[1]
Addr[1] – номер страницы;
Data[4] – записываемые данные.

3.6.4 0x2C Запись ключа аутентификации

0x2C CLSCRF_MifareUltralightC_WriteKey
IN: KeyFlashAddress[2]
OUT: ACK[1]
KeyFlashAddress[2] - адрес блока во flash-памяти считывателя (0..239), откуда следует взять ключ для записи в карту.

3.6.5 0x2D Аутентификация карты

0x2D CLSCRF_MifareUltralightC_Authenticate
IN: KeyFlashAddress[2]
OUT: ACK[1]
KeyFlashAddress[2] - адрес блока во flash-памяти считывателя (0..239), откуда следует взять ключ для аутентификации.

3.7 Команды обмена данными с картой Mifare Plus

3.7.1 Таблица формирования параметра типа значения

Назначение блока данных	Значение параметра
Блоки и данные	
MIFARE Data/Value Blocks MIFARE Sector Trailers	0x00
Специализированные блоки	
MFP Configuration Block	0xB0
Installation Identifier	0xB1
ATS Information	0xB2
Field Configuration Block	0xB3
Ключи секторов	

AES Sector Keys (Key A)	0x4A
AES Sector Keys (Key B)	0x4B
Специализированные ключи	
Originality Key	0x80
Card Master Key	0x90
Card Configuration Key	0x91
Level 2 Switch Key	0x92
Level 3 Switch Key	0x93
SL1 Card Authentication Key	0x94
Ключи виртуальных карт	
Select VC Key	0xA0
Proximity Check Key	0xA1
VC Polling ENC Key	0xA2
VC Polling MAC Key	0xA3

3.7.2 Таблица формирования параметра режима защиты передачи данных

MAC в комманде	Шифрование AES	MAC в ответе	Значение параметра
Да	Да	Нет	0x00
Да	Да	Да	0x01
Да	Нет	Нет	0x02
Да	Нет	Да	0x03
Нет	Да	Нет	0x04
Нет	Да	Да	0x05
Нет	Нет	Нет	0x06
Нет	Нет	Да	0x07

3.7.3 Таблица допустимых режимов защиты передачи данных

Операция	MAC в комманде	Шифрование AES	MAC в ответе

Read Data	Да/Нет	Да/Нет	Да/Нет
Write Data	Строго Да	Да/Нет	Да/Нет
Increment	Строго Да	Строго Да	Да/Нет
Decrement	Строго Да	Строго Да	Да/Нет
Transfer	Строго Да	Строго Нет	Да/Нет
Increment Transfer	Строго Да	Строго Да	Да/Нет
Decrement Transfer	Строго Да	Строго Да	Да/Нет
Restore	Строго Да	Да/Нет*	Да/Нет*

* Должны быть либо оба Да, либо оба Нет.

3.7.4 0xA8 Запись данных персонализации в карту

0xA8 CLSCRF_MifarePlus_WritePersoExplicit
IN: ValueType[1]; SectorNumber[1]; BlockNumber[1]; RFU[2]; Data[16]
OUT: ACK[1]

ValueType[1] - тип записываемого значения (см. таблицу);

SectorNumber[1] - номер сектора (используется при записи блока данных или ключа, относящегося к определенному сектору);

BlockNumber[1] - номер блока (используется при записи блока данных или ключа, относящегося к определенному блоку в секторе);

RFU[2] – должно быть 0xFFFF;

Data[16] – данные для записи.

3.7.5 0xAA Персонализация карты

0xAA CLSCRF_MifarePlus_CommitPerso
IN: 0x21; 0x43; 0x65; 0x87;
OUT: ACK[1];
Пример: aa 21 43 65 87
Отклик: 00

3.7.6 0xA0 Управление аутентификацией

0xA0 CLSCRF_MifarePlus_Authenticate
IN: AuthType[1]; KeyType[1]; SectorNumber[1]; KeyFlashAddress[2]; LenCap[1]; PCDCap2[0..6]
OUT: ACK[1]

AuthType[1] - тип операции аутентификации (0x01 – первичная, 0x0F – последующая, 0x00 – сброс аутентификации);

KeyType[1] - тип ключа для аутентификации (см. таблицу);

SectorNumber[1] - номер сектора для аутентификации;
KeyFlashAddress[2] - адрес ключа во flash-памяти считывателя для аутентификации;
LenCap[1] - длина блока характеристик считывателя (0..6, установить в 0);
PCDCap2[0..6] - блок характеристик считывателя (пока отсутствует).

3.7.7 0xA6 Чтение нескольких блоков SL2

0xA6 CLSCRF_MifarePlus_MultiBlockRead
IN: AbsBlockNo[1]; BlocksCount[1]
OUT: ACK[1]; DataLength[2]; Data[16/32/48]
AbsBlockNo[1] - абсолютный номер блока;
BlocksCount[1] - количество читаемых блоков;
DataLength[2] - длина прочитанного массива данных;
Data[16/32/48] – прочитанный массив данных (1,2 или 3 блока).

3.7.8 0xA7 Запись нескольких блоков SL2

0xA7 CLSCRF_MifarePlus_MultiBlockWrite
IN: AbsBlockNo[1]; BlocksCount[1]; Data[16/32/48]
OUT: ACK[1]
AbsBlockNo[1] - абсолютный номер блока;
BlocksCount[1] - количество записываемых блоков;
Data[16/32/48] - массив данных для записи (1,2 или 3 блока).

3.7.9 0xA4 Чтение данных

0xA4 CLSCRF_MifarePlus_ReadData
IN: EncryptionMode[1]; ValueType[1]; SectorNumber[1]; BlockNumber[1]; BlocksCount[1]
OUT: ACK[1]; DataLength[2]; Data[16/32/48]
EncryptionMode[1] – режим защиты обмена данными (см. таблицы);
ValueType[1] - тип читаемого значения (см. таблицу);
SectorNumber[1] – номер сектора, в котором нужно производить чтение;
BlockNumber[1] – номер блока, с которого требуется начать считывание данных;
BlocksCount[1] – количество блоков данных, которое нужно прочитать (1..3);
DataLength[2] - длина прочитанного массива данных (байт);
Data[16/32/48] – прочитанный массив данных (1,2 или 3 блока).

3.7.10 0xA5 Запись данных

0xA5 CLSCRF_MifarePlus_WriteData
IN: EncryptionMode[1]; ValueType[1]; SectorNumber[1]; BlockNumber[1]; Data[16/32/48]

OUT: ACK[1]
EncryptionMode[1] – режим защиты обмена данными (см. таблицы);
ValueType[1] - тип записываемого значения (см. таблицу);
SectorNumber[1] – номер сектора для записи;
BlockNumber[1] – номер начального блока для записи;
BlocksCount[1] – количество блоков данных, которое нужно записать (1..3);
Data[16/32/48] – записываемый массив данных (1,2 или 3 блока).

3.7.11 0xA1 Прибавление значения

0xA1 CLSCRF_MifarePlus_Increment
IN: OperationType[1]; EncryptionMode[1]; SourceSectorNumber[1]; SourceBlockNumber[1]; Value[4]
OUT: ACK[1]
OperationType[1] – тип операции (0xB0);
EncryptionMode[1] – режим защиты обмена данными (см. таблицы);
SourceSectorNumber[1] – номер исходного сектора;
SourceBlockNumber[1] – номер исходного блока;
Value[4] – значение, на которое требуется прирастить блок значения.

3.7.12 0xA1 Вычитание значения

0xA1 CLSCRF_MifarePlus_Decrement
IN: OperationType[1]; EncryptionMode[1]; SourceSectorNumber[1]; SourceBlockNumber[1]; Value[4]
OUT: ACK[1]
OperationType[1] – тип операции (0xB2);
EncryptionMode[1] – режим защиты обмена данными (см. таблицы);
SourceSectorNumber[1] – номер исходного сектора;
SourceBlockNumber[1] – номер исходного блока;
Value[4] – значение, которое требуется вычесть из блока значения.

3.7.13 0xA1 Запись данных из буфера переноса в блок

0xA1 CLSCRF_MifarePlus_Transfer
IN: OperationType[1]; EncryptionMode[1]; DestinationSectorNumber[1]; DestinationBlockNumber[1]
OUT: ACK[1]
OperationType[1] – тип операции (0xB4);
EncryptionMode[1] – режим защиты обмена данными (см. таблицы);
DestinationSectorNumber[1] – номер сектора для записи;
DestinationBlockNumber[1] – номер блока для записи.

3.7.14 0xA1 Прибавление значения с последующей записью данных из буфера переноса в блок

0xA1 CLSCRF_MifarePlus_IncrementTransfer

IN: OperationType[1]; EncryptionMode[1]; SourceSectorNumber[1]; SourceBlockNumber[1]; Value[4]; DestinationSectorNumber[1]; DestinationBlockNumber[1]

OUT: ACK[1]

OperationType[1] – тип операции (0xB6);
EncryptionMode[1] – режим защиты обмена данными (см. таблицы);
SourceSectorNumber[1] – номер исходного сектора;
SourceBlockNumber[1] – номер исходного блока;
Value[4] – значение, на которое требуется прирастить блок значения;
DestinationSectorNumber[1] – номер сектора для записи;
DestinationBlockNumber[1] – номер блока для записи.

3.7.15 0xA1 Вычитание значения с последующей записью данных из буфера переноса в блок

0xA1 CLSCRF_MifarePlus_DecrementTransfer

IN: OperationType[1]; EncryptionMode[1]; SourceSectorNumber[1]; SourceBlockNumber[1]; Value[4]; DestinationSectorNumber[1]; DestinationBlockNumber[1]

OUT: ACK[1]

OperationType[1] – тип операции (0xB8);
EncryptionMode[1] – режим защиты обмена данными (см. таблицы);
SourceSectorNumber[1] – номер исходного сектора;
SourceBlockNumber[1] – номер исходного блока;
Value[4] – значение, которое требуется вычесть из блока значения;
DestinationSectorNumber[1] – номер сектора для записи;
DestinationBlockNumber[1] – номер блока для записи.

3.7.16 0xA1 Запись данных блока значения в буфер переноса

0xA1 CLSCRF_MifarePlus_Restore

IN: OperationType[1]; EncryptionMode[1]; SourceSectorNumber[1]; SourceBlockNumber[1]

OUT: ACK[1]

OperationType[1] – тип операции (0xC2);
EncryptionMode[1] – режим защиты обмена данными (см. таблицы);
SourceSectorNumber[1] – номер исходного сектора;
SourceBlockNumber[1] – номер исходного блока.

3.7.17 0xA2 Начальный и промежуточный запрос поддержки виртуальных карт

0xA2 CLSCRF_MifarePlus_VirtualCardSupport
IN: OperationType[1]; IID[16]
OUT: ACK[1]
OperationType[1] – тип операции (0x00);
IID[16] – идентификатор инсталляции.

3.7.18 0xA2 Завершающий запрос поддержки виртуальных карт

0xA2 CLSCRF_MifarePlus_VirtualCardSupportLast
IN: OperationType[1]; IID[16]; KencFlashAddress[2]; KmacFlashAddress[2]; LenCap[1]; PCDCap[0..3]
OUT: ACK[1]; Info[1]; PICCCap[2]; PaddedUID[13]
OperationType[1] – тип операции (0x01);
IID[16] – идентификатор инсталляции;
KencFlashAddress[2] - адрес ключа VC Polling ENC Key во flash-памяти считывателя;
KmacFlashAddress[2] - адрес ключа VC Polling MAC Key во flash-памяти считывателя;
LenCap[1] - длина блока характеристик считывателя (0..3, установить в 0);
PCDCap[0..3] - блок характеристик считывателя (пока отсутствует);
Info[1] – информация о карте (0x83 – 4 байт UID, 0x03 – 7 байт UID);
PICCCap[2] – характеристики карты;
PaddedUID[13] – идентификатор карты (4-байтовых или 7-байтовый, в зависимости от Info), паддированный до длины 13 байт.

3.7.19 0xA3 Выбор виртуальной карты

0xA3 CLSCRF_MifarePlus_VirtualCardSelect
IN: OperationType[1]; KselFlashAddress[2] PICCCap[2]; PaddedUID[13]
OUT: ACK[1]
OperationType[1] – тип операции (0x01);
KselFlashAddress[2] - адрес ключа Select VC Key во flash-памяти считывателя;
PICCCap[2] – характеристики карты;
PaddedUID[13] – идентификатор карты (4-байтовых или 7-байтовый), паддированный до длины 13 байт.

3.7.20 0xA3 Снятие выбора виртуальной карты

0xA3 CLSCRF_MifarePlus_VirtualCardDeselect
IN: OperationType[1]
OUT: ACK[1]
OperationType[1] – тип операции (0x00);

3.7.21 0xA9 Поиск релейной атаки

0xA9 CLSCRF_MifarePlus_ProximityCheck
IN: KproxFlashAddress[2]
OUT: ACK[1]
KproxFlashAddress[2] - адрес ключа Proximity Check Key во flash-памяти считывателя.

3.8 Обмен данными с картой Mifare DES Fire

3.8.1 Таблица константных значений

Наименования констант	Значения
Типы аутентификации карты или приложения на карте	
DES	0x00
3DES	0x01
3K3DES	0x02
AES	0x03
Методы шифрации данных при транзакциях	
DES или 3DES	0x00
3K3DES	0x01
AES	0x02
Виды передачи данных	
Открытая передача	0x00
Использование MAC-подписи	0x01
Передача с шифрованием	0x03
Типы файлов	
Файл данных с одномоментным обновлением	0x00

Файл данных с резервным копированием	0x01
Файл, хранящий значение (32-битное число) с резервным копированием	0x02
Файл, хранящий линейную последовательность записей с резервным копированием	0x03
Файл, хранящий циклическую последовательность записей с резервным копированием	0x04
Требования по доступу для выполнения операции	
Требуется аутентификация по мастер-ключу приложения	0x00
Требуется аутентификация по ключу приложения 1 - 13	0x01 – 0x0D
Свободный доступ (или требуется аутентификация по аналогичному ключу – для команды ChangeKey)	0x0E
Доступ закрыт (или все ключи приложения, кроме мастер-ключа заморожены – для команды ChangeKey)	0x0F
Типы ключей	
DES или 3DES	0x00
3K3DES	0x01
AES	0x02

3.8.2 Формат команд для управления считывателем при помощи микропрограммы внешнего контроллера.

Система команд для работы с картами Mifare DES Fire (EV1) описана в соответствующих спецификациях от NXP (http://www.ru.nxp.com/products/identification_and_security/smart_card_ics/mifare_smart_card_ics/mifare_desfire/). Для их выполнения, войдите в режим T=CL и сформируйте соответствующие командные посылки.

3.9 Команда непосредственного обмена данными с картой

3.9.1 0x48 Непосредственный обмен с картой

0x48 CLSCRF_DirectIO_Card
IN: tx_len[2]; tx_data[tx_len]; timeout[4]
OUT: ACK[1]; rx_len[2]; rx_data[rx_len]

tx_len[2] – длина передаваемых в карту данных;
tx_data[tx_len] – передаваемые в карту данные;
timeout[4] – таймаут операции, единица измерения = 128 / 13.56 МГц = около 9.439528 мкс;
rx_len[2] – длина принятых от карты данных;
rx_data[rx_len] – принятые от карты данные.

3.10 Команды конфигурации устройств на шине RS485

3.10.1 0x7A Чтение адреса устройства

0x7A CLSCRF_ReadDeviceAddress
OUT: ACK[1]; Addr[1]
Addr[1] – адрес устройства.

Эта команда используется для определения «потерянного» адреса считывателя. На шине RS485 оставляют один считыватель и выдают эту команду по адресу 0.

3.10.2 0x77 Запись адреса устройства

0x77 CLSCRF_WriteDeviceAddress
IN: Addr[1]
OUT: ACK[1]
Addr[1] – новый адрес устройства.

Время выполнения команды – 400 мс. Эта команда используется для назначения нового адреса считывателю. Поскольку при изготовлении устройство получает адрес 0, то считыватель оставляют на шине RS485 единственным и выдают эту команду по адресу 0. Пользователь должен сам следить за тем, чтобы на шине RS485 не оказалось двух устройств с одинаковыми адресами.

3.11 Команды для работы со считывателем NFC663

3.11.1 0x11 Активация устройства

0x11 CLSCRF_NFC663_ActivateCard
IN: Nfcid3i[10],Did[1],NadEnable[1],Nad[1],Dsi[1],Dri[1],Fsl[1],GiLength[1], Gi[GiLength]

OUT: ACK[1]; RespLen[2]; Resp[RespLen]

Nfcid3i[10] – массив из 10 байт: при начальной скорости в 106kbps - NFCID3, генерируется случайным образом; при скоростях 212/424kbps - байты 0-7 соответствуют NFCID2, а байты 8-9 должны быть установлены в 0.
Did[1] – идентификатор устройства, "0" - не использовать, либо 1-14;
NadEnable[1] – включение использования адреса шины, для включения установить НЕ в "0";
Nad[1] – адрес узла: игнорируется, если bNadEnabled = 0;
Dsi[1] – индекс делителя отправки (от цели к инициатору) 0-2 (PHPAL_I18092MPI_DATARATE_106, PHPAL_I18092MPI_DATARATE_212, PHPAL_I18092MPI_DATARATE_424);
Dri[1] – индекс делителя приема (от инициатора к цели) 0-2 (PHPAL_I18092MPI_DATARATE_106, PHPAL_I18092MPI_DATARATE_212, PHPAL_I18092MPI_DATARATE_424);
Fsl[1] – байт длины кадра 0-3 (PHPAL_I18092MPI_FRAMESIZE_64, PHPAL_I18092MPI_FRAMESIZE_128, PHPAL_I18092MPI_FRAMESIZE_192, PHPAL_I18092MPI_FRAMESIZE_254);
GiLength[1] – количество байт общей информации;
Gi[1] – опционально, байты общей информации;
RespLen[2] – длина считанных атрибутов в байтах;
Resp[RespLen] – буфер, куда будут записаны байты ATR (ответа с атрибутаи), должен быть не меньше 64 байт;

3.11.2 0x12 Отмена выбора

0x12 CLSCRF_NFC663_Deselect
IN: DeselectCommand[1]
OUT: ACK[1]

DeselectCommand[1] – запрос на отправку: PHPAL_I18092MPI_DESELECT_DSL, либо PHPAL_I18092MPI_DESELECT_RLS

3.11.3 0x13 Обмен

0x13 CLSCRF_NFC663_Exchange
IN: Option[2]; send_len[2]; send_data[send_len]
OUT: ACK[1]; rec_len[2]; rec_data[rec_len]

Option[2] – параметр опций:
одно из PH_EXCHANGE_DEFAULT, PH_EXCHANGE_TXCHAINING, PH_EXCHANGE_RXCHAINING, PH_EXCHANGE_RXCHAINING_BUFSIZE,
сложенное с любой комбинацией из PH_EXCHANGE_TX_CRC, PH_EXCHANGE_RX_CRC, PH_EXCHANGE_PARITY,
сложенное с любой комбинацией из PH_EXCHANGE_LEAVE_BUFFER_BIT, PH_EXCHANGE_BUFFERED_BIT;
send_len[2] – количество байт для передачи;
send_data[send_len] – буфер с данными для передачи;
rec_len[2] – количество принятых байт;

rec_data[rec_len] – буфер для размещения принятых байт;

3.11.4 0x40 Сброс протокола

0x40 CLSCRF_NFC663_ResetProtocol
OUT: ACK[1]

3.11.5 0x33 Запрос атрибутов

0x33 CLSCRF_NFC663_AttributeRequest
IN: Nfcid3i[10],Did[1],NadEnable[1],Nad[1],Fsl[1],GiLength[1],Gi[GiLength]
OUT: ACK[1]; RespLen[2]; Resp[RespLen]

Nfcid3i[10] – массив из 10 байт: при начальной скорости в 106kbps - NFCID3, генерируется случайным образом; при скоростях 212/424kbps - байты 0-7 соответствуют NFCID2, а байты 8-9 должны быть установлены в 0.
Did[1] – идентификатор устройства, "0" - не использовать, либо 1-14;
NadEnable[1] – включение использования адреса шины, для включения установить НЕ в "0";
Nad[1] – адрес узла: игнорируется, если bNadEnabled = 0;
Fsl[1] – байт длины кадра 0-3 (PHPAL_I18092MPI_FRAMESIZE_64, PHPAL_I18092MPI_FRAMESIZE_128, PHPAL_I18092MPI_FRAMESIZE_192, PHPAL_I18092MPI_FRAMESIZE_254);
GiLength[1] – количество байт общей информации;
Gi[1] – опционально, байты общей информации;
RespLen[2] – длина считанных атрибутов в байтах;
Resp[RespLen] – буфер, куда будут записаны байты ATR (ответа с атрибутаи), должен быть не меньше 64 байт;

3.11.6 0x34 Выбор параметров

0x34 CLSCRF_NFC663_ParameterSelect
IN: Dsi[1],Dri[1],Fsl[1]
OUT: ACK[1]

Dsi[1] – индекс делителя отправки (от цели к инициатору) 0-2 (PHPAL_I18092MPI_DATARATE_106, PHPAL_I18092MPI_DATARATE_212, PHPAL_I18092MPI_DATARATE_424);
Dri[1] – индекс делителя приема (от инициатора к цели) 0-2 (PHPAL_I18092MPI_DATARATE_106, PHPAL_I18092MPI_DATARATE_212, PHPAL_I18092MPI_DATARATE_424);
Fsl[1] – байт длины кадра 0-3 (PHPAL_I18092MPI_FRAMESIZE_64, PHPAL_I18092MPI_FRAMESIZE_128, PHPAL_I18092MPI_FRAMESIZE_192, PHPAL_I18092MPI_FRAMESIZE_254);

3.11.7 0x1F Проверка присутствия

0x1F CLSCRF_NFC663_PresenceCheck
OUT: ACK[1]

3.11.8 0x41 Установка конфигурации

0x41 CLSCRF_NFC663_SetConfig
IN: ParameterNumber[2],ParameterValue[2]
OUT: ACK[1]
ParameterNumber[2] – идентификатор параметра - одно из:
PHPAL_I18092MPI_CONFIG_PACKETNO, PHPAL_I18092MPI_CONFIG_DID, PHPAL_I18092MPI_CONFIG_NAD, PHPAL_I18092MPI_CONFIG_WT, PHPAL_I18092MPI_CONFIG_FSL, PHPAL_I18092MPI_CONFIG_MAXRETRYCOUNT;
ParameterValue[2] – значение параметра;

3.11.9 0x42 Чтение конфигурации

0x42 CLSCRF_NFC663_GetConfig
IN: ParameterNumber[2]
OUT: ACK[1]; ParameterValue[2]
ParameterNumber[2] – идентификатор параметра - одно из:
PHPAL_I18092MPI_CONFIG_PACKETNO, PHPAL_I18092MPI_CONFIG_DID, PHPAL_I18092MPI_CONFIG_NAD, PHPAL_I18092MPI_CONFIG_WT, PHPAL_I18092MPI_CONFIG_FSL, PHPAL_I18092MPI_CONFIG_MAXRETRYCOUNT;
ParameterValue[2] – указатель на значение параметра;

3.11.10 0x28 Чтение серийного номера (NFCID3)

0x28 CLSCRF_NFC663_GetSerialNo
OUT: ACK[1]; NFCID3[10]
NFCID3[10] – серийный номер NFCID3 - 10 байт;

3.11.11 0x23 Чтение заданного количества байт из указанного адреса EEPROM

0x23 CLSCRF_NFC663_E2_Read
IN: Addr[2]; Length[1];
OUT: ACK[1]; Data[Length]
Addr[2] – адрес байта 192(0x00C0)..6143(0x17FF);
Length[1] – количество вычитываемых байт данных 1..127;
Data[Length] – массив, куда будут скопированы прочитанные байты данных;

3.11.12 0x24 Запись одного байта данных в указанный адрес EEPROM

0x24 CLSCRF_NFC663_E2_Write
IN: Addr[2]; Data[1];
OUT: ACK[1]

Addr[2] – адрес байта 192(0x00C0)..6143(0x17FF);
Data[1] – записываемый байт данных;

3.11.13 0x2A Запись нескольких байт данных в указанную страницу EEPROM

0x2A CLSCRF_NFC663_E2_WritePage
IN: PageAddr[2]; Length[1]; Data[Length];
OUT: ACK[1]

PageAddr[2] – адрес страницы 3..95(0x5F);
Length[1] – количество записываемых байт данных 1..64;
Data[Length] – указатель на массив байт данных;

МикроЭМ

Руководство программиста

Часть

4 Библиотека Clscrfl.dll

4.1 Функции управления интерфейсом

4.1.1 CLSCRF_Create

LONG CLSCRF_Create(IN OUT LPVOID* ppReader);

Создает объект-интерфейс. Эта функция должна вызываться 1 раз в начале работы приложения.

ppReader – адрес переменной, в которую будет помещена ссылка на созданный
объект-интерфейс и которая будет использоваться в вызовах всех остальных функций.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.1.2 CLSCRF_Destroy

LONG CLSCRF_Destroy(IN LPVOID* ppReader);

Уничтожает объект-интерфейс. Эта функция должна вызываться 1 раз в конце работы приложения.

ppReader – адрес переменной, которая содержит ссылку на уничтожаемый объект-интерфейс (см. функцию CLSCRF_Create).
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.1.3 CLSCRF_Open

LONG CLSCRF_Open(IN LPVOID pReader,
IN DWORD dwPortNumber = 0,
IN DWORD dwBaudrate = 9600,
IN DWORD dwLogFile = 0);

Устаревшая функция, оставлена только для совместимости с уже существующими приложениями. Для вновь разрабатываемых приложений следует пользоваться функциями CLSCRF_OpenRS и CLSCRF_OpenUSB.

Открывает интерфейс. Эта функция должна вызываться 1 раз перед началом обмена со считывателем и каждый раз при изменении скорости обмена по COM-порту.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwPortNumber – выбор интерфейса:
0 - интерфейс USB;
1 - последовательный порт COM1;
2 - последовательный порт COM2;
3 - последовательный порт COM3 и т.д.
dwBaudrate – скорость обмена по COM-порту;
dwLogFile – вывод в файл хронологии обмена:
1 - создавать файл;
0 - не создавать файл.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.1.4 CLSCRF_OpenRS

```
LONG CLSCRF_OpenRS(     IN LPVOID pReader,
                        IN DWORD dwIndex = 0,
                        IN DWORD dwBaudrate = 9600,
                        IN DWORD dwLogFile = 0 );
```

Открывает COM-порт (интерфейс RS232 или RS485). Эта функция должна вызываться 1 раз перед началом обмена со считывателем и каждый раз при изменении скорости обмена по COM-порту.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwIndex – номер COM-порта от 0 (0 - COM1, 1 - COM2 и т.д.);
dwBaudrate – скорость обмена по COM-порту;
dwLogFile – вывод в файл хронологии обмена:
1 - создавать файл;
0 - не создавать файл.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.1.5 CLSCRF_OpenUSB

```
LONG CLSCRF_OpenUSB(    IN LPVOID pReader,
                        IN DWORD  dwIndex = 0,
                        IN DWORD  dwLogFile = 0 );
```

Открывает USB-интерфейс. Эта функция должна вызываться 1 раз перед

началом обмена со считывателем.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwIndex – номер USB-устройства в дереве устройств от 0;
dwLogFile – вывод в файл хронологии обмена:
1 - создавать файл;
0 - не создавать файл.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.1.6 CLSCRF_IsOpened

BOOL CLSCRF_IsOpened(IN LPVOID pReader);

Проверяет доступность интерфейса.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create).
Возвращаемое значение:
0 – интерфейс закрыт,
1 – интерфейс открыт.

4.1.7 CLSCRF_Close

LONG CLSCRF_Close(IN LPVOID pReader);

Закрывает интерфейс. Эта функция должна вызываться 1 раз перед уничтожением интерфейса и каждый раз при изменении скорости обмена по COM-порту.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create).
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.1.8 CLSCRF_GetIOTimeout

LONG CLSCRF_GetIOTimeout(IN LPVOID pReader,
OUT LPDWORD pdwTimeout);

Выдает текущий интервал ожидания выполнения функций.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pdwTimeout – адрес переменной, в которую будет помещено значение таймаута в миллисекундах.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.1.9 CLSCRF_SetIOTimeout

```
LONG CLSCRF_SetIOTimeout(      IN LPVOID pReader,
                               IN DWORD dwTimeout );
```

Время выполнения функций библиотеки складывается из следующих компонентов:

- передача команды из компьютера в считыватель
- выполнение команды в считывателе
- передача ответа из считывателя в компьютер
- накладные расходы операционной системы

При создании интерфейса устанавливается таймаут, равный 1000 мс. С помощью этой функции можно изменить интервал ожидания выполнения функций.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

dwTimeout – значение таймаута в миллисекундах. Если это значение равно 0, то устанавливается таймаут по умолчанию (1000 мс).

Минимальное отличное от 0 значение таймаута 50 мс.

Максимальное значение таймаута 86400000 мс.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.1.10 CLSCRF_GetLastInternalError

```
LONG CLSCRF_GetLastInternalError(      IN  LPVOID pReader,
                                       OUT LPBYTE pbError );
```

Если предыдущая функция данной библиотеки завершилась с кодом 0x80100001, значит произошла ошибка выполнения операции внутри считывателя. Данная функция выдает значение кода этой внутренней ошибки. Наиболее часто из внутренних ошибок встречается 0xFF - карта не отвечает.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

pbError – адрес переменной, в которую будет помещено значение кода внутренней ошибки считывателя.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.2 Функции управления считывателем

4.2.1 CLSCRF_GetState

LONG CLSCRF_GetState(IN LPVOID pReader,
OUT LPDWORD pdwState);

Выдает состояние считывателя.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pdwState – ссылка на переменную, в которую будет помещено значение состояния устройства:
биты 31-16 - резерв,
бит 15 - микросхема-считыватель включена,
бит 14 - электромагнитное поле включено,
бит 13 - считыватель поддерживает режимы ICODE,
бит 12 - считыватель поддерживает режимы 14443-B,
биты 11-7 - резерв,
биты 6-4 - текущий режим электромагнитного поля:
000 - ISO 14443-A (скорость см. биты 3-0)
001 - ISO 14443-B (скорость см. биты 3-0)
100 - ICODE SLI ISO 15693
110 - ICODE EPC ISO 18000-3
111 - ICODE UID ISO 18000-3
биты 3-2 - текущая скорость приема в режимах ISO 14443 (поток данных от карты к считывателю):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод
биты 1-0 - текущая скорость передачи в режимах ISO 14443 (поток данных от считывателя к карте):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.2 CLSCRF_Mfrc_On

LONG CLSCRF_Mfrc_On(IN LPVOID pReader);

Включает микросхему-считыватель.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

Примечание. При включенной микросхеме-считывателе в рабочем режиме

устройство потребляет ток до 120 mA.

4.2.3 CLSCRF_Mfrc_Off

LONG CLSCRF_Mfrc_Off(IN LPVOID pReader);

Выключает микросхему-считыватель.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении..

Примечание. При выключенной микросхеме-считывателе устройство воспринимает команды, но не выполняет те из них, которые касаются этой микросхемы. В этом состоянии устройство потребляет ток около 50 mA.

4.2.4 CLSCRF_Get_Mfrc_Version

LONG CLSCRF_Get_Mfrc_Version(IN LPVOID pReader,
OUT LPBYTE pbMfrcVersion);

Выдает идентификационный код продукта микросхемы-считывателя и версии микросхемы и микропрограммы.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pbMfrcVersion – адрес массива, в который будет помещено 6 байтов версии считывателя, из которых
4 байта - код продукта микросхемы-считывателя,
1 байт - версия микросхемы-считывателя,
1 байт - версия микропрограммы считывателя.
Примеры:
30 88 FE 03 04 20 - считыватель базируется на микросхеме MF RC530,
30 CC FF 0F 04 20 - считыватель базируется на микросхеме MF RC531,
30 FF FF 0F 04 20 - считыватель базируется на микросхеме CL RC632,
где версия микросхемы 4, версия микропрограммы 2.0.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.5 CLSCRF_Get_Mfrc_Serial_Number

LONG CLSCRF_Get_Mfrc_Serial_Number(IN LPVOID pReader,
OUT LPBYTE pbMfrcSN);

Выдает серийный (уникальный) номер микросхемы-считывателя.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pbMfrcSN – адрес массива, в который будет помещено 4 байта серийного номера.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.6 CLSCRF_Mfrc_Rf_Off_On

```
LONG CLSCRF_Mfrc_Rf_Off_On(  IN LPVOID pReader,
                             IN USHORT usDelay );
```

Сбрасывает или выключает электромагнитное поле (ЭМП) считывателя. Сброс ЭМП необходим для рестарта микросхемы транспондера.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
usDelay – интервал времени [мс], в течение которого ЭМП выключено.
Если этот параметр равен 0, то ЭМП остается выключенным.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.7 CLSCRF_Mfrc_Set_Rf_Mode

```
LONG CLSCRF_Mfrc_Set_Rf_Mode( IN LPVOID pReader,
                              IN BYTE  bRfMode );
```

Устанавливает режим модуляции электромагнитного поля считывателя на передачу команды и последующий прием ответа от карты.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bRfMode – байт режима модуляции ЭМП:
бит 7 - резерв
биты 6-4 - устанавливаемый режим электромагнитного поля:
000 - ISO 14443-A (скорость см. биты 3-0)
001 - ISO 14443-B (скорость см. биты 3-0)
100 - ICODE SLI ISO 15693
110 - ICODE EPC ISO 18000-3
111 - ICODE UID ISO 18000-3
биты 3-2 - устанавливаемая скорость приема в режимах ISO 14443 (поток данных от карты к считывателю):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод
биты 1-0 - устанавливаемая скорость передачи в режимах ISO 14443 (поток данных от считывателя к карте):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод
Возвращаемое значение:
0 – успешное выполнение,

иначе – ошибка при выполнении.

4.2.8 CLSCRF_Sound

```
LONG CLSCRF_Sound( IN LPVOID pReader,
                   IN BYTE  nBeepCount );
```

Подает звуковой сигнал.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
nBeepCount – количество одиночных сигналов длительностью 100 мс с промежутками такой же длительности.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.9 CLSCRF_Led

```
LONG CLSCRF_Led(  IN LPVOID pReader,
                  IN BYTE  bBlinkColor,
                  IN BYTE  bBlinkCount,
                  IN BYTE   bPostColor );
```

Мигает двухцветным светодиодом, затем гасит или зажигает его постоянно. Частота миганий - 2 Гц.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bBlinkColor – цвет мигающего светодиода:
0 - не мигать,
1 - мигать красным,
2 - мигать зеленым,
3 - мигать желто-оранжевым;
bBlinkCount – количество миганий (если 0 - не мигать);
bPostColor – последующее состояние светодиода:
0 - погасить,
1 - зажечь красным,
2 - зажечь зеленым,
3 - зажечь желто-оранжевым.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.10 CLSCRF_UART_Baudrate

```
LONG CLSCRF_UART_Baudrate(  IN LPVOID pReader,
                            IN DWORD dwBaudrate );
```

Устанавливает скорость обмена со считывателем по COM-порту.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

dwBaudrate – новая скорость обмена по COM-порту; возможные значения:
9600,
14400,
19200,
38400,
57600,
115200,
1500000.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.11 CLSCRF_EraseFlash

LONG CLSCRF_EraseFlash(IN LPVOID pReader);

Очищает flash-память считывателя для последующей записи блоков данных и ключей.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.12 CLSCRF_WriteFlashValue

LONG CLSCRF_WriteFlashValue(IN LPVOID pReader,
IN DWORD dwAddress,
IN LPBYTE pbValue);
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwAddress – адрес блока flash-памяти, в который следует записать данные (0..239);
pbValue – указатель на массив данных (16 байт), содержащий данные для записи в блок flash-памяти считывателя.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.13 CLSCRF_CheckFlashValueFilled

LONG CLSCRF_CheckFlashValueFilled(IN LPVOID pReader,
IN DWORD dwFlashAddr,
OUT LPBYTE pbStatus);
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwFlashAddr – адрес блока данных во flash {0 ...239};
pbStatus – код возврата:
0x00 – результат команды успешный и блок заполнен;

0xFF – результат команды успешный и блок пуст, другие значения – ошибка при выполнении команды.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.3 Функции управления защищенным режимом

4.3.1 CLSCRF_Crypto_SaveAESKeysToFile

```
LONG CLSCRF_Crypto_SaveAESKeysToFile( IN LPSTR lsFileName,
                                      IN LPBYTE pbIV,
                                      IN LPBYTE pbKeys,
                                      IN DWORD dwKeysCount,
                                      IN LPSTR lsPassword);
```

lsFileName – ASCII-строка с именем файла, должна заканчиваться нулем;

pbIV – указатель на массив (16 байт), содержащий вектор инициализации записываемый в файл ключей и служащий для его за/расшифровки;

pbKeys – указатель на массив ключей (каждый по 16 байт), записываемый в файл;

dwKeysCount – количество записываемых в файл ключей;

lsPassword – строка с паролем, используемым для шифрации файла ключей (ASCII-строка, завершающаяся нулем).

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.3.2 CLSCRF_Crypto_GenerateAndSaveAESKeysToFile

```
LONG CLSCRF_Crypto_GenerateAndSaveAESKeysToFile(IN LPSTR lsFileName,
                                                IN DWORD dwKeysCount,
                                                IN LPSTR lsPassword);
```

lsFileName – ASCII-строка с именем файла, должна заканчиваться нулем;

dwKeysCount – количество генерируемых и записываемых в файл ключей;

lsPassword – строка с паролем, используемым для шифрации файла ключей (ASCII-
строка, завершающаяся нулем).

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.3.3 CLSCRF_Crypto_LoadAESKeysFromFile

LONG CLSCRF_Crypto_LoadAESKeysFromFile(IN LPVOID pReader,
IN LPSTR lsFileName,
IN LPSTR lsPassword,
OUT LPDWORD ldKeysCount);

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
lsFileName – ASCII-строка с именем файла, должна заканчиваться нулем;
lsPassword – строка с паролем, которым зашифрован файл ключей (ASCII-строка, завершающаяся нулем);
ldKeysCount – количество ключей, которое требуется загрузить и количество по факту загруженных ключей после операции.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.3.4 CLSCRF_Crypto_WriteFlash_AESKeys

LONG CLSCRF_Crypto_WriteFlash_AESKeys(IN LPVOID pReader,
IN DWORD dwStartKey,
IN DWORD dwKeysCount);

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwStartKey – номер ключа, с которого требуется начать запись;
dwKeysCount – количество ключей, которое требуется записать во flash-память считывателя (запись произведется в аналогичные блоки flash-памяти).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.3.5 CLSCRF_Crypto_AuthenticateReader

LONG CLSCRF_Crypto_AuthenticateReader(IN LPVOID pReader,
IN BYTE ucAuthType,
IN DWORD dwKeyNumber);

Производит аутентификацию (сброс аутентификации) считывателя и хоста по указанному ключу и включает режим шифрованного обмена между ними. Ключ должен быть предварительно загружен во flash считывателя и во внутренний массив ключей библиотеки. Номера ключей должны совпадать.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucAuthType – тип аутентификации (0x00 – сброс аутентификации, 0x01 - аутентификация);
dwKeyNumber – номер блока flash-памяти считывателя и одновременно номер записи во внутреннем массиве ключей библиотеки, из которого требуется взять значение в качестве ключа для аутентификации и последующего вычисления сессионного ключа шифрования данных в защищенном режиме.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.3.6 CLSCRF_Crypto_SetEncryptionMode

LONG CLSCRF_Crypto_SetEncryptionMode(IN LPVOID pReader,
IN BOOL fEncryptionMode);
Принудительно устанавливает режим обмена на стороне хоста.
Вызывается в случае, если считыватель вышел в открытый режим обмена и требуется вручную изменить режим обмена хоста.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
fEncryptionMode – режим обмена (1 - шифрованный, 0 - открытый)
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.3.7 CLSCRF_Crypto_GetEncryptionMode

LONG CLSCRF_Crypto_GetEncryptionMode(IN LPVOID pReader,
OUT PBOOL pfEncryptionMode);
Считывает текущий режим обмена на стороне хоста.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pfEncryptionMode – указатель на переменную типа BOOL, в которую будет помещён прочитанный режим обмена (1 - шифрованный, 0 - открытый)
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4 Функции работы со списком карт

4.4.1 CLSCRF_Cards_Add

LONG CLSCRF_Cards_Add(IN CLSCRF_CARD* pCard);
Добавляет указатель на объект карты в список карт.
pCard – указатель на объект карты.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.2 CLSCRF_Cards_GetFirst

LONG CLSCRF_Cards_GetFirst(OUT CLSCRF_CARD** ppCard);

Возвращает указатель на первую карту в списке карт. Возвращает NULL, если такой карты не в списке.

ppCard – указатель на указатель на объект карты.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.4.3 CLSCRF_Cards_GetLast

LONG CLSCRF_Cards_GetLast(OUT CLSCRF_CARD** ppCard);

Возвращает указатель на последнюю карту в списке карт. Возвращает NULL, если такой карты не в списке.

ppCard – указатель на указатель на объект карты.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.4.4 CLSCRF_Cards_Get

LONG CLSCRF_Cards_Get(IN int iCardIndex, OUT CLSCRF_CARD** ppCard);

Возвращает указатель на карту с заданным индексом в списке карт. Возвращает NULL, если такой карты не в списке.

iCardIndex – индекс карты в списке карт;

ppCard – указатель на указатель на объект карты.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.4.5 CLSCRF_Cards_GetCount

LONG CLSCRF_Cards_GetCount(OUT int* piCardsCount);

Возвращает количество карт в списке.

piCardsCount – указатель на неременную, принимающую значение количества.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.4.6 CLSCRF_Cards_Clear

LONG CLSCRF_Cards_Clear();

Очищает список карт (но не удаляет из памяти объекты карт).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.7 CLSCRF_Cards_SetCurrent

LONG CLSCRF_Cards_SetCurrent(IN CLSCRF_CARD* pCard);
Устанавливает карту, как текущую.
pCard – указатель на объект карты.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.8 CLSCRF_Cards_GetCurrent

LONG CLSCRF_Cards_GetCurrent(OUT CLSCRF_CARD** ppCard);
Возвращает указатель на текущую карту.
ppCard – указатель на указатель на объект карты.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.9 CLSCRF_Cards_Search

LONG CLSCRF_Cards_Search(IN int UID_Len,
IN PBYTE pUID,
OUT CLSCRF_CARD** ppCard);
Выполняет поиск карты по заданному UID. Возвращает NULL, если такой карты не в списке.
UID_Len – длина UID карты;
pUID – указатель на массив байт с UID карты;
ppCard – указатель на указатель на объект карты.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.10 CLSCRF_Cards_CreateNew_A

LONG CLSCRF_Cards_CreateNew_A(IN BYTE bRfMode,
IN PBYTE pbATQ,
IN PBYTE pbSAK,
IN PBYTE pbUID,
IN DWORD UID_Len,
OUT CLSCRF_CARD** ppCard);
Создаёт объект карты ISO14443A.

bRfMode – режим RF;
pbATQ – указатель на массив, содержащий ATQ карты;
pbSAK – указатель на массив, содержащий SAK карты;
pbUID – указатель на массив байт с UID карты;
UID_Len – длина UID карты;
ppCard – указатель на указатель на объект карты.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.11 CLSCRF_Cards_CreateNew_B

```
LONG CLSCRF_Cards_CreateNew_B(      IN BYTE bRfMode,
                                    IN DWORD* pdwPUPI,
                                    IN DWORD* pdwAppData,
                                    IN DWORD* pdwProtInfo,
                                   IN int   CardCount,
                                    OUT CLSCRF_CARD**  ppCard );
```

Создаёт объекты карт ISO14443B и автоматически добавляет их в список карт.
bRfMode – режим RF;
pdwPUPI – указатель на массив, содержащий PUPI карт;
pdwAppData – указатель на массив, содержащий AppData карт;
pdwProtInfo – указатель на массив, содержащий ProtInfo карт;
CardCount – количество создаваемых объектов;
ppCard – указатель на указатель на объект последней созданной карты.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.12 CLSCRF_Cards_CreateNew15693

```
LONG CLSCRF_Cards_CreateNew15693(   IN BYTE bRfMode,
                                    IN BYTE* pbData,
                                   IN DWORD dwCount,
                                    OUT CLSCRF_CARD**  ppCard );
```

Создаёт объекты карт ISO15693 и автоматически добавляет их в список карт.
bRfMode – режим RF;
pbData – указатель на массив, содержащий DSFID, UID карт;
dwCount – количество карт;
ppCard – указатель на указатель на объект последней созданной карты.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.13 CLSCRF_Cards_CreateNewEPCUID

```
LONG CLSCRF_Cards_CreateNewEPCUID(      IN BYTE bRfMode,
                                        IN DWORD dwUID_Len,
                                        IN BYTE* pbData,
                                      IN DWORD dwCount,
                                       OUT CLSCRF_CARD** ppCard );
```

Создаёт объекты карт ISO15693 и автоматически добавляет их в список карт.

bRfMode – режим RF;
dwUID_Len – длина UID карты;
pbData – указатель на массив, содержащий UID, CRC16 карт;
dwCount – количество карт;
ppCard – указатель на указатель на объект последней созданной карты.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.14 CLSCRF_Cards_Delete

```
LONG CLSCRF_Cards_Delete(           IN int UID_Len,
                                    IN PBYTE pUID );
```

Удаляет карту по заданному UID.

UID_Len – длина UID карты;
pUID – указатель на массив байт с UID карты.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.15 CLSCRF_CARD

```
typedef struct _CLSCRF_CARD
{
  unsigned int  Timeout;  // etu =  128/fc = 9,4 us
  unsigned int  CardType;
  int         UID_Len;
  unsigned char UID[32];
  union {
    struct {    // 14443-A
      unsigned char ATQA [2];
      unsigned char SAK;
      unsigned char ATS  [37];    // if  14443-4 only
    };
    struct {    // 14443-B
      unsigned char AppData [4];
      unsigned char  ProtInf[3];
```

```
        unsigned char Attrib  [33];
      };
      struct {    // 15693
        unsigned char DSFID;
        unsigned char SLI_Data[39]; // not used
      };
      struct {    //ICODE EPC
        unsigned char CRC16[2];
        unsigned char EPC_Data[38]; // not used
      };
    } ATQ;
    unsigned char CID;
    unsigned char Active;
    unsigned char RfMode;
    unsigned char BlockNumberBit;

} CLSCRF_CARD;
```

- информация о карте:

Timeout – таймаут ответа карты = <период> / 9,4us (6800 для периода 64мс – по умолчанию);

CardType – тип карты:

```
#define CLSCRF_ICC_TYPE_14443A      0x01
#define CLSCRF_ICC_TYPE_14443B      0x02
#define CLSCRF_ICC_TYPE_15693       0x04
#define CLSCRF_ICC_TYPE_ICODE_1     0x08
#define CLSCRF_ICC_TYPE_ICODE_EPC   0x20
#define CLSCRF_ICC_TYPE_ICODE_UID   0x40
#define CLSCRF_ICC_TYPE_NEXT_RFU    0x80
```

UID_Len – длина идентификатора карты;
UID – идентификатор карты;
ATQ.ATQA – соответственно, ATQ карты ISO14443A;
ATQ.SAK – соответственно, SAK карты ISO14443A;
ATQ.ATS – соответственно, ATS карты ISO14443A;
ATQ.AppData – данные приложения карты ISO14443B;
ATQ.ProtInf – информация о протоколе карты ISO14443B;
ATQ.Attrib – аттрибуты карты ISO14443B;
ATQ.DSFID – DSFID карты ISO15693;
ATQ.SLI_Data – данные карты ISO15693;
ATQ.CRC16 – CRC16 карты ICode EPC;
ATQ.EPC_Data – код карты ICode EPC;
CID – сессионный идентификатор карты;
Active – признак активности карты;
RfMode – режим коммуникации карты;
BlockNumberBit – бит чётности номера блока.

4.5 Функции управления картами типа A стандарта ISO 14443

4.5.1 CLSCRF_Activate_Idle_A

LONG CLSCRF_Activate_Idle_A(IN LPVOID pReader,
OUT LPBYTE pbATQ,
OUT LPBYTE pbSAK,
OUT LPBYTE pbUID,
IN OUT LPDWORD pdwUIDLength);

Активирует карту 14443-A из состояния IDLE (см. ISO 14443-3 п.6.3).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pbATQ – ссылка на массив (2 байта), в которые будет помещен ATQ карты;
pbSAK – ссылка на массив (1 байт), в который будет помещен SAK карты;
pbUID – ссылка на массив не менее 10 байтов, в который будет помещен уникальный идентификатор карты (UID);
pdwUIDLength – ссылка на переменную, которая перед вызовом функции должна содержать размер массива pbUID, а на выходе будет содержать действительную длину уникального идентификатора карты, помещенного в массив pbUID (возможные значения - 4, 7 или 10).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.5.2 CLSCRF_Halt_A

LONG CLSCRF_Halt_A(IN LPVOID pReader);

Деактивирует карту 14443-A в состояние HALT (см. ISO 14443-3 п.6.3).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.5.3 CLSCRF_Activate_Wakeup_A

LONG CLSCRF_Activate_Wakeup_A(IN LPVOID pReader,
IN LPBYTE pbUID,
IN DWORD UIDLength,
OUT LPBYTE pbATQ,
OUT LPBYTE pbSAK);

Активирует карту 14443-A из состояния HALT (см. ISO 14443-3 п.6.3).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

pbUID – ссылка на массив не менее 10 байтов, в котором перед вызовом функции должен быть помещен уникальный идентификатор карты (UID);
UIDLength – длина уникального идентификатора карты, который содержится в массиве pbUID;
pbATQ – ссылка на массив (2 байта), в которые будет помещен ATQ карты;
pbSAK – ссылка на массив (1 байт), в который будет помещен SAK карты.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.5.4 CLSCRF_ISO14443A_4_RATS

```
LONG CLSCRF_ISO14443A_4_RATS( IN LPVOID pReader,
                              IN BYTE ucCID,
                              OUT LPBYTE pbATS );
```

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucCID – логический идентификатор карты для обмена по протоколу T = CL (см. ISO 14443-3 7.10.6);
pbATS – указатель на массив данных, в который будет записан прочитанный ATS карты.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.5.5 CLSCRF_ISO14443A_4_PPS

```
LONG CLSCRF_ISO14443A_4_PPS( IN LPVOID pReader,
                             IN BYTE ucCID,
                             IN BYTE ucBaudrate );
```

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucCID – логический идентификатор карты для обмена по протоколу T = CL (см. ISO 14443-3 7.10.6);
ucBaudrate – устанавливаемая скорость обмена с картой:

биты 7-4 - резерв
биты 3-2 - устанавливаемая скорость приема в режимах ISO 14443 (поток данных от карты к считывателю):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод
биты 1-0 - устанавливаемая скорость передачи в режимах ISO 14443 (поток данных от считывателя к карте):
00 - 106 кбод
01 - 212 кбод
10 - 424 кбод
11 - 848 кбод

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.6 Функции управления картами типа B стандарта ISO 14443

4.6.1 CLSCRF_Activate_Idle_B

```
LONG CLSCRF_Activate_Idle_B( IN    LPVOID  pReader,
                             IN    BYTE    bAfi,
                             IN    BYTE    bSni,
                             OUT   LPDWORD pdwPUPI,
                             OUT   LPDWORD pdwAppData,
                             OUT   LPDWORD pdwProtInfo,
                             IN OUT LPDWORD pdwCardCount );
```

Активирует карты 14443-B из состояния IDLE (см. ISO 14443-3 п.7.7).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

bAfi – идентификатор семейства приложений;

bSni – код количества временнЫх слотов:
0 => SlotQuantity = 1 слот;
1 => SlotQuantity = 2 слота;
2 => SlotQuantity = 4 слота;
3 => SlotQuantity = 8 слотов;
4 => SlotQuantity = 16 слотов;

pdwPUPI – ссылка на массив типа DWORD размером 16, в который будут помещены псевдоуникальные идентификаторы обнаруженных карт, по 4 байта для одной карты на элемент массива;

pdwAppData – ссылка на массив типа DWORD размером 16, в который будет помещена информация о приложениях в обнаруженных картах, по 4 байта для одной карты на элемент массива;

pdwProtInfo – ссылка на массив типа DWORD размером 16, в который будет помещена информация о протоколах обнаруженных карт, по 3 байта для одной карты на элемент массива в битах с 0 по 23;

pdwCardCount – ссылка на переменную, которая перед вызовом функции должна содержать размер массивов pdwPUPI, pdwAppData и pdwProtInfo, а на выходе будет содержать количество обнаруженных карт (от 0 до 16).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.6.2 CLSCRF_Attrib_B

```
LONG CLSCRF_Attrib_B( IN    LPVOID  pReader,
                      IN    LPBYTE  pbPUPI,
```

```
                                    IN    BYTE    bParam1,
                                    IN    BYTE    bParam2,
                                    IN    BYTE    bParam3,
                                    IN    BYTE    bParam4,
                                    IN OUT LPBYTE pbHigherLayerBuf,
                                    IN    DWORD   dwHLBufSize,
                                    IN OUT LPDWORD pdwHLInfLength,
                                    OUT LPBYTE pbMbliCid );
```

Выбирает карту 14443-B (см. ISO 14443-3 п.7.10) и назначает ей логический идентификатор CID для обмена по протоколу T = CL (см. ISO 14443-4).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

pbPUPI – ссылка на массив (4 байта), в котором находится псевдоуникальный идентификатор выбираемой карты;

bParam1 – формат кадра от считывателя к карте;

bParam2 – максимальные размер кадра и скорости обмена считывателя;

bParam3 – поддержка считывателем протокола ISO 14443-4;

bParam4 – логический идентификатор карты (CID);

pbHigherLayerBuf – буфер для Higher layer – INF на входе и Higher layer Response на выходе;

dwHLBufSize – длина массива pbHigherLayerBuf;

pdwHLInfLength – ссылка на переменную, в которой на входе указывают длину Higher layer – INF в массиве pbHigherLayerBuf а на выходе в ней будет помещена длина Higher layer Response, записанного в массив pbHigherLayerBuf;

pbMbliCid – ссылка на переменную, в которую будет помещено значение MBLI CID.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.6.3 CLSCRF_Halt_B

```
LONG CLSCRF_Halt_B( IN LPVOID pReader,
                    IN LPBYTE pbPUPI );
```

Деактивирует карту 14443-B в состояние HALT (см. ISO 14443-3 п.7.12).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

pbPUPI – ссылка на массив (4 байта), в котором находится псевдоуникальный идентификатор деактивируемой карты.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.6.4 CLSCRF_Activate_Wakeup_B

```
LONG CLSCRF_Activate_Wakeup_B( IN    LPVOID  pReader,
                               IN    BYTE    bAfi,
```

```
                                    IN    BYTE    bSni,
                                    OUT LPDWORD pdwPUPI,
                                    OUT LPDWORD pdwAppData,
                                    OUT LPDWORD pdwProtInfo,
                                    IN OUT LPDWORD pdwCardCount );
```

Активирует карты 14443-B из состояния HALT (см. ISO 14443-3 п.7.7).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bAfi – идентификатор семейства приложений;
bSni – код количества временнЫх слотов: 0 => SlotQuantity = 1 слот;
1 => SlotQuantity = 2 слота;
2 => SlotQuantity = 4 слота;
3 => SlotQuantity = 8 слотов;
4 => SlotQuantity = 16 слотов;
pdwPUPI - ссылка на массив типа DWORD размером 16, в который будут помещены псевдоуникальные идентификаторы обнаруженных карт, по 4 байта для одной карты на элемент массива;
pdwAppData – ссылка на массив типа DWORD размером 16, в который будет помещена информация о приложениях в обнаруженных картах, по 4 байта для одной карты на элемент массива;
pdwProtInfo – ссылка на массив типа DWORD размером 16, в который будет помещена информация о протоколах обнаруженных карт, по 3 байта для одной карты на элемент массива в битах с 0 по 23;
pdwCardCount – ссылка на переменную, которая перед вызовом функции должна содержать размер массивов pdwPUPI, pdwAppData и pdwProtInfo, а на выходе будет содержать количество обнаруженных карт (от 0 до 16).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.7 Функции управления метками стандарта ISO 15693

4.7.1 CLSCRF_FindAllTags_15693

```
LONG CLSCRF_FindAllTags_15693 ( IN    LPVOID  pReader,
                                IN    BYTE    bFlags,
                                IN    BYTE    bAfi,
                                OUT LPBYTE pbRecvBuffer,
                                IN OUT LPDWORD pdwRecvLength );
```

Производит рекурсивную инвентаризацию меток стандарта ISO 15693 (см. ISO 15693-3 п.10.3.1).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – установить в 0x00 (зарезервировано на будущее);
bAfi – идентификатор семейства приложений;

pbRecvBuffer – массив для ответа от считывателя;
pdwRecvLength – ссылка на переменную, которая перед вызовом функции должна содержать размер массива pbRecvBuffer, а на выходе будет содержать количество принятых от считывателя байтов.

Для каждой найденной метки в массиве pbRecvBuffer расположена следующая информация:

RetCode – код завершения запроса в этом слоте - 1 байт
Count – количество байтов, полученных от метки - 1 байт

Если Count отличен от 0, то далее следует ответ от метки (см. ISO 15693 п.10.3.1):

Flags - 1 байт
DSFID - 1 байт
UID - 8 байтов

Если RetCode отличен от 0, то далее может присутствовать

CRC16 - 2 байта

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.7.2 CLSCRF_Inventory_15693

```
LONG CLSCRF_Inventory_15693( IN    LPVOID  pReader,
                             IN    BYTE    bFlags,
                             IN    BYTE    bInventory,
                             IN    BYTE    bAfi,
                             IN    BYTE    bMaskLen,
                             IN    LPCBYTE pbMaskVal,
                             OUT   LPBYTE  pbRecvBuffer,
                             IN OUT LPDWORD pdwRecvLength );
```

Производит единичную инвентаризацию меток стандарта ISO 15693 (см. ISO 15693-3 п.10.3.1).
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bInventory – код команды Inventory (всегда равен 0x01);
bAfi – идентификатор семейства приложений;
bMaskLen – количество битов маски;
pbMaskVal – массив байтов, содержащий маску;
pbRecvBuffer – массив для ответа от считывателя;
pdwRecvLength – ссылка на переменную, которая перед вызовом функции должна содержать размер массива pbRecvBuffer, а на выходе будет содержать количество принятых от считывателя байтов по 1 или 16 временнЫм слотам.

Для каждого временнОго слота в массиве pbRecvBuffer расположена следующая информация:

RetCode – код завершения запроса в этом слоте - 1 байт
Count – количество байтов, полученных от метки - 1 байт

Если Count отличен от 0, то далее следует ответ от метки (см. ISO 15693 п.10.3.1):

Flags - 1 байт
DSFID - 1 байт
UID - 8 байтов

Если RetCode отличен от 0, то далее может присутствовать

CRC16 - 2 байта

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.7.3 CLSCRF_Stay_Quiet_15693

```
LONG CLSCRF_Stay_Quiet_15693( IN LPVOID pReader,
                              IN BYTE   bFlags,
                              IN BYTE   bStayQuiet,
                              IN LPBYTE pbUID );
```

Переводит метку в состояние Quiet (см. ISO 15693-3 п.10.3.2).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bStayQuiet – код команды Stay Quiet (всегда равен 0x02);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.7.4 CLSCRF_Select_15693

```
LONG CLSCRF_Select_15693( IN    LPVOID pReader,
                          IN    BYTE   bFlags,
                          IN    BYTE   bSelect,
                          IN    LPBYTE pbUID,
                          OUT LPBYTE pbFlags,
                          OUT LPBYTE pbErrorCode );
```

Переводит метку в состояние Selected (см. ISO 15693-3 п.10.4.6).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bSelect – код команды Select (всегда равен 0x25);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,

иначе – ошибка при выполнении.

4.7.5 CLSCRF_ResetToReady_15693

LONG CLSCRF_ResetToReady_15693(IN LPVOID pReader,
IN BYTE bFlags,
IN BYTE bResetToReady,
IN LPBYTE pbUID,
OUT LPBYTE pbFlags,
OUT LPBYTE pbErrorCode);

Переводит метку в состояние Ready (см. ISO 15693-3 п.10.4.7).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1)
(рекомендуется включить флаг Select_flag);
bResetToReady – код команды Reset to ready (всегда равен 0x26);
pbUID – ссылка на массив (8 байтов), в котором находится
уникальный идентификатор метки, может быть равен NULL при включенном флаге Select_flag (рекомендуется);
pbFlags – ссылка на переменную, в которую будут
помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет
помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.8 Функции обмена данными с картами Mifare Classic

4.8.1 (обратная совместимость) CLSCRF_MifareStandard_HostCodeKey

LONG CLSCRF_MifareStandard_HostCodeKey(IN LPVOID pReader,
IN LPBYTE pbUncoded,
OUT LPBYTE pbCoded);

Кодирует ключ Mifare Standard.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pbUncoded – массив (6 байтов) исходного (некодированного) ключа;
pbCoded – массив (12 байтов) кодированного ключа.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.8.2 (обратная совместимость) CLSCRF_MifareStandard_AuthKey

```
LONG CLSCRF_MifareStandard_AuthKey( IN LPVOID pReader,
                                    IN BYTE  bKeyType,
                                    IN LPBYTE pbUID,
                                    IN DWORD  dwSector,
                                    IN LPBYTE pbCodedKey );
```

Производит аутентификацию сектора непосредственно заданным ключом.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bKeyType – тип ключа: 0x60 - Key A,
0x61 - Key B;
pbUID – массив (4 байта), содержащий уникальный идентификатор карты (для карт с 7-байтовым UID[0..6], поддерживающих протокол Mifare Standard, в качестве snr[0..3] использовать байты UID[3..6]);
dwSector – номер аутентифицируемого сектора;
pbCodedKey – массив (12 байтов) кодированного ключа.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.8.3 CLSCRF_MifareStandard_AuthKeyDirect

```
LONG CLSCRF_MifareStandard_AuthKeyDirect(  IN LPVOID pReader,
                                           IN BYTE  bKeyType,
                                           IN LPBYTE pbUID,
                                           IN DWORD  dwSector,
                                           IN LPBYTE pbKey );
```

Производит аутентификацию сектора непосредственно заданным ключом.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bKeyType – тип ключа: 0x60 - Key A,
0x61 - Key B;
pbUID – массив (4 байта), содержащий уникальный идентификатор карты (для карт с 7-байтовым UID[0..6], поддерживающих протокол Mifare Standard, в качестве snr[0..3] использовать байты UID[3..6]);
dwSector – номер аутентифицируемого сектора;
pbKey – массив (6 байт) ключа.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.8.4 CLSCRF_MifareStandard_WriteKeyToE2

```
LONG CLSCRF_MifareStandard_WriteKeyToE2( IN LPVOID pReader,
                                         IN BYTE  bKeyType,
```

IN DWORD dwSector,
IN LPBYTE pbUncoded);

Записывает ключ заданного типа для заданного сектора в EEPROM считывателя.

Считыватель поддерживает в данном режиме работу с секторами с индексами 0..15, то есть с первым килобайтом памяти карты.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bKeyType – тип ключа: 0x60 - Key A,
0x61 - Key B;
dwSector – номер сектора, для которого записывается ключ;
pbUncoded – массив (6 байтов) некодированного ключа.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.8.5 CLSCRF_MifareStandard_AuthE2

LONG CLSCRF_MifareStandard_AuthE2(IN LPVOID pReader,
IN BYTE bKeyType,
IN LPBYTE pbUID,
IN DWORD dwSector);

Производит аутентификацию сектора ключом, находящимся в EEPROM считывателя.

Считыватель поддерживает в данном режиме работу с секторами с индексами 0..15, то есть с первым килобайтом памяти карты.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bKeyType – тип ключа: 0x60 - Key A,
0x61 - Key B;
pbUID – массив (4 байта), содержащий уникальный идентификатор карты (для карт с 7-байтовым UID[0..6], поддерживающих протокол Mifare Standard, в качестве snr[0..3] использовать байты UID[3..6]);
dwSector – номер аутентифицируемого сектора.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.8.6 CLSCRF_MifareStandard_Read

LONG CLSCRF_MifareStandard_Read(IN LPVOID pReader,
IN DWORD dwSector,
IN DWORD dwBlock,
OUT LPBYTE pbRecvBuffer);

Читает 16 байтов из заданного блока в заданном секторе.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwSector – номер сектора, содержащего читаемый блок;
dwBlock – номер читаемого блока;
pbRecvBuffer – массив (не менее 16 байтов) для прочитанного блока.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

Перед вызовом этой функции должна быть проведена аутентификация сектора.

4.8.7 CLSCRF_MifareStandard_Write

```
LONG CLSCRF_MifareStandard_Write( IN LPVOID pReader,
                                  IN DWORD  dwSector,
                                  IN DWORD  dwBlock,
                                  IN LPBYTE pbSendBuffer );
```

Записывает 16 байтов в заданный блок в заданном секторе.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwSector – номер сектора, содержащего записываемый блок;
dwBlock – номер записываемого блока;
pbSendBuffer – массив 16 байтов данных, записываемых в блок.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

Перед вызовом этой функции должна быть проведена аутентификация сектора.

4.8.8 CLSCRF_MifareStandard_Decrement

```
LONG CLSCRF_MifareStandard_Decrement( IN LPVOID pReader,
                                      IN DWORD  dwSector,
                                      IN DWORD  dwSourceBlock,
                                      IN DWORD  dwValue,
                                      IN DWORD  dwTargetBlock );
```

Уменьшает значение блока типа Value.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwSector – номер сектора, содержащего обрабатываемый блок;
dwSourceBlock – номер исходного блока;
dwValue – вычитаемое, на которое уменьшается значение;
dwTargetBlock – номер блока-результата, может быть равен dwSourceBlock.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

Перед вызовом этой функции должна быть проведена аутентификация сектора. Блоки dwSourceBlock и dwTargetBlock должны принадлежать сектору dwSector.

4.8.9 CLSCRF_MifareStandard_Increment

```
LONG CLSCRF_MifareStandard_Increment( IN LPVOID pReader,
                                      IN DWORD  dwSector,
                                      IN DWORD  dwSourceBlock,
                                      IN DWORD  dwValue,
                                      IN DWORD  dwTargetBlock );
```

Увеличивает значение блока типа Value.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwSector – номер сектора, содержащего обрабатываемый блок;
dwSourceBlock – номер исходного блока;
dwValue – слагаемое, на которое увеличивается значение;
dwTargetBlock – номер блока-результата, может быть равен dwSourceBlock.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

Перед вызовом этой функции должна быть проведена аутентификация сектора. Блоки dwSourceBlock и dwTargetBlock должны принадлежать сектору dwSector.

4.8.10 CLSCRF_MifareStandard_Restore

```
LONG CLSCRF_MifareStandard_Restore( IN LPVOID pReader,
                                    IN DWORD  dwSector,
                                    IN DWORD  dwSourceBlock,
                                    IN DWORD  dwTargetBlock );
```

Копирует значение из одного блока типа Value в другой в заданном секторе.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwSector – номер сектора, содержащего оба блока;
dwSourceBlock – номер исходного блока;
dwTargetBlock – номер блока-результата.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

Перед вызовом этой функции должна быть проведена аутентификация сектора. Блоки dwSourceBlock и dwTargetBlock должны принадлежать сектору dwSector.

4.8.11 CLSCRF_MifareStandard_EV1_PersonalizeUid

```
LONG CLSCRF_MifareStandard_EV1_PersonalizeUid( IN LPVOID pReader,
                                               IN BYTE
ucUidMode );
```

Команда выполняется один раз в период жизни карты, устанавливая один из режимов UID.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

ucUidMode – режим UID (режим, в который переводится карта)
Допустимо любое из значений:
UIDF0 = 0x00,
UIDF1 = 0x40,
UIDF2 = 0x20,
UIDF3 = 0x60.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

Перед вызовом этой функции должна быть проведена аутентификация сектора 0.

4.8.12 CLSCRF_MifareStandard_EV1_SetLoadModulationType

LONG CLSCRF_MifareStandard_EV1_SetLoadModulationType(IN LPVOID pReader,
IN BYTE ucModType);

Команда измененяет нагрузку антенны карт.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucModType – тип нагрузки: 0x01 - Сильная (по умолчанию), 0x00 - нормальная.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

Перед вызовом этой функции должна быть проведена аутентификация сектора 0.

4.9 Функции обмена данными с картами Mifare Ultralight (C)

4.9.1 CLSCRF_MifareUltralight_Read

LONG CLSCRF_MifareUltralight_Read(IN LPVOID pReader,
IN BYTE bPage,
OUT LPBYTE pbRecvBuffer);

Читает 4 страницы (16 байтов), начиная с заданной страницы.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bPage – номер читаемой страницы;
pbRecvBuffer – массив (не менее 16 байтов) для прочитанной страницы.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.9.2 CLSCRF_MifareUltralight_Write

LONG CLSCRF_MifareUltralight_Write(IN LPVOID pReader,
IN BYTE bPage,
IN LPBYTE pbSendBuffer);

Записывает 4 байта в заданную страницу.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bPage – номер записываемой страницы;
pbSendBuffer – массив 4 байтов данных, записываемых в страницу.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.9.3 CLSCRF_MifareUltralightC_WriteKey

LONG CLSCRF_MifareUltralight_Write(IN LPVOID pReader,
IN DWORD dwKeyFlashAddress);

Записывает в карту ключ из flash считывателя.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwValueFlashAddress - адрес блока во flash-памяти считывателя, откуда следует взять ключ для записи.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.9.4 CLSCRF_MifareUltralightC_Authenticate

LONG CLSCRF_MifareUltralight_Authenticate(IN LPVOID pReader,
IN DWORD dwKeyFlashAddress);

Производит аутентификацию карты по указанному ключу.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwValueFlashAddress - адрес блока во flash-памяти считывателя, откуда следует взять ключ для аутентификации.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10 Функции обмена данными с метками стандарта ISO 15693

4.10.1 CLSCRF_ReadSingleBlock_15693

```
LONG CLSCRF_ReadSingleBlock_15693( IN    LPVOID pReader,
                                   IN    BYTE   bFlags,
                                   IN    BYTE   bReadSingleBlock,
                                   IN    LPBYTE pbUID,
                                   IN    BYTE   bBlockNumber,
                                   OUT   LPBYTE pbFlags,
                                   OUT   LPBYTE pbBlockSecurityStatus,
                                   OUT   LPBYTE pbData,
                                   OUT   LPBYTE pbErrorCode );
```

Читает 4 байта из заданного блока (см. ISO 15693-3 п.10.4.1).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bReadSingleBlock – код команды Read Single Block (всегда равен 0x20);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
bBlockNumber – номер читаемого блока (от 0);
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbBlockSecurityStatus – ссылка на переменную, в которую будет помещен статус защиты блока от записи;
pbData – ссылка на массив (не менее 4 байтов) для прочитанного блока;
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.2 CLSCRF_WriteSingleBlock_15693

```
LONG  CLSCRF_WriteSingleBlock_15693( IN    LPVOID pReader,
                                     IN    BYTE   bFlags,
                                     IN    BYTE   bWriteSingleBlock,
                                     IN    LPBYTE pbUID,
                                     IN    BYTE   bBlockNumber,
                                     IN    LPBYTE pbData,
                                     OUT   LPBYTE pbFlags,
                                     OUT   LPBYTE pbErrorCode );
```

Записывает 4 байта в заданный блок (см. ISO 15693-3 п.10.4.2).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bWriteSingleBlock – код команды Write Single Block (всегда равен 0x21);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки
bBlockNumber – номер записываемого блока (от 0);

pbData – ссылка на массив 4 байтов данных, записываемых в блок;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.3 CLSCRF_LockBlock_15693

```
LONG CLSCRF_LockBlock_15693( IN    LPVOID pReader,
                             IN    BYTE   bFlags,
                             IN    BYTE   bLockBlock,
                             IN    LPBYTE pbUID,
                             IN    BYTE   bBlockNumber,
                             OUT   LPBYTE pbFlags,
                             OUT   LPBYTE pbErrorCode );
```

Предохраняет заданный блок от перезаписи (см. ISO 15693-3 п.10.4.3).
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bLockBlock – код команды Lock Block (всегда равен 0x22);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
bBlockNumber – номер закрываемого блока (от 0);
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.4 CLSCRF_ReadMultipleBlocks_15693

```
LONG CLSCRF_ReadMultipleBlocks_15693( IN    LPVOID pReader,
                                      IN    BYTE   bFlags,
                                      IN    BYTE   bReadMultipleBlock,
                                      IN    LPBYTE pbUID,
                                      IN    BYTE   bFirstBlockNumber,
                                      IN          OUT       LPBYTE
                                      pbNumberOfBlocks,
                                      OUT   LPBYTE pbFlags,
                                      OUT         LPBYTE
                                      pbBlockSecurityStatus,
                                      OUT   LPBYTE pbData,
                                      OUT   LPBYTE pbErrorCode );
```

Читает несколько блоков (см. ISO 15693-3 п.10.4.4).
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bReadMultipleBlock – код команды Read Multiple Block (всегда равен 0x23);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
bFirstBlockNumber – номер первого читаемого блока (от 0);
pbNumberOfBlocks – ссылка на переменную, которая перед вызовом функции должна содержать количество (от 0) читаемых блоков, а на выходе будет содержать количество (тоже от 0) действительно прочитанных блоков;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbBlockSecurityStatus – ссылка на массив (до 256 байтов), в который будут помещены значения статуса защиты блоков от записи (только при наличии флага Option_flag);
pbData – ссылка на массив (до 8192 байтов) для прочитанных данных (по 4 байта на каждый блок);
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2);
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.5 CLSCRF_WriteAFI_15693

```
LONG CLSCRF_WriteAFI_15693( IN    LPVOID pReader,
                            IN    BYTE   bFlags,
                            IN    BYTE   bWriteAFI,
                            IN    LPBYTE pbUID,
                            IN    BYTE   bAFI,
                            OUT   LPBYTE pbFlags,
                            OUT   LPBYTE pbErrorCode );
```

Записывает 1 байт AFI (см. ISO 15693-3 п.10.4.8).
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bWriteAFI – код команды Write AFI (всегда равен 0x27);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
bAFI – записываемое значение AFI;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).
Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.6 CLSCRF_LockAFI_15693

```
LONG CLSCRF_LockAFI_15693( IN    LPVOID pReader,
                           IN    BYTE   bFlags,
                           IN    BYTE   bLockAFI,
                           IN    LPBYTE pbUID,
                           OUT   LPBYTE pbFlags,
                           OUT   LPBYTE pbErrorCode );
```

Предохраняет AFI от перезаписи (см. ISO 15693-3 п.10.4.9).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bLockAFI – код команды Lock AFI (всегда равен 0x28);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.7 CLSCRF_WriteDSFID_15693

```
LONG CLSCRF_WriteDSFID_15693( IN    LPVOID pReader,
                              IN    BYTE   bFlags,
                              IN    BYTE   bWriteDSFID,
                              IN    LPBYTE pbUID,
                              IN    BYTE   bDSFID,
                              OUT   LPBYTE pbFlags,
                              OUT   LPBYTE pbErrorCode );
```

Записывает 1 байт DSFID (см. ISO 15693-3 п.10.4.10).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bWriteDSFID – код команды Write DSFID (всегда равен 0x29);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
bDSFID – записываемое значение DSFID;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,

иначе – ошибка при выполнении.

4.10.8 CLSCRF_LockDSFID_15693

```
LONG CLSCRF_LockDSFID_15693( IN    LPVOID pReader,
                             IN    BYTE   bFlags,
                             IN    BYTE   bLockDSFID,
                             IN    LPBYTE pbUID,
                               OUT LPBYTE pbFlags,
                               OUT LPBYTE pbErrorCode );
```

Предохраняет DSFID от перезаписи (см. ISO 15693-3 п.10.4.11).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bLockDSFID – код команды Lock DSFID (всегда равен 0x2A);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.9 CLSCRF_GetSystemInfo_15693

```
LONG CLSCRF_GetSystemInfo_15693( IN    LPVOID pReader,
                                 IN    BYTE   bFlags,
                                 IN    BYTE   bGetSystemInfo,
                                 IN OUT LPBYTE pbUID,
                                 OUT LPBYTE pbFlags,
                                 OUT LPBYTE pbInfoFlags,
                                 OUT LPBYTE pbDSFID,
                                 OUT LPBYTE pbAFI,
                                 OUT LPWORD pbMemorySize,
                                 OUT LPBYTE pbICReference,
                                 OUT LPBYTE pbErrorCode );
```

Выдает значения системной информации (см. ISO 15693-3 п.10.4.12).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bGetSystemInfo – код команды Get System Information (всегда равен 0x2B);
pbUID – ссылка на массив (8 байтов), который перед вызовом функции может содержать уникальный идентификатор метки, а на выходе будет содержать уникальный идентификатор метки, прочитанный как часть системной информации;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);

pbInfoFlags – ссылка на переменную, в которую будут помещены флаги системной информации;
pbDSFID – ссылка на переменную, в которую будет помещено значение DSFID;
pbAFI – ссылка на переменную, в которую будет помещено значение AFI;
pbMemorySize – ссылка на переменную, в которую будет помещено значение Information on VICC memory size;
pbICReference – ссылка на переменную, в которую будет помещено значение Information on IC reference;
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.10 CLSCRF_GetMultipleBSS_15693

```
LONG CLSCRF_GetMultipleBSS_15693( IN    LPVOID pReader,
                                  IN    BYTE   bFlags,
                                  IN    BYTE   bGetMultipleBSS,
                                  IN    LPBYTE pbUID,
                                  IN    BYTE   bFirstBlockNumber,
                                  IN OUT LPBYTE pbNumberOfBlocks,
                                  OUT   LPBYTE pbFlags,
                                  OUT   LPBYTE pbBlockSecurityStatus,
                                  OUT   LPBYTE pbErrorCode );
```

Выдает значения статуса защиты блоков от записи (block security status) (см. ISO 15693-3 п.10.4.13).
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bGetMultipleBSS – код команды Get Multiple Block Security Status (всегда равен 0x2C);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
bFirstBlockNumber – номер первого читаемого блока (от 0);
pbNumberOfBlocks – ссылка на переменную, которая перед вызовом функции должна содержать количество (от 0) читаемых значений статуса защиты блоков от записи, а на выходе будет содержать количество (тоже от 0) действительно прочитанных значений;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbBlockSecurityStatus – ссылка на массив (до 256 байтов), в который будут помещены значения статуса защиты блоков от записи;
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.11 CLSCRF_SetEAS_15693

```
LONG CLSCRF_SetEAS_15693( IN    LPVOID pReader,
                          IN    BYTE   bFlags,
                          IN    BYTE   bSetEAS,
                          IN    BYTE   bICMfgCode,
                          IN    LPBYTE pbUID,
                          OUT   LPBYTE pbFlags,
                          OUT   LPBYTE pbErrorCode );
```

Устанавливает EAS в 1.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bSetEAS – код команды Set EAS (всегда равен 0xA2);
bICMfgCode – код производителя микросхемы метки;
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.12 CLSCRF_ResetEAS_15693

```
LONG CLSCRF_ResetEAS_15693( IN    LPVOID pReader,
                            IN    BYTE   bFlags,
                            IN    BYTE   bResetEAS,
                            IN    BYTE   bICMfgCode,
                            IN    LPBYTE pbUID,
                            OUT   LPBYTE pbFlags,
                            OUT   LPBYTE pbErrorCode );
```

Сбрасывает EAS в 0.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bResetEAS – код команды Reset EAS (всегда равен 0xA3);
bICMfgCode – код производителя микросхемы метки;
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет

помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.13 CLSCRF_LockEAS_15693

```
LONG CLSCRF_LockEAS_15693( IN    LPVOID pReader,
                           IN    BYTE   bFlags,
                           IN    BYTE   bLockEAS,
                           IN    BYTE   bICMfgCode,
                           IN    LPBYTE pbUID,
                           OUT   LPBYTE pbFlags,
                           OUT   LPBYTE pbErrorCode );
```

Предохраняет EAS от перезаписи.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bLockEAS – код команды Lock EAS (всегда равен 0xA4);
bICMfgCode – код производителя микросхемы метки;
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
pbFlags – ссылка на переменную, в которую будут помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.10.14 CLSCRF_EASAlarm_15693

```
LONG CLSCRF_EASAlarm_15693( IN    LPVOID pReader,
                            IN    BYTE   bFlags,
                            IN    BYTE   bEASAlarm,
                            IN    BYTE   bICMfgCode,
                            IN    LPBYTE pbUID,
                            OUT   LPBYTE pbFlags,
                            OUT   LPBYTE pbEASData,
                            OUT   LPBYTE pbErrorCode );
```

Читает EAS-последовательность, если бит EAS установлен в 1.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);
bEASAlarm – код команды EAS Alarm (всегда равен 0xA5);
pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;
bICMfgCode – код производителя микросхемы метки;
pbFlags – ссылка на переменную, в которую будут

помещены флаги ответа (см. ISO 15693-3 п.7.4.1);
pbEASData – ссылка на массив (32 байта) для прочитанной EAS-последовательности;
pbErrorCode – ссылка на переменную, в которую будет помещен код ошибки (см. ISO 15693-3 п.7.4.2).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11 Функции обмена данными с картами Mifare Plus

4.11.1 CLSCRF_MifarePlus_WritePersoExplicit

```
LONG CLSCRF_MifarePlus_WritePersoExplicit(  IN LPVOID pReader,
                                            IN BYTE ucValueType,
                                            IN BYTE ucSectorNumber,
                                            IN BYTE ucBlockNumber,
                                            IN LPBYTE pData );
```

Записывает данные персонализации в карту. Данные задаются явно.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucValueType – тип записываемого значения (см. таблицу);
ucSectorNumber – номер сектора (используется при записи блока данных или ключа, относящегося к определенному сектору);
ucBlockNumber – номер блока (используется при записи блока данных или ключа,
относящегося к определенному блоку в секторе);
pData – данные блока для записи;

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.2 CLSCRF_MifarePlus_CommitPerso

```
LONG CLSCRF_MifarePlus_CommitPerso(  IN LPVOID pReader );
```

Завершает персонализацию карты.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.3 CLSCRF_MifarePlus_Authenticate

LONG CLSCRF_MifarePlus_Authenticate(IN LPVOID pReader,
IN BYTE ucAuthType,
IN BYTE ucKeyType,
IN BYTE ucSectorNumber,
IN DWORD dwKeyFlashAddress,
IN BYTE ucLenCap,
IN LPBYTE pbPcdCap);

Выполняет первичную аутентификацию карты.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucAuthType – тип операции аутентификации (0x01 – первичная, 0x0F – последующая, 0x00 – сброс аутентификации);
ucKeyType – тип ключа для аутентификации (см. таблицу);
ucSectorNumber – номер сектора для аутентификации;
dwKeyFlashAddress – адрес ключа во flash-памяти считывателя для аутентификации;
ucLenCap – длина блока характеристик считывателя (0..6, установить в 0);
pbPcdCap – указатель на байтовый массив - блок характеристик считывателя (установить в NULL).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.4 CLSCRF_MifarePlus_MultiBlockRead

LONG CLSCRF_MifarePlus_MultiBlockRead(IN LPVOID pReader,
IN BYTE ucSectorNumber,
IN BYTE ucBlockNumber,
IN BYTE ucBlocksCount,
OUT LPBYTE pbData);

Для карты в режиме SL2, производит множественное чтение блоков.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucSectorNumber – порядковый номер сектора (от 0);
ucBlockNumber – порядковый номер блока (от 0);
ucBlocksCount – количество читаемых блоков;
pbData – указатель н байтовый массив, в который будут прочитаны данные.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.5 CLSCRF_MifarePlus_MultiBlockWrite

LONG CLSCRF_MifarePlus_MultiBlockWrite(IN LPVOID pReader,
IN BYTE ucSectorNumber,
IN BYTE ucBlockNumber,

IN BYTE ucBlocksCount,
IN LPBYTE pbData);

Для карты в режиме SL2, производит множественную запись блоков.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucSectorNumber – порядковый номер сектора (от 0);
ucBlockNumber – порядковый номер начального блока для записи (от 0);
ucBlocksCount – количество записываемых блоков;
pbData – указатель н байтовый массив с данными для записи.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.6 CLSCRF_MifarePlus_ReadData

LONG CLSCRF_MifarePlus_ReadData(IN LPVOID pReader,
IN BYTE ucEncryptionMode,
IN BYTE ucValueType,
IN BYTE ucSectorNumber,
IN BYTE ucBlockNumber,
IN BYTE ucBlocksCount,
OUT LPBYTE pbData);

Для карты в режиме SL3, производит чтение данных.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucEncryptionMode - режим защиты обмена данными (см. таблицы);
ucValueType - тип читаемого значения (см. таблицу);
ucSectorNumber - номер сектора, в котором нужно производить чтение;
ucBlockNumber - номер блока, с которого требуется начать считывание данных;
ucBlocksCount - количество блоков данных, которое нужно прочитать (1..3);
pbData – указатель н байтовый массив, в который будут прочитаны данные.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.7 CLSCRF_MifarePlus_WriteData

LONG CLSCRF_MifarePlus_WriteData(IN LPVOID pReader,
IN BYTE ucEncryptionMode,
IN BYTE ucValueType,
IN BYTE ucSectorNumber,
IN BYTE ucBlockNumber,
IN BYTE ucBlocksCount,
IN LPBYTE pbData);

Для карты в режиме SL3, производит запись данных.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucEncryptionMode - режим защиты обмена данными (см. таблицы);
ucValueType - тип записываемого значения (см. таблицу);
ucSectorNumber - номер сектора для записи;

ucBlockNumber - номер начального блока для записи;
ucBlocksCount - количество блоков данных, которое нужно записать (1..3);
pbData – указатель на байтовый массив с данными для записи.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.8 CLSCRF_MifarePlus_Increment

LONG CLSCRF_MifarePlus_Increment(IN LPVOID pReader,
IN BYTE ucEncryptionMode,
IN BYTE ucSourceSectorNumber,
IN BYTE ucSourceBlockNumber,
IN DWORD dwValue);

Для карты в режиме SL3, производит увеличение значения счетчика и последующую запись увеличенного значения в буфер переноса (Transfer Buffer).
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucEncryptionMode - режим защиты обмена данными (см. таблицы);
ucSourceSectorNumber - номер исходного сектора;
ucSourceBlockNumber - номер исходного блока;
dwValue - значение, на которое требуется прирастить блок значения.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.9 CLSCRF_MifarePlus_Decrement

LONG CLSCRF_MifarePlus_Decrement(IN LPVOID pReader,
IN BYTE ucEncryptionMode,
IN BYTE ucSourceSectorNumber,
IN BYTE ucSourceBlockNumber,
IN DWORD dwValue);

Для карты в режиме SL3, производит уменьшение значения счетчика и последующую запись уменьшенного значения в буфер переноса (Transfer Buffer).
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucEncryptionMode - режим защиты обмена данными (см. таблицы);
ucSourceSectorNumber - номер исходного сектора;
ucSourceBlockNumber - номер исходного блока;
dwValue - значение, которое требуется вычесть из блока значения.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.10 CLSCRF_MifarePlus_Transfer

LONG CLSCRF_MifarePlus_Transfer(IN LPVOID pReader,
IN BYTE ucEncryptionMode,
IN BYTE
ucDestinationSectorNumber,
IN BYTE
ucDestinationBlockNumber);

Для карты в режиме SL3, записывает данные буфера переноса (Transfer Buffer) в указанный блок.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucEncryptionMode - режим защиты обмена данными (см. таблицы);
ucDestinationSectorNumber - номер сектора для записи;
ucDestinationBlockNumber - номер блока для записи.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.11 CLSCRF_MifarePlus_IncrementTransfer

LONG CLSCRF_MifarePlus_IncrementTransfer(IN LPVOID pReader,
IN BYTE ucEncryptionMode,
IN BYTE ucSourceSectorNumber,
IN BYTE ucSourceBlockNumber,
IN BYTE
ucDestinationSectorNumber,
IN BYTE
ucDestinationBlockNumber,
IN DWORD dwValue);

Для карты в режиме SL3, производит увеличение значения счетчика, запись увеличенного значения в буфер переноса (Transfer Buffer) и затем в указанный блок.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucEncryptionMode - режим защиты обмена данными (см. таблицы);
ucSourceSectorNumber - номер исходного сектора;
ucSourceBlockNumber - номер исходного блока;
ucDestinationSectorNumber - номер сектора для записи;
ucDestinationBlockNumber - номер блока для записи;
dwValue - значение, на которое требуется прирастить блок значения.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.12 CLSCRF_MifarePlus_DecrementTransfer

LONG CLSCRF_MifarePlus_DecrementTransfer(IN LPVOID pReader,
IN BYTE ucEncryptionMode,
IN BYTE ucSourceSectorNumber,
IN BYTE ucSourceBlockNumber,
IN BYTE ucDestinationSectorNumber,
IN BYTE ucDestinationBlockNumber,
IN DWORD dwValue);

Для карты в режиме SL3, производит уменьшение значения счетчика, запись уменьшенного значения в буфер переноса (Transfer Buffer) и затем в указанный блок.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucEncryptionMode - режим защиты обмена данными (см. таблицы);
ucSourceSectorNumber - номер исходного сектора;
ucSourceBlockNumber - номер исходного блока;
ucDestinationSectorNumber - номер сектора для записи;
ucDestinationBlockNumber - номер блока для записи;
dwValue - значение, которое требуется вычесть из блока значения.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.13 CLSCRF_MifarePlus_Restore

LONG CLSCRF_MifarePlus_Restore(IN LPVOID pReader,
IN BYTE ucEncryptionMode,
IN BYTE ucSourceSectorNumber,
IN BYTE ucSourceBlockNumber);

Для карты в режиме SL3, производит запись значения указанного блока-значения в буфер переноса (Transfer Buffer).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucEncryptionMode - режим защиты обмена данными (см. таблицы);
ucSourceSectorNumber - номер исходного сектора;
ucSourceBlockNumber - номер исходного блока.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.14 CLSCRF_MifarePlus_VirtualCardSupport

LONG CLSCRF_MifarePlus_VirtualCardSupport(IN LPVOID pReader,
IN LPBYTE pIID);

Передает карте очередной идентификатор инфраструктуры.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pIID – указатель на идентификатор инфраструктуры (16 байт).
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.15 CLSCRF_MifarePlus_VirtualCardSupportLast

```
LONG CLSCRF_MifarePlus_VirtualCardSupportLast(
                                        IN LPVOID pReader,
                                        IN LPBYTE pIID,
                                        IN DWORD dwKencFlashAddress,
                                        IN DWORD dwKmacFlashAddress,
                                        IN BYTE ucLenCap,
                                        IN LPBYTE pbPcdCap,
                                        OUT PBYTE pucInfo,
                                        OUT PBYTE pPiccCap,
                                        OUT PBYTE pPaddedUID);
```

Передает карте последний идентификатор инфраструктуры, а также собственные характеристики и принимает характиристики карты.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pIID – указатель на идентификатор инфраструктуры (16 байт);
dwKencFlashAddress – адрес ключа VC Polling ENC Key во flash-памяти считывателя;
dwKmacFlashAddress – адрес ключа VC Polling MAC Key во flash-памяти считывателя;
ucLenCap – длина блока характеристик считывателя (0..3, установить в 0);
pbPcdCap – указатель на массив (3 байта) с блоком характеристик считывателя (пока отсутствует);
pucInfo – указатель на байт, в который будет записана информация о карте (0x83 – 4 байт UID, 0x03 – 7 байт UID);
pPiccCap – указатель на массив (2 байта), в который будут записаны прочитанные характеристики карты;
pPaddedUID – указатель на массив (13 байт), в который будет записан полученный идентификатор карты (4-байт или 7-байт, в зависимости от *pucInfo), дополненный до 13 байт данными паддировки.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.16 CLSCRF_MifarePlus_VirtualCardSelect

```
LONG CLSCRF_MifarePlus_VirtualCardSelect( IN LPVOID pReader,
                                        IN DWORD dwKselFlashAddress,
                                        IN PBYTE pPiccCap,
                                        IN PBYTE pPaddedUID );
```

Выбирает виртуальную карту.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

dwKselFlashAddress – адрес ключа Select VC Key во flash-памяти считывателя;

pPiccCap – указатель на массив (2 байта), из которого будут прочитаны характеристики карты;

pPaddedUID – указатель на массив (13 байт), из которого будет взят идентификатор карты длиной ucUidLen, дополненный до 13 байт данными паддировки.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.11.17 CLSCRF_MifarePlus_VirtualCardDeselect

```
LONG CLSCRF_MifarePlus_VirtualCardDeselect( IN LPVOID pReader );
```

Отменяет выбор виртуальной карты.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create).

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.11.18 CLSCRF_MifarePlus_ProximityCheck

```
LONG CLSCRF_MifarePlus_ProximityCheck(IN LPVOID pReader,
                                      IN DWORD dwKproxFlashAddress
);
```

Выполняет проверку релейной атаки.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

dwKproxFlashAddress – адрес ключа Proximity Check Key во flash-памяти считывателя.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.12 Функции обмена данными с картами Mifare DES Fire (EV1)

4.12.1 CLSCRF_MifareDesFire_Authenticate

```
LONG CLSCRF_MifareDesFire_Authenticate( IN LPVOID pReader,
                                        IN BYTE ucAuthType,
                                        IN PBYTE pKey,
                                        IN BYTE ucKeyNumber );
```

Производит аутентификацию (карты или приложения) указанного типа и с

указанным ключом.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucAuthType – тип аутентификации (0x00 – DES, 0x01 – 3DES, 0x02 – 3K3DES, 0x03 - AES);
pKey – указатель на массив байт, содержащий ключ (8 – 24 байта);
ucKeyNumber – номер ключа на карте, который будет использован при аутентификации.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении..

4.12.2 CLSCRF_MifareDesFire_SetTransferType

LONG CLSCRF_MifareDesFire_SetTransferType(IN LPVOID pReader,
IN BYTE ucTransferType);

Устанавливает формат обмена считыватель - карта.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucTransferType – тип передачи данных (0x00 – открытая, 0x01 – MAC, 0x03 - шифрованная).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.3 CLSCRF_MifareDesFire_ChangeKeySettings

LONG CLSCRF_MifareDesFire_ChangeKeySettings(IN LPVOID pReader,
IN CLSCRF_DESFIRE_MASTER_KEY_SETTINGS* pMasterKeySettings);

Меняет параметры мастер-ключа для текущего приложения.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pMasterKeySettings – указатель на структуру CLSCRF_DESFIRE_MASTER_KEY_SETTINGS с параметрами ключа.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.4 CLSCRF_MifareDesFire_GetKeySettings

LONG CLSCRF_MifareDesFire_GetKeySettings(IN LPVOID pReader,
OUT CLSCRF_DESFIRE_MASTER_KEY_SETTINGS_AND_LENGTH* pKeyData);

Считывает параметры мастер-ключа для текущего приложения.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pKeyData – указатель на структуру CLSCRF_DESFIRE_MASTER_KEY_SETTINGS_AND_LENGTH, в которую будут записаны считанные параметры мастер-ключа.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.5 CLSCRF_MifareDesFire_ChangeKey

LONG CLSCRF_MifareDesFire_ChangeKey(IN LPVOID pReader,
IN CLSCRF_DESFIRE_KEY_DATA*
pKeyData);

Меняет ключ в текущем приложении.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pKeyData – указатель на структуру CLSCRF_DESFIRE_KEY_DATA, в которой хранятся параметры нового ключа.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.6 CLSCRF_MifareDesFire_GetKeyVersion

LONG CLSCRF_MifareDesFire_GetKeyVersion(IN LPVOID pReader,
IN BYTE ucKeyNumber,
OUT PBYTE
pucKeyVersion);

Считывает версию ключа.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucKeyNumber – номер ключа карты;
pucKeyVersion – указатель на переменную, в которую будет записана версия ключа.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.7 CLSCRF_MifareDesFire_CreateApplication

LONG CLSCRF_MifareDesFire_CreateApplication(IN LPVOID pReader,
IN PBYTE pAID,
IN CLSCRF_DESFIRE_APPLICATION_MASTER_KEY_SETTINGS* pKeySett1,
IN CLSCRF_DESFIRE_NEW_APPLICATION_KEY_SETTINGS* pKeySett2,
IN PBYTE pIsoFileID,
IN PBYTE pIso7816DfName,
IN int iIso7816DfNameLength);

Создаёт приложение.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pAID – указатель на массив (3 байта), содержащий номер создаваемого приложения;
pKeySett1 – указатель на структуру CLSCRF_DESFIRE_APPLICATION_MASTER_KEY_SETTINGS, содержащую

параметры мастер-ключа приложения;
pKeySett2– указатель на структуру
CLSCRF_DESFIRE_NEW_APPLICATION_KEY_SETTINGS, содержащую параметры ключей приложения;
pIsoFileID – указатель на массив (2 байта), ISO-идентификатор файла (NULL, если не использовать);
pIso7816DfName – указатель на массив (размером **iIso7816DfNameLength**), содержащий ISO7816 Df-имя создаваемого приложения (NULL, если не использовать);
iIso7816DfNameLength – размер ISO7816 Df-имени создаваемого приложения в байтах.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.8 CLSCRF_MifareDesFire_DeleteApplication

```
LONG CLSCRF_MifareDesFire_DeleteApplication(    IN LPVOID pReader,
                                                IN PBYTE pAID );
```

Удаляет приложение.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pAID – указатель на массив (3 байта), содержащий номер удаляемого приложения.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.9 CLSCRF_MifareDesFire_GetApplicationIDs

```
LONG CLSCRF_MifareDesFire_GetApplicationIDs(  IN LPVOID pReader,
                                              OUT PBYTE pAIDs,
                                              OUT PBYTE
pucApplicationsCount);
```

Считывает идентификаторы приложений на карте.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pAIDs – указатель на массив, в который будут последовательно записаны 3-байтовые идентификаторы приложений;
pucApplicationsCount – указатель на переменную, в которую будет записано количество имеющихся на карте приложений.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.10 CLSCRF_MifareDesFire_GetDFNames

LONG CLSCRF_MifareDesFire_GetDFNames(IN LPVOID pReader,
OUT CLSCRF_DESFIRE_DFNAME* pDFNames,
OUT PBYTE pucDFNamesCount);

Считывает Df-имена приложений карты.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pDFNames – указатель на массив, в который будут последовательно записаны структры CLSCRF_DESFIRE_DFNAME, содержащие Df-имена приложений;
pucDFNamesCount – указатель на переменную, в которую будет записано количество имеющихся на карте приложений с Df-именами.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.11 CLSCRF_MifareDesFire_SelectApplication

LONG CLSCRF_MifareDesFire_SelectApplication(IN LPVOID pReader,
IN PBYTE pAID);

Переключает текущее приложение карты.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pAID – указатель на массив (3 байта), содержащий номер приложения.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.12 CLSCRF_MifareDesFire_FormatPICC

LONG CLSCRF_MifareDesFire_FormatPICC(IN LPVOID pReader);

Форматирование карты (требуется предварительная аутентификация мастер-ключом карты).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.13 CLSCRF_MifareDesFire_GetVersion

LONG CLSCRF_MifareDesFire_GetVersion(IN LPVOID pReader,
OUT CLSCRF_DESFIRE_HW_SW_INFO* pHWInfo,
OUT CLSCRF_DESFIRE_HW_SW_INFO* pSWInfo,
OUT CLSCRF_DESFIRE_MORE_INFO* pMoreInfo);

Считывает информацию о карте.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

pHWInfo – указатель на структуру CLSCRF_DESFIRE_HW_SW_INFO, в которую будет записана информация по Hardware карты;
pSWInfo – указатель на структуру CLSCRF_DESFIRE_HW_SW_INFO, в которую будет записана информация по Software карты;
pMoreInfo – указатель на структуру CLSCRF_DESFIRE_MORE_INFO, в которую будет записана дополнительная информация по версии карты.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.14 CLSCRF_MifareDesFire_FreeMemory

LONG CLSCRF_MifareDesFire_FreeMemory(IN LPVOID pReader,
OUT PDWORD pdwMemSize);

Освобождает память карты.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pdwMemSize – указатель на переменную, в которую будет записан размер памяти карты.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.15 CLSCRF_MifareDesFire_SetConfiguration

LONG CLSCRF_MifareDesFire_SetConfiguration(IN LPVOID pReader,
IN CLSCRF_DESFIRE_CONFIGURATION* pCfg);

Устанавливает конфигурацию карты.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pCfg – указатель на структуру CLSCRF_DESFIRE_CONFIGURATION, содержащую устанавливаемые параметры конфигурации.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.16 CLSCRF_MifareDesFire_GetCardUID

LONG CLSCRF_MifareDesFire_GetCardUID(IN LPVOID pReader,
OUT PBYTE pUID);

Считывает UID карты.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pUID – указатель на массив байт, к который будет записан UID карты.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.17 CLSCRF_MifareDesFire_GetFileIDs

LONG CLSCRF_MifareDesFire_GetFileIDs(IN LPVOID pReader,
OUT PBYTE pFileIDs,
OUT PBYTE pucFilesCount);

Считывает номера файлов в текущем приложении.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pFileIDs – указатель на массив байт, в котрый будут последовательно записаны номера файлов (1 байт на каждый) в текущем приложении;
pucFilesCount – указатель на переменную, в которую будет записано количество файлов в текущем приложении.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.18 CLSCRF_MifareDesFire_GetISOFileIDs

LONG CLSCRF_MifareDesFire_GetISOFileIDs(IN LPVOID pReader,
OUT PWORD pISOFileIDs,
OUT PBYTE pucFilesCount
);

Считывает ISO-номера файлов в текущем приложении.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pISOFileIDs – указатель на массив байт, в котрый будут последовательно записаны ISO-номера файлов (2 байта на каждый) в текущем приложении;
pucFilesCount – указатель на переменную, в которую будет записано количество файлов, имеющих ISO-идентификатор, в текущем приложении.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.19 CLSCRF_MifareDesFire_GetFileSettings

LONG CLSCRF_MifareDesFire_GetFileSettings(IN LPVOID pReader,
IN BYTE ucFileNumber,
OUT CLSCRF_DESFIRE_FILE_SETTINGS* pFileSettings);

Считывает паметры указанного файла.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
pFileSettings – указатель на структуру CLSCRF_DESFIRE_FILE_SETTINGS, в которую будут записаны параметры файла.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.20 CLSCRF_MifareDesFire_ChangeFileSettings

LONG CLSCRF_MifareDesFire_ChangeFileSettings(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN BYTE ucCommSettings,
IN CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS* pAccessRights);

Изменяет параметры указанного файла.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
ucCommSettings – настройки канала обмена (0x00 – открытый, 0x01 – MAC, 0x03 - шифрованный);
pAccessRights – указатель на структуру CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS, содержащую настройки доступа.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.21 CLSCRF_MifareDesFire_CreateStdDataFile

LONG CLSCRF_MifareDesFire_CreateStdDataFile(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN PBYTE pISOFileID,
IN BYTE ucCommSettings,
IN CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS* pAccessRights,
IN DWORD dwFileSize);

Создаёт стандартный файл данных.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
pISOFileID – указатель на массив, содержащий ISO-номер файла (2 байта, если NULL, то пропускается);
ucCommSettings – настройки канала обмена (0x00 – открытый, 0x01 – MAC, 0x03 - шифрованный);
pAccessRights – указатель на структуру CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS, содержащую настройки доступа;
dwFileSize – размер файла.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.22 CLSCRF_MifareDesFire_CreateBackupDataFile

LONG CLSCRF_MifareDesFire_CreateBackupDataFile(IN LPVOID pReader,
IN BYTE

ucFileNumber,
IN PBYTE pISOFileID,
IN BYTE ucCommSettings,
IN CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS*
pAccessRights,
IN DWORD dwFileSize);

Создаёт файл данных с резервным хранилищем.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
pISOFileID – указатель на массив, содержащий ISO-номер файла (2 байта, если NULL, то пропускается);
ucCommSettings – настройки канала обмена (0x00 – открытый, 0x01 – MAC, 0x03 - шифрованный);
pAccessRights – указатель на структуру CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS, содержащую настройки доступа;
dwFileSize – размер файла.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.23 CLSCRF_MifareDesFire_CreateValueFile

LONG CLSCRF_MifareDesFire_CreateValueFile(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN BYTE ucCommSettings,
IN CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS* pAccessRights,
IN DWORD dwLowerLimit,
IN DWORD dwUpperLimit,
IN DWORD dwValue,
IN CLSCRF_DESFIRE_LIMITED_CREDIT_ENABLED* pLimitedCreditEnabled);

Создаёт файл значения с резервным хранилищем.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
ucCommSettings – настройки канала обмена (0x00 – открытый, 0x01 – MAC, 0x03 - шифрованный);
pAccessRights – указатель на структуру CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS, содержащую настройки доступа;
dwLowerLimit – нижний предел значения;
dwUpperLimit – верхний предел значения;
dwValue – начальное значение;
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.24 CLSCRF_MifareDesFire_CreateLinearRecordFile

LONG CLSCRF_MifareDesFire_CreateLinearRecordFile(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN PBYTE pISOFileID,
IN BYTE ucCommSettings,
IN CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS* pAccessRights,
IN DWORD dwRecordSize,
IN DWORD dwMaxNumOfRecords);

Создаёт линейный файл записей с резервным хранилищем.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
pISOFileID – указатель на массив, содержащий ISO-номер файла (2 байта, если NULL, то пропускается);
ucCommSettings – настройки канала обмена (0x00 – открытый, 0x01 – MAC, 0x03 - шифрованный);
pAccessRights – указатель на структуру CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS, содержащую настройки доступа;
dwRecordSize – размер записи;
dwMaxNumOfRecords – максимальное количество записей.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.25 CLSCRF_MifareDesFire_CreateCyclicRecordFile

LONG CLSCRF_MifareDesFire_CreateCyclicRecordFile(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN PBYTE pISOFileID,
IN BYTE ucCommSettings,
IN CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS* pAccessRights,
IN DWORD dwRecordSize,
IN DWORD dwMaxNumOfRecords);

Создаёт циклический файл записей с резервным хранилищем.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
pISOFileID – указатель на массив, содержащий ISO-номер файла (2 байта, если NULL, то пропускается);
ucCommSettings – настройки канала обмена (0x00 – открытый, 0x01 – MAC, 0x03 - шифрованный);
pAccessRights – указатель на структуру CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS, содержащую настройки доступа;
dwRecordSize – размер записи;
dwMaxNumOfRecords – максимальное количество записей.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.26 CLSCRF_MifareDesFire_DeleteFile

```
LONG CLSCRF_MifareDesFire_DeleteFile(   IN LPVOID pReader,
                                        IN BYTE ucFileNumber );
```

Удаляет файл.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.27 CLSCRF_MifareDesFire_ReadData

```
LONG CLSCRF_MifareDesFire_ReadData(   IN LPVOID pReader,
                                      IN BYTE ucFileNumber,
                                      IN DWORD dwOffset,
                                      IN DWORD dwLength,
                                      OUT PBYTE pData,
                                      OUT PDWORD pdwDataLength);
```

Считывает данные из файла данных.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
dwOffset – адрес начального байта;
dwLength – количество считываемых байт;
pData – указатель на массив байт, в который будут считаны данные;
pdwDataLength – указатель на переменную, в которую будет записано количество считанных байт.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.28 CLSCRF_MifareDesFire_WriteData

LONG CLSCRF_MifareDesFire_WriteData(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN DWORD dwOffset,
IN DWORD dwLength,
IN PBYTE pData);

Записывает данные в файл данных.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
dwOffset – адрес начального байта;
dwLength – количество записываемых байт;
pData – указатель на массив байт, содержащий записываемые данные.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.29 CLSCRF_MifareDesFire_GetValue

LONG CLSCRF_MifareDesFire_GetValue(IN LPVOID pReader,
IN BYTE ucFileNumber,
OUT PDWORD pdwValue);

Считывает значение из файла-значения.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
pdwValue – указатель на переменную, в которую будет записано считанное значение.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.30 CLSCRF_MifareDesFire_Credit

LONG CLSCRF_MifareDesFire_Credit(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN DWORD dwValue);

Увеличивает значение в файле-значении на заданную величину.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
dwValue – значение, на которое нужно увеличить значение (простите за тафталогию).
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.31 CLSCRF_MifareDesFire_Debit

LONG CLSCRF_MifareDesFire_Debit(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN DWORD dwValue);

Уменьшает значение в файле-значении на заданную величину.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
dwValue – значение, на которое нужно уменьшить значение (простите за тафталогию).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.32 CLSCRF_MifareDesFire_LimitedCredit

LONG CLSCRF_MifareDesFire_LimitedCredit(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN DWORD dwValue);

Ограниченно увеличивает значение в файле-значении на заданную величину.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
dwValue – значение, на которое нужно увеличить значение (простите за тафталогию).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.33 CLSCRF_MifareDesFire_WriteRecord

LONG CLSCRF_MifareDesFire_WriteRecord(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN DWORD dwOffset,
IN DWORD dwLength,
IN PBYTE pData);

Записывает данные в файл записей.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
dwOffset – адрес начального байта;
dwLength – количество записываемых байт;
pData – указатель на массив байт, содержащий записываемые данные.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.34 CLSCRF_MifareDesFire_ReadRecords

LONG CLSCRF_MifareDesFire_ReadRecords(IN LPVOID pReader,
IN BYTE ucFileNumber,
IN DWORD dwRecordOffset,
IN DWORD dwRecordsCount,
IN DWORD dwRecordSize,
OUT PBYTE pData,
OUT PDWORD pdwDataLength);

Считывает данные из файла записей.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла;
dwRecordOffset – адрес начальной записи;
dwRecordsCount – количество считываемых записей;
dwRecordSize – размер считываемой записи;
pData – указатель на массив байт, в который будут считаны данные;
pdwDataLength – указатель на переменную, в которую будет записано количество считанных байт.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.35 CLSCRF_MifareDesFire_ClearRecordFile

LONG CLSCRF_MifareDesFire_ClearRecordFile(IN LPVOID pReader,
IN BYTE ucFileNumber);

Очищает файл записей.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucFileNumber – номер файла.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.36 CLSCRF_MifareDesFire_CommitTransaction

LONG CLSCRF_MifareDesFire_CommitTransaction(IN LPVOID pReader);

Копирует данные из резервного буфера файла в основной.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.37 CLSCRF_MifareDesFire_AbortTransaction

LONG CLSCRF_MifareDesFire_AbortTransaction(IN LPVOID pReader);
Копирует данные из основного буфера файла в резервный.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create).
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12.38 CLSCRF_DESFIRE_HW_SW_INFO

```
typedef struct _CLSCRF_DESFIRE_HW_SW_INFO
{
        BYTE ucVendorID;
        BYTE ucType;
        BYTE ucSubType;
        BYTE ucMajorVer;
        BYTE ucMinorVer;
        BYTE ucStorageSize;
        BYTE  ucProtocol;
} CLSCRF_DESFIRE_HW_SW_INFO;
```

- информация об аппаратном/программном обеспечении карты:
ucVendorID – идентификатор производителя;
ucType – тип карты;
ucSubType – подтип карты;
ucMajorVer – старшая версия;
ucMinorVer – младшая версия;
ucStorageSize – размер памяти;
ucProtocol – протокол.

4.12.39 CLSCRF_DESFIRE_MORE_INFO

```
typedef struct _CLSCRF_DESFIRE_MORE_INFO
{
        BYTE pUID[7];
        BYTE pBatchNumber[5];
        BYTE ucCwProd;
        BYTE ucYearProd;
} CLSCRF_DESFIRE_MORE_INFO;
```

- дополнительная информация о карте:
pUID – идентификатор карты;
pBatchNumber – номер партии (серии);
ucCwProd – календарная неделя производства;
ucYearProd – год производства.

4.12.40 CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS

```
typedef struct _CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS
{
        BYTE ChangeAccessRights :4;
        BYTE ReadAndWriteAccess :4;
        BYTE WriteAccess :4;
        BYTE ReadAccess :4;
} CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS;
```

- права доступа к файлу:

ChangeAccessRights – права на смену ключей;
ReadAndWriteAccess – права на чтение и запись;
WriteAccess – права на запись;
ReadAccess – права на чтение.
Значения: 0 – по мастер-ключу; 1-13 – по ключу 1-13, 14 – свободный доступ (по соответствующему ключу для операций смены ключа), 15 – полный запрет.

4.12.41 CLSCRF_DESFIRE_FILE_SETTINGS

```
typedef struct _CLSCRF_DESFIRE_FILE_SETTINGS
{
        BYTE ucFileType;
        BYTE ucCommSettings;
        CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS AccessRights;
        union
        {
                struct
                {
                        DWORD dwFileSize :24;
                } DataFile;
                struct
                {
                        DWORD dwLowerLimit;
                        DWORD dwUpperLimit;
                        DWORD dwLimitedCreditValue;
                        BYTE ucLimitedCreditEnabled;
                } ValueFile;
                union
                {
                        struct
                        {
                                DWORD dwRecordSize :24;
                        };
                        struct
                        {
                                BYTE Padding1[3];
```

```
                DWORD dwMaxNumberOfRecords :24;
            };
            struct
            {
                BYTE Padding2[6];
                DWORD dwCurrentNumberOfRecords :24;
            };
        } RecordFile;
    };
} CLSCRF_DESFIRE_FILE_SETTINGS;
```

- параметры файла:

DataFile.dwFileSize – размер файла данных;
ValueFile.dwLowerLimit – нижний предел значения файла записей;
ValueFile.dwUpperLimit – верхний предел значения файла записей;
ValueFile.dwLimitedCreditValue – максимальное значение для команды LimitedCredit;
RecordFile.dwRecordSize – размер записи в байтах;
RecordFile.dwMaxNumberOfRecords – максимальное количество записей;
RecordFile.dwCurrentNumberOfRecords – текущее количество записей.

4.12.42 CLSCRF_DESFIRE_CONFIGURATION

```
typedef struct _CLSCRF_DESFIRE_CONFIGURATION
{
    BYTE ucOption;
    union
    {
        union
        {
            BYTE ucConfigurationByte;
            struct
            {
                BYTE fFormatCardDisabled    :1;
                BYTE fRandomIDEnabled       :1;
                BYTE RFU                    :6;
            };
        };
        struct
        {
            BYTE Key[24];
            BYTE KeyVersion;
        };
        struct
        {
            BYTE ATS[32];
            BYTE ATSSize;
        };
```

```
        };
} CLSCRF_DESFIRE_CONFIGURATION;
```

- конфигурация карты:

ucOption – тип опции (0 – конфигурационный байт; 1 – ключ и его версия; 2 – ATS и его длина);
ucConfigurationByte – конфигурационный байт целиком;
fFormatCardDisabled – отключение возможности форматирования карты;
fRandomIDEnabled – включение случайного идентификатора карты;
Key – массив с ключом;
KeyVersion – версия ключа;
ATS – ATS карты;
ATSSize – размер ATS карты в байтах.

4.12.43 CLSCRF_DESFIRE_DFNAME

```
typedef union _CLSCRF_DESFIRE_DFNAME
{
        DWORD AID:24;
        struct
        {
                BYTE Padding[3];
                WORD FID;
                BYTE DFName[16];
                BYTE DFNameSize;
        };
} CLSCRF_DESFIRE_DFNAME;
```

- данные, получаемые вместе с DfName приложения:

AID – номер приложения;
FID – номер файла;
DFName – Df-имя приложения;
DFNameSize – размер Df-имени приложения.

4.12.44 CLSCRF_DESFIRE_LIMITED_CREDIT_ENABLED

```
typedef struct _CLSCRF_DESFIRE_LIMITED_CREDIT_ENABLED
{
        BYTE fEnableLimitedCredit :1;
        BYTE fEnableFreeGetValue :1;
        BYTE RFU :6;
} CLSCRF_DESFIRE_LIMITED_CREDIT_ENABLED;
```

- управление параметрами файла-значения:

fEnableLimitedCredit – разрешить команду LimitedCredit;
fEnableFreeGetValue – отклчить шифрацию данных в команде GetValue..

4.12.45 CLSCRF_DESFIRE_PICC_MASTER_KEY_SETTINGS

```
typedef struct _CLSCRF_DESFIRE_PICC_MASTER_KEY_SETTINGS
{
        BYTE fAllowChangingPiccMasterKey:1;
        BYTE  fFreeDirectoryListAccessWithoutPiccMasterKey:1;
        BYTE  fCreateAndDeleteWithoutPiccMasterKey:1;
        BYTE fConfigurationChangeable:1;
        BYTE RFU:4;
} CLSCRF_DESFIRE_PICC_MASTER_KEY_SETTINGS;
```

- параметры мастер-ключа карты:

fAllowChangingPiccMasterKey – разрешить смену мастер-ключа карты;
fFreeDirectoryListAccessWithoutPiccMasterKey – разрешить доступ к списку приложений и файлов без аутентификации мастер-ключом карты;
fCreateAndDeleteWithoutPiccMasterKey – разрешить создание и удаление приложений без аутентификации мастер-ключом карты;
fConfigurationChangeable – разрешить менять конфигурацию карты.

4.12.46 CLSCRF_DESFIRE_APPLICATION_MASTER_KEY_SETTINGS

```
typedef struct _CLSCRF_DESFIRE_APPLICATION_MASTER_KEY_SETTINGS
{
        BYTE fAllowChangingApplicationMasterKey:1;
        BYTE  fFreeDirectoryListAccessWithoutApplicationMasterKey:1;
        BYTE fCreateAndDeleteWithoutApplicationMasterKey:1;
        BYTE fConfigurationChangeable:1;
        BYTE ChangeKeyAccessRights:4;
} CLSCRF_DESFIRE_APPLICATION_MASTER_KEY_SETTINGS;
```

- параметры мастер-ключа приложения:

fAllowChangingApplicationMasterKey – разрешить смену мастер-ключа приложения;
fFreeDirectoryListAccessWithoutApplicationMasterKey – разрешить доступ к списку приложений и файлов без аутентификации мастер-ключом приложения;
fCreateAndDeleteWithoutPiccMasterKey – разрешить создание и удаление приложений без аутентификации мастер-ключом приложения;
fConfigurationChangeable – разрешить менять конфигурацию приложения;
ChangeKeyAccessRights – права доступа для смены ключа (0 – по мастер-ключу; 1-13 – по ключу 1-13, 14 – по соответствующему ключу, 15 – полный запрет.).

4.12.47 CLSCRF_DESFIRE_NEW_APPLICATION_KEY_SETTINGS

```
typedef struct _CLSCRF_DESFIRE_NEW_APPLICATION_KEY_SETTINGS
{
        BYTE MaxNumberOfKeys :4;
        BYTE RFU :1;
        BYTE Supported2ByteFileIDs :1;
        BYTE CryptoMethod :2;
} CLSCRF_DESFIRE_NEW_APPLICATION_KEY_SETTINGS;
```

- параметры ключей создаваемого приложения:
MaxNumberOfKeys – максимальное количество ключей;
Supported2ByteFileIDs – разрешить поддержку 2-байтовых ISO-идентификаторов файлов;
CryptoMethod – режим шифрации (0 – DES или 3DES, 1 – 3K3DES, 2 - AES).

4.12.48 CLSCRF_DESFIRE_MASTER_KEY_SETTINGS

```
typedef union _CLSCRF_DESFIRE_MASTER_KEY_SETTINGS
{
        CLSCRF_DESFIRE_PICC_MASTER_KEY_SETTINGS
PiccMasterKeySettings;
        CLSCRF_DESFIRE_APPLICATION_MASTER_KEY_SETTINGS
ApplicationMasterKeySettings;
} CLSCRF_DESFIRE_MASTER_KEY_SETTINGS;
```

- параметры мастер-ключей:
PiccMasterKeySettings – для карты;
ApplicationMasterKeySettings – для приложения.

4.12.49 CLSCRF_DESFIRE_KEY_DATA

```
typedef struct _CLSCRF_DESFIRE_KEY_DATA
{
        union
        {
                struct
                {
                        BYTE SetToZero :6;
                        BYTE KeyType :2;
                } PiccMasterKey;
                BYTE ApplicationKeyNumber;
        } KeyNumber;
        BYTE Key[24];
        BYTE ucKeySize;
```

```
        BYTE ucAESKeyVersion;
        bool fIsAESKey;
} CLSCRF_DESFIRE_KEY_DATA;
```

- параметры ключа:

KeyNumber.PiccMasterKey.KeyType – тип ключа (0 – DES или 3DES, 1 – 3K3DES, 2 - AES);
KeyNumber.ApplicationKeyNumber – количество ключей в приложении;
Key – ключ;
ucKeySize – размер ключа;
ucAESKeyVersion – версия ключа AES;
fIsAESKey – true, если ключ AES.

4.12.50 CLSCRF_DESFIRE_MASTER_KEY_SETTINGS_AND_LENGTH

```
typedef struct _CLSCRF_DESFIRE_MASTER_KEY_SETTINGS_AND_LENGTH
{
        CLSCRF_DESFIRE_MASTER_KEY_SETTINGS MasterKeySettings;
        union
        {
                struct
                {
                        BYTE MaxPiccKeysNumber :6;
                        BYTE KeyType :2;
                } PiccMasterKey;
                BYTE MaxApplicationKeysNumber;
        } MaxNumberOfKeys;
} CLSCRF_DESFIRE_MASTER_KEY_SETTINGS_AND_LENGTH;
```

- параметры и длина мастер-ключа:

MasterKeySettings – установки мастер-ключа;
MaxNumberOfKeys.MaxApplicationKeysNumber – максимальное количество ключей в приложении;
MaxNumberOfKeys.PiccMasterKey – мастер-ключ карты;
MaxNumberOfKeys.PiccMasterKey.MaxPiccKeysNumber – максимальное количество ключей в карте;
MaxNumberOfKeys.PiccMasterKey.KeyType – тип ключа (0 – DES или 3DES, 1 – 3K3DES, 2 - AES).

4.13 Функции работы со считывателем NFC663

В данной группе объединены функции работы со считывателем, работаюим на микросхеме CLRC663.

4.13.1 CLSCRF_NFC663_ActivateCard

```
LONG CLSCRF_NFC663_ActivateCard( IN LPVOID pReader,
                                 IN LPBYTE Nfcid3i,
                                 IN BYTE Did,
                                 IN BYTE NadEnable,
                                 IN BYTE Nad,
                                 IN BYTE Dsi,
                                 IN BYTE Dri,
                                 IN BYTE Fsl,
                                 IN BYTE GiLength,
                                 IN LPBYTE Gi,
                                 OUT LPBYTE pbATR,
                                 IN OUT LPDWORD pdwATRLength );
```

Выполняет команды ISO18092 ATR и PSL.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
Nfcid3i – массив из 10 байт: при начальной скорости в 106kbps - NFCID3, генерируется случайным образом; при скоростях 212/424kbps - байты 0-7 соответствуют NFCID2, а байты 8-9 должны быть установлены в 0.
Did – идентификатор устройства, "0" - не использовать, либо 1-14;
NadEnable – включение использования адреса шины, для включения установить НЕ в "0";
Nad – адрес узла: игнорируется, если bNadEnabled = 0;
Dsi – индекс делителя отправки (от цели к инициатору) 0-2 (PHPAL_I18092MPI_DATARATE_106, PHPAL_I18092MPI_DATARATE_212, PHPAL_I18092MPI_DATARATE_424);
Dri – индекс делителя приема (от инициатора к цели) 0-2 (PHPAL_I18092MPI_DATARATE_106, PHPAL_I18092MPI_DATARATE_212, PHPAL_I18092MPI_DATARATE_424);
Fsl – байт длины кадра 0-3 (PHPAL_I18092MPI_FRAMESIZE_64, PHPAL_I18092MPI_FRAMESIZE_128, PHPAL_I18092MPI_FRAMESIZE_192, PHPAL_I18092MPI_FRAMESIZE_254);
GiLength – количество байт общей информации;
Gi – опционально, байты общей информации;
pbATR – буфер, куда будут записаны байты ATR (ответа с атрибутаи), должен быть не меньше 64 байт;
pdwATRLength – длина считанных атрибутов в байтах;

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.2 CLSCRF_NFC663_Deselect

```
LONG CLSCRF_NFC663_Deselect(  IN LPVOID pReader,
                              IN BYTE DeselectCommand );
```

Отмена выбора цели ISO18092 путем отправки запросов DSL либо RLS.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
DeselectCommand – запрос на отправку, PHPAL_I18092MPI_DESELECT_DSL, либо PHPAL_I18092MPI_DESELECT_RLS
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.3 CLSCRF_NFC663_Exchange

```
LONG CLSCRF_NFC663_Exchange(    IN LPVOID pReader,
                                IN DWORD dwOption,
                                IN LPCBYTE pbSendBuffer,
                                IN DWORD dwSendLength,
                                OUT LPBYTE pbRecvBuffer,
                                IN OUT LPDWORD pdwRecvLength );
```

Выполняет обмен данными ISO18092 с целью.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwOption – параметр опций:
одно из PH_EXCHANGE_DEFAULT, PH_EXCHANGE_TXCHAINING, PH_EXCHANGE_RXCHAINING, PH_EXCHANGE_RXCHAINING_BUFSIZE,
сложенное с любой комбинацией из PH_EXCHANGE_TX_CRC, PH_EXCHANGE_RX_CRC, PH_EXCHANGE_PARITY,
сложенное с любой комбинацией из PH_EXCHANGE_LEAVE_BUFFER_BIT, PH_EXCHANGE_BUFFERED_BIT;
pbSendBuffer – буфер с данными для передачи;
dwSendLength – количество байт для передачи;
pbRecvBuffer – буфер для размещения принятых байт;
pdwRecvLength – количество принятых байт;
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.4 CLSCRF_NFC663_ResetProtocol

```
LONG CLSCRF_NFC663_ResetProtocol( IN LPVOID pReader );
```

Сбрасывает параметры протокола ISO18092.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.5 CLSCRF_NFC663_AttributeRequest

```
LONG CLSCRF_NFC663_AttributeRequest(    IN LPVOID pReader,
                                        IN LPBYTE Nfcid3i,
                                        IN BYTE Did,
```

IN BYTE NadEnable,
IN BYTE Nad,
IN BYTE Fsl,
IN BYTE GiLength,
IN LPBYTE Gi,
OUT LPBYTE pbATR,
IN OUT LPDWORD
pdwATRLength);

Выполняет команду ISO18092 "Attribute Request".
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
Nfcid3i – массив из 10 байт: при начальной скорости в 106kbps - NFCID3, генерируется случайным образом; при скоростях 212/424kbps - байты 0-7 соответствуют NFCID2, а байты 8-9 должны быть установлены в 0.
Did – идентификатор устройства, "0" - не использовать, либо 1-14;
NadEnable – включение использования адреса шины, для включения установить НЕ в "0";
Nad – адрес узла: игнорируется, если bNadEnabled = 0;
Fsl – байт длины кадра 0-3 (PHPAL_I18092MPI_FRAMESIZE_64, PHPAL_I18092MPI_FRAMESIZE_128, PHPAL_I18092MPI_FRAMESIZE_192, PHPAL_I18092MPI_FRAMESIZE_254);
GiLength – количество байт общей информации;
Gi – опционально, байты общей информации;
pbATR – буфер, куда будудт записаны байты ATR (ответа с атрибутаи), должен быть не меньше 64 байт;
pdwATRLength – длина считанных атрибутов в байтах;

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.6 CLSCRF_NFC663_ParameterSelect

LONG CLSCRF_NFC663_ParameterSelect(IN LPVOID pReader,
IN BYTE Dsi,
IN BYTE Dri,
IN BYTE Fsl);

Выполняет команду ISO18092 "Parameter Select".
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
Dsi – индекс делителя отправки (от цели к инициатору) 0-2 (PHPAL_I18092MPI_DATARATE_106, PHPAL_I18092MPI_DATARATE_212, PHPAL_I18092MPI_DATARATE_424);
Dri – индекс делителя приема (от инициатора к цели) 0-2 (PHPAL_I18092MPI_DATARATE_106, PHPAL_I18092MPI_DATARATE_212, PHPAL_I18092MPI_DATARATE_424);
Fsl – байт длины кадра 0-3 (PHPAL_I18092MPI_FRAMESIZE_64, PHPAL_I18092MPI_FRAMESIZE_128, PHPAL_I18092MPI_FRAMESIZE_192, PHPAL_I18092MPI_FRAMESIZE_254);

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.7 CLSCRF_NFC663_PresenceCheck

LONG CLSCRF_NFC663_PresenceCheck(IN LPVOID pReader);
Выполняет проверку присутствия для текущей цели.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.8 CLSCRF_NFC663_SetConfig

LONG CLSCRF_NFC663_SetConfig(IN LPVOID pReader,
IN DWORD dwParameterNumber,
IN DWORD pdwParameterValue);
Выполняет установку конфигурационного параметра.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwParameterNumber – идентификатор параметра - одно из:
PHPAL_I18092MPI_CONFIG_PACKETNO, PHPAL_I18092MPI_CONFIG_DID,
PHPAL_I18092MPI_CONFIG_NAD, PHPAL_I18092MPI_CONFIG_WT,
PHPAL_I18092MPI_CONFIG_FSL,
PHPAL_I18092MPI_CONFIG_MAXRETRYCOUNT;
pdwParameterValue – значение параметра;
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.9 CLSCRF_NFC663_GetConfig

LONG CLSCRF_NFC663_GetConfig(IN LPVOID pReader,
IN DWORD dwParameterNumber,
OUT LPDWORD pdwParameterValue);
Выполняет чтение конфигурационного параметра.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwParameterNumber – идентификатор параметра - одно из:
PHPAL_I18092MPI_CONFIG_PACKETNO, PHPAL_I18092MPI_CONFIG_DID,
PHPAL_I18092MPI_CONFIG_NAD, PHPAL_I18092MPI_CONFIG_WT,
PHPAL_I18092MPI_CONFIG_FSL,
PHPAL_I18092MPI_CONFIG_MAXRETRYCOUNT;
pdwParameterValue – указатель на значение параметра;
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.10 CLSCRF_NFC663_GetSerialNo

LONG CLSCRF_NFC663_GetSerialNo(IN LPVOID pReader,
OUT LPBYTE NFCID3);

Выполняет чтение конфигурационного параметра.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
NFCID3 – серийный номер NFCID3 - 10 байт;
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.11 CLSCRF_NFC663_E2_Read

LONG CLSCRF_NFC663_E2_Read(IN LPVOID pReader,
IN DWORD dwAddr,
IN BYTE ucLength,
OUT LPBYTE pData);

Выполняет чтение заданного количества байт из указанного адреса EEPROM.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwAddr – адрес байта 192(0x00C0)..6143(0x17FF);
ucLength – количество вычитываемых байт данных 1..127;
pData – массив, куда будут скопированы прочитанные байты данных;
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.12 CLSCRF_NFC663_E2_Write

LONG CLSCRF_NFC663_E2_Write(IN LPVOID pReader,
IN DWORD dwAddr,
IN BYTE ucData);

Выполняет запись одного байта данных в указанный адрес EEPROM.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwAddr – адрес байта 192(0x00C0)..6143(0x17FF);
ucData – записываемый байт данных;
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.13 CLSCRF_NFC663_E2_WritePage

LONG CLSCRF_NFC663_E2_WritePage(IN LPVOID pReader,
IN DWORD dwAddr,
IN BYTE ucLength,
IN LPBYTE pData);

Выполняет запись нескольких байт данных в указанную страницу EEPROM.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
dwAddr – адрес страницы 3..95(0x5F);
ucLength – количество записываемых байт данных 1..64;
pData – указатель на массив байт данных;
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.14 PHPAL_I18092MPI_DATARATE_106

```
#define PHPAL_I18092MPI_DATARATE_106  0x00U
```

Значение DRI/DSI для 106 kBit/s.

4.13.15 PHPAL_I18092MPI_DATARATE_212

```
#define PHPAL_I18092MPI_DATARATE_212  0x01U
```

Значение DRI/DSI для 212 kBit/s.

4.13.16 PHPAL_I18092MPI_DATARATE_424

```
#define PHPAL_I18092MPI_DATARATE_424  0x02U
```

Значение DRI/DSI для 424 kBit/s.

4.13.17 PHPAL_I18092MPI_FRAMESIZE_64

```
#define PHPAL_I18092MPI_FRAMESIZE_64  0x00U
```

Значение FSL для макс. размера кадра в 64 байта.

4.13.18 PHPAL_I18092MPI_FRAMESIZE_128

```
#define PHPAL_I18092MPI_FRAMESIZE_128  0x01U
```

Значение FSL для макс. размера кадра в 128 байт.

4.13.19 PHPAL_I18092MPI_FRAMESIZE_192

```
#define PHPAL_I18092MPI_FRAMESIZE_192  0x02U
```

Значение FSL для макс. размера кадра в 192 байта.

4.13.20 PHPAL_I18092MPI_FRAMESIZE_254

#define PHPAL_I18092MPI_FRAMESIZE_254 0x03U

Значение FSL для макс. размера кадра в 254 байта.

4.13.21 PHPAL_I18092MPI_DESELECT_DSL

#define PHPAL_I18092MPI_DESELECT_DSL 0x08U

DSL отправляется для отмена выбора цели.

4.13.22 PHPAL_I18092MPI_DESELECT_RLS

#define PHPAL_I18092MPI_DESELECT_RLS 0x0AU

RLS отправляется для отмены выбора цели.

4.13.23 PH_EXCHANGE_DEFAULT

#define PH_EXCHANGE_DEFAULT 0x0000U

Режим обмена по умолчанию.
Для выполнения буферизации комбинируйте с PH_EXCHANGE_BUFFERED_BIT и PH_EXCHANGE_LEAVE_BUFFER_BIT
Специфично для ISO14443-4:
Выполняет цепочную передачу Tx/Rx с картой.
Возвращает PH_ERR_SUCCESS_CHAINING, если RxBuffer полон и не отправляет подтверждение (ACK) для последнего принятого блока.

4.13.24 PH_EXCHANGE_TXCHAINING

#define PH_EXCHANGE_TXCHAINING 0x0001U

Специфично для ISO14443-4:
Передает данные в карту цепочной передачей.
Кобинируется с PH_EXCHANGE_BUFFERED_BIT и PH_EXCHANGE_LEAVE_BUFFER_BIT для выполнения буферизации.
Не принимает никаких данных.

4.13.25 PH_EXCHANGE_RXCHAINING

#define PH_EXCHANGE_RXCHAINING 0x0002U

Специфично для ISO14443-4:
Начинает передачу с блоком R(ACK) и выполняет цепочную передачу Rx с картой.
Возвращает PH_ERR_SUCCESS_CHAINING, если RxBuffer полон и не подтверждает (ACK) последний принятый блок.

4.13.26 PH_EXCHANGE_RXCHAINING_BUFSIZE

```
#define PH_EXCHANGE_RXCHAINING_BUFSIZE 0x0003U
```

Специфично для ISO14443-4:
Начинает передачу с блоком R(ACK) и выполняет цепочную передачу Rx с картой.
Завершает цепочную передачу Rx с картой, если RxBuffer полон.

4.13.27 PH_EXCHANGE_TX_CRC

```
#define PH_EXCHANGE_TX_CRC          0x0010U
```

Добавлять TX CRC.

4.13.28 PH_EXCHANGE_RX_CRC

```
#define PH_EXCHANGE_RX_CRC          0x0020U
```

Добавлять RX CRC.

4.13.29 PH_EXCHANGE_PARITY

```
#define PH_EXCHANGE_PARITY          0x0040U
```

Добавлять признак четности.

4.13.30 PH_EXCHANGE_LEAVE_BUFFER_BIT

```
#define PH_EXCHANGE_LEAVE_BUFFER_BIT  0x4000U
```

Не производит очистки внутреннего буфера перед выполнением операции.
Если этот бит установлен и данные переданы, то содержимое внутреннего буфера отправляются в первую очередь.

4.13.31 PH_EXCHANGE_BUFFERED_BIT

```
#define PH_EXCHANGE_BUFFERED_BIT      0x8000U
```

Буферизует Tx-Data во внутренний буфер вместо его передачи.

4.13.32 PHPAL_I18092MPI_CONFIG_PACKETNO

#define PHPAL_I18092MPI_CONFIG_PACKETNO 0x0000U

Задать / получить номер пакета.

4.13.33 PHPAL_I18092MPI_CONFIG_DID

#define PHPAL_I18092MPI_CONFIG_DID 0x0001U

Задать / получить идентификатор устройства.

4.13.34 PHPAL_I18092MPI_CONFIG_NAD

#define PHPAL_I18092MPI_CONFIG_NAD 0x0002U

Задать / получить адрес узла.

4.13.35 PHPAL_I18092MPI_CONFIG_WT

#define PHPAL_I18092MPI_CONFIG_WT 0x0003U

Задать / получить время ожидания.

4.13.36 PHPAL_I18092MPI_CONFIG_FSL

#define PHPAL_I18092MPI_CONFIG_FSL 0x0004U

Задать / получить длину кадра.

4.13.37 PHPAL_I18092MPI_CONFIG_MAXRETRYCOUNT

#define PHPAL_I18092MPI_CONFIG_MAXRETRYCOUNT 0x0005U

Задать / получить максимальное количество повторных попыток.

4.14 Функции демонстрационные для работы со стандартом NFC

В данной группе собраны демонстрационные функции, предназначенные для поверхностного тестирования поддержки считывателем технологии NFC (Near Field Communication). Использование функций вы можете посмотреть на примерах.

4.14.1 Функции работы с NDEF сообщениями

В данной группе собраны функции NFC, формирующие и обрабатывающие сообщения в формате NDEF.

4.14.1.1 CLSCRF_NFC_NDEF_Message_Create

LONG CLSCRF_NFC_NDEF_Message_Create(
IN LPBYTE lpMsgDataBytes,
IN DWORD dwMsgDataLength,
OUT LPHANDLE
phNDEFMessage);
Создаёт объект сообщения в формате NDEF (NFC Data Exchange Format).
lpMsgDataBytes – указатель на инициализационный массив байт, на основе которого будет создано сообщение в формате NDEF. Если NULL (0x00000000), то создаётся пустое сообщение;
dwMsgDataLength – количество байт в инициализационном массиве. Если 0x00000000, то создаётся пустое сообщение;
phNDEFMessage – указатель на переменную типа DWORD, куда будет помещён созданный указатель на объект сообщения в формате NDEF (при успешной отработке функции).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.1.2 CLSCRF_NFC_NDEF_Message_GetRecordsCount

LONG CLSCRF_NFC_NDEF_Message_GetRecordsCount(IN HANDLE hNDEFMessage,
OUT LPDWORD
pdwRecordsCount);
Возвращает количество записей в сообщении в формате NDEF.
hNDEFMessage – указатель на объект сообщения в формате NDEF;
pdwRecordsCount – указатель на переменную типа DWORD, куда будет записано количество записей, входящих в сообщение NDEF.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.1.3 CLSCRF_NFC_NDEF_Message_GetDataSize

LONG CLSCRF_NFC_NDEF_Message_GetDataSize(IN HANDLE hNDEFMessage,
OUT LPDWORD
pdwDataSize);
Возвращает размер сообщения формата NDEF в байтах.
hNDEFMessage – указатель на объект сообщения в формате NDEF;
pdwDataSize – указатель на переменную типа DWORD, куда будет записан

размер сообщения NDEF в байтах.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.1.4 CLSCRF_NFC_NDEF_Message_GetData

LONG CLSCRF_NFC_NDEF_Message_GetData(IN HANDLE hNDEFMessage,
OUT LPBYTE lpData);

Возвращает массив байт сообщения в формате NDEF.

hNDEFMessage – указатель на объект сообщения в формате NDEF;

lpData – указатель на массив байт, в который будут записаны все данные сообщения NDEF.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.1.5 CLSCRF_NFC_NDEF_Message_AddRecord

LONG CLSCRF_NFC_NDEF_Message_AddRecord(IN HANDLE hNDEFMessage,
IN HANDLE hNDEFRecord);

Добавляет запись в сообщение в формате NDEF.

hNDEFMessage – указатель на объект сообщения в формате NDEF;

hNDEFRecord – указатель на объект записи сообщения в формате NDEF, которую следует добавить к указанному сообщению.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.1.6 CLSCRF_NFC_NDEF_Message_AddRecordEmpty

LONG CLSCRF_NFC_NDEF_Message_AddRecordEmpty(IN HANDLE hNDEFMessage);

Добавляет пустую запись в сообщение в формате NDEF.

hNDEFMessage – указатель на объект сообщения в формате NDEF.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.1.7 CLSCRF_NFC_NDEF_Message_AddRecordText

LONG CLSCRF_NFC_NDEF_Message_AddRecordText(IN HANDLE hNDEFMessage,
IN LPWSTR lpwstrText,
IN LPWSTR lpwstrLanguage);

Добавляет текстовую запись в сообщение в формате NDEF.
hNDEFMessage – указатель на объект сообщения в формате NDEF;
lpwstrText – указатель на текстовую строку в формате Unicode, из которой следует сформировать запись типа текст и добавить её к указанному сообщению NDEF;
lpwstrLanguage – указатель на строку в формате Unicode, содержащую кодировку языка, на котором написана добавляемая к сообщению NDEF в качестве записи строка.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.1.8 CLSCRF_NFC_NDEF_Message_AddRecordUri

LONG CLSCRF_NFC_NDEF_Message_AddRecordUri(IN HANDLE hNDEFMessage,
IN LPWSTR lpwstrUri);

Добавляет запись типа URI (Universal Resource Identifier) в сообщение в формате NDEF.
hNDEFMessage – указатель на объект сообщения в формате NDEF;
lpwstrUri – указатель на строку в формате Unicode, содержащую URI, из которой следует сформировать запись типа URI и добавить её к указанному сообщению NDEF.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.1.9 CLSCRF_NFC_NDEF_Message_AddRecordMimeMedia

LONG CLSCRF_NFC_NDEF_Message_AddRecordMimeMedia(IN HANDLE hNDEFMessage,
IN LPWSTR lpwstrMimeType,
IN LPBYTE lpPayload,
IN DWORD dwPayloadLength);

Добавляет запись типа MIME в сообщение в формате NDEF.
hNDEFMessage – указатель на объект сообщения в формате NDEF;
lpwstrMimeType – указатель на строку в формате Unicode, содержащую тип MIME, определяющий подтип добавляемой записи типа MIME;
lpPayload – указатель на массив байт, из которого следует сформировать запись типа MIME и добавить её к указанному сообщению NDEF.
dwPayloadLength – количество байт в массиве данных lpPayload.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.1.10 CLSCRF_NFC_NDEF_Message_AddRecordMimeMediaText

LONG CLSCRF_NFC_NDEF_Message_AddRecordMimeMediaText(IN HANDLE hNDEFMessage,
IN LPWSTR lpwstrMimeType,
IN LPWSTR lpwstrText);

Добавляет запись типа MIME в сообщение в формате NDEF.

hNDEFMessage – указатель на объект сообщения в формате NDEF;

lpwstrMimeType – указатель на строку в формате Unicode, содержащую тип MIME, определяющий подтип добавляемой записи типа MIME;

lpwstrText – указатель на строку в формате Unicode, из которой следует сформировать запись типа MIME и добавить её к указанному сообщению NDEF.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.1.11 CLSCRF_NFC_NDEF_Message_GetRecord

LONG CLSCRF_NFC_NDEF_Message_GetRecord(IN HANDLE hNDEFMessage,
IN DWORD dwRecordIndex,
OUT LPHANDLE phNDEFRecord);

Возвращает определённую порядковым номером запись в сообщении в формате NDEF.

hNDEFMessage – указатель на объект сообщения в формате NDEF;

dwDataSize – количество байт из массива данных, которые нужно записать в метку.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.1.12 CLSCRF_NFC_NDEF_Message_Destroy

LONG CLSCRF_NFC_NDEF_Message_Destroy(IN HANDLE hNDEFMessage);

Удаляет объект сообщения в формате NDEF (NFC Data Exchange Format).

hNDEFMessage – указатель на объект сообщения в формате NDEF.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.2 Функции работы с NDEF записями

В данной группе собраны функции NFC, формирующие и обрабатывающие записи сообщений в формате NDEF (NFC Data Exchange Format).

4.14.2.1 CLSCRF_NFC_NDEF_Record_Create

LONG CLSCRF_NFC_NDEF_Record_Create(OUT LPHANDLE phNDEFRecord);
Создаёт объект записи, входящей в формат NDEF.
phNDEFRecord – указатель на переменную типа DWORD, куда будет помещён созданный указатель на объект записи сообщения в формате NDEF (при успешной отработке функции).

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.2 CLSCRF_NFC_NDEF_Record_GetDataSize

LONG CLSCRF_NFC_NDEF_Record_GetDataSize(IN HANDLE hNDEFRecord,
OUT LPDWORD pdwDataSize);
Возвращает размер данных, хранящихся в записи NDEF.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
pdwDataSize – указатель на переменную типа DWORD, куда будет помещён размер записи сообщения NDEF в байтах.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.3 CLSCRF_NFC_NDEF_Record_GetTnf

LONG CLSCRF_NFC_NDEF_Record_GetTnf(IN HANDLE hNDEFRecord,
OUT LPBYTE pucTnf);
Возвращает TNF (Type Name Format) записи, входящей в состав NDEF.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
pucTnf – указатель на байт, в который будет помещён TNF (Type Name Format) записи.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.4 CLSCRF_NFC_NDEF_Record_SetTnf

LONG CLSCRF_NFC_NDEF_Record_SetTnf(IN HANDLE hNDEFRecord,
IN BYTE ucTnf);
Устанавливает TNF (Type Name Format) записи, входящей в состав NDEF.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;

ucTnf – TNF (Type Name Format) записи.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.2.5 CLSCRF_NFC_NDEF_Record_GetTypeLength

LONG CLSCRF_NFC_NDEF_Record_GetTypeLength(IN HANDLE hNDEFRecord,
OUT LPDWORD pdwTypeLength);

Возвращает размер типа записи, входящей в состав NDEF.

hNDEFRecord – указатель на объект записи сообщения в формате NDEF;

pdwTypeLength – указатель на переменную типа DWORD, в которую будет помещён размер массива типа записи сообщения.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.2.6 CLSCRF_NFC_NDEF_Record_GetType

LONG CLSCRF_NFC_NDEF_Record_GetType(IN HANDLE hNDEFRecord,
OUT LPBYTE lpTypeData);

Возвращает тип записи, входящей в состав NDEF, в виде массива данных.

hNDEFRecord – указатель на объект записи сообщения в формате NDEF;

lpTypeData – указатель на массив байт, в который будет скопирован тип записи сообщения NDEF.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.2.7 CLSCRF_NFC_NDEF_Record_GetTypeStr

LONG CLSCRF_NFC_NDEF_Record_GetTypeStr(IN HANDLE hNDEFRecord,
OUT LPWSTR lpwstrType);

Возвращает тип записи, входящей в состав NDEF, в виде текстовой строки.

hNDEFRecord – указатель на объект записи сообщения в формате NDEF;

lpwstrType – указатель строку в формате Unicode, в которую будет помещён тип записи сообщения NDEF. Размер буфера под строку должен быть достаточным для копирования в него типа, а также завершающего символа 0x00 0x00.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.2.8 CLSCRF_NFC_NDEF_Record_SetType

LONG CLSCRF_NFC_NDEF_Record_SetType(IN HANDLE hNDEFRecord,
IN LPBYTE lpTypeData,
IN DWORD dwTypeLength);

Устанавливает тип записи, входящей в состав NDEF, в виде массива данных.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
lpTypeData – указатель на массив байт, содержащий тип записи;
dwTypeLength – длина массива lpTypeData.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.9 CLSCRF_NFC_NDEF_Record_SetTypeStr

LONG CLSCRF_NFC_NDEF_Record_SetTypeStr(IN HANDLE hNDEFRecord,
IN LPWSTR lpwstrType);

Устанавливает тип записи, входящей в состав NDEF, в виде текстовой строки.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
lpwstrType – указатель на строку типав формате Unicode, которую следует установить.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.10 CLSCRF_NFC_NDEF_Record_GetPayloadLength

LONG CLSCRF_NFC_NDEF_Record_GetPayloadLength(IN HANDLE hNDEFRecord,
OUT LPDWORD pdwPayloadLength);

Возвращает размер данных записи, входящей в состав NDEF.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
pdwPayloadLength – указатель на переменную типа DWORD, в которую будет записана длина данных записи сообщения NDEF в байтах.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.11 CLSCRF_NFC_NDEF_Record_GetPayload

LONG CLSCRF_NFC_NDEF_Record_GetPayload(IN HANDLE hNDEFRecord,
OUT LPBYTE lpPayloadData);

Возвращает данные записи, входящей в состав NDEF, в виде массива данных.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
lpPayloadData – указатель на массив байт, в который будут скопированы данные записи сообщения NDEF.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.12 CLSCRF_NFC_NDEF_Record_GetPayloadText

LONG CLSCRF_NFC_NDEF_Record_GetPayloadText(IN HANDLE hNDEFRecord,
OUT LPWSTR lpwstrText);
Возвращает данные записи, входящей в состав NDEF, в виде текстовой строки (при типе записи - текст).
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
lpwstrText – указатель на строку в формате Unicode, в которую будут поещены данные записи сообщения NDEF. Размер буфера под строку должен быть достаточным для копирования в него данных, а также завершающего символа 0x00 0x00.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.13 CLSCRF_NFC_NDEF_Record_GetPayloadUri

LONG CLSCRF_NFC_NDEF_Record_GetPayloadUri(IN HANDLE hNDEFRecord,
OUT LPWSTR lpwstrUri);
Возвращает данные записи, входящей в состав NDEF, в виде строки URI (Universal Resource Identifier, при типе записи - URI).
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
lpwstrUri – указатель на строку в формате Unicode, в которую будет помещён URI из записи сообщения NDEF. Размер буфера под строку должен быть достаточным для копирования в него URI, а также завершающего символа 0x00 0x00.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.14 CLSCRF_NFC_NDEF_Record_SetPayload

LONG CLSCRF_NFC_NDEF_Record_SetPayload(IN HANDLE hNDEFRecord,
IN LPBYTE lpPayloadData,
IN DWORD dwPayloadLength);
Устанавливает данные записи, входящей в состав NDEF, в виде массива данных.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
lpPayloadData – указатель на массив байт, из которого будут скопированы данные записи сообщения NDEF;
dwPayloadLength – длина данных массива в байтах.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.15 CLSCRF_NFC_NDEF_Record_GetIdLength

LONG CLSCRF_NFC_NDEF_Record_GetIdLength(IN HANDLE hNDEFRecord,
OUT LPDWORD pdwIdLength);
Возвращает размер идентификатора записи, входящей в состав NDEF.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
pdwIdLength – указатель на переменную типа DWORD, в которую будет помещен размер идентификатора записи в байтах.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.16 CLSCRF_NFC_NDEF_Record_GetId

LONG CLSCRF_NFC_NDEF_Record_GetId(IN HANDLE hNDEFRecord,
OUT LPBYTE lpIdData);
Возвращает идентификатор записи, входящей в состав NDEF, в виде массива данных.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
lpIdData – указатель на массив байт, в который будет помещен идентификатор записи сообщения NDEF.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.17 CLSCRF_NFC_NDEF_Record_GetIdStr

LONG CLSCRF_NFC_NDEF_Record_GetIdStr(IN HANDLE hNDEFRecord,
OUT LPWSTR lpwstrId);
Возвращает идентификатор записи, входящей в состав NDEF, в виде текстовой строки.
hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
lpwstrId – указатель на строку в формате Unicode, в которую будет помещен идентификатор записи сообщения NDEF. Размер буфера под строку должен быть достаточным для копирования в него идентификатора, а также завершающего символа 0x00 0x00.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.18 CLSCRF_NFC_NDEF_Record_SetId

LONG CLSCRF_NFC_NDEF_Record_SetId(IN HANDLE hNDEFRecord,
IN LPBYTE lpIdData,
IN DWORD dwIdLength);

Устанавливает идентификатор записи, входящей в состав NDEF, в виде массива данных.

hNDEFRecord – указатель на объект записи сообщения в формате NDEF;
lpIdData – указатель на массив байт, содержащий устанавливаемый идентификатор записи;
dwIdLength – длина идентификатора записи в байтах.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2.19 CLSCRF_NFC_NDEF_Record_Destroy

LONG CLSCRF_NFC_NDEF_Record_Destroy(IN HANDLE hNDEFRecord);

Удаляет объект записи, входящей в формат NDEF.

hNDEFRecord – указатель на объект записи сообщения в формате NDEF.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3 Функции работы с протоколом LLCP

В данной группе собраны функции NFC, реализующие протокол LLCP (Logical Link Control Protocol) в упрощённом демонстрационном формате.

4.14.3.1 CLSCRF_NFC_LLCP_Create

LONG CLSCRF_NFC_LLCP_Create(IN LPVOID pReader,
OUT LPHANDLE phLLCPLink);

Создаёт объект интерфейса протокола LLCP.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
phLLCPLink – ссылка на переменную типа DWORD, в которую будет помещен указатель на созданный объект протокола LLCP.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.2 CLSCRF_NFC_LLCP_SetTimeout

LONG CLSCRF_NFC_LLCP_SetTimeout(IN HANDLE hLLCPLink,
IN DWORD dwTimeoutMs);

Устанавливает таймаут соединения по протоколу LLCP.
Следует вызывать до открытия соединения.

hLLCPLink – указатель на объект протокола LLCP;

dwTimeoutMs – таймаут соединения в миллисекундах.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.3 CLSCRF_NFC_LLCP_SetBaudrate

LONG CLSCRF_NFC_LLCP_SetBaudrate(IN HANDLE hLLCPLink,
IN BYTE ucDsiDri);
Устанавливает скорость обмена по протоколу LLCP.
Следует вызывать до открытия соединения.
hLLCPLink – указатель на объект протокола LLCP;
ucDsiDri – индекс делителя скорости обмена 0-2 (PHPAL_I18092MPI_DATARATE_106, PHPAL_I18092MPI_DATARATE_212, PHPAL_I18092MPI_DATARATE_424).
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.4 CLSCRF_NFC_LLCP_SetGeneralInformation

LONG CLSCRF_NFC_LLCP_SetGeneralInformation(IN HANDLE hLLCPLink,
IN PBYTE lpGI,
IN DWORD dwGILength);
Устанавливает байты общей информации, используемые при соединении по протоколу LLCP.
Следует вызывать до открытия соединения.
hLLCPLink – указатель на объект протокола LLCP;
lpGI – указатель на массив, содержащий байты общей информации;
dwGILength – размер массива байт общей информации.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.5 CLSCRF_NFC_LLCP_SetNFCID3Information

LONG CLSCRF_NFC_LLCP_SetNFCID3Information(IN HANDLE hLLCPLink,
IN PBYTE lpNFCID3i);
Устанавливает байты идентификатора NFCID3, используемые при установке соединения по протоколу LLCP.
Следует вызывать до открытия соединения.
hLLCPLink – указатель на объект протокола LLCP;
lpNFCID3i – указатель на массив байт, содержащий идентификатор NFCID3.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.6 CLSCRF_NFC_LLCP_ServerOpen

LONG CLSCRF_NFC_LLCP_ServerOpen(IN HANDLE hLLCPLink);
Открывает соединение типа сервер.
hLLCPLink – указатель на объект протокола LLCP.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.7 CLSCRF_NFC_LLCP_ServerReceive

LONG CLSCRF_NFC_LLCP_ServerReceive(IN HANDLE hLLCPLink,
OUT LPBYTE lpData,
OUT DWORD lpdwDataLength);
Запускает ожидание приёма данных от устройства-клиента при соединении типа сервер.
hLLCPLink – указатель на объект протокола LLCP;
lpData – указатель на массив байт, в который будут скопированы полученные данные;
lpdwDataLength – указатель на переменную типа DWORD, в которую будет помещен размер полученных данных в байтах.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.8 CLSCRF_NFC_LLCP_ServerClose

LONG CLSCRF_NFC_LLCP_ServerClose(IN HANDLE hLLCPLink);
Закрывает соединение типа сервер.
hLLCPLink – указатель на объект протокола LLCP.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.9 CLSCRF_NFC_LLCP_ClientOpen

LONG CLSCRF_NFC_LLCP_ClientOpen(IN HANDLE hLLCPLink);
Открывает соединение типа клиент с удалённым сервером.
hLLCPLink – указатель на объект протокола LLCP.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.10 CLSCRF_NFC_LLCP_ClientTransmit

LONG CLSCRF_NFC_LLCP_ClientTransmit(IN HANDLE hLLCPLink,
IN LPBYTE lpData,
IN DWORD dwDataLength);
Выполняет передачу данных на сервер при соединении типа клиент.
hLLCPLink – указатель на объект протокола LLCP;

lpData – указатель на массив байт, из которого нужно взять данные для передачи;
dwDataLength – количество передаваемых из массива байт.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.11 CLSCRF_NFC_LLCP_ClientClose

LONG CLSCRF_NFC_LLCP_ClientClose(IN HANDLE hLLCPLink);
Закрывает соединение типа клиент.
hLLCPLink – указатель на объект протокола LLCP.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3.12 CLSCRF_NFC_LLCP_Destroy

LONG CLSCRF_NFC_LLCP_Destroy(IN HANDLE hLLCPLink);
Удаляет объект интерфейса связи по протоколу LLCP.
hLLCPLink – указатель на объект протокола LLCP.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.4 Функции работы с протоколом SNEP

В данной группе собраны функции NFC, реализующие протокол SNEP (Simple NDEF Exchange Protocol) в упрощённом демонстрационном формате.

4.14.4.1 CLSCRF_NFC_SNEP_Create

LONG CLSCRF_NFC_SNEP_Create(IN HANDLE hLLCPLink,
OUT LPHANDLE phSNEP);
Создаёт объект интерфейса протокола SNEP.
hLLCPLink – указатель на объект протокола LLCP;
phSNEP – ссылка на переменную типа DWORD, в которую будет помещен указатель на созданный объект протокола SNEP.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.4.2 CLSCRF_NFC_SNEP_Receive

LONG CLSCRF_NFC_SNEP_Receive(IN HANDLE hSNEP,
OUT LPBYTE lpData,
OUT LPDWORD pdwDataLength);
Открывает LLCP соединение типа Server и принимает через него данные по протоколу SNEP.

hSNEP – указатель на объект протокола SNEP;
lpData – указатель на массив байт, в который будут помещены полученные данные;
pdwDataLength – указатель на переменную типа DWORD, в которую будет помещен размер полученных данных в байтах.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.4.3 CLSCRF_NFC_SNEP_Transmit

LONG CLSCRF_NFC_SNEP_Transmit(IN HANDLE hSNEP,
IN LPBYTE lpData,
IN DWORD dwDataLength);

Открывает LLCP соединение типа Client и передаёт через него данные по протоколу SNEP.
hSNEP – указатель на объект протокола SNEP;
lpData – указатель на массив байт, из которого будут взяты данные для передачи;
dwDataLength – количество передаваемых байт.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.4.4 CLSCRF_NFC_SNEP_Destroy

LONG CLSCRF_NFC_SNEP_Destroy(IN HANDLE hSNEP);
Удаляет объект интерфейса связи по протоколу SNEP.
hSNEP – указатель на объект протокола SNEP.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.5 Функции работы с NFC метками

В данной группе собраны функции NFC, предназначенные для работы с метками стандарта NFC Forum Tags 2, 4 в упрощённом демонстрационном формате.

4.14.5.1 CLSCRF_NFC_ForumTags_SupposeTypeISO14443A

LONG CLSCRF_NFC_ForumTags_SupposeTypeISO14443A(IN LPVOID pReader,
IN LPBYTE ATQ,
IN BYTE SAK,
OUT LPBYTE
pucTagType);
Выполняет команды ISO18092 ATR и PSL.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

ATQ – ссылка на массив (2 байта), содержащий ATQ карты, полученный ранее посредством вызова функции Activate Idle;
SAK – 1 байт, содержащий SAK карты, полученный ранее посредством вызова функции Activate Idle;
pucTagType – указатель на переменную типа BYTE, куда будет записан тип метки по классификации NFC Forum Tag Type. Поддерживаемые значения:

- 2, 4 - типы меток:
  - NFC Forum Tag Type 2 - Mifare Ultralaight, Mifare Ultralight C, NTAG 203, NTAG 213, NTAG 210, NTAG 216;
  - NFC Forum Tag Type 4 - Mifare DESFire (EV1);
- 0 - в поле устройство NFC P2P (ISO18092);
- 255 - тип метки не поддерживается функционалом NFC библиотеки.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.5.2 CLSCRF_NFC_ForumTags_BeginType4

LONG CLSCRF_NFC_ForumTags_BeginType4(IN LPVOID pReader);
Переводит метку типа NFC Forum Tag Type 4 в режим T=CL.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.5.3 CLSCRF_NFC_ForumTags_Format

LONG CLSCRF_NFC_ForumTags_Format(IN LPVOID pReader,
IN BYTE ucTagType);
Выполняет форматирование (инициализацию) метки типа NFC Forum Tag Type 2 или 4, если метка не была уже отформатирована.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucTagType – тип метки по классификации NFC Forum Tag Type. Поддерживаемые значения - 2,4.
Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.5.4 CLSCRF_NFC_ForumTags_Read

LONG CLSCRF_NFC_ForumTags_Read(IN LPVOID pReader,
IN BYTE ucTagType,
OUT LPBYTE lpData,
IN OUT LPDWORD lpdwDataSize,
OUT LPBYTE lpucPermissions);
Выполняет чтение данных в формате NDEF из метки типа NFC Forum Tag.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucTagType – тип метки по классификации NFC Forum Tag Type. Поддерживаемые значения - 2,4.

lpData – указатель на массив байт, куда будут записаны прочитанные данные в формате NDEF из метки NFC;
lpdwDataSize – указатель на переменную типа DWORD, куда будет записано количество вычитанных из метки байт данных;
lpucPermissions – указатель на переменную типа BYTE, куда будет записан байт разрешений:

бит 0 - доступ для записи в метку (0 - полный, 1 - запрещён),
биты 1-7 - зарезервированы.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.5.5 CLSCRF_NFC_ForumTags_Write

LONG CLSCRF_NFC_ForumTags_Write(IN LPVOID pReader,
IN BYTE ucTagType,
IN LPBYTE lpData,
IN DWORD dwDataSize);

Выполняет запись данных в формате NDEF в метку типа NFC Forum Tag.
pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
ucTagType – тип метки по классификации NFC Forum Tag Type. Поддерживаемые значения - 2,4.
lpData – указатель на массив байт, содержащих данные в формате NDEF для записи в метку NFC;
dwDataSize – количество байт из массива данных, которые нужно записать в метку.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.6 Примеры работы с демонстрационными функциями NFC

В данной группе приведены примеры работы с демонстрационными функциями NFC.

Примеры сделаны на основе кода проекта TestNFC663, входящего в дистрибутив комплекта поставки считывателя.

В начале работы выполняем все необходимые процедуры инициализации.

```
LONG StopInterface(void *pReader)
{
	LONG Status;

	// Выключаем микросхему радиоинтерфейса считывателя
	Status = CLSCRF_Mfrc_Off(pReader);
```

```
        if (Status != SCARD_S_SUCCESS)
        {
                m_strResult = "Ошибка обращения к считывателю!";
                return Status;
        }

        // Закрываем соединение USB
        Status = CLSCRF_Close(pReader);
        if (Status != SCARD_S_SUCCESS)
        {
                m_strResult = "Ошибка вызова библиотеки!";
                CLSCRF_Destroy(&pReader);
                return Status;
        }

        // Удаляем из памяти объект считывателя
        Status = CLSCRF_Destroy(&pReader);
        if (Status != SCARD_S_SUCCESS)
        {
                m_strResult = "Ошибка вызова библиотеки!";
                return Status;
        }

        return Status;
}

// Переменная для хранения статуса, возвращаемого функцией
LONG Status;
// Переменная для хранения указателя на объект считывателя
void *pReader = NULL;

// Создаем объект считывателя
Status = CLSCRF_Create(&pReader);
if (Status != SCARD_S_SUCCESS)
{
        m_strResult = "Ошибка вызова библиотеки!";
        return Status;
}

// Открываем соединения со считывателем по USB
Status = CLSCRF_OpenUSB(pReader);
if (Status != SCARD_S_SUCCESS)
{
        m_strResult = "Не найден считыватель!";
        CLSCRF_Destroy(pReader);
        return Status;
}

// Включаем микросхему радиообмена
Status = CLSCRF_Mfrc_On(pReader);
if (Status != SCARD_S_SUCCESS)
{
        m_strResult = "Ошибка обращения к считывателю!";
        StopInterface(pReader);
        return Status;
}
```

```
	// Пауза нужна для того, чтобы метка успела войти в рабочий режим после
включения радиополя
	Sleep(50);

	DWORD dwState = 0;

	// Получаем состояние считывателя
	Status = CLSCRF_GetState(pReader, &dwState);
	if (Status != SCARD_S_SUCCESS)
	{
		m_strResult = "Ошибка обращения к считывателю!";
		StopInterface(pReader);
		return Status;
	}

	// Если текущий протокол обмена - не ISO14443A
	if ((dwState & 0xFFUL) != 0x00)
	{
		// Устанавливаем протокол ISO14443A
		Status = CLSCRF_Mfrc_Set_Rf_Mode(pReader, 0x00);
		if (Status != SCARD_S_SUCCESS)
		{
			m_strResult = "Ошибка обращения к считывателю!";
			StopInterface(pReader);
			return Status;
		}
	}

	BYTE m_UID[10];
	DWORD m_dwUIDLen;
	BYTE m_ATQ[2];
	BYTE m_SAK;

	// Попытка активировать метку в поле
	Status = CLSCRF_Activate_Idle_A(pReader,
		m_ATQ,// pbATQ 2 bytes
		&m_SAK,// pbSAK 1 byte
		m_UID,// pbUID max 10 bytes
		&m_dwUIDLen);

	if (Status != SCARD_S_SUCCESS)
	{
		m_strResult = "Ошибка чтения карты!";
		StopInterface(pReader);
		return Status;
	}

	BYTE m_ucTagType;
	// Определяем тип метки/устройства NFC
	Status = CLSCRF_NFC_ForumTags_SupposeTypeISO14443A(*ppReader, m_ATQ, m_SAK,
&m_ucTagType);
	if (Status != SCARD_S_SUCCESS)
	{
		m_strResult = "Ошибка обращения к библиотеке!";
		UpdateData(FALSE);
		StopInterface(*ppReader);
		return Status;
```

```
	}

	if (m_ucTagType == 0xFF)
	{
		m_strResult = "Тип метки не поддерживается!";
		UpdateData(FALSE);
		StopInterface(*ppReader);
		return SCARD_E_NO_SMARTCARD;
	}

	// Если найденная метка - типа NFC Forum Tag Type 4, то нужно предварительно
перевести её в режим T=CL
	if (m_ucTagType == 4)
	{
		CLSCRF_NFC_ForumTags_BeginType4(*ppReader);
		if (Status != SCARD_S_SUCCESS)
		{
			m_strResult = "Ошибка активации режима T=CL!";
			UpdateData(FALSE);
			StopInterface(*ppReader);
			return Status;
		}
	}
```

4.14.6.1 Приём сообщения NDEF с текстовой записью

Инициализируем массив для принимаемого сообщения.

```
BYTE lpMsgData[1024] = { 0 };
DWORD dwMsgSize = sizeof(lpMsgData);
```

4.14.6.1.1 Чтение сообщения из меток типа NFC Forum Tag 2, 4

Выполняется, если `m_ucTagType` == 2 или 4

```
	BYTE ucPermissions = 0x00;
	// Пытаемся почитать NDEF из метки
	Status = CLSCRF_NFC_ForumTags_Read(pReader, m_ucTagType, lpMsgData, &
dwMsgSize, &ucPermissions);
	if (Status != SCARD_S_SUCCESS)
	{
		m_strResult = "Ошибка чтения метки!";
		StopInterface(pReader);
		return;
	}
```

4.14.6.1.2 Получение сообщения от удалённого устройства в режиме P2P

Выполняется, если `m_ucTagType == 0`

```
	HANDLE hLLCPLink;
	// Создаём объект протокола LLCP
	Status = CLSCRF_NFC_LLCP_Create(pReader, &hLLCPLink);
	if (Status != SCARD_S_SUCCESS)
	{
		m_strResult = "Ошибка!";
		StopInterface(pReader);
		return;
	}

	// Устанавливаем скорость обмена по протоколу LLCP
	Status = CLSCRF_NFC_LLCP_SetBaudrate(hLLCPLink, PHPAL_I18092MPI_DATARATE_106
);
	if (Status != SCARD_S_SUCCESS)
	{
		m_strResult = "Ошибка!";
		StopInterface(pReader);
		return;
	}

	HANDLE hSNEP;
	// Создаём объект протокола SNEP
	Status = CLSCRF_NFC_SNEP_Create(hLLCPLink, &hSNEP);
	if (Status != SCARD_S_SUCCESS)
	{
		m_strResult = "Ошибка!";
		StopInterface(pReader);
		return;
	}


	// Открываем соединение LLCP типа Server и принимаем через него данные по
протоколу SNEP
	Status = CLSCRF_NFC_SNEP_Receive(hSNEP, lpMsgData, &dwMsgSize);
	if (Status != SCARD_S_SUCCESS)
	{
		m_strResult = "Ошибка!";
		StopInterface(pReader);
		return;
	}

	// Удаляем объект протокола SNEP
	Status = CLSCRF_NFC_SNEP_Destroy(hSNEP);
	if (Status != SCARD_S_SUCCESS)
	{
		m_strResult = "Ошибка!";
		StopInterface(pReader);
		return;
	}

	// Удаляем объект протокола LLCP
```

```
        Status = CLSCRF_NFC_LLCP_Destroy(hLLCPLink);
        if (Status != SCARD_S_SUCCESS)
        {
            m_strResult = "Ошибка!";
            StopInterface(pReader);
            return;
        }
```

4.14.6.1.3 Разбор принятого сообщения

```
        HANDLE hNDEFMessage;
        // Создаём объект сообщения NDEF на основе принятых данных
        Status = CLSCRF_NFC_NDEF_Message_Create(lpMsgData, dwMsgSize, &hNDEFMessage
);
        if (Status != SCARD_S_SUCCESS)
        {
            m_strResult = "Ошибка!";
            StopInterface(pReader);
            return;
        }

        DWORD dwRecCount = 0;
        // Получение количества записей в сообщении NDEF
        Status = CLSCRF_NFC_NDEF_Message_GetRecordsCount(hNDEFMessage, &dwRecCount);
        if (Status != SCARD_S_SUCCESS)
        {
            m_strResult = "Ошибка!";
            StopInterface(pReader);
            return;
        }

        HANDLE hNDEFRecord;
        if (dwRecCount > 0)
        {
            // Получение ссылки на объект первой записи в сообщении NDEF
            Status = CLSCRF_NFC_NDEF_Message_GetRecord(hNDEFMessage, 0, &
hNDEFRecord);
            if (Status != SCARD_S_SUCCESS)
            {
                m_strResult = "Ошибка!";
                StopInterface(pReader);
                return;
            }

            TCHAR lpStrType[16] = { 0 };
            // Считываем тип записи в строку
            Status = CLSCRF_NFC_NDEF_Record_GetTypeStr(hNDEFRecord, lpStrType);
            if (Status != SCARD_S_SUCCESS)
            {
                m_strResult = "Ошибка!";
                StopInterface(pReader);
                return;
            }

            if ((CString(lpStrType) == CString("T")) || (CString(lpStrType) ==
CString("U")))
```

```
                {
                        TCHAR lpStrText[1024] = { 0 };
                        // Считываем текст из данных записи в строку
                        Status = CLSCRF_NFC_NDEF_Record_GetPayloadText(hNDEFRecord,
lpStrText);
                        if (Status != SCARD_S_SUCCESS)
                        {
                                m_strResult = "Ошибка!";
                                StopInterface(pReader);
                                return;
                        }

                        m_strNDEFText = lpStrText;
                }
        }

        // Удаляем объект сообщения NDEF
        Status = CLSCRF_NFC_NDEF_Message_Destroy(hNDEFMessage);
        if (Status != SCARD_S_SUCCESS)
        {
                m_strResult = "Ошибка!";
                StopInterface(pReader);
                return;
        }

        m_strResult.Format(L"Сообщение успешно прочитано! Найдено записей: %d.",
dwRecCount);

        StopInterface(pReader);
```

4.14.6.2 Отправка сообщения NDEF с текстовой записью

Инициализируем массив для отправляемого сообщения.

```
BYTE lpMsgData[1024] = { 0 };
DWORD dwMsgSize = sizeof(lpMsgData);
```

4.14.6.2.1 Формирование отправляемого сообщения

Формируем объект сообщения NDEF.

```
HANDLE hNDEFMessage;
// Создаём объект сообщения NDEF
Status = CLSCRF_NFC_NDEF_Message_Create(NULL, 0, &hNDEFMessage);
if (Status != SCARD_S_SUCCESS)
{
        m_strResult = "Ошибка!";
        StopInterface(pReader);
        return;
}
```

```
        // Добавляем запись типа ТЕКСТ к сообщению NDEF
        Status = CLSCRF_NFC_NDEF_Message_AddRecordText(hNDEFMessage, L"Привет, мир!"
, L"ru");
        if (Status != SCARD_S_SUCCESS)
        {
                m_strResult = "Ошибка!";
                StopInterface(pReader);
                return;
        }

        // Определяем размер сообщения NDEF в байтах
        Status = CLSCRF_NFC_NDEF_Message_GetDataSize(hNDEFMessage, &dwMsgSize);
        if (Status != SCARD_S_SUCCESS)
        {
                m_strResult = "Ошибка!";
                UpdateData(FALSE);
                StopInterface(pReader);
                return;
        }

        // Копируем в массив данные сообщения NDEF
        Status = CLSCRF_NFC_NDEF_Message_GetData(hNDEFMessage, lpMsgData);
        if (Status != SCARD_S_SUCCESS)
        {
                m_strResult = "Ошибка!";
                UpdateData(FALSE);
                StopInterface(pReader);
                return;
        }
```

4.14.6.2.2 Запись сообщения в метки типа NFC Forum Tag 2, 4

Выполняется, если `m_ucTagType` == 2 или 4

```
        // Пытаемся записать сообщение NDEF в метку
        Status = CLSCRF_NFC_ForumTags_Write(pReader, m_ucTagType, lpMsgData,
dwMsgSize);
        if (Status != SCARD_S_SUCCESS)
        {
                m_strResult = "Ошибка записи в метку!";
                StopInterface(pReader);
                return;
        }

        StopInterface(pReader);

        m_strResult = "Сообщение успешно записано!";
```

4.14.6.2.3 Отправка сообщения на удалённое устройство в режиме P2P

Выполняется, если `m_ucTagType` == 0

```
HANDLE hLLCPLink;
// Создаём объект протокола LLCP
Status = CLSCRF_NFC_LLCP_Create(pReader, &hLLCPLink);
if (Status != SCARD_S_SUCCESS)
{
	m_strResult = "Ошибка!";
	StopInterface(pReader);
	return;
}

// Устанавливаем скорость обмена по протоколу LLCP
Status = CLSCRF_NFC_LLCP_SetBaudrate(hLLCPLink, PHPAL_I18092MPI_DATARATE_106
);
if (Status != SCARD_S_SUCCESS)
{
	m_strResult = "Ошибка!";
	StopInterface(pReader);
	return;
}

HANDLE hSNEP;
// Создаём объект протокола SNEP
Status = CLSCRF_NFC_SNEP_Create(hLLCPLink, &hSNEP);
if (Status != SCARD_S_SUCCESS)
{
	m_strResult = "Ошибка!";
	StopInterface(pReader);
	return;
}

// Открываем соединение типа Client по протоколу LLCP и
// передаем массив байт сообщения NDEF по протоколу SNEP
Status = CLSCRF_NFC_SNEP_Transmit(hSNEP, lpMsgData, dwMsgSize);
if (Status != SCARD_S_SUCCESS)
{
	m_strResult = "Ошибка!";
	StopInterface(pReader);
	return;
}

// Удаляем объект протокола SNEP
Status = CLSCRF_NFC_SNEP_Destroy(hSNEP);
if (Status != SCARD_S_SUCCESS)
{
	m_strResult = "Ошибка!";
	StopInterface(pReader);
	return;
}

// Удаляем объект протокола LLCP
Status = CLSCRF_NFC_LLCP_Destroy(hLLCPLink);
```

```
if (Status != SCARD_S_SUCCESS)
{
        m_strResult = "Ошибка!";
        StopInterface(pReader);
        return;
}

// Удаляем объект сообщения NDEF
Status = CLSCRF_NFC_NDEF_Message_Destroy(hNDEFMessage);
if (Status != SCARD_S_SUCCESS)
{
        m_strResult = "Ошибка!";
        StopInterface(pReader);
        return;
}

StopInterface(pReader);

m_strResult = "Сообщение успешно отправлено!";
```

4.15 Функции непосредственного обмена данными с картой

4.15.1 CLSCRF_DirectIO_Card

LONG CLSCRF_DirectIO_Card(IN LPVOID pReader,
IN LPCBYTE pbSendBuffer,
IN DWORD dwSendLength,
IN DWORD dwTimeout,
OUT LPBYTE pbRecvBuffer,
IN OUT LPDWORD pdwRecvLength);

Осуществляет передачу данных карте без предварительной обработки и последующий прием данных от карты, в том числе при выполнении команд карте по протоколу ISO 14443-4 (T=CL).

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pbSendBuffer – массив байтов, передаваемых карте;
dwSendLength – количество передаваемых в карту байтов;
dwTimeout – величина таймаута карты в единицах (128 / 13,56)[мкс];
pbRecvBuffer – массив для ответа от карты;
pdwRecvLength – ссылка на переменную, которая перед вызовом функции должна содержать размер массива pbRecvBuffer, а на выходе будет содержать количество принятых от карты байтов.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.16 Функции конфигурации устройств на шине RS485

4.16.1 CLSCRF_GetIOAddress

LONG CLSCRF_GetIOAddress(IN LPVOID pReader,
OUT LPBYTE pbIOAddr);

Выдает адрес устройства, с которым производится обмен данными.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pbIOAddr – адрес переменной, в которую будет помещен адрес устройства, с которым производится обмен данными.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.16.2 CLSCRF_SetIOAddress

LONG CLSCRF_SetIOAddress(IN LPVOID pReader,
IN BYTE bIOAddr);

Задает адрес устройства, с которым будет производиться обмен данными. Значение адреса 0 допустимо лишь при условии, что на шине RS485 находится только один считыватель.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
bIOAddr – адрес устройства, с которым будет производиться обмен данными.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.16.3 CLSCRF_ReadDeviceAddress

LONG CLSCRF_ReadDeviceAddress(IN LPVOID pReader,
OUT LPBYTE pbDevAddr);

Выдает адрес устройства. Если адрес устройства неизвестен, необходимо оставить устройство на шине RS485 единственным, задать адрес обмена 0 (см. CLSCRF_SetIOAddress()) и вызвать данную функцию.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);
pbDevAddr – адрес переменной, в которую будет помещен адрес устройства.

Возвращаемое значение:
0 – успешное выполнение,
иначе – ошибка при выполнении.

4.16.4 CLSCRF_WriteDeviceAddress

LONG CLSCRF_WriteDeviceAddress(IN LPVOID pReader,
IN BYTE bDevAddr);

Задает устройству новый адрес и записывает его во flash-памяти.

pReader – ссылка на объект-интерфейс (см. функцию CLSCRF_Create);

bDevAddr – новый адрес устройства.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.17 Обработка ошибок при использовании библиотеки Clscrfl.dll

В процессе работы со считывателем с использованием библиотеки Clscrfl.dll успешное выполнение вызываемой функции сопровождается кодом возврата, равным 0x00000000.

Среди других кодов возврата наиболее распространенным является код 0x80100001. Он означает отсутствие ошибок в канале связи и отличный от нуля код завершения при выполнении команды считывателем. Код завершения можно получить путем вызова функции CLSCRF_GetLastInternalError Возвращаемое значение: 0 – успешное выполнение,

иначе – ошибка при выполнении.

CLSCRF_GetLastInternalError(…). Перечень кодов завершения приведен в разделе Коды завершения команды.

При искажении кадра запроса в канале связи устройство в соответствии с протоколом обмена может не отвечать на команду. В этом случае код возврата равен 0x000005B4, что означает истечение времени ожидания ответа (таймаут). По умолчанию величина таймаута равна 1 секунде.

Разные приложения могут потребовать различных значений таймаута. Для этого библиотека Clscrfl.dll содержит две функции обслуживания таймаута:

Возвращаемое значение: 0 – успешное выполнение,

иначе – ошибка при выполнении.

CLSCRF_GetIOTimeout(…)

Возвращаемое значение: 0 – успешное выполнение,

иначе – ошибка при выполнении.

CLSCRF_SetIOTimeout(…)

С помощью этих функций пользователь может узнать текущее значение таймаута, а при необходимости – установить новое значение.

Рекомендации по обработке ошибок сводятся к следующему.

Ошибка с кодом 0x80100001 исправляется путем повторения последней команды. Если 3-кратное повторение команды не приводит к успеху, необходимо уточнить следующее:

- исправность текущей карты;
- правильное расположение RFID-карты в зоне обслуживания считывателя;
- совместимость текущей карты с версией считывателя;

- соответствие режима работы считывателя типу текущей карты;
- соблюдение порядка работы с текущей картой (например, своевременное проведение аутентификации сектора в карте Mifare);

При возникновении ошибки с кодом 0x000005B4 необходимо:

1. Выждать интервал времени, равный 200 мс;
2. Вызвать функцию CLSCRF_GetState(…);
3. Если результатом является та же ошибка – повторить пункты 1, 2. Иначе – продолжить работу в обычном режиме.
4. Если количество повторений превысило 4, можно предположить неисправность канала связи или считывателя.

МикроЭМ

Руководство программиста

Часть V

5 Рекомендации по работе с картами в протоколе T=CL

Работа с картами, поддерживающими протокол T=CL (ISO 14443-4), происходит по следующему сценарию.

1. Активация карты (из состояния IDLE или HALT)
2. Осуществление одной или нескольких операций обмена
3. Деактивация карты (перевод карты в состояние HALT)

Пункты 2 и 3 идентичны для карт обоих типов (A и B).

Активация должна происходить на минимальной скорости (106 Кбод). После активации карты необходимо присвоить ей логический идентификатор, получить от карты ее параметры (размер буфера, скорость, таймауты) и установить желаемые параметры обмена (скорость).

Пункт 1 существенно различается для карт типа A и для карт типа B.

5.1 Активация карты типа A из состояния IDLE

5.1.1 Переключить режим Rf на тип A скорость 106.

Команда 51 00
Ответ 00

5.1.2 Активировать карту типа A из состояния IDLE.

Команда 43
Ответ 00 44 03 20 07 04 5E 32 41 B4 C4 40

Код завершения = 0, поэтому продолжаем работу и разбираем ответ:

44 03 – ATQ = 0344 – тип карты Mifare DESFire. Есть тонкость: если карт несколько, все их ATQ складываются операцией логическое ИЛИ. Поэтому проверка на тип DESFire должна выглядеть так:
if((ATQ & 0x0344) == 0x0344) …

20 – SAK = 20 - антиколлизия прошла успешно, карта поддерживает протокол T=CL (ISO 14443-4).

07 – длина UID

04 5E 32 41 B4 C4 40 – UID

5.1.3 Запросить параметры протокола карты.

Команда 48 02 00 E0 50 90 1A 00 00

02 00 – длина посылки в карту равна 2.

E0 – стартовый байт запроса к карте о ее параметрах.
50 – в соответствии с ISO 14443-4 п.5.1 выбираем свободный CID (здесь 0), размер буфера считывателя равен 64 байта, поэтому значение байта, следующего за E0, будет равно 50.
90 1A 00 00 – значение таймаута у считывателя на выполнение данной команды.
Ответ 00 06 00 06 75 77 81 02 80
Код завершения = 0, поэтому продолжаем работу и разбираем ответ.
06 00 – длина ответа от карты – 6 байтов
В соответствии с ISO 14443-4 п.5.1
06 75 77 81 02 80 – это ATS
06 – длина ATS вместе с самой длиной
75 – T0 : далее следуют TA(1), TB(1), TC(1), а размер буфера карты равен 64.
77 – TA(1) : карта поддерживает все 4 скорости обмена в обоих направлениях, причем скорости в разных направлениях могут быть разными.
81 – TB(1) : таймаут карты 77 мс, а задержка перед следующей командой должна быть не менее 604 мкс.
02 – карта поддерживает обращение к ней по логическому идентификатору CID.
80 – historical byte.

5.1.4 Установить текущие параметры протокола обмена с картой типа A.

Команда 48 03 00 D0 11 00 90 1A 00 00
03 00 – длина посылки карте равна 3
D0 – команда карте с логическим идентификатором 0 установить параметры
11 – всегда означает, что следующий байт присутствует в команде.
00 – дальнейший обмен будет вестись на скорости 106 Кбод в обоих направлениях.
90 1A 00 00 – таймаут.
Ответ 00 01 00 D0
Код завершения = 0
01 00 – длина ответа от карты – 1 байт
D0 – параметры протокола установлены, кроме того, подтверждается, что в данном сеансе связи логический идентификатор карты 0.

5.1.5 Переключить режим Rf на тип A скорость 106.

Команда 51 00
Ответ 00

5.2 Активация карты типа A из состояния HALT

5.2.1 Переключить режим Rf на тип A скорость 106.

Команда 51 00
Ответ 00

5.2.2 Активировать карту типа A из состояния HALT.

Команда 44 07 04 5E 32 41 B4 C4 40
07 – длина UID
04 5E 32 41 B4 C4 40 – UID
Ответ 00 44 03 20
Код завершения = 0, поэтому продолжаем работу и разбираем ответ:
44 03 – ATQ = 0344 – тип карты Mifare DESFire. Есть тонкость: если карт несколько, все их ATQ складываются операцией логическое ИЛИ. Поэтому проверка на тип DESFire должна выглядеть так:
if((ATQ & 0x0344) == 0x0344) …
20 – SAK = 20 - антиколлизия прошла успешно, карта поддерживает протокол T=CL (ISO 14443-4).

5.2.3 Запросить параметры протокола карты.

Команда 48 02 00 E0 50 90 1A 00 00
02 00 – длина посылки в карту равна 2.
E0 – стартовый байт запроса к карте о ее параметрах.
50 – в соответствии с ISO 14443-4 п.5.1 выбираем свободный CID (здесь 0), размер буфера считывателя равен 64 байта, поэтому значение байта, следующего за E0, будет равно 50.
90 1A 00 00 – значение таймаута у считывателя на выполнение данной команды.
Ответ 00 06 00 06 75 77 81 02 80
Код завершения = 0, поэтому продолжаем работу и разбираем ответ.
06 00 – длина ответа от карты – 6 байтов
В соответствии с ISO 14443-4 п.5.1
06 75 77 81 02 80 – это ATS
06 – длина ATS вместе с самой длиной
75 – T0 : далее следуют TA(1), TB(1), TC(1), а размер буфера карты равен 64.
77 – TA(1) : карта поддерживает все 4 скорости обмена в обоих направлениях, причем скорости в разных направлениях могут быть разными.
81 – TB(1) : таймаут карты 77 мс, а задержка перед следующей командой должна быть не менее 604 мкс.
02 – карта поддерживает обращение к ней по логическому идентификатору CID.
80 – historical byte.

5.2.4 Установить текущие параметры протокола обмена с картой типа A.

Команда 48 03 00 D0 11 0A 90 1A 00 00
03 00 – длина посылки карте равна 3
D0 – команда карте с логическим идентификатором 0 установить параметры
11 – всегда, означает, что следующий байт присутствует в команде.
0A – дальнейший обмен будет вестись на скорости 424 Кбод в обоих направлениях.
90 1A 00 00 – таймаут.
Ответ 00 01 00 D0
Код завершения = 0, поэтому продолжаем работу и разбираем ответ.
01 00 – длина ответа от карты – 1 байт
D0 – параметры протокола установлены, кроме того, подтверждается, что в данном сеансе связи логический идентификатор карты 0.

5.2.5 Переключить режим Rf на тип A скорость 424.

Команда 51 0A
Ответ 00

5.3 Активация карты типа B из состояния IDLE

5.3.1 Переключить режим Rf на тип B скорость 106.

Команда 51 10
Ответ 00

5.3.2 Активировать карту типа B из состояния IDLE.

Команда 56 00 02
00 – фильтр приложения
02 – 4 слота антиколлизии
Ответ 00 9C 01 F5 B8 00 00 00 00 33 81 B3
Код завершения = 0, поэтому продолжаем работу и разбираем ответ:
PUPI = 9C 01 F5 B8
AppData = 00 00 00 00
ProtInfo = 33 81 B3
В соответствии с ISO 14443-3 п.7.9.4
33 – карта поддерживает 3 скорости обмена до 424 Кбод включительно в обоих направлениях, причем скорости в разных направлениях могут быть разными.

81 – размер буфера карты равен 256, карта поддерживает протокол T=CL (ISO 14443-4).
B3 – таймаут карты 618 мс, карта поддерживает обращение к ней по логическим идентификаторам NAD и CID.

5.3.3 Установить текущие параметры протокола обмена с картой типа B.

Команда 54 08 9C 01 F5 B8 00 05 01 00
08 – длина передаваемых данных (PUPI+ Param1+ Param2+ Param3+ Param4)
9C 01 F5 B8 – PUPI
00 – Param1 (см. ISO 14443-3 п. 7.10.3) (все по умолчанию)
05 – Param2 (см. ISO 14443-3 п. 7.10.4) (106 Кбод)
01 – Param3 (см. ISO 14443-3 п. 7.10.5) (ISO 14443-4)
00 – Param4 (см. ISO 14443-3 п. 7.10.6) (CID)
Ответ 00 00 80 31 80 66 40 90 89 12 08 05 83 01 90 00
Код завершения = 0, поэтому продолжаем работу и разбираем ответ:
00 – нет информации об ограничениях для приема цепочки кадров, кроме того, подтверждается, что в данном сеансе связи логический идентификатор карты равен 0.
80 31 80 66 40 90 89 12 08 05 83 01 90 00 – дополнительные данные, которых в общем случае может не быть.

5.3.4 Переключить режим Rf на тип B скорость 106.

Команда 51 10
Ответ 00

5.4 Активация карты типа B из состояния HALT

5.4.1 Переключить режим Rf на тип B скорость 106.

Команда 51 10
Ответ 00

5.4.2 Активировать карту типа B из состояния HALT.

Команда 57 00 02
00 – фильтр приложения
02 – 4 слота антиколлизии
Ответ 00 9C 01 F5 B8 00 00 00 00 33 81 B3
Код завершения = 0, поэтому продолжаем работу и разбираем ответ:

PUPI = 9C 01 F5 B8
AppData = 00 00 00 00
ProtInfo = 33 81 B3
В соответствии с ISO 14443-3 п.7.9.4
33 – карта поддерживает 3 скорости обмена до 424 Кбод включительно в обоих направлениях, причем скорости в разных направлениях могут быть разными.
81 – размер буфера карты равен 256, карта поддерживает протокол T=CL (ISO 14443-4).
B3 – таймаут карты 618 мс, карта поддерживает обращение к ней по логическим идентификаторам NAD и CID.

5.4.3 Установить текущие параметры протокола обмена с картой типа B.

Команда 54 08 9C 01 F5 B8 00 55 01 00
08 – длина передаваемых данных (PUPI+ Param1+ Param2+ Param3+ Param4)
9C 01 F5 B8 – PUPI
00 – Param1 (см. ISO 14443-3 п. 7.10.3) (все по умолчанию)
55 – Param2 (см. ISO 14443-3 п. 7.10.4) (212 Кбод)
01 – Param3 (см. ISO 14443-3 п. 7.10.5) (ISO 14443-4)
00 – Param4 (см. ISO 14443-3 п. 7.10.6) (CID)
Ответ 00 00 80 31 80 66 40 90 89 12 08 05 83 01 90 00
Код завершения = 0, поэтому продолжаем работу и разбираем ответ:
00 – нет информации об ограничениях для приема цепочки кадров, кроме того, подтверждается, что в данном сеансе связи логический идентификатор карты равен 0.
80 31 80 66 40 90 89 12 08 05 83 01 90 00 – дополнительные данные, которых в общем случае может не быть.

5.4.4 Переключить режим Rf на тип B скорость 212.

Команда 51 15
Ответ 00

5.5 Деактивация карты

Деактивация карты любого типа производится командой DESELECT (см. ISO 14443-4 п. 8).

5.5.1 Деактивировать карту, работающую в протоколе T=CL.

Команда 48 02 00 CA 00 90 1A 00 00
02 00 – длина посылки карте равна 2
CA – команда DESELECT

00 – логический идентификатор карты
90 1A 00 00 – таймаут.
Ответ 00 02 00 CA 00
Код завершения = 0, поэтому продолжаем работу и разбираем ответ:
02 00 – длина ответа от карты – 2 байта
CA – команда DESELECT выполнена
00 – логический идентификатор карты 0 освобожден.