

Считыватель бесконтактный Руководство программиста

Версия 9.2.1



© 2021 Акционерное общество “МикроЭМ”

Москва

Содержание

Часть I Общая информация	16
1 Перечень нормативных документов.....	16
2 Список сокращений.....	16
3 Предупреждение.....	17
Часть II Взаимодействие считывателя с управляющим устройством	20
1 Интерфейсы.....	20
USB	20
RS232	21
RS485	21
2 Протокол обмена данными.....	21
Кадр запроса	22
Кадр запроса шифрованный	23
Кадр ответа	24
Кадр ответа шифрованный	25
Коды завершения команды	26
Байтстаффинг	29
Контрольная сумма кадра	30
Многобайтовые переменные	31
Часть III Система команд	34
1 Управление считывателем.....	34
0x0E Выдача состояния считывателя	34
0x10 Инициализация микросхемы-считывателя	35
0x04 Выключение микросхемы-считывателя	35
0x64 Выдача версии считывателя	35
0x22 Выдача серийного номера микросхемы-считывателя	35
0x58 0x58 Выдача уникального идентификатора считывателя	35
0x58 0x77 Перезагрузка считывателя	36
0x20 Сброс/выключение электромагнитного поля (RF)	36
0x51 Переключение режима электромагнитного поля (RF)	36
0x05 Подача звукового сигнала	37
0x07 Управление светодиодом	37
0x0F Изменение скорости обмена по COM-порту	38
0x70 Очистка flash-памяти с ключами	38
0x79 Запись во flash-память блока с ключом	38
0x6E Проверка заполнения блока flash-памяти считывателя	39
0x6F (Сброс) аутентификация считывателя	39
0x68 Работа с конфигурацией считывателя	40
0x75 Комплексная активация карт ISO 14443-A и ISO 14443-B из состояния IDLE	42
0x47 Обмен в режиме T=CL	43
0x37 Управление выходными линиями	43
Конфигурация устройств на шине RS485	44
0x7A Чтение адреса устройства.....	44
0x77 Запись адреса устройства.....	44
2 Управление картами типа A стандарта ISO 14443	45
0x43 Активация карты типа A, находящейся в состоянии Idle	45

0x1D Перевод активной карты типа A в состояние Halt	45
0x44 Активация карты типа A, находящейся в состоянии Halt	45
0x35 Вход в режим T=CL и чтение информации ATS из карты	45
0x36 Установка протокола и параметров работы с картой по протоколу T=CL	46
0x2B Активировать режим T=CL: получение доступных скоростей и установка скорости работы с картой	
3 Управление картами типа B стандарта ISO 14443	47
0x56 Активация карт типа B, находящихся в состоянии Idle	47
0x54 Установка параметров протокола в данном сеансе связи с картой типа B	47
0x55 Перевод активной карты типа B в состояние Halt	48
0x57 Активация карт типа B, находящихся в состоянии Halt	48
4 Управление метками стандарта ISO 15693	48
0x30 Инвентаризация меток 15693	48
0x32 Перевод метки 15693 в состояние QUIET	49
5 Обмен данными с картой Mifare Classic	50
Вычисление абсолютного номера блока	50
0x16 (обратная совместимость) Кодирование ключа	50
0x18 (обратная совместимость) Аутентификация ключом, заданным в команде	50
0x14 Аутентификация ключом, заданным в команде	50
0x17 Запись ключа в EEPROM считывателя	51
0x15 Аутентификация ключом, находящимся в EEPROM считывателя	51
0x19 Чтение блока	51
0x1A Запись блока	52
0x1B Операция Value	52
0x1C Персонализация UID	52
0x26 Изменение нагрузки антенны карт	52
0x46 Настраиваемый обмен с картой	53
6 Обмен данными с картой Mifare UltraLight (C)	53
0x25 (устарела) Чтение одной страницы	53
0x19 Чтение четырёх страниц	53
0x1E Запись страницы	54
0x2C Запись ключа аутентификации	54
0x2D Аутентификация карты	54
7 Обмен данными с картой Mifare UltraLight EV1	54
0xC460 Чтение версии карты	54
0xC430 Чтение четырёх страниц	55
0xC43A Чтение диапазона страниц	55
0xC4A2 Запись страницы	55
0xC41B Верификация пароля	55
0xC439 Прочитать значение счетчика	56
0xC4A5 Увеличить значение счетчика	56
0xC43C Прочитать подпись ECC	56
0xC43E Check if a tearing event	56
0xC44B Заключительный запрос поддержки виртуальных карт	57
8 Обмен данными с картой Mifare Plus	57
Таблица формирования параметра типа значения	57
Таблица формирования параметра режима защиты передачи данных	58
Таблица допустимых режимов защиты передачи данных	58
0xA8 Запись данных персонализации в карту	59
0xAA Персонализация карты	59
0xA0 Управление аутентификацией	59
0xE92D Управление аутентификацией с ключом в команде	59
0xA6 Чтение нескольких блоков SL2	60
0xA7 Запись нескольких блоков SL2	60
0xA4 Чтение данных	60
0xA5 Запись данных	61
0xA1 Прибавление значения	61

0xA1 Вычитание значения	61
0xA1 Запись данных из буфера переноса в блок	61
0xA1 Прибавление значения с последующей записью данных из буфера переноса в блок	62
0xA1 Вычитание значения с последующей записью данных из буфера переноса в блок	62
0xA1 Запись данных блока значения в буфер переноса	62
0xA2 Начальный и промежуточный запрос поддержки виртуальных карт	63
0xA2 Завершающий запрос поддержки виртуальных карт	63
0xA3 Выбор виртуальной карты	63
0xA3 Снятие выбора виртуальной карты	64
0xA9 Поиск релейной атаки	64
9 Обмен данными с картой Mifare Plus EV1	64
Таблица формирования параметра типа значения	64
Таблица формирования параметра режима защиты передачи данных	66
Таблица допустимых режимов защиты передачи данных	67
0xE8A8 Запись данных персонализации в карту	67
0xE8AA Персонализация карты	67
0xE872 Управление аутентификацией	68
0xE82D Управление аутентификацией с ключом в команде	68
0xE830 Чтение данных	69
0xE8A0 Запись данных	69
0xE8B0 Прибавление значения	70
0xE8B2 Вычитание значения	70
0xE8B4 Запись данных из буфера переноса в блок	70
0xE8B6 Прибавление значения с последующей записью данных из буфера переноса в блок	71
0xE8B8 Вычитание значения с последующей записью данных из буфера переноса в блок	71
0xE8C2 Запись данных блока значения в буфер переноса	71
0xE84B Завершающий запрос поддержки виртуальных карт	72
0xE8F2 Поиск релейной атаки	72
0xE8C8 Установка идентификатора считывателя	72
0xE882 Выбор карты ISO, внешняя аутентификация	72
0xE8A4 Выбор карты ISO	73
0xE87A Аутентификация переключения сектора	73
0xE840 Смена типа идентификатора карты Mifare Classic EV1	73
0xE844 Конфигурация карты в SL1	74
0xE860 Чтение версии карты	74
0xE83C Вычитать подпись карты	74
10 Обмен данными с картой Mifare DES Fire	74
Таблица константных значений	74
Формат команд для управления считывателем при помощи микропрограммы внешнего контроллера.	
11 Непосредственный обмен данными с картой	76
0x48 Непосредственный обмен с картой	76
12 Работа с SAM-модулем	76
0xB2 Смена рабочего SAM-модуля	76
0xB4 Получение номера рабочего SAM-модуля	79
0xB6 Выполнение команды в формате APDU	79
0xBA Выполнение сброса (рестарт) SAM	79
0xB6 Halt для карты ISO14443A через SAM	80
13 Работа с контактными картами стандарта ISO7816	80
0xB2 Смена рабочей контактной карты	80
0xB4 Получение номера рабочей контактной карты	80
0xB1 Запросить текущее состояние холдеров	80
0xB6 Выполнение команды в формате APDU	81
0xBA Выполнение сброса (рестарт) контактной карты	82
14 Работа SAM-модуля с Mifare Classic	82
0xAB Выполнение аутентификации карты Mifare Classic	82
0xAC Чтение блока карты Mifare Classic	82

0xAD Запись блока карты Mifare Classic	83
0xAE Работа с блоком Value карты Mifare Classic	83
0xAF Смена ключей карты Mifare Classic	83
15 Работа SAM-модуля с Mifare Ultralight C.....	84
0x2E Выполнение аутентификации карты Mifare Ultralight C	84
0x2F Смена ключей карты Mifare Ultralight C	84
16 Работа SAM-модуля с Mifare Plus.....	85
0xF8 Запись данных персонализации в карту	85
0xFA Персонализация карты	85
0xF0 Управление аутентификацией	85
0xF6 Чтение нескольких блоков SL2	86
0xF7 Запись нескольких блоков SL2	86
0xF4 Чтение данных	86
0xF5 Запись данных	87
0xFD Смена ключа	87
0xF1 Прибавление значения	88
0xF1 Вычитание значения	88
0xF1 Запись данных из буфера переноса в блок	88
0xF1 Прибавление значения с последующей записью данных из буфера переноса в блок ..	89
0xF1 Вычитание значения с последующей записью данных из буфера переноса в блок	89
0xF1 Запись данных блока значения в буфер переноса	90
0xF2 Запрос поддержки виртуальных карт	90
0xF3 Выбор виртуальной карты	90
0xF3 Снятие выбора виртуальной карты	91
0xF9 Поиск релейной атаки	91
17 Управление устройствами и метками стандарта ISO 18092.....	91
0x11 Активация устройства ISO18092	91
0x12 Отмена выбора ISO18092	92
0x13 Обмен ISO18092	92
0x40 Сброс протокола ISO18092	93
0x33 Запрос атрибутов ISO18092	93
0x34 Выбор параметров ISO18092	93
0x1F Проверка присутствия ISO18092	93
0x41 Установка конфигурации	94
0x42 Чтение конфигурации	94
0x28 Чтение серийного номера (NFCID3)	94
0x23 Чтение заданного количества байт из указанного адреса EEPROM	94
0x24 Запись одного байта данных в указанный адрес EEPROM	95
0x2A Запись нескольких байт данных в указанную страницу EEPROM	95
18 Работа с TDA8029.....	95
0x85 Непосредственный обмен с TDA	95
0x86 0xBA Сброс TDA	96

Часть IV Библиотека Clscrfl.dll

98

1 Управление интерфейсом.....	98
CLSCRF_Create	98
CLSCRF_Destroy	98
CLSCRF_Open	98
CLSCRF_OpenRS	99
CLSCRF_OpenUSB	99
CLSCRF_OpenETH	100
CLSCRF_IsOpened	100
CLSCRF_Close	100
CLSCRF_GetIOTimeout	101
CLSCRF_SetIOTimeout	101
CLSCRF_GetLastError	101

2 Управление считывателем.....	102
CLSCRF_GetState	102
CLSCRF_Mfrc_On	103
CLSCRF_Mfrc_Off	103
CLSCRF_Get_Mfrc_Version.....	103
CLSCRF_Get_Mfrc_Serial_Number.....	104
CLSCRF_Get_ReaderID	104
CLSCRF_Reader_Restart	104
CLSCRF_Mfrc_Rf_Off_On	104
CLSCRF_Mfrc_Set_Rf_Mode	105
CLSCRF_Sound	105
CLSCRF_Led	106
CLSCRF_Switches	106
CLSCRF_UART_Baudrate	107
CLSCRF_EraseFlash	108
CLSCRF_WriteFlashValue	108
CLSCRF_CheckFlashValueFilled	108
CLSCRF_DirectIO_Reader	109
CLSCRF_WriteReaderConfig	109
CLSCRF_ReadReaderConfig	110
CLSCRF_ResetReaderConfig	111
CLSCRF_ISO14443_ActivateEx	111
CLSCRF_ISO14443_4_Exchange	113
3 Конфигурация устройств на шине RS485.....	114
CLSCRF_GetIOAddress	114
CLSCRF_SetIOAddress	114
CLSCRF_ReadDeviceAddress	114
CLSCRF_WriteDeviceAddress	115
4 Управление защищенным режимом.....	115
CLSCRF_Crypto_SaveAESKeysToFile	115
CLSCRF_Crypto_GenerateAndSaveAESKeysToFile.....	115
CLSCRF_Crypto_LoadAESKeysFromFile	116
CLSCRF_Crypto_WriteFlash_AESKeys	116
CLSCRF_Crypto_AuthenticateReader.....	116
CLSCRF_Crypto_SetEncryptionMode	117
CLSCRF_Crypto_GetEncryptionMode	117
5 Работа со списком карт.....	117
CLSCRF_Cards_Add	117
CLSCRF_Cards_GetFirst	118
CLSCRF_Cards_GetLast	118
CLSCRF_Cards_Get	118
CLSCRF_Cards_GetCount	118
CLSCRF_Cards_Clear	118
CLSCRF_Cards_SetCurrent.....	119
CLSCRF_Cards_GetCurrent.....	119
CLSCRF_Cards_Search	119
CLSCRF_Cards_CreateNew_A.....	119
CLSCRF_Cards_CreateNew_B.....	120
CLSCRF_Cards_CreateNew15693.....	120
CLSCRF_Cards_Delete	121
CLSCRF_CARD	121
6 Управление картами типа А стандарта ISO 14443	122
CLSCRF_Activate_Idle_A	122
CLSCRF_Halt_A	123
CLSCRF_Activate_Wakeup_A	123
CLSCRF_ISO14443A_4_RATS	123

CLSCRF_ISO14443A_4_PPS	124
CLSCRF_ISO14443A_4_Activate	124
7 Управление картами типа B стандарта ISO 14443	125
CLSCRF_Activate_Idle_B	125
CLSCRF_Attrib_B	126
CLSCRF_Halt_B	126
CLSCRF_Activate_Wakeup_B	127
8 Управление метками стандарта ISO 15693	127
CLSCRF_FindAllTags_15693	127
CLSCRF_Inventory_15693	128
CLSCRF_Stay_Quiet_15693	129
CLSCRF_Select_15693	129
CLSCRF_ResetToReady_15693	130
9 Обмен данными с картами Mifare Classic	130
(обратная совместимость) CLSCRF_MifareStandard_HostCodeKey	130
(обратная совместимость) CLSCRF_MifareStandard_AuthKey	131
CLSCRF_MifareStandard_AuthKeyDirect	131
CLSCRF_MifareStandard_WriteKeyToE2	132
CLSCRF_MifareStandard_AuthE2	132
CLSCRF_MifareStandard_AuthE2Ex	132
CLSCRF_MifareStandard_Read	133
CLSCRF_MifareStandard_Write	133
CLSCRF_MifareStandard_Decrement	134
CLSCRF_MifareStandard_Increment	134
CLSCRF_MifareStandard_Restore	135
CLSCRF_MifareStandard_EV1_PersonalizeUid	135
CLSCRF_MifareStandard_EV1_SetLoadModulationType	135
CLSCRF_MifareStandard_CustomExchange	136
10 Обмен данными с картами Mifare Ultralight (C)	137
CLSCRF_MifareUltralight_Read	137
CLSCRF_MifareUltralight_Write	137
CLSCRF_MifareUltralightC_WriteKey	137
CLSCRF_MifareUltralightC_Authenticate	138
11 Обмен данными с картами Mifare Ultralight EV1	138
CLSCRF_MifareUltralightEV1_GetVersion	138
CLSCRF_MifareUltralightEV1_Read	138
CLSCRF_MifareUltralightEV1_ReadFast	139
CLSCRF_MifareUltralightEV1_Write	139
CLSCRF_MifareUltralightEV1_AuthenticatePassword	139
CLSCRF_MifareUltralightEV1_ReadCounter	140
CLSCRF_MifareUltralightEV1_IncrementCounter	140
CLSCRF_MifareUltralightEV1_ReadSignature	140
CLSCRF_MifareUltralightEV1_CheckTearingEvent	141
CLSCRF_MifareUltralightEV1_VirtualCardSupportLast	141
12 Обмен данными с метками стандарта ISO 15693	141
CLSCRF_ReadSingleBlock_15693	141
CLSCRF_WriteSingleBlock_15693	142
CLSCRF_LockBlock_15693	143
CLSCRF_ReadMultipleBlocks_15693	143
CLSCRF_WriteAFI_15693	144
CLSCRF_LockAFI_15693	144
CLSCRF_WriteDSFID_15693	145
CLSCRF_LockDSFID_15693	145
CLSCRF_GetSystemInfo_15693	146
CLSCRF_GetMultipleBSS_15693	147
CLSCRF_SetEAS_15693	147

CLSCRF_ResetEAS_15693	148
CLSCRF_LockEAS_15693	148
CLSCRF_EASAlarm_15693	149
CLSCRF_15693_ICode_GetRandomNumber	149
CLSCRF_15693_ICode_SetPassword	150
CLSCRF_15693_ICode_WritePassword	151
CLSCRF_15693_ICode_LockPassword	151
CLSCRF_15693_ICode_PasswordProtectEAS_AFI	152
CLSCRF_15693_ICode_64BitProtection	152
CLSCRF_15693_ICode_ProtectPage	153
CLSCRF_15693_ICode_LockPageProtectionCondition	153
CLSCRF_15693_ICode_Destroy	154
CLSCRF_15693_ICode_EnablePrivacy	155
13 Обмен данными с картами Mifare Plus	155
CLSCRF_MifarePlus_WritePersonExplicit	155
CLSCRF_MifarePlus_CommitPerson	156
CLSCRF_MifarePlus_Authenticate	156
CLSCRF_MifarePlus_AuthenticateDirect	156
CLSCRF_MifarePlus_MultiBlockRead	157
CLSCRF_MifarePlus_MultiBlockWrite	157
CLSCRF_MifarePlus_ReadData	158
CLSCRF_MifarePlus_WriteData	158
CLSCRF_MifarePlus_Increment	159
CLSCRF_MifarePlus_Decrement	159
CLSCRF_MifarePlus_Transfer	159
CLSCRF_MifarePlus_IncrementTransfer	160
CLSCRF_MifarePlus_DecrementTransfer	160
CLSCRF_MifarePlus_Restore	161
CLSCRF_MifarePlus_VirtualCardSupport	161
CLSCRF_MifarePlus_VirtualCardSupportLast	161
CLSCRF_MifarePlus_VirtualCardSelect	162
CLSCRF_MifarePlus_VirtualCardDeselect	163
CLSCRF_MifarePlus_ProximityCheck	163
14 Обмен данными с картами Mifare Plus EV1	163
CLSCRF_MifarePlusEV1_WritePerson	163
CLSCRF_MifarePlusEV1_CommitPerson	164
CLSCRF_MifarePlusEV1_Authenticate	164
CLSCRF_MifarePlusEV1_AuthenticateDirect	165
CLSCRF_MifarePlusEV1_ReadData	166
CLSCRF_MifarePlusEV1_WriteData	166
CLSCRF_MifarePlusEV1_Increment	167
CLSCRF_MifarePlusEV1_Decrement	167
CLSCRF_MifarePlusEV1_Transfer	167
CLSCRF_MifarePlusEV1_IncrementTransfer	168
CLSCRF_MifarePlusEV1_DecrementTransfer	169
CLSCRF_MifarePlusEV1_Restore	169
CLSCRF_MifarePlusEV1_VirtualCardSupportLastISOL3	170
CLSCRF_MifarePlusEV1_ProximityCheck	170
CLSCRF_MifarePlusEV1_CommitReaderID	170
CLSCRF_MifarePlusEV1_ISOSelectExternalAuthenticate	171
CLSCRF_MifarePlusEV1_ISOSelect	172
CLSCRF_MifarePlusEV1_AuthenticateSectorSwitch	172
CLSCRF_MifarePlusEV1_MF_PersonalizeUID	173
CLSCRF_MifarePlusEV1_SetConfigSL1	173
CLSCRF_MifarePlusEV1_GetVersion	173
CLSCRF_MifarePlusEV1_ReadSignature	174
15 Обмен данными с картами Mifare DES Fire (EV1)	174

CLSCRF_MifareDesFire_Authenticate.....	174
CLSCRF_MifareDesFire_SetTransferType.....	174
CLSCRF_MifareDesFire_ChangeKeySettings.....	175
CLSCRF_MifareDesFire_GetKeySettings.....	175
CLSCRF_MifareDesFire_ChangeKey.....	175
CLSCRF_MifareDesFire_GetKeyVersion.....	176
CLSCRF_MifareDesFire_CreateApplication.....	176
CLSCRF_MifareDesFire_DeleteApplication.....	177
CLSCRF_MifareDesFire_GetApplicationIDs.....	177
CLSCRF_MifareDesFire_GetDFNames.....	177
CLSCRF_MifareDesFire_SelectApplication.....	178
CLSCRF_MifareDesFire_FormatPICC.....	178
CLSCRF_MifareDesFire_GetVersion.....	178
CLSCRF_MifareDesFire_FreeMemory.....	178
CLSCRF_MifareDesFire_SetConfiguration.....	179
CLSCRF_MifareDesFire_GetCardUID.....	179
CLSCRF_MifareDesFire_GetFileIDs.....	179
CLSCRF_MifareDesFire_GetISOFileIDs.....	180
CLSCRF_MifareDesFire_GetFileSettings.....	180
CLSCRF_MifareDesFire_ChangeFileSettings.....	180
CLSCRF_MifareDesFire_CreateStdDataFile.....	181
CLSCRF_MifareDesFire_CreateBackupDataFile.....	181
CLSCRF_MifareDesFire_CreateValueFile.....	182
CLSCRF_MifareDesFire_CreateLinearRecordFile.....	182
CLSCRF_MifareDesFire_CreateCyclicRecordFile.....	183
CLSCRF_MifareDesFire_DeleteFile.....	183
CLSCRF_MifareDesFire_ReadData.....	184
CLSCRF_MifareDesFire_WriteData.....	184
CLSCRF_MifareDesFire_GetValue.....	184
CLSCRF_MifareDesFire_Credit.....	185
CLSCRF_MifareDesFire_Debit.....	185
CLSCRF_MifareDesFire_LimitedCredit.....	185
CLSCRF_MifareDesFire_WriteRecord.....	186
CLSCRF_MifareDesFire_ReadRecords.....	186
CLSCRF_MifareDesFire_ClearRecordFile.....	187
CLSCRF_MifareDesFire_CommitTransaction.....	187
CLSCRF_MifareDesFire_AbortTransaction.....	187
CLSCRF_DESFIRE_HW_SW_INFO.....	187
CLSCRF_DESFIRE_MORE_INFO.....	188
CLSCRF_DESFIRE_FILE_ACCESS_RIGHTS.....	188
CLSCRF_DESFIRE_FILE_SETTINGS.....	188
CLSCRF_DESFIRE_CONFIGURATION.....	189
CLSCRF_DESFIRE_DFNAME.....	190
CLSCRF_DESFIRE_LIMITED_CREDIT_ENABLED.....	191
CLSCRF_DESFIRE_PICC_MASTER_KEY_SETTINGS.....	191
CLSCRF_DESFIRE_APPLICATION_MASTER_KEY_SETTINGS.....	191
CLSCRF_DESFIRE_NEW_APPLICATION_KEY_SETTINGS.....	192
CLSCRF_DESFIRE_MASTER_KEY_SETTINGS.....	192
CLSCRF_DESFIRE_KEY_DATA.....	193
CLSCRF_DESFIRE_MASTER_KEY_SETTINGS_AND_LENGTH.....	193
16 Работа с SAM-модулем.....	194
CLSCRF_SAM_GetStatusDescription.....	194
CLSCRF_SAM_GetCurrentHolder.....	194
CLSCRF_SAM_SetCurrentHolder.....	195
CLSCRF_SAM_SetCurrentHolderGetATR.....	195
CLSCRF_SAM_SetCurrentHolderEx.....	195
CLSCRF_SAM_APDU.....	197
CLSCRF_SAM_Lock.....	198

CLSCRF_SAM_Unlock	199
CLSCRF_SAM_AuthenticateHost	199
CLSCRF_SAM_GetVersion	200
CLSCRF_SAM_SwitchToAV2Mode	200
CLSCRF_SAM_Reset	201
CLSCRF_SAM_KillAuth	201
CLSCRF_SAM_Halt_A	202
17 Работа с контактными картами стандарта ISO7816.....	202
CLSCRF_ISO7816_GetCurrentHolder	202
CLSCRF_ISO7816_SetCurrentHolder	202
CLSCRF_ISO7816_SetCurrentHolderGetATR	202
CLSCRF_ISO7816_SetCurrentHolderEx	203
CLSCRF_ISO7816_HOLDERS_INFO	203
CLSCRF_ISO7816_GetHoldersInfo	204
CLSCRF_ISO7816_APDU	204
CLSCRF_ISO7816_Reset	204
18 Работа с ключами SAM-модуля.....	205
CLSCRF_SAM_GetKeyEntry	205
CLSCRF_SAM_ChangeKeyEntry	206
SAM_SETTINGS	207
SAM_EXT_SETTINGS	208
SAM_NV_PROGRAMMING_MASK	209
SAM_VERSION	209
19 Работа с картами Mifare Classic при помощи SAM-модуля.....	211
CLSCRF_SAM_MifareAuthenticate	211
CLSCRF_SAM_MifareRead	212
CLSCRF_SAM_MifareWrite	212
CLSCRF_SAM_MifareIncrement	212
CLSCRF_SAM_MifareDecrement	213
CLSCRF_SAM_MifareRestore	213
CLSCRF_SAM_MifareChangeKey	214
20 Работа с картами Mifare Ultralight C при помощи SAM-модуля.....	215
CLSCRF_SAM_MifareUltralightC_Authenticate	215
CLSCRF_SAM_MifareUltralightC_WriteKey	215
21 Работа с картами Mifare Plus при помощи SAM-модуля.....	216
CLSCRF_SAM_MifarePlus_WritePerso	216
CLSCRF_SAM_MifarePlus_CommitPerso	216
CLSCRF_SAM_MifarePlus_Authenticate	217
CLSCRF_SAM_MifarePlus_MultiBlockRead	218
CLSCRF_SAM_MifarePlus_MultiBlockWrite	218
CLSCRF_SAM_MifarePlus_ReadData	219
CLSCRF_SAM_MifarePlus_WriteData	219
CLSCRF_SAM_MifarePlus_ChangeKey	220
CLSCRF_SAM_MifarePlus_Increment	221
CLSCRF_SAM_MifarePlus_Decrement	221
CLSCRF_SAM_MifarePlus_Transfer	222
CLSCRF_SAM_MifarePlus_IncrementTransfer	222
CLSCRF_SAM_MifarePlus_DecrementTransfer	223
CLSCRF_SAM_MifarePlus_Restore	224
CLSCRF_SAM_MifarePlus_VirtualCardSupport	224
CLSCRF_SAM_MifarePlus_VirtualCardSelect	225
CLSCRF_SAM_MifarePlus_VirtualCardDeselect	226
CLSCRF_SAM_MifarePlus_ProximityCheck	226
22 Работа со считывателем NFC663	227
CLSCRF_NFC663_ActivateCard	227
CLSCRF_NFC663_Deselect	228

CLSCRF_NFC663_Exchange	228
CLSCRF_NFC663_ResetProtocol	229
CLSCRF_NFC663_AttributeRequest	229
CLSCRF_NFC663_ParameterSelect	230
CLSCRF_NFC663_PresenceCheck	230
CLSCRF_NFC663_SetConfig	230
CLSCRF_NFC663_GetConfig	231
CLSCRF_NFC663_GetSerialNo	231
CLSCRF_NFC663_E2_Read	231
CLSCRF_NFC663_E2_Write	232
CLSCRF_NFC663_E2_WritePage	232
PHPAL_I18092MPI_DATARATE_106	232
PHPAL_I18092MPI_DATARATE_212	232
PHPAL_I18092MPI_DATARATE_424	233
PHPAL_I18092MPI_FRAME_SIZE_64	233
PHPAL_I18092MPI_FRAME_SIZE_128	233
PHPAL_I18092MPI_FRAME_SIZE_192	233
PHPAL_I18092MPI_FRAME_SIZE_254	233
PHPAL_I18092MPI_DESELECT_DSL	233
PHPAL_I18092MPI_DESELECT_RLS	233
PH_EXCHANGE_DEFAULT	234
PH_EXCHANGE_TXCHAINING	234
PH_EXCHANGE_RXCHAINING	234
PH_EXCHANGE_RXCHAINING_BUFSIZE	234
PH_EXCHANGE_TX_CRC	234
PH_EXCHANGE_RX_CRC	235
PH_EXCHANGE_PARTY	235
PH_EXCHANGE_LEAVE_BUFFER_BIT	235
PH_EXCHANGE_BUFFERED_BIT	235
PHPAL_I18092MPI_CONFIG_PACKETNO	235
PHPAL_I18092MPI_CONFIG_DID	235
PHPAL_I18092MPI_CONFIG_NAD	235
PHPAL_I18092MPI_CONFIG_WT	236
PHPAL_I18092MPI_CONFIG_FSL	236
PHPAL_I18092MPI_CONFIG_MAXRETRYCOUNT	236
23 Демонстрация работы со стандартом NFC	236
Функции работы с NDEF сообщениями	236
CLSCRF_NFC_NDEF_Message_Create	236
CLSCRF_NFC_NDEF_Message_GetRecordsCount	237
CLSCRF_NFC_NDEF_Message_GetDataSize	237
CLSCRF_NFC_NDEF_Message_GetData	237
CLSCRF_NFC_NDEF_Message_AddRecord	238
CLSCRF_NFC_NDEF_Message_AddRecordEmpty	238
CLSCRF_NFC_NDEF_Message_AddRecordText	238
CLSCRF_NFC_NDEF_Message_AddRecordUri	238
CLSCRF_NFC_NDEF_Message_AddRecordMimeMedia	239
CLSCRF_NFC_NDEF_Message_AddRecordMimeMediaText	239
CLSCRF_NFC_NDEF_Message_GetRecord	240
CLSCRF_NFC_NDEF_Message_Destroy	240
Функции работы с NDEF записями	240
CLSCRF_NFC_NDEF_Record_Create	240
CLSCRF_NFC_NDEF_Record_GetDataSize	240
CLSCRF_NFC_NDEF_Record_GetTrf	241
CLSCRF_NFC_NDEF_Record_SetTrf	241
CLSCRF_NFC_NDEF_Record_GetTypeLength	241
CLSCRF_NFC_NDEF_Record_GetType	241
CLSCRF_NFC_NDEF_Record_GetTypeStr	242
CLSCRF_NFC_NDEF_Record_SetType	242

CLSCRF_NFC_NDEF_Record_SetTypeStr	242
CLSCRF_NFC_NDEF_Record_GetPayloadLength	243
CLSCRF_NFC_NDEF_Record_GetPayload	243
CLSCRF_NFC_NDEF_Record_GetPayloadText	243
CLSCRF_NFC_NDEF_Record_GetPayloadUri	244
CLSCRF_NFC_NDEF_Record_SetPayload	244
CLSCRF_NFC_NDEF_Record_GetIdLength	244
CLSCRF_NFC_NDEF_Record_GetId	244
CLSCRF_NFC_NDEF_Record_GetIdStr	245
CLSCRF_NFC_NDEF_Record_SetId	245
CLSCRF_NFC_NDEF_Record_Destroy	245
Функции работы с протоколом LLCP	246
CLSCRF_NFC_LLCP_Create	246
CLSCRF_NFC_LLCP_SetTimeout	246
CLSCRF_NFC_LLCP_SetBaudrate	246
CLSCRF_NFC_LLCP_SetGeneralInformation	246
CLSCRF_NFC_LLCP_SetNFCD3Information	247
CLSCRF_NFC_LLCP_ServerOpen	247
CLSCRF_NFC_LLCP_ServerReceive	247
CLSCRF_NFC_LLCP_ServerClose	248
CLSCRF_NFC_LLCP_ClientOpen	248
CLSCRF_NFC_LLCP_ClientTransmit	248
CLSCRF_NFC_LLCP_ClientClose	248
CLSCRF_NFC_LLCP_Destroy	248
Функции работы с протоколом SNEP	249
CLSCRF_NFC_SNEP_Create	249
CLSCRF_NFC_SNEP_Receive	249
CLSCRF_NFC_SNEP_Transmit	249
CLSCRF_NFC_SNEP_Destroy	250
Функции работы с NFC метками	250
CLSCRF_NFC_ForumTags_SupposeTypeISO14443A	250
CLSCRF_NFC_ForumTags_BeginType4	250
CLSCRF_NFC_ForumTags_Format	251
CLSCRF_NFC_ForumTags_Read	251
CLSCRF_NFC_ForumTags_Write	251
Примеры работы с демонстрационными функциями NFC	252
Приём сообщения NDEF с текстовой записью	255
Чтение сообщения из меток типа NFC Forum Tag 2, 4	255
Получение сообщения от удалённого устройства в режиме P2P	255
Разбор принятого сообщения	256
Отправка сообщения NDEF с текстовой записью	258
Формирование отправляемого сообщения	258
Запись сообщения в метки типа NFC Forum Tag 2, 4	259
Отправка сообщения на удалённое устройство в режиме P2P	259
24 Непосредственный обмен данными с картой	260
CLSCRF_DirectIO_Card	261
25 Работа с TDA8029	261
CLSCRF_TDA8029_Exchange	261
CLSCRF_TDA8029_APDU	262
CLSCRF_TDA8029_Reset	262
CLSCRF_TDA8029_GetStatus	263
CLSCRF_TDA8029_GetATR	263
26 Обработка ошибок при использовании библиотеки Clscrfl.dll	263

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

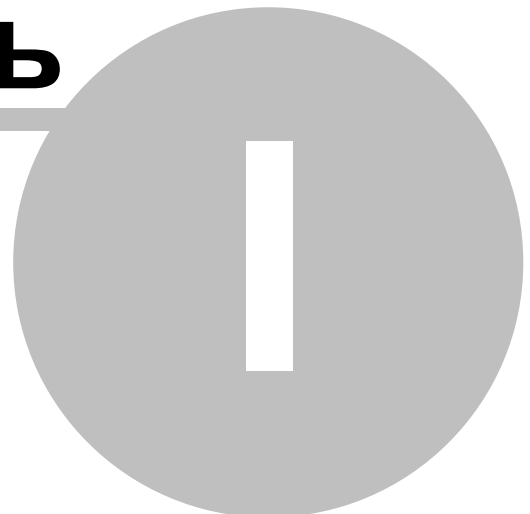
266

1 Активация карты типа А из состояния IDLE	266
Переключить режим Rf на тип А скорость 106.	266
Активировать карту типа А из состояния IDLE	266
Запросить параметры протокола карты.	266
Установить текущие параметры протокола обмена с картой типа А.	267
Переключить режим Rf на тип А скорость 106.	267
2 Активация карты типа А из состояния HALT	267
Переключить режим Rf на тип А скорость 106.	268
Активировать карту типа А из состояния HALT.	268
Запросить параметры протокола карты.	268
Установить текущие параметры протокола обмена с картой типа А.	269
Переключить режим Rf на тип А скорость 424.	269
3 Активация карты типа В из состояния IDLE	269
Переключить режим Rf на тип В скорость 106.	269
Активировать карту типа В из состояния IDLE	269
Установить текущие параметры протокола обмена с картой типа В.	270
Переключить режим Rf на тип В скорость 106.	270
4 Активация карты типа В из состояния HALT	270
Переключить режим Rf на тип В скорость 106.	270
Активировать карту типа В из состояния HALT.	270
Установить текущие параметры протокола обмена с картой типа В.	271
Переключить режим Rf на тип В скорость 212.	271
5 Деактивация карты	271
Деактивировать карту, работающую в протоколе T=CL.	271

МикроЭМ

Руководство программиста

Часть



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
ISO 18092

1.2 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 управления логическими соединениями между устройствами, работающими по протоколу ISO18092 (NFC P2P)

SNEP (Simple NDEF Exchange Protocol) – протокол обмена сообщениями NFC в формате NDEF

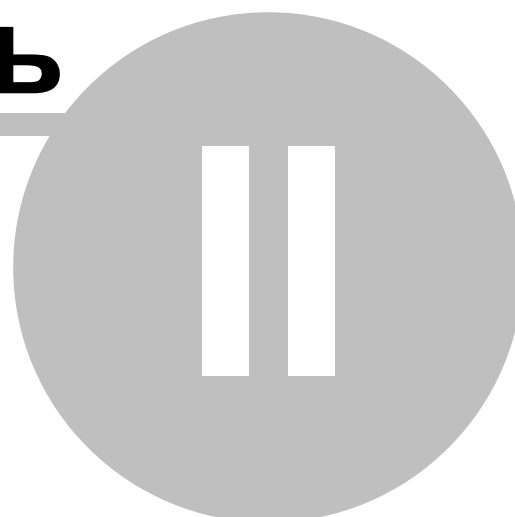
1.3 Предупреждение

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



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.

Для считывателей UEM доступны два вида драйверов: стандартные USB и PC/SC.

Чтобы установить любой из видов USB-драйверов считывателя для операционной системы Windows, необходимо выполнить следующее.

1. Скачайте с сайта microem.ru и распакуйте архив с программным обеспечением, 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 с подходящим драйвером из родительской папки Driver в поставляемом со считывателем комплекте ПО. Нажмите "ОК".

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.

В комплекте ПО также доступны исходные коды драйвера под Linux, находящиеся в папке Drivers/Usb/Linux.

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, считыватель поддерживает работу с шифрованным каналом обмена.

Для интерфейсов RS232 и RS485 пользователь должен обеспечить интервал между соседними кадрами размером не менее 30 битов спокойного состояния линии.

Для интерфейсов RS232 и RS485 важную роль играет признак завершения кадра, который определяется как промежуток спокойного состояния шины, измеряемый в байтах (10 бит = старт + 8 бит + стоп). По умолчанию это значение равно 3.

В микросекундах этот признак будет иметь разную продолжительность в зависимости от скорости обмена данными (для скорости 9600 бод это 3125 мкс).

Каждое устройство на шине перед началом передачи данных должно убедиться в наличии признака завершения предыдущего кадра и, кроме того, обязано не допускать внутри своего кадра промежутков спокойного состояния шины более одного байта.

Если ошибки обмена с хостом обусловлены присутствием на шине устройства, не соблюдающего последнее условие, признак завершения кадра можно увеличить с помощью команды 0x68. Это несколько замедлит обмен данными, но исключит подобные ошибки.

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*n	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:

$$\text{EncData}[16*n] = \text{EncAES128CBC}(\{\text{OpCode}[1] \parallel \text{Data}[XX] \parallel \text{CRC32}(\{\text{OpCode}[1] \parallel \text{Data}[XX]\}), 0x80, \{0x00, 0x00, \dots\}, \text{SessionKey}[16], \text{IV}[16])$$

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, байт	Значение
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:
 $\{OpCode[1] \parallel Data[XX] \parallel CRC32(\{OpCode[1] \parallel Data[XX]\}), 0x80, \{0x00, 0x00, \dots\}\}$
 $= DecAES128CBC(EncData[16*m]), SessionKey[16], IV[16])$

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 таймута 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 пуганию.

2.2.5 Коды завершения команды

Все ошибки, связанные с интерфейсом, возникают на беспроводном участке передачи данных между картой и считывателем. Исключение составляет ошибка 0xFE, которая может возникать также и на проводном интерфейсе (RS232 или RS485 в случае если адрес устройства на шине - 0x00).

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_PROTOCOLERR	(-7)	0xF9	Ошибка в протоколе

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_BITCOUNTERERR	(-11)	0xF5	Неверное количество принятых битов
MI_BYTECOUNTERERR	(-12)	0xF4	Неверное количество принятых байтов
MI_WRITEERR	(-15)	0xF1	Ошибка записи данных
MI_INCRERR	(-16)	0xF0	Ошибка инкремента
MI_DECRERR	(-17)	0xEF	Ошибка декремента
MI_READERR	(-18)	0xEE	Ошибка чтения 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_CRYPTOMODE	(-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 В не перешла в режим HALT
UEM_FLASH_OPERATING_ERROR	(-50)	0xCE	Ошибка работы флеш-памяти
MI_INTERNAL_CALL_ERR	(-51)	0xCD	Ошибка внутреннего вызова
MI_CASCLEVEEX	(-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
MI_MIFARE_ERR_NAK_(F...0)	(-65 ... - 80)	(0xBF ... 0xB0)	Внутренняя ошибка Mifare Classic NAK (F...0)
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_EVI_FUNC_NOT_SUPPORTED	(-83)	0xAD	Функция EV1 не поддерживается
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_MFP_EV1_TMAC_ERR	(-91)	0xA5	Ошибка с TMAC
MI_NY_IMPLEMENTED	(-100)	0x9C	Неизвестная ошибка
MI_INVALID_CRC16	(-101)	0x9B	Неверный код CRC16
MI_RECBUF_OVERFLOW	(-112)	0x90	Переполнение буфера приемника
INTERNAL_RD_LIB_ERR	(-123)	0x85	Внутренняя ошибка библиотеки считывателя
MI_VALERR	(-124)	0x84	Неверный формат блока VALUE
UEM_UNSUPPORTED_PARAMETER	(-125)	0x83	Параметр не поддерживается
UEM_CHAINING_NOT_COMPLETE	(-126)	0x82	Цепочка приема не завершена
UEM_TEMPERATURE_ERROR	(-127)	0x81	Перегрев микросхемы считывателя
UEM_UNKNOWN_ERROR	(-128)	0x80	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 );
}
```


}

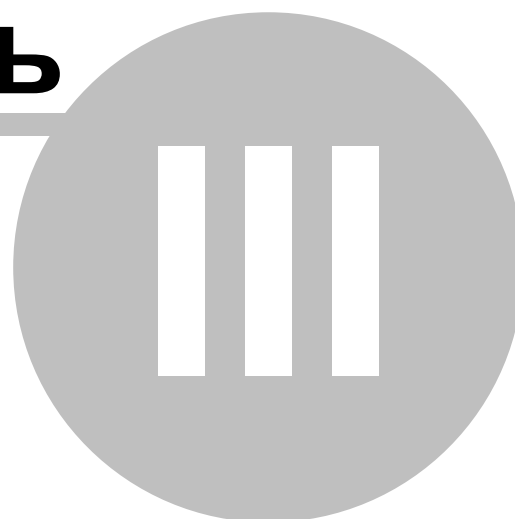
2.2.8 Многобайтовые переменные

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 [здесь](#) (п.2.2.5).

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

3.1 Управление 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,

биты 11-7 - резерв,

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

*** - резерв для будущего использования

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 Выключение 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]; Version[6]

Version[6] – версия считывателя.

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 0x58 0x58 Выдача уникального идентификатора считывателя

0x58 0x58 [CLSCRF Get ReaderID](#)

OUT: ACK[1]; ReaderID[16]

ReaderID[16] – уникальный идентификатор считывателя.

3.1.7 0x58 0x77 Перезагрузка считывателя

0x58 0x77 CLSCRF Reader Restart

IN: ReaderID[16]

OUT: ACK[1]

ReaderID[16] – уникальный идентификатор считывателя.

Пример: 58 77 22 C0 02 0C C9 80 35 AF 29 7F 27 5B C1 1B 00 F5

Ответ: C4 или нет ответа

3.1.8 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.

3.1.9 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

*** - 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

3.1.10 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.

3.1.11 0x07 Управление светодиодом

0x07 [CLSCRF Led](#)

IN: BlinkColor[1]; BlinkCount[1]; PostColor[1]

OUT: ACK[1]

Структура байтов **BlinkColor** и **PostColor**:

- Бит 0 - красный;
- Бит 1 - зеленый;
- Бит 2 - синий;
- Биты 3...7 - не используются.

BlinkColor[1] – цвет мигания:

- 0 – не мигать;
- 1 – мигать красным цветом;
- 2 – мигать зеленым цветом;
- 3 – мигать желто-оранжевым цветом;
- 4 – мигать синим цветом(при наличии);
- 5 – мигать фиолетовым цветом(при наличии);
- 6 – мигать бирюзовым цветом(при наличии);
- 7 – мигать белым цветом(при наличии);

(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;
- 4 – зажечь синим цветом(при наличии);
- 5 – зажечь фиолетовым цветом(при наличии);
- 6 – зажечь бирюзовым цветом(при наличии);
- 7 – зажечь белым цветом(при наличии).

3.1.12 0x0F Изменение скорости обмена по COM-порту

0x0F [CLSCRF UART Baudrate](#)

IN: divisor[1]

OUT: ACK[1]

divisor[1] = целая часть от $(1500000 / \text{BaudRate})$,

где $\text{BaudRate} = \{9600, 14400, 19200, 38400, 57600, 115200\}$.

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

3.1.13 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 ($\{0x21, 0x43, 0x65, 0x87\}$)

Время выполнения команды - 250мс.

3.1.14 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] – записываемое значение ключа

3.1.15 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.

3.1.16 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:

$DataA \parallel DataB$ – 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 шифрованном обмене.

Подготовка шифрованного обмена

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

SessionKey[16] = $RndA[0..3] \parallel RndB[0..3] \parallel RndA[12..15] \parallel RndB[12..15]$

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.

3.1.17 0x68 Работа с конфигурацией считывателя

0x68 [CLSCRF_ReadReaderConfig](#) [CLSCRF_WriteReaderConfig](#)
[CLSCRF_ResetReaderConfig](#)

IN: Cmd[1]; Address[1]; Length[1]; (Data[Length])

OUT: ACK[1]; (Data[Length])

Cmd[1]: 0x00 - Чтение конфигурации из RAM

0x01 - Чтение конфигурации из Flash с записью в RAM

0x10 - Запись конфигурации в RAM

0x11 - Запись конфигурации во Flash и RAM

0x80 - Сброс конфигурации в RAM к заводским настройкам

0x81 - Сброс конфигурации во Flash и в RAM к заводским

настройкам

Address[1] – адрес начального байта конфигурации {0..7};

Length[1] – длина записываемой/читаемой последовательности байт конфигурации {1..8};

Data[Length] – записываемые/вычитанные байты конфигурации.

Address:

Адрес	Длина	Название
0x00	1 байт	Rc663_14443A_mks
0x01	1 байт	Rc663_14443B_mks
0x02	1 байт	Rc663_15693_mks
0x03	1 байт	RFU (Rc663_18000P3M3_mks)
0x04	1 байт	RFU (Rc663_18092mPI_mks)
0x05	1 байт	Char_Timeout_bytes

Параметры по адресам (Offset) 0x00..0x02 предназначены для настройки качества обмена данными с картами соответственно стандартов 14443-A, 14443-B, 15693.

По этим адресам расположены байты конфигурации, определяющие внутреннюю задержку считывателя при работе с картами соответствующих стандартов в микросекундах. По умолчанию они имеют значение 0xFF. Если оно задано, то считыватель устанавливает значение задержки автоматически, на основе определенного по различным признакам типа карты. Если значение задержки в конфигурации находится в диапазоне 0x01..0xFE (0x00 - резерв), то автоматическое определение отключается, и это значение будет использовано принудительно для всех операций с картой соответствующего типа. В случае, если карта считывается нестабильно, то, возможно, ручная регулировка данного параметра позволит уменьшить или убрать нестабильность.

Параметры по адресам 0x03..0x04 зарезервированы для использования в будущем.

Параметр по адресу 0x05 предназначен для настройки признака завершения кадра при обмене с хостом по интерфейсам RS232 и RS485.

Значение 0xFF для каждого параметра означает, что пользователь не менял этот параметр, а используется значение по умолчанию.

На стабильность обмена с картами влияет множество факторов, включая наличие металлических предметов, поверхностей вблизи считывателя, проводов, электромагнитных полей, качество источника питания. Кроме того, сами карты могут иметь дефекты. На предельно малых и больших расстояниях до считывателя карты могут читаться, либо записываться нестабильно. Хотя регулировка параметров внутренней задержки и может немного повлиять на качество обмена, однако первостепенным является влияние перечисленных выше факторов.

Признак завершения кадра (параметр 5) - это промежуток спокойного состояния шины, измеряемый в байтах (см. описание протокола).

По умолчанию это значение равно 3. Возможные значения для этого параметра лежат в пределах от 4 до 254.

3.1.18 0x75 Комплексная активация карт ISO 14443-A и ISO 14443-B из состояния IDLE

0x75 CLSCRF ISO14443 ActivateEx

IN: Type_Baudrate[1], RfReset[1], DisableTCL_CID[1]; [если Type_Baudrate = 0x1X, то дополнительно: Afi[1], Param[1]];

OUT: ACK[1]; [если Type_Baudrate = 0x0X, то: Atq[2], Sak[1], UIDlen[1], UID[UIDlen], Ats[Ats[0]]]; [если Type_Baudrate = 0x1X, то: Mbli[1]; AtqbLen[1]; Atqb[AtqbLen]];

Type_Baudrate[1] = Type (бит 4) + Baudrate (биты 0..3):

- Type - тип карты: 0 - ISO14443A; 1 - ISO14443B;
- Baudrate – значение устанавливаемой скорости обмена данными с

картой:

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;

RfReset[1] - длительность [ms] отключения поля перед включением (0..127) - биты 7..4 + дополнительная пауза [ms] после включения поля (0..127) - биты 3..0

DisableTCL_CID[1] = DisableTCL (бит 7) + CID (биты 0..3):

- DisableTCL - отключение автоматического перехода в T=CL (0 - не отключать, 1 - отключить);
- CID - логический идентификатор карты для обмена по протоколу T=CL – любое число в диапазоне 0..14 (см. ISO 14443-3 7.10.6);

Afi[1] - Application Family Identifier (для ISO14443B);

Param[1] - дополнительные параметры (для ISO14443B):

- биты 7..5 = 0;
- бит 4 = 1 - считыватель требует использовать расширенный ATQB;
- бит 3 = 0;

- биты 2..0 - количество временных слотов;
Atq[2] - ATQ карты типа ISO14443A;
Sak[1] - SAK карты типа ISO14443A;
UIDlen[1] – длина уникального номера карты (может быть 4, 7 или 10);
UID[UIDlen] – уникальный номер карты;
ATS[ATS[0]] – структура Answer To Select (для карт типа ISO14443A, см. ISO 14443-4 5.1, 5.2), первый байт которой содержит ее длину;
Mbli[1] - идентификатор размера буфера (для ISO14443B);
AtqbLen[1] - размер считанного ATQ карты (для ISO14443B);
Atqb[AtqbLen] - ATQ карты (для ISO14443B);

3.1.19 0x47 Обмен в режиме T=CL

0x47 CLSCRF ISO14443 4 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;

3.1.20 0x37 Управление выходными линиями

0x37 CLSCRF_Switches

IN: Control[1];

OUT: ACK[1];

На плате расположены три контакта с надписями:

- "Out-" - может быть подключен к земле считывателя;
- "Out+" - может быть подключен к +5В;
- "Т перевернутое" - земля считывателя.

При подаче питания (по умолчанию) Out- и Out+ не подключены.

Control[1] – управление контактами Out- и Out+

- 0x1X - управление контактом Out-
- 0x2X - управление контактом Out+
- 0xX1 - Out- подключить к земле считывателя
- 0xX2 - Out+ подключить к +5В

Примеры

Одновременное управление контактами Out- и Out+:

- 37 33 => Out- подключить к земле считывателя, Out+ подключить к +5В
- 37 32 => Out- отключить, Out+ подключить к +5В
- 37 31 => Out- подключить к земле считывателя, Out+ отключить
- 37 30 => отключить Out- и Out+ (состояние по умолчанию)

Индивидуальное управление Out+ (при этом Out- остается в прежнем состоянии):

- 37 22 => Out+ подключить к +5В
- 37 20 => отключить Out+

Индивидуальное управление Out- (при этом Out+ остается в прежнем состоянии):

- 37 11 => Out- подключить к земле считывателя
- 37 10 => отключить Out-

Примечание

При питании от USB ограничение на общее потребление тока считывателем 300 мА.

На контакт Out+ при полной нагрузке считывателя остается не более 150 мА.

3.1.21 Конфигурация устройств на шине RS485

3.1.21.1 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.

3.1.21.2 0x77 Запись адреса устройства

0x77 [CLSCRF_WriteDeviceAddress](#)

IN: Addr[1]

OUT: ACK[1]

Addr[1] – новый адрес устройства.

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.

3.2 Управление 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 Вход в режим T=CL и 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 по протоколу T=CL

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

3.2.6 0x2B Активировать режим T=CL: получение доступных скоростей и установка скорости работы с картой

0x2B [CLSCRF ISO14443A 4 Activate](#)

IN: CID[1] Baudrate[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);

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

DataLength[2] – количество принятых от карты байт данных (ATS);

ATS[DataLength] – структура Answer To Select (см. ISO 14443-4 5.1, 5.2)

3.3 Управление 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 AttrB B](#)

IN: ParamLen[1]; Params[ParamLen]

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

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

3.3.3 0x55 Перевод активной карты типа В 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 В, 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 Управление 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 Обмен данными с картой Mifare Classic

3.5.1 Вычисление абсолютного номера блока

Секторы за номерами 0..31 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. $\text{<абсолютный номер блока>} = \text{<номер сектора>} * 4 + \text{<номер блока в секторе>}$
2. Если номер сектора > 32 , то прибавляем $(\text{<номер сектора>} - 32) * 12$.

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]; key_num[1]; AbsBlockNo[1]
OUT: ACK[1]
key_type[1] – тип ключа:
0x60 - Key A,
0x61 - Key B;
snr[4] – UID карты;
key_num[1] – номер ключа в EEPROM считывателя;
AbsBlockNo[1] – абсолютный номер блока: $4 * \text{Sector} + \text{Block}$;
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.

3.5.12 0x46 Настраиваемый обмен с картой

0x46 [CLSCRF MifareStandard CustomExchange](#)

IN: Timeout[2]; Options[2]; TxLastBitsCount[1]; TxLength[2]; TxBuffer[TxLength]

OUT: ACK[1]; RxLastBitsCount[1]; RxLength[2]; RxBuffer[RxLength]

Timeout[2] – таймаут операции, единица измерения = 128 / 13.56 МГц = около 9.439528 мкс;

Options[2] – настройки обмена данными с картой:

биты 0-3 – установить в 0;

бит 4 – флаг добавления контрольной суммы к запросу на карту (высчитывается автоматически);

бит 5 – флаг ожидания контрольной суммы в ответе от карты (высчитывается автоматически);

бит 6 – флаг использования бита четности при обмене (высчитывается автоматически);

биты 7, 8-15 – установить в 0.

TxLastBitsCount[1] – количество значащих бит в завершающем байте запроса (0..7, 0 - все биты значащие);

TxLength[2] – длина запроса на карту в байтах;

TxBuffer[TxLength] – данные для отправки на карту;

RxLastBitsCount[1] – количество значащих бит в завершающем байте ответа (0..7, 0 - все биты значащие);

RxLength[2] – длина ответа от карты в байтах;

RxBuffer[RxLength] – принятые от карты данные;

3.6 Обмен 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 Обмен данными с картой Mifare UltraLight EV1

3.7.1 0xC460 Чтение версии карты

0xC460 [CLSCRF MifareUltralightEV1 GetVersion](#)

IN:

OUT: ACK[1]; VerInfo[8]

VerInfo[8] – данные версии карты.

Пример: C4 60

Ответ: 00 00 04 03 01 01 00 0E 03 (128 bytes)

Ответ: 00 00 04 03 03 01 00 0B 03 (48 bytes)

Ответ: 00 00 34 21 01 01 00 0E 03 (Mikron)

3.7.2 0xC430 Чтение четырёх страниц

0xC430 [CLSCRF MifareUltralightEV1 Read](#)

IN: Addr[1]

OUT: ACK[1]; Data[16]

Addr[1] – номер страницы;

Data[16] – прочитанные данные.

Пример: C4 30 00

Ответ: 00 04 C8 10 54 EA FC 38 80 AE 48 00 00 00 00 00 00 (clean)

Ответ: 00 04 BD B6 87 32 AE 41 80 5D 48 F0 00 FF FF FF FC (in use)

3.7.3 0xC43A Чтение диапазона страниц

0xC43A [CLSCRF MifareUltralightEV1 ReadFast](#)

IN: PageStart[1]; PageEnd[1]

OUT: ACK[1]; Size[2]; Data[Size]

PageStart[1] – номер начальной страницы;

PageEnd[1] – номер конечной страницы;

Size[2] – количество прочитанных байт;

Data[Size] – прочитанные данные.

Пример: C4 3A 02 06

Ответ: 00 14 00 5D 48 F0 00 FF FF FF FC 45 D9 A0 15 D3 4C ED 00 22 E0
00 00

3.7.4 0xC4A2 Запись страницы

0xC4A2 [CLSCRF MifareUltralightEV1 Write](#)

IN: Addr[1]; Data[4]

OUT: ACK[1]

Addr[1] – номер страницы;

Data[4] – записываемые данные.

Пример: C4 A2 04 A0A1A2A3

Ответ: 00

3.7.5 0xC41B Верификация пароля

0xC41B [CLSCRF MifareUltralightEV1 AuthenticatePassword](#)

IN: Password[4]

OUT: ACK[1]; PasswordAcknowledge[2]

Password[4] - пароль.

PasswordAcknowledge[2] - подтверждение пароля.

Пример: C4 1B FFFFFFFF

Ответ: 00 00 00

3.7.6 0xC439 Прочитать значение счетчика

0xC439 [CLSCRF MifareUltralightEV1 ReadCounter](#)

IN: CounterNumber[1]

OUT: ACK[1]; Value[3]

CounterNumber[1] – номер счетчика: 0..2;

Value[3] – значение счетчика LSB вперед.

Пример: C4 39 02

Ответ: 00 01 00 00

3.7.7 0xC4A5 Увеличить значение счетчика

0xC4A5 [CLSCRF MifareUltralightEV1 IncrementCounter](#)

IN: CounterNumber[1]; Value[3]

OUT: ACK[1]

CounterNumber[1] – номер счетчика: 0..2;

Value[3] – прибавляемое значение счетчика LSB вперед.

Пример: C4 A5 02 010000

Отв: 00

3.7.8 0xC43C Прочитать подпись ECC

0xC43C [CLSCRF MifareUltralightEV1 ReadSignature](#)

IN: Addr[1]

OUT: ACK[1]; Signature[32]

Addr[1] – зарезервировано, должно быть 0x00;

Signature[32] – прочитанная подпись.

Пример: C4 3C 00

Ответ: 00 1A CB 7F 6B 83 B0 9C 9A 6A 36 81 82 34 C0 84 8F F4 63 60 FD
0D 66 DA 65 66 B3 E0 C6 31 AC A8 F6

Ответ: 00 8B 34 F9 F2 3B 55 6F D9 AC 7A 6B FF 83 AD 1B A0 6E 17 B2
D0 7E 95 A5 D9 2C E1 5E D0 E6 BB 54 84

3.7.9 0xC43E Check if a tearing event

0xC43E [CLSCRF MifareUltralightEV1 CheckTearingEvent](#)

IN: CounterNumber[1]

OUT: ACK[1]; Data[1]

CounterNumber[1] – номер счетчика: 0..2;

Data[1] – байт флагов.

Пример: C4 3E 02

Ответ: 00 BD

3.7.10 0xC44B Заключительный запрос поддержки виртуальных карт

0xC44B [CLSCRF](#) [MifareUltralightEV1](#) [VirtualCardSupportLast](#)

IN: IID[16]; PCDCAPS[4]

OUT: ACK[1]; VirtualCardTypeIdentifier[1]

IID[16] – идентификатор инфраструктуры;

PCDCAPS[4] – характеристики считывателя;

VirtualCardTypeIdentifier[1] – идентификатор типа виртуальной карты.

Пример: C4 4B A0A1A2A3A4A5A6A7A8A9AAABACADAEAF 00112233

Ответ: 00 05

3.8 Обмен данными с картой Mifare Plus

3.8.1 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

3.8.2 Таблица формирования параметра режима защиты передачи данных

MAC в команде	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

3.8.3 Таблица допустимых режимов защиты передачи данных

Операция	MAC в команде	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.

3.8.4 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.

3.8.5 0xAA Персонализация карты

0xAA [CLSCRF MifarePlus CommitPerso](#)

IN: 0x21; 0x43; 0x65; 0x87;

OUT: ACK[1];

Пример: aa 21 43 65 87

Ответ: 00

3.8.6 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).

3.8.7 0xE92D Управление аутентификацией с ключом в команде

0xE92D [CLSCRF MifarePlus AuthenticateDirect](#)

IN: AuthType[1]; KeyType[1]; SectorNumber[1]; Key[16]; 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;
Key[16] - ключ AES-128 для аутентификации;
LenCap[1] - длина блока характеристик считывателя (0..6, установить в 0);
PCDCap2[0..6] - блок характеристик считывателя (пока отсутствует).

3.8.8 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).

3.8.9 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).

3.8.10 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).

3.8.11 0xA5 Запись данных

0xA5 [CLSCRF MifarePlus WriteData](#)

IN: EncryptionMode[1]; ValueType[1]; SectorNumber[1]; BlockNumber[1];
BlocksCount[1]; Data[16/32/48]

OUT: ACK[1]

EncryptionMode[1] – режим защиты обмена данными (см. [таблицы](#));

ValueType[1] – тип записываемого значения (см. [таблицы](#));

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).

3.8.12 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] – слагаемое, на которое увеличивается значение.

3.8.13 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] – вычитаемое, на которое уменьшается значение.

3.8.14 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.

3.8.15 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] – слагаемое, на которое увеличивается значение;

DestinationSectorNumber[1]– номер сектора для записи;

DestinationBlockNumber[1]– номер блока для записи.

3.8.16 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] – вычитаемое, на которое уменьшается значение;

DestinationSectorNumber[1]– номер сектора для записи;

DestinationBlockNumber[1]– номер блока для записи.

3.8.17 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.

3.8.18 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.

3.8.19 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.

3.8.20 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.

3.8.21 0xA3 Снятие выбора виртуальной карты

0xA3 [CLSCRF MifarePlus VirtualCardDeselect](#)

IN: OperationType[1]

OUT: ACK[1]

OperationType[1] – тип операции (0x00);

3.8.22 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.

3.9 Обмен данными с картой Mifare Plus EV1

3.9.1 Таблица формирования параметра типа значения

Назначение блока данных	Значение параметра Value/Key Type	Значение параметров Sector Number, Block Number
Блоки и данные		
MIFARE Data/Value Blocks MIFARE Sector Trailers	0x00	XX, XX
Специализированные блоки		
MFP Configuration Block	0xB0	0x00, 0x00
VC Installation Identifier	0xB1	0x00, 0x00
ATS Information	0xB2	0x00, 0x00
Field Configuration Block	0xB3	0x00, 0x00
Anti Tearing Configuration Block	0xB4	0x00, 0x00
Ключи секторов		
AES Sector Keys (Key A)	0x4A	XX, 0x00
AES Sector Keys (Key B)	0x4B	XX, 0x00
Специализированные ключи		

Card Master Key	0x90	0x00, 0x00
Card Configuration Key	0x91	0x00, 0x00
Level 3 Switch Key	0x93	0x00, 0x00
SL1 Card Authentication Key	0x94	0x00, 0x00
L3 Sector Switch Key	0x96	0x00, 0x00
L1 L3 Mix Sector Switch Key	0x97	0x00, 0x00
Ключи виртуальных карт		
VC Proximity Key	0xA1	0x00, 0x00
VC Select ENC Key	0xA2	0x00, 0x00
VC Select MAC Key	0xA3	0x00, 0x00
Блоки Transaction MAC		
Transaction MAC Key 1	0xC1	0x00, 0x00
Transaction MAC Conf. Key 1	0xC1	0x01, 0x00
Transaction MAC Block 1	0xC1	0x02, 0x00
Commit Reader ID Block 1	0xC1	0x03, 0x00
TM Configuration Block 1.0	0xC1	0x04, 0x00
TM Configuration Block 1.1	0xC1	0x05, 0x00
TM Configuration Block 1.2	0xC1	0x06, 0x00
Transaction MAC Key 2	0xC2	0x00, 0x00
Transaction MAC Conf. Key 2	0xC2	0x01, 0x00
Transaction MAC Block 2	0xC2	0x02, 0x00
Commit Reader ID Block 2	0xC2	0x03, 0x00
TM Configuration Block 2.0	0xC2	0x04, 0x00
TM Configuration Block 2.1	0xC2	0x05, 0x00
TM Configuration Block 2.2	0xC2	0x06, 0x00

Transaction MAC Key 3	0xC3	0x00, 0x00
Transaction MAC Conf. Key 3	0xC3	0x01, 0x00
Transaction MAC Block 3	0xC3	0x02, 0x00
Commit Reader ID Block 3	0xC3	0x03, 0x00
TM Configuration Block 3.0	0xC3	0x04, 0x00
TM Configuration Block 3.1	0xC3	0x05, 0x00
TM Configuration Block 3.2	0xC3	0x06, 0x00
Transaction MAC Key 4	0xC4	0x00, 0x00
Transaction MAC Conf. Key 4	0xC4	0x01, 0x00
Transaction MAC Block 4	0xC4	0x02, 0x00
Commit Reader ID Block 4	0xC4	0x03, 0x00
TM Configuration Block 4.0	0xC4	0x04, 0x00
TM Configuration Block 4.1	0xC4	0x05, 0x00
TM Configuration Block 4.2	0xC4	0x06, 0x00

3.9.2 Таблица формирования параметра режима защиты передачи данных

MAC в команде	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
-----	-----	----	------

3.9.3 Таблица допустимых режимов защиты передачи данных

Операция	MAC в команде	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.

3.9.4 0xE8A8 Запись данных персонализации в карту

0xE8A8 CLSCRF MifarePlusEV1 WritePerso

IN: T_CL[1]; ValueType[1]; SectorNumber[1]; BlockNumber[1]; BlocksCount[1]; Data[BlocksCount * 16]

OUT: ACK[1]

T_CL[1] - по какому протоколу выполнить команду (0x00: ISO14443-3; 0x01: ISO14443-4);

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 начального блока;

BlocksCount[1] - количество записываемых блоков;

Data[BlocksCount * 16] - данные для записи.

Пример: E8 A8 00 C0 00 02 01 00112233445566778899AABBCCDDEEFF

3.9.5 0xE8AA Персонализация карты

0xE8AA CLSCRF MifarePlusEV1 CommitPerso

IN: Options[1]; T_CL[1]; 0x21; 0x43; 0x65; 0x87;

OUT: ACK[1];

Options[1] - дополнительные опции команды (0x00 или 0x01 - переключить карту в режим SL1; 0x03 - переключить карту в режим SL3);

T_CL[1] - по какому протоколу выполнить команду (0x00: ISO14443-3; 0x01: ISO14443-4);

Пример: e8 aa 03 01 21 43 65 87

Ответ: 00

3.9.6 0xE872 Управление аутентификацией

0xE872 CLSCRF MifarePlusEV1 Authenticate

IN: SL[1]; T_CL[1]; AuthType[1]; KeyType[1]; SectorNumber[1]; KeyFlashAddress[2]; LenDIV[1]; DIV[LenDIV]; LenCap[1]; PCDCap[1..6]

OUT: ACK[1]; RCOut[6]; TCOut[6]

SL[1] - текущий уровень SL у сектора (0x00 - SL0, 0x01 - SL1, 0x03 - SL3);

T_CL[1] - по какому протоколу выполнить команду (0x00: ISO14443-3; 0x01: ISO14443-4);

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;

LenDIV[1] - длина вектора диверсификации;

DIV[LenDIV] - данные диверсификации;

LenCap[1] - длина блока характеристик считывателя (1..6);

PCDCap[1..6] - блок характеристик считывателя (бит 0 байта 0 хранит тип шифрованного обмена: 0 - EV0, 1 - EV1; остальные биты зарезервированы для будущего использования и могут иметь любые значения);

RCOut[6] - блока характеристик считывателя из ответа карты;

TCOut[6] - блок характеристик карты из ответа карты.

Пример: E8 72 03 01 01 4A 00 0300 00 01 01

Пример 2: E8 72 03 01 01 4B 00 1300 00 01 01

Ответ: 00 01 00 00 00 00 00 01 00 00 00 00 00

3.9.7 0xE82D Управление аутентификацией с ключом в команде

0xE82D CLSCRF MifarePlusEV1 AuthenticateDirect

IN: SL[1]; T_CL[1]; AuthType[1]; KeyType[1]; SectorNumber[1]; Key[16]; LenDIV[1]; DIV[LenDIV]; LenCap[1]; PCDCap[1..6]

OUT: ACK[1]; RCOut[6]; TCOut[6]

SL[1] - текущий уровень SL у сектора (0x00 - SL0, 0x01 - SL1, 0x03 - SL3);

T_CL[1] - по какому протоколу выполнить команду (0x00: ISO14443-3; 0x01: ISO14443-4);

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;
Key[16] - ключ AES-128 для аутентификации;
LenDIV[1] - длина вектора диверсификации;
DIV[LenDIV] - данные диверсификации;
LenCap[1] - длина блока характеристик считывателя (1..6);
PCDCap[1..6] - блок характеристик считывателя (бит 0 байта 0 хранит тип шифрованного обмена: 0 - EV0, 1 - EV1; остальные биты зарезервированы для будущего использования и могут иметь любые значения);
RCOut[6] - блока характеристик считывателя из ответа карты;
TCOut[6] - блок характеристик карты из ответа карты.
 Пример: E8 72 03 01 01 4A 00
 01B48C7B65D302A647705D25D13ACDED 00 01 01
 Пример 2: E8 72 03 01 01 4B 00
 01B48C7B65D302A647705D25D13ACDED 00 01 01
 Ответ: 00 01 00 00 00 00 00 01 00 00 00 00 00

3.9.8 0xE830 Чтение данных

0xE830 CLSCRF MifarePlusEV1 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 прочесть (1..3);
DataLength[2] - длина прочитанного массива данных (байт);
Data[16/32/48] – прочитанный массив данных (1,2 или 3 блока).
 Пример: E8 30 01 00 02 00 03

3.9.9 0xE8A0 Запись данных

0xE8A0 CLSCRF MifarePlusEV1 WriteData

IN: EncryptionMode[1]; ValueType[1]; SectorNumber[1]; BlockNumber[1]; BlocksCount[1]; Data[16/32/48]

OUT: ACK[1]; TMACCounter[4]; TMACValue[8];

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);
TMACCounter[4] – значение счетчика TMAC;
TMACValue[8] – значение TMAC.

Пример: E8 A0 02 00 00 00 01 00112233445566778899AABBCCDDEEFF

3.9.10 0xE8B0 Прибавление значения

0xE8B0 [CLSCRF MifarePlusEV1 Increment](#)

IN: EncryptionMode[1]; SourceSectorNumber[1]; SourceBlockNumber[1]; Value[4]

OUT: ACK[1]

EncryptionMode[1] – режим защиты обмена данными (0x00 - без защиты, 0x01 - MAC в ответе);

SourceSectorNumber[1] – номер исходного сектора;

SourceBlockNumber[1] – номер исходного блока;

Value[4] – слагаемое, на которое увеличивается значение.

Пример: E8 B0 01 00 01 00112233

3.9.11 0xE8B2 Вычитание значения

0xE8B2 [CLSCRF MifarePlusEV1 Decrement](#)

IN: EncryptionMode[1]; SourceSectorNumber[1]; SourceBlockNumber[1]; Value[4]

OUT: ACK[1]

EncryptionMode[1] – режим защиты обмена данными (0x00 - без защиты, 0x01 - MAC в ответе);

SourceSectorNumber[1] – номер исходного сектора;

SourceBlockNumber[1] – номер исходного блока;

Value[4] – вычитаемое, на которое уменьшается значение.

Пример: E8 B2 01 00 01 00112233

3.9.12 0xE8B4 Запись данных из буфера переноса в блок

0xE8B4 [CLSCRF MifarePlusEV1 Transfer](#)

IN: EncryptionMode[1]; DestinationSectorNumber[1]; DestinationBlockNumber[1];

OUT: ACK[1]; TMACCounter[4]; TMACValue[8];

EncryptionMode[1] – режим защиты обмена данными (0x00 - без защиты, 0x01 - MAC в ответе);

DestinationSectorNumber[1] – номер сектора для записи;

DestinationBlockNumber[1] – номер блока для записи;

TMACCounter[4] – значение счетчика TMAC;

TMACValue[8] – значение TMAC.

Пример: E8 B4 01 00 01

3.9.13 0xE8B6 Прибавление значения с последующей записью данных из буфера переноса в блок

0xE8B6 CLSCRF MifarePlusEV1 IncrementTransfer

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

OUT: ACK[1]; TMACCounter[4]; TMACValue[8];

EncryptionMode[1] – режим защиты обмена данными (0x00 - без защиты, 0x01 - MAC в ответе);

SourceSectorNumber[1] – номер исходного сектора;

SourceBlockNumber[1] – номер исходного блока;

Value[4] – слагаемое, на которое увеличивается значение;

DestinationSectorNumber[1] – номер сектора для записи;

DestinationBlockNumber[1] – номер блока для записи;

TMACCounter[4] – значение счетчика TMAC;

TMACValue[8] – значение TMAC.

Пример: E8 B6 01 00 01 00112233 00 01

3.9.14 0xE8B8 Вычитание значения с последующей записью данных из буфера переноса в блок

0xE8B8 CLSCRF MifarePlusEV1 DecrementTransfer

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

OUT: ACK[1]; TMACCounter[4]; TMACValue[8];

EncryptionMode[1] – режим защиты обмена данными (0x00 - без защиты, 0x01 - MAC в ответе);

SourceSectorNumber[1] – номер исходного сектора;

SourceBlockNumber[1] – номер исходного блока;

Value[4] – вычитаемое, на которое уменьшается значение;

DestinationSectorNumber[1] – номер сектора для записи;

DestinationBlockNumber[1] – номер блока для записи;

TMACCounter[4] – значение счетчика TMAC;

TMACValue[8] – значение TMAC.

Пример: E8 B8 01 00 01 00112233 00 01

3.9.15 0xE8C2 Запись данных блока значения в буфер переноса

0xE8C2 CLSCRF MifarePlusEV1 Restore

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

OUT: ACK[1]

EncryptionMode[1] – режим защиты обмена данными (0x00 - без защиты, 0x01 - MAC в ответе);

SourceSectorNumber[1] – номер исходного сектора;

SourceBlockNumber[1] – номер исходного блока.

Пример: E8 C2 01 00 01

3.9.16 0xE84B Завершающий запрос поддержки виртуальных карт

0xE84B CLSCRF MifarePlusEV1 VirtualCardSupportLastISOL3

IN: IID[16]; PCDCapL3[4]

OUT: ACK[1]; Info[1]

IID[16] – идентификатор инсталляции;

PCDCapL3[4] – блок характеристик считывателя (пока отсутствует);

Info[1] – информация о карте (0x83 – 4 байт UID, 0x03 – 7 байт UID);

Пример: E8 4B 00112233445566778899AABBCCDDEEFF 00112233

3.9.17 0xE8F2 Поиск релейной атаки

0xE8F2 CLSCRF MifarePlusEV1 ProximityCheck

IN: Steps[1]; PcKeyNr[2]; GenRndC[1]; RndC[8]

OUT: ACK[1]

Steps[1] – количество циклов Proximity Check – должно быть 0x01.

PcKeyNr[2] – адрес ключа Flash Proximity Check MAC Key (LSB) во flash-памяти считывателя.

GenRndC[1] – 0x00: RndC задается в команде; 0x01: сгенерировать RndC.

RndC[8] – заданный RndC (если GenRndC == 0x00).

Пример: E8 F2 01 0600 01

3.9.18 0xE8C8 Установка идентификатора считывателя

0xE8C8 CLSCRF MifarePlusEV1 CommitReaderID

IN: Sector[1]; Block[1], TMAcNr[1], KeyNr[2]

OUT: ACK[1]; PrevTMRI[16]

Sector[1] – номер сектора, где будет храниться идентификатор считывателя;

Block[1] – номер блока в секторе, где будет храниться идентификатор считывателя;

TMAcNr[1] – номер кластера TMAc в карте (1..4);

KeyNr[2] – номер блока во флеш считывателя, где содержится ключ вычисления TMAc;

PrevTMRI[16] – расшифрованный идентификатор считывателя от предыдущей транзакции в ответе от карты.

Пример: E8 C8 01 00 01 0900

3.9.19 0xE882 Выбор карты ISO, внешняя аутентификация

0xE882 CLSCRF MifarePlusEV1 ISOSelectExternalAuthenticate

IN: EncKeyNr[2]; MacKeyNr[2]; IIDL[1]; IID[IIDL]

OUT: ACK[1]; VCDataLength[2]; VCData[VCDataLength]
EncKeyNr[2] - адрес ключа VC Select ENC Key во flash-памяти считывателя;
MacKeyNr[2] - адрес ключа VC Select MAC Key во flash-памяти считывателя;
IDDL[1] - длина IID в байтах;
IDD[IDDL] - Installation Identifier (DF Name или идентификатор приложения ISO);
VCDataLength[2] - количество байт VCData, принятых от карты;
VCData[VCDataLength] - данные виртуальной карты.
 Пример: E8 82 0400 0500 10 A00000039656434101344157799DCA86

3.9.20 0xE8A4 Выбор карты ISO

0xE8A4 CLSCRF MifarePlusEV1 ISOSelect

IN: EncKeyNr[2]; IIDL[1]; IID[IDDL]
 OUT: ACK[1]; VCDataLength[2]; VCData[VCDataLength]
EncKeyNr[2] - адрес ключа VC Select ENC Key во flash-памяти считывателя;
IDDL[1] - длина IID в байтах;
IDD[IDDL] - Installation Identifier (DF Name или идентификатор приложения ISO);
VCDataLength[2] - количество байт VCData, принятых от карты;
VCData[VCDataLength] - данные виртуальной карты.
 Пример: E8 A4 0400 10 00112233445566778899AABBCCDDEEFF

3.9.21 0xE87A Аутентификация переключения сектора

0xE87A CLSCRF MifarePlusEV1 AuthenticateSectorSwitch

IN: L1L3[1]; SSKeyNr[2]; Sector[1]; SKeyBNr[2]; LSS[1]; DISS[LSS]; LSK[1]; DISK[LSK]
 OUT: ACK[1]
L1L3[1] - 0x01: L3SectorSwitch; 0x02: L1L3MixSectorSwitch;
SSKeyNr[2] - адрес ключа Sector Switch Key во flash-памяти считывателя;
Sector[1] - номер переключаемого сектора считывателя;
SKeyBNr[2] - адрес ключа В сектора во flash-памяти считывателя;
LSS[1] - длина вектора диверсификации для ключа Sector Switch Key;
DISS[LSS] - вектор диверсификации для Sector Switch Key;
LSK[1] - длина вектора диверсификации для ключа В сектора;
DISK[LSK] - вектор диверсификации для ключа В сектора.
 Пример: E8 7A 02 0900 07 0B00 00 00

3.9.22 0xE840 Смена типа идентификатора карты Mifare Classic EV1

0xE840 CLSCRF MifarePlusEV1 MF PersonalizeUID

IN: UidType[1]
 OUT: ACK[1]
UidType[1] - режим UID (режим, в который переводится карта)

Допустимо любое из значений:

UIDF0 = 0x00,

UIDF1 = 0x40,

UIDF2 = 0x20,

UIDF3 = 0x60.

Пример: E8 40 01

3.9.23 0xE844 Конфигурация карты в SL1

0xE844 [CLSCRF MifarePlusEV1 SetConfigSL1](#)

IN: T_CL_Disable[1]

OUT: ACK[1]

T_CL_Disable[1] - отключение T=CL для карты в режиме SL1: 0x00 или 0x01 - отключить.

Пример: E8 44 01

3.9.24 0xE860 Чтение версии карты

0xE860 [CLSCRF MifarePlusEV1 GetVersion](#)

IN: VerInfo[28]

OUT: ACK[1]

VerInfo[28] - байты информации о карте.

Пример: E8 60

3.9.25 0xE83C Вычитать подпись карты

0xE83C [CLSCRF MifarePlusEV1 ReadSignature](#)

IN: T_CL[1]; Addr[1]

OUT: ACK[1]; Sign[56]

T_CL[1] - по какому протоколу выполнить команду (0x00: ISO14443-3; 0x01: ISO14443-4);

Addr[1] - 0x00 (зарезервировано);

Sign[56] - подпись проверки оригинальности ECC.

Пример: E8 3C 01 00

3.10 Обмен данными с картой Mifare DES Fire

3.10.1 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

3.10.2 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/) (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 послышки.

3.11 Непосредственный обмен данными с картой

Внимание! Для режима ISO14443 T=CL используйте связку команд [0x75](#) и [0x47](#).

3.11.1 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.

3.12 Работа с SAM-модулем

Важно! Все команды работы с SAM-модулем возвращают в ответе 2 байта SW1, SW2 - статусы ответа от SAM-модуля.

Но эти два байта следует обрабатывать только в случае, если общий статус ответа ACK[1] равен UEM_SAM_APDU_ERR (0xC8).

3.12.1 0xB2 Смена рабочего SAM-модуля

[0xB2 CLSCRF SAM SetCurrentHolder](#)

[CLSCRF SAM SetCurrentHolderGetATR](#) [CLSCRF SAM SetCurrentHolderEx](#)

IN: Options_Num[1]; [WT10[1]]; F_D[1]

OUT: ACK[1], [LenATR[2], ATR[LenATR]];

Options_Num[1] – опции и номер холдера:

биты 7..4 - опции - независимы и могут устанавливаться в любых сочетаниях.

бит 7 - Opt_WT=1 - время ожидания ATR кодируется в следующем байте;
бит 6 - Opt_PPS1=1 - значение PPS1 задается в следующем байте;
бит 5 - Opt_Deactivate=1 - выключить предыдущую рабочую контактную карту;
бит 4 - Opt_SendATR=1 - выдать ATR контактной карты, назначенной рабочей;
биты 3..0 - NumHolder - номер холдера = 1, 2... (0 - особое значение).
WT10[1] - нестандартное время ожидания ATR в десятках миллисекунд (ISO 7816-3, п.8.1) (255 = RFU).
F_D[1] - значение PPS1, отличное от глобального байта TA1 (ISO 7816-3, п.9.2, п.8.3).
LenATR[2] – длина принятого от контактной карты ATR (при SendATR = 1).
ATR[LenATR] – принятый от контактной карты ATR (при SendATR = 1).

При подаче питания все контактные карты активируются, и одна из них является рабочей.

Если значение NumHolder превышает количество имеющихся холдеров, выдается ошибка.

Иначе, Если NumHolder > 0

Контактная карта в холдере NumHolder становится рабочей.
Все последующие команды будут восприниматься именно этой картой.
Переключение на холдер с выключенной картой занимает не менее 50 мс.
Переключение между активированными контактными картами занимает менее 1 мс.
Попытка переключиться на холдер с отсутствующей или неисправной контактной картой занимает время WT (ожидание ATR от карты) согласно ISO 7816-3, п.8.1.
Для выбора нестандартного WT служат бит Opt_WT и FlashParam[20]:
Если Opt_WT = 1
WT = WT10 * 10 мс (до 2540 мс).
иначе (при Opt_WT = 0)
Если FlashParam[20] = 0xFF
WT соответствует стандарту (ISO 7816-3, п.8.1) (около 750 мс).
иначе (при FlashParam[20] < 0xFF)
WT = FlashParam[20] * 10 мс (до 2540 мс).

Производители некоторых контактных карт для повышения скорости обмена

допускают выбор нестандартных параметров, не совпадающих с указанными в ATR.

Например, для SAM AV2 от NXP при следовании стандарту

PPS1 = 0x18 (fKHz = 4800, Baudrate = 154838 бод),

Однако, в нарушение стандарта, допустимы следующие значения:

PPS1 = 0x96 (fKHz = 4800, Baudrate = 300000 бод),

PPS1 = 0xD7 (fKHz = 12000, Baudrate = 375000 бод).

Для выбора нестандартных параметров служат бит Opt_PPS1 и FlashParam [21].

```

Если Opt_PPS1=1
    PPS1 = F_D;
иначе (при Opt_PPS1 = 0)
    Если FlashParam[21] = 0xFF
        PPS1 = TA1
        (для SAM AV2 TA1 = 0x18 (fKHz = 4800, Baudrate =
154838));
    иначе (при FlashParam[21] < 0xFF)
        PPS1 = FlashParam[21].

```

```

Если Opt_Deactivate = 1
    Предыдущая рабочая контактная карта (при наличии) выключается.
иначе (при Opt_Deactivate = 0)
    Предыдущая рабочая контактная карта (при наличии) остается
активированной.

```

Иначе (при NumHolder = 0)

Рабочей контактной карты не будет (все активированные контактные карты выключаются), значения битов 7..4 игнорируются.

Примеры:

Выключить все контактные карты

Команда: B2 00

Ответ: 00 - все контактные карты выключены

D8 - нет ни одного холдера

Назначить рабочей контактную карту в холдере 5 (несуществующем)

Команда: B2 05

Ответ: D8 - нет такого холдера

Назначить рабочей контактную карту в холдере 2 и установить параметры обмена (PPS1) F=9, D=6

Команда: B2 42 96

Ответ: 00

C9 - таймаут при выполнении команды PPS (параметры не поддерживаются)

Назначить рабочей контактную карту в холдере 2 с ожиданием ATR 50 мс и установить параметры обмена (PPS1) F=1, D=3

Команда: B2 C2 05 13

Ответ: 00

Назначить рабочей контактную карту в холдере 2

Команда: B2 02

Ответ: 00

FF - не получен ATR

Назначить рабочей контактную карту в холдере 2 и выдать ее ATR

Команда: B2 12

Ответ: 00 11 00 3B 7C 95 00 00 45 53 4D 41 52 54 47 4F 53 54 31 30

Выключить текущую рабочую карту, назначить рабочей контактную карту в холдере 1,

и выдать ее ATR

Команда: B2 31

Ответ: 00 1C 00 3B DF 18 FF 81 F1 FE 43 00 3F 03 83 4D 49 46 41 52 45 20
50 6C 75 73 20 53 41 4D 3B

3.12.2 0xB4 Получение номера рабочего SAM-модуля

0xB4 [CLSCRF SAM GetCurrentHolder](#)

OUT: ACK[1]; NumSAM[1];

NumSAM[1] – номер модуля от 1 до 4 (если нет SAM-модуля ни в одном из холдеров, то NumSAM = 0).

Пример: b4

Ответ: 00 01

3.12.3 0xB6 Выполнение команды в формате APDU

0xB6 [CLSCRF SAM APDU](#)

IN: send_len[2]; send_data[send_len];

OUT: ACK[1], rec_len[2], rec_data[rec_len];

send_len[2] - длина команды;

send_data[send_len] – команда;

rec_len[2] - длина ответа;

rec_data[rec_len] - ответ, в том числе SW1 и SW2

Пример: b6 05 00 80 60 00 00 00

Ответ: 00 21 00 04 01 01 03 02 28 01 04 01 01 03 02 28 01 04 25 31 01
D5 28 80 91 57 55 00 00 0C 08 0A 00 A2 90 00

3.12.4 0xBA Выполнение сброса (рестарт) SAM

0xBA [CLSCRF SAM Reset](#) , [CLSCRF SAM KillAuth](#)

IN: Type[1]; 0x43; 0x65; 0x87;

Type[1] – 0x21 - холодный сброс модуля; 0x01 - мягкий сброс аутентификации карты и активации OfflineChange или Offline Crypto ключей, 0x00 - мягкий сброс аутентификации всего модуля.

OUT: ACK[1];

Пример: ba 21 43 65 87

Ответ: 00

3.12.5 0xB6 Halt для карты ISO14443A через SAM

0xB6 [CLSCRF SAM Halt A](#)

IN: LC[1];

OUT: ACK[1];

LC[1] - логический канал от 0 до 3;

Пример: 58 50 00

Ответ: 00

3.13 Работа с контактными картами стандарта ISO7816

Важно! Все команды работы с контактными картами возвращают в ответе 2 байта SW1, SW2 - статусы ответа от контактной карты.

Но эти два байта следует обрабатывать только в случае, если общий статус ответа ACK[1] равен UEM_SAM_APDU_ERR (0xC8).

3.13.1 0xB2 Смена рабочей контактной карты

[Аналогична команде SAM-модуля.](#)

3.13.2 0xB4 Получение номера рабочей контактной карты

[Аналогична команде SAM-модуля.](#)

3.13.3 0xB1 Запросить текущее состояние холдеров

0xB1 [CLSCRF ISO7816 GetHoldersInfo](#)

IN: NumHolder[1];

Запрашивает текущее состояние холдеров.

Важно! Работает только для считывателей с детектором контакта - с холдерами на отдельной плате!!!

NumHolder[1] - номер холдера = 1, 2... (0 - особое значение).

Если значение NumHolder превышает количество имеющихся холдеров, выдается ошибка.

Если NumHolder = 0, то

OUT: ACK[1], Count[1], Active[1], CurNo[1], WTI[1]

Count - количество холдеров.

Active - маска активированных контактных карт:

0x01 - контактная карта 1 активирована;

0x02 - контактная карта 2 активирована;

0x04 - контактная карта 3 активирована;

0x08 - контактная карта 4 активирована.

CurNo - номер холдера рабочей контактной карты (0..Count).

WPI - максимальное время ожидания ATR в десятках мс.

Если NumHolder > 0, то

OUT: ACK[1], State[1], Prot[1], F_D[1], fKHz[2], BR[4]

State - состояние контактной карты

Биты 7..2 - не используются;

Бит 1 - контактная карта является рабочей;

Бит 0 - контактная карта активирована.

Prot - протокол T=(0 или 1).

F_D - параметры обмена (значение PPS1 из ISO 7816-3, п.9.2, п.8.3).

fKHz - частота тактирования (кГц).

BR - скорость обмена (бод).

Примеры:

Выдать общую информацию о холдерах

Команда: B1 00

Ответ: 00 04 03 01 4B

04 - в считывателе 4 холдера;

03 - активированы карты в холдерах 1 и 2;

01 - рабочая контактная карта в 1-м холдере;

05 - максимальное время ожидания ATR 50 мс.

Выдать информацию о холдере 1

Команда: B1 01

Ответ: 00 03 01 18 C0 12 D6 5C 02 00

03 - карта в этом холдере является рабочей

01 - протокол T=1

18 - параметры обмена F=1, D=8

C0 12 - частота тактирования 4800 кГц

D6 5C 02 00 - скорость обмена 154838 бод.

3.13.4 0xB6 Выполнение команды в формате APDU

[Аналогична команде SAM-модуля.](#)

3.13.5 0xBA Выполнение сброса (рестарт) контактной карты

Аналогична команде SAM-модуля.

3.14 Работа SAM-модуля с Mifare Classic

3.14.1 0xAB Выполнение аутентификации карты Mifare Classic

0xAB CLSCRF SAM MifareAuthenticate

IN: LC[1], AuthType[1], UID[4], KeyNo[1], KeyVer[1], KeyType[1], BlockNo[1];

OUT: ACK[1], SW1[1], SW2[1];

LC[1] - логический канал от 0 до 3;

AuthType[1] - тип аутентификации: 1 - First Authenticate, иначе - Following Authenticate;

UID[4] - уникальный номер карты;

KeyNo[1] - номер Mifare-ключа в SAM-модуле от 1 до 127;

KeyVer[1] - версия Mifare-ключа в SAM-модуле от 0 до 255;

KeyType[1] - тип Mifare-ключа: 0x0a - KeyA, иначе KeyB;

BlockNo[1] - абсолютный номер аутентифицируемого блока в карте;

SW1[1] - байт статуса;

SW2[1] - байт статуса.

Пример: ab 00 01 fc b0 a8 3c 20 02 0a 09

Ответ: 00 90 00

3.14.2 0xAC Чтение блока карты Mifare Classic

0xAC CLSCRF SAM MifareRead

IN: LC[1], BlockNo[1];

OUT: ACK[1], SW1[1], SW2[1], Data[16];

LC[1] - логический канал от 0 до 3;

BlockNo[1] - абсолютный номер читаемого блока в карте;

SW1[1] - байт статуса;

SW2[1] - байт статуса;

Data[16] - данные прочитанного блока.

Пример: ac 00 09

Ответ: 00 90 00 30 00 00 00 CF FF FF FF 30 00 00 00 0A F5 0A F5

00 90 00 28 00 00 00 D7 FF FF FF 28 00 00 00 0A F5 0A F5 (после Decrement)

3.14.3 0xAD Запись блока карты Mifare Classic

0xAD [CLSCRF SAM MifareWrite](#)

IN: LC[1], BlockNo[1], Data[16];

OUT: ACK[1], SW1[1], SW2[1];

LC[1] - логический канал от 0 до 3;

BlockNo[1]- абсолютный номер читаемого блока в карте;

Data[16] – записываемые данные.

SW1[1] - байт статуса;

SW2[1] - байт статуса;

Пример: ad 00 08 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f

Ответ: 00 90 00

3.14.4 0xAE Работа с блоком Value карты Mifare Classic

0xAE [CLSCRF SAM MifareValue](#)

IN: LC[1], Cmd[1], BlkSrc[1], Delta[4], BlkTgt[1];

OUT: ACK[1], SW1[1], SW2[1];

LC[1] - логический канал от 0 до 3;

Cmd[1] - код операции:

0xc0 – CLSCRF_SAM_MifareDecrement;

0xc1 – CLSCRF_SAM_MifareIncrement;

0xc2 – CLSCRF_SAM_MifareRestore;

BlkSrc[1]- абсолютный номер исходного блока в карте;

Delta[4] - аргумент операции Decrement или Increment;

BlkTgt[1] - абсолютный номер блока результата в карте;

SW1[1] - байт статуса;

SW2[1] - байт статуса.

Возможны операции Decrement, Increment или Restore, включая Transfer.

Пример: ae 00 c0 09 08 00 00 00 09

Ответ: 00 90 00

3.14.5 0xAF Смена ключей карты Mifare Classic

0xAF [CLSCRF SAM MifareChangeKey](#)

IN: LC[1], Crypto[1], KeyNo[1], KeyVerA[1], KeyVerB[1], BlockNo[1], Access[4], UID[4];

OUT: ACK[1], SW1[1], SW2[1];

LC[1] - логический канал от 0 до 3;

Crypto[1] - маска метода компиляции ключей;

KeyNo[1] - номер нового Mifare-ключа в SAM-модуле от 1 до 127;

KeyVerA[1] - версия нового Mifare-ключа KeyA в SAM-модуле от 0 до 255;

KeyVerB[1] - версия нового Mifare-ключа KeyB в SAM-модуле от 0 до 255;

BlockNo[1]- абсолютный номер блока-трейлера в карте;

Access[4] - новые биты доступа в трейлере;

UID[4] - уникальный номер карты (не требуется, если Crypto равен 0);

SW1[1] - байт статуса;

SW2[1] - байт статуса.

Пример: af 00 00 20 01 01 0f ff 07 80 69

Пример: af 00 01 20 01 01 0f ff 07 80 69 fc b0 a8 3c (если Crypto)

Ответ: 00 90 00

3.15 Работа SAM-модуля с Mifare Ultralight C

Команды [чтения](#) и [записи](#) смотрите в разделе стандартных команд для работы с картами Mifare Ultralight.

3.15.1 0x2E Выполнение аутентификации карты Mifare Ultralight C

0x2E [CLSCRF SAM MifareUltralightC Authenticate](#)

IN: LC[1], RFU[1], KeyNo[1], KeyVer[1], UID[7];

OUT: ACK[1], SW1[1], SW2[1];

LC[1] - логический канал от 0 до 3;

RFU[1] - диверсификация (зарезервировано на будущее, установить в 0x00);

KeyNo[1] - номер Mifare-ключа в SAM-модуле от 1 до 127;

KeyVer[1] - версия Mifare-ключа в SAM-модуле от 0 до 255;

UID[7] - уникальный номер карты;

SW1[1] - байт статуса;

SW2[1] - байт статуса.

Пример: 2E 03 00 03 02 04 72 63 E1 ED 25 80

Ответ: 00 90 00

3.15.2 0x2F Смена ключей карты Mifare Ultralight C

0x2F [CLSCRF SAM MifareUltralightC WriteKey](#)

IN: LC[1], RFU[1], KeyNo[1], KeyVer[1], UID[7];

OUT: ACK[1], SW1[1], SW2[1];

LC[1] - логический канал от 0 до 3;

RFU[1] - диверсификация (зарезервировано на будущее, установить в 0x00);

KeyNo[1] - номер Mifare-ключа в SAM-модуле от 1 до 127;

KeyVer[1] - версия Mifare-ключа в SAM-модуле от 0 до 255;

UID[7] - уникальный номер карты;

SW1[1] - байт статуса;

SW2[1] - байт статуса.

Пример: 2F 03 00 05 02 04 72 63 E1 ED 25 80

Ответ: 00 90 00

3.16 Работа SAM-модуля с Mifare Plus

3.16.1 0xF8 Запись данных персонализации в карту

0xF8 [CLSCRF SAM MifarePlus WritePerso](#)

IN: LogicalChannel[1]; ValueType[1]; SectorNumber[1]; BlockNumber[1];
KeyNumber[1]; KeyVersion[1]; LenDiv[1]; Div[LenDiv]

OUT: ACK[1]; SW1[1]; SW2[1]

LogicalChannel[1] – логический канал от 0 до 3;

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);

KeyNumber[1] - номер Mifare-ключа в SAM-модуле от 1 до 127;

KeyVersion[1] - версия Mifare-ключа в SAM-модуле от 0 до 255;

LenDiv[1] - длина входного вектора диверсификации ключа;

Div[LenDiv] - входной вектор диверсификации ключа;

SW1[1] – байт статуса 1;

SW2[1] – байт статуса 2.

3.16.2 0xFA Персонализация карты

0xFA [CLSCRF SAM MifarePlus CommitPerso](#)

IN: LogicalChannel[1]; PassPhrase[4]= 0x87654321

OUT: ACK[1]; SW1[1]; SW2[1]

LogicalChannel[1] – логический канал от 0 до 3;

PassPhrase[4] – константное выражение для защиты от случайной персонализации ({0x21,0x43,0x65,0x87});

SW1[1] – байт статуса 1;

SW2[1] – байт статуса 2.

3.16.3 0xF0 Управление аутентификацией

0xF0 [CLSCRF SAM MifarePlus Authenticate](#)

IN: LogicalChannel[1]; AuthType[1]; KeyType[1]; SectorNumber[1];
KeyNumber[1]; KeyVersion[1]; LenCap[1]; PCDCap2[0..6]; LenDiv[1]; DivInput
[LenDiv]

OUT: ACK[1]; SW1[1]; SW2[1]

LogicalChannel[1] – логический канал от 0 до 3;

AuthType[1] - тип операции аутентификации (0x00 – первичная, 0x02 – последующая), сложенный с уровнем безопасности карты (0x00 - SL1, 0x04 -

SL2, 0x0C - SL3);

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

SectorNumber[1] - номер сектора для аутентификации;

KeyNumber[1] - номер Mifare-ключа в SAM-модуле от 1 до 127 (если равен 0x00 - сброс аутентификации карты);

KeyVersion [1] - версия Mifare-ключа в SAM-модуле от 0 до 255;

LenCap[1] - длина блока характеристик считывателя (0..6, установить в 0);

PCDCap2[0..6] - блок характеристик считывателя (пока отсутствует);

LenDiv[1] - длина входного вектора диверсификации ключа;

DivInput[LenDiv] - входной вектор диверсификации ключа;

SW1[1] – байт статуса 1;

SW2[1] – байт статуса 2.

Пример: F0 00 0C 4A 00 10 00 00 00

Ответ: 00 90 00

3.16.4 0xF6 Чтение нескольких блоков SL2

0xF6 CLSCRF SAM MifarePlus MultiBlockRead

IN: LogicalChannel[1]; AbsBlockNo[1]; BlocksCount[1]

OUT: ACK[1]; SW1[1]; SW2[1]; DataLength[2]; Data[16/32/48]

LogicalChannel[1] – логический канал от 0 до 3;

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);

SW1[1] – байт статуса 1;

SW2[1] – байт статуса 2.

3.16.5 0xF7 Запись нескольких блоков SL2

0xF7 CLSCRF SAM MifarePlus MultiBlockWrite

IN: LogicalChannel[1]; AbsBlockNo[1]; BlocksCount[1]; Data[16/32/48]

OUT: ACK[1]; SW1[1]; SW2[1]

LogicalChannel[1] – логический канал от 0 до 3;

AbsBlockNo[1] - абсолютный номер блока;

BlocksCount[1] - количество записываемых блоков;

Data[16/32/48] - массив данных для записи (1,2 или 3 блока);

SW1[1] – байт статуса 1;

SW2[1] – байт статуса 2.

3.16.6 0xF4 Чтение данных

0xF4 CLSCRF SAM MifarePlus ReadData

IN: LogicalChannel[1]; EncryptionMode[1]; ValueType[1]; SectorNumber[1];
BlockNumber[1]; BlocksCount[1]

OUT: ACK[1]; SW1[1]; SW2[1]; DataLength[2]; Data[16/32/48]

LogicalChannel[1] – логический канал от 0 до 3;
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 прочесть (1..3);
DataLength[2] – длина прочитанного массива данных (байт);
Data[16/32/48] – прочитанный массив данных (1,2 или 3 блока);
SW1[1] – байт статуса 1;
SW2[1] – байт статуса 2.
 Пример: F4 00 06 00 00 00 01
 Ответ: 00 90 00 10 00 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

3.16.7 0xF5 Запись данных

0xF5 CLSCRF SAM MifarePlus WriteData
 IN: LogicalChannel[1]; EncryptionMode[1]; ValueType[1]; SectorNumber[1]; BlockNumber[1]; Data[16/32/48]
 OUT: ACK[1]; SW1[1]; SW2[1]
LogicalChannel[1] – логический канал от 0 до 3;
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);
SW1[1] – байт статуса 1;
SW2[1] – байт статуса 2.
 Пример: F5 00 01 00 02 00 01 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
 Ответ: 00 90 00

3.16.8 0xFD Смена ключа

0xFD CLSCRF SAM MifarePlus ChangeKey
 IN: LogicalChannel[1]; EncryptionMode[1]; KeyType[1]; SectorNumber[1]; KeyNumber[1]; KeyVersion[1]; LenDiv[1]; DivInput[LenDiv]
 OUT: ACK[1]; SW1[1]; SW2[1]
LogicalChannel[1] – логический канал от 0 до 3;
EncryptionMode[1] – режим защиты обмена данными (0x00 - шифрование, MAC в запросе; 0x01 - шифрование, MAC в запросе, MAC в ответе);
KeyType[1] – тип ключа (см. [таблицу](#));
SectorNumber[1] – номер сектора;
KeyNumber[1] – номер Mifare-ключа в SAM-модуле от 1 до 127;
KeyVersion[1] – версия Mifare-ключа в SAM-модуле от 0 до 255;
LenDiv[1] – длина входного вектора диверсификации ключа;

DivInput[LenDiv] - входной вектор диверсификации ключа;

SW1[1] – байт статуса 1;

SW2[1] – байт статуса 2.

Пример: FD 00 01 4A 01 10 02 00

Ответ: 00 90 00

3.16.9 0xF1 Прибавление значения

0xF1 [CLSCRF SAM MifarePlus Increment](#)

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

OUT: ACK[1]; SW1[1]; SW2[1]

LogicalChannel[1] – логический канал от 0 до 3;

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;

SW1[1] – байт статуса 1;

SW2[1] – байт статуса 2.

3.16.10 0xF1 Вычитание значения

0xF1 [CLSCRF SAM MifarePlus Decrement](#)

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

OUT: ACK[1]; SW1[1]; SW2[1]

LogicalChannel[1] – логический канал от 0 до 3;

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;

SW1[1] – байт статуса 1;

SW2[1] – байт статуса 2.

3.16.11 0xF1 Запись данных из буфера переноса в блок

0xF1 [CLSCRF SAM MifarePlus Transfer](#)

IN: LogicalChannel[1]; OperationType[1]; EncryptionMode[1];
DestinationSectorNumber[1]; DestinationBlockNumber[1]

OUT: ACK[1]; SW1[1]; SW2[1]

LogicalChannel[1] – логический канал от 0 до 3;

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;
SW1[1] – байт статуса 1;
SW2[1] – байт статуса 2.

3.16.12 0xF1 Прибавление значения с последующей записью данных из буфера переноса в блок

0xF1 [CLSCRF SAM MifarePlus IncrementTransfer](#)

IN: LogicalChannel[1]; OperationType[1]; EncryptionMode[1];
SourceSectorNumber[1]; SourceBlockNumber[1]; Value[4];
DestinationSectorNumber[1]; DestinationBlockNumber[1]
OUT: ACK[1]; SW1[1]; SW2[1]
LogicalChannel[1]– логический канал от 0 до 3;
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;
SW1[1] – байт статуса 1;
SW2[1] – байт статуса 2.

3.16.13 0xF1 Вычитание значения с последующей записью данных из буфера переноса в блок

0xF1 [CLSCRF SAM MifarePlus DecrementTransfer](#)

IN: LogicalChannel[1]; OperationType[1]; EncryptionMode[1];
SourceSectorNumber[1]; SourceBlockNumber[1]; Value[4];
DestinationSectorNumber[1]; DestinationBlockNumber[1]
OUT: ACK[1]; SW1[1]; SW2[1]
LogicalChannel[1]– логический канал от 0 до 3;
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;
SW1[1] – байт статуса 1;
SW2[1] – байт статуса 2.

3.16.14 0xF1 Запись данных блока значения в буфер переноса

0xF1 [CLSCRF SAM MifarePlus Restore](#)

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

OUT: ACK[1]; SW1[1]; SW2[1]

LogicalChannel[1] – логический канал от 0 до 3;

OperationType[1] – тип операции (0xC2);

EncryptionMode[1] – режим защиты обмена данными (см. [таблицы](#));

SourceSectorNumber[1] – номер исходного сектора;

SourceBlockNumber[1] – номер исходного блока;

SW1[1] – байт статуса 1;

SW2[1] – байт статуса 2.

3.16.15 0xF2 Запрос поддержки виртуальных карт

0xF2 [CLSCRF SAM MifarePlus VirtualCardSupport](#)

IN: LogicalChannel[1]; IIDCount[1]; IIDs[16*IIDCount]; LenCap[1]; PCDCap
[LenCap]; DuosCount[1]; Duos[4*DuosCount];

OUT: ACK[1]; SW1[1]; SW2[1]; RndQ[1]; Info[1]; PICCCap[2]; UID[7];

LogicalChannel[1] – логический канал от 0 до 3;

IIDCount[1] – количество IID в команде;

IIDs[16*IIDCount] – идентификаторы инсталляции;

LenCap[1] – длина блока характеристик считывателя (0..3, установить в 0);

PCDCap[LenCap] – блок характеристик считывателя (пока отсутствует);

DuosCount[1] – количество пар записей ключей;

Duos[4*DuosCount] – указатели на пары записей ключей KencNum[1];
KencVer[1]; KmacNum[1]; KmacVer[1] в SAM-модуле, где Kenc – ключ
шифрования, Kmac – ключ подписи;

RndQ[1] – случайный вектор, соответствующий в SAM-модуле записи карты;

Info[1] – информация о карте (0x83 – 4 байт UID, 0x03 – 7 байт UID);

PICCCap[2] – характеристики карты;

UID[7] – идентификатор карты (4-байтовый или 7-байтовый, в зависимости от
Info);

SW1[1] – байт статуса 1;

SW2[1] – байт статуса 2.

3.16.16 0xF3 Выбор виртуальной карты

0xF3 [CLSCRF SAM MifarePlus VirtualCardSelect](#)

IN: LogicalChannel[1]; OperationType[1]; KeyNumber[1]; KeyVersion[1];
RndQ[12]; LenDiv[1]; DivInput[LenDiv];

OUT: ACK[1]; SW1[1]; SW2[1]

LogicalChannel[1] – логический канал от 0 до 3;

OperationType[1] – тип операции (0x01);
KeyNumber[1] – номер записи ключа Virtual Card Select Key;
KeyVersion[1] – версия записи ключа Virtual Card Select Key;
RndQ[12] – случайная последовательность, соответствующая записи о карте в SAM-модуле (получена в результате выполнения команды VirtualCardSupport);
LenDiv[1] - длина входного вектора диверсификации ключа;
DivInput[LenDiv] - входной вектор диверсификации ключа;
SW1[1] – байт статуса 1;
SW2[1] – байт статуса 2.

3.16.17 0xF3 Снятие выбора виртуальной карты

0xF3 [CLSCRF SAM MifarePlus VirtualCardDeselect](#)

IN: LogicalChannel[1]; OperationType[1]
OUT: ACK[1]; SW1[1]; SW2[1]
LogicalChannel[1] – логический канал от 0 до 3;
OperationType[1] – тип операции (0x00);
SW1[1] – байт статуса 1;
SW2[1] – байт статуса 2.

3.16.18 0xF9 Поиск релейной атаки

0xF9 [CLSCRF SAM MifarePlus ProximityCheck](#)

IN: LogicalChannel[1]; KeyNumber[1]; KeyVersion[1]; LenDiv[1]; DivInput[LenDiv];
OUT: ACK[1]; SW1[1]; SW2[1]
LogicalChannel[1] – логический канал от 0 до 3;
KeyNumber[1] – номер записи ключа Proximity Check Key;
KeyVersion[1] – версия записи ключа Proximity Check Key;
LenDiv[1] - длина входного вектора диверсификации ключа;
DivInput[LenDiv] - входной вектор диверсификации ключа;
SW1[1] – байт статуса 1;
SW2[1] – байт статуса 2.

3.17 Управление устройствами и метками стандарта ISO 18092

3.17.1 0x11 Активация устройства ISO18092

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 HE 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 атрибутом), должен быть не меньше 64 байт;

3.17.2 0x12 Отмена выбора ISO18092

0x12 [CLSCRF NFC663 Deselect](#)

IN: DeselectCommand[1]

OUT: ACK[1]

DeselectCommand[1] – запрос на отправку:

[PHPAL I18092MPI DESELECT DSL](#), либо

[PHPAL I18092MPI DESELECT RLS](#)

3.17.3 0x13 Обмен ISO18092

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;

3.17.4 0x40 Сброс протокола ISO18092

0x40 [CLSCRF NFC663 ResetProtocol](#)

OUT: ACK[1]

3.17.5 0x33 Запрос атрибутов ISO18092

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 HE 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;

3.17.6 0x34 Выбор параметров ISO18092

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](#));

3.17.7 0x1F Проверка присутствия ISO18092

0x1F [CLSCRF NFC663 PresenceCheck](#)

OUT: ACK[1]

3.17.8 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;

3.17.9 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;

3.17.10 0x28 Чтение серийного номера (NFCID3)

0x28 [CLSCRF NFC663 GetSerialNo](#)

OUT: ACK[1]; NFCID3[10]

NFCID3[10] – серийный номер NFCID3 - 10 байт;

3.17.11 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;

3.17.12 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;

3.17.13 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;

3.18 Работа с TDA8029

Важно! Все команды работы с SAM-модулем возвращают в ответе 2 байта SW1, SW2 - статусы ответа от SAM-модуля.

Но эти два байта следует обрабатывать только в случае, если общий статус ответа ACK[1] равен UEM_SAM_APDU_ERR (0xC8).

3.18.1 0x85 Непосредственный обмен с TDA

0x85 CLSCRF TDA8029 Exchange

Отправляет команду в формате ALPARI на микросхему TDA8029.

IN: CmdCode[1]; send_len[2]; send_data[send_len];

OUT: ACK[1], rec_len[2], rec_data[rec_len];

CmdCode[1] – код команды, отправляемой на TDA8029 (см. документацию на TDA8029).

send_len[2] – длина команды (не включая заголовок и байт контрольной суммы LRC, младший байт - вперед);

send_data[send_len] – команда (не включает заголовок и байт контрольной суммы LRC);

rec_len[2] – длина ответа (младший байт вперед);

rec_data[rec_len] – ответ от TDA8029 (не включает заголовок и байт контрольной суммы LRC).

Если ACK[1] = 0x80, то TDA8029 вернула ошибку. Код этой ошибки будет сохранен в rec_data, а rec_len будет = 0x0001.

Пример: 85 00 05 00 80 60 00 00 00
Ответ: 00 21 00 04 01 01 03 02 28 01 04 01 01 03 02 28 01 04 25 31 01
D5 28 80 91 57 55 00 00 0C 08 0A 00 A2 90 00

3.18.2 0x86 0xBA Сброс TDA

0x86 0xBA [CLSCRF TDA8026 Reset](#)

Выполняет перезапуск микросхемы.

IN: Interval[1];

OUT: ACK[1];

Interval[1] – интервал выключенного состояния, мс. Если равен 0, то TDA не включается.

Пример: 86 BA 0A

Ответ: 00

МикроЭМ

Руководство программиста

Часть



IV

4 Библиотека Clscrfl.dll

4.1 Управление 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_OpenETH

```
LONG CLSCRF_OpenETH(    IN LPVOID pReader,  
                        IN DWORD dwPort,  
                        IN PCHAR szIP,  
                        IN DWORD dwLogFile);
```

Открывает Ethernet-интерфейс. Эта функция должна вызываться 1 раз перед началом обмена со считывателем.

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

dwPort – порт считывателя

szIP – IP адрес считывателя в формате "aaa.bbb.ccc.ddd"

dwLogFile– вывод в файл хронологии обмена:

1 - создавать файл;

0 - не создавать файл.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.1.7 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.

4.1.8 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.

4.1.9 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.

4.1.10 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.

4.1.11 CLSCRF_GetLastError

LONG CLSCRF_GetLastError(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 Управление 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:

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 - 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
*** - резерв для будущего использования

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

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_Get_ReaderID

```
LONG CLSCRF_Get_ReaderID( IN LPVOID pReader,  
                           OUT LPBYTE pbReaderID );
```

Выдает уникальный идентификатор считывателя.

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

pbReaderID – адрес массива, в который будет помещено 16 байт уникального идентификатора.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.7 CLSCRF_Reader_Restart

```
LONG CLSCRF_Reader_Restart( IN LPVOID pReader,  
                             IN LPBYTE pbReaderID );
```

Перезагружает считыватель.

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

pbReaderID – адрес массива, в котором содержится 16 байт уникального идентификатора считывателя.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.8 CLSCRF_Mfrc_Rf_Off_On

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

Сбрасывает или выключает электромагнитное поле (RF) считывателя. Сброс RF необходим для рестарта микросхемы транспондера.

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

usDelay – интервал времени [мс], в течение которого RF выключено.

Если этот параметр равен 0, то RF остается выключенным.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.9 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 RF:

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

*** - 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.

4.2.10 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.

4.2.11 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.

Структура байтов **bBlinkColor** и **bPostColor**:

- Бит 0 - красный;
- Бит 1 - зеленый;
- Бит 2 - синий;
- Биты 3...7 - не используются.

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;
- 4 – мигать синим цветом(при наличии);
- 5 – мигать фиолетовым цветом(при наличии);
- 6 – мигать бирюзовым цветом(при наличии);
- 7 – мигать белым цветом(при наличии);

bBlinkCount – количество миганий (если 0 - не мигать);

bPostColor – последующее состояние светодиода:

- 0 – погасить;
- 1 – зажечь красным;
- 2 – зажечь зеленым;
- 3 – зажечь желто-оранжевым;
- 4 – зажечь синим цветом(при наличии);
- 5 – зажечь фиолетовым цветом(при наличии);
- 6 – зажечь бирюзовым цветом(при наличии);
- 7 – зажечь белым цветом(при наличии).

Возвращаемое значение:

- 0 – успешное выполнение,
- иначе – ошибка при выполнении.

4.2.12 CLSCRF_Switches

```
LONG CLSCRF_Switches(    IN LPVOID pReader,
                        IN BYTE  ucControl );
```

Устанавливает состояние выходных линий.

На плате расположены три контакта с надписями:

- "Out-" - может быть подключен к земле считывателя;

- "Out+" - может быть подключен к +5В;
 - "Т перевернутое" - земля считывателя.
- При подаче питания (по умолчанию) Out- и Out+ не подключены.

ucControl – управление контактами Out- и Out+

- 0x1X - управление контактом Out-
- 0x2X - управление контактом Out+
- 0xX1 - Out- подключить к земле считывателя
- 0xX2 - Out+ подключить к +5В

Примеры

Одновременное управление контактами Out- и Out+:

- 37 33 => Out- подключить к земле считывателя, Out+ подключить к +5В
- 37 32 => Out- отключить, Out+ подключить к +5В
- 37 31 => Out- подключить к земле считывателя, Out+ отключить
- 37 30 => отключить Out- и Out+ (состояние по умолчанию)

Индивидуальное управление Out+ (при этом Out- остается в прежнем состоянии):

- 37 22 => Out+ подключить к +5В
- 37 20 => отключить Out+

Индивидуальное управление Out- (при этом Out+ остается в прежнем состоянии):

- 37 11 => Out- подключить к земле считывателя
- 37 10 => отключить Out-

Примечание

При питании от USB ограничение на общее потребление тока считывателем 300 мА.

На контакт Out+ при полной нагрузке считывателя остается не более 150 мА.

Возвращаемое значение:

- 0 – успешное выполнение,
- иначе – ошибка при выполнении.

4.2.13 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.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.14 CLSCRF_EraseFlash

LONG CLSCRF_EraseFlash(IN LPVOID pReader);

Очищает flash-память считывателя для последующей записи блоков данных и ключей.

Время выполнения команды - 250мс.

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

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.15 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.

4.2.16 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.

4.2.17 CLSCRF_DirectIO_Reader

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

Отправляет на считыватель [команду управления](#).

Функция позволяет программисту самостоятельно формировать [команду](#) на считыватель, без вызова специализированных функций библиотеки.

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

pbSendBuffer – массив байтов, передаваемых на считыватель;

dwSendLength – количество передаваемых на считыватель байтов;

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 ответа.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.2.18 CLSCRF_WriteReaderConfig

```
LONG CLSCRF_WriteReaderConfig( IN LPVOID pReader,
                               IN BYTE ucMemType,
                               IN BYTE ucAddress,
                               IN BYTE ucLength,
                               IN LPBYTE pbValue );
```

Пишет в память считывателя по указанному адресу ряд байтов конфигурации.

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

ucMemType – тип памяти: 0x00 - RAM; 0x01 - Flash и RAM;

ucAddress – адрес начального байта конфигурации (0x00..0x07);

ucLength – длина записываемой последовательности байт конфигурации (0x01..0x08);

pbValue – указатель на массив данных, содержащий данные конфигурации.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

ucAddress:

Адрес	Длина	Название
0x00	1 байт	Rc663 14443A mks
0x01	1 байт	Rc663 14443B mks
0x02	1 байт	Rc663 15693 mks

0x03	1 байт	RFU (Rc663 18000P3M3 mks)
0x04	1 байт	RFU (Rc663 18092mPI mks)
0x05	1 байт	Char Timeout bytes

Параметры по адресам (Offset) 0x00..0x02 предназначены для настройки качества обмена данными с картами соответственно стандартов 14443-A, 14443-B, 15693.

По этим адресам расположены байты конфигурации, определяющие внутреннюю задержку считывателя при работе с картами соответствующих стандартов в микросекундах. По умолчанию они имеют значение 0xFF. Если оно задано, то считыватель устанавливает значение задержки автоматически, на основе определенного по различным признакам типа карты. Если значение задержки в конфигурации находится в диапазоне 0x01..0xFE (0x00 - резерв), то автоматическое определение отключается, и это значение будет использовано принудительно для всех операций с картой соответствующего типа. В случае, если карта считывается нестабильно, то, возможно, ручная регулировка данного параметра позволит уменьшить или убрать нестабильность.

Параметры по адресам 0x03..0x04 зарезервированы для использования в будущем.

Параметр по адресу 0x05 предназначен для настройки признака завершения кадра при обмене с хостом по интерфейсам RS232 и RS485.

Значение 0xFF для каждого параметра означает, что пользователь не менял этот параметр, а используется значение по умолчанию.

На стабильность обмена с картами влияет множество факторов, включая наличие металлических предметов, поверхностей вблизи считывателя, проводов, электромагнитных полей, качество источника питания. Кроме того, сами карты могут иметь дефекты. На предельно малых и больших расстояниях до считывателя карты могут читаться, либо записываться нестабильно. Хотя регулировка параметров внутренней задержки и может немного повлиять на качество обмена, однако первостепенным является влияние перечисленных выше факторов.

Признак завершения кадра (параметр 5) - это промежуток спокойного состояния шины, измеряемый в байтах (см. описание протокола).

По умолчанию это значение равно 3. Возможные значения для этого параметра лежат в пределах от 4 до 254.

4.2.19 CLSCRF_ReadReaderConfig

```
LONG CLSCRF_ReadReaderConfig( IN LPVOID pReader,
                              IN BYTE ucMemType,
                              IN BYTE ucAddress,
```


IN BYTE ucLength,
OUT LPBYTE pbValue);

Вычитывает из памяти считывателя по указанному адресу ряд байтов конфигурации.

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

ucMemType – тип памяти: 0x00 - RAM; 0x01 - Flash с записью в RAM;

ucAddress – адрес начального байта конфигурации (0x00..0x07);

ucLength – длина вычитываемой последовательности байт конфигурации (0x01..0x08);

pbValue – указатель на массив данных, в который будут вычитаны данные конфигурации.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.2.20 CLSCRF_ResetReaderConfig

LONG CLSCRF_ResetReaderConfig(IN LPVOID pReader,
IN BYTE ucMemType,
IN BYTE ucAddress,
IN BYTE ucLength);

Сбрасывает конфигурацию в считывателе по указанному адресу в значения по умолчанию.

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

ucMemType – тип памяти: 0x00 - RAM; 0x01 - Flash и RAM;

ucAddress – адрес начального байта конфигурации (0x00..0x07);

ucLength – длина сбрасываемой последовательности байт конфигурации (0x01..0x08);

pbValue – указатель на массив данных, в который будут вычитаны данные конфигурации.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.2.21 CLSCRF_ISO14443_ActivateEx

LONG CLSCRF_ISO14443_ActivateEx(IN LPVOID pReader,
IN BYTE ucCardStandard,
IN BYTE ucCID,
IN BYTE ucBaudrate,
IN BYTE ucRfOffTime,
IN BYTE ucRfOnPause,
IN BYTE ucDisableTCLSwitch,

IN BYTE ucAFI,
IN BYTE ucExtATQ,
IN BYTE ucSni,


```

OUT LPBYTE pbATQ,
OUT LPBYTE pucSAK,
OUT LPBYTE pbATS,
OUT LPBYTE pbUID,
IN OUT LPBYTE pucUIDLength,

```

```

OUT LPBYTE pucMBLI,
OUT LPBYTE pbATQB,
IN OUT LPBYTE pucATQBLength);

```

Активирует карты ISO 14443-A или -B из состояния IDLE и переводит их в режим T=CL.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRF Create](#));

ucCardStandard – используемый стандарт (ISO14443A - 0x00, ISO14443B - 0x01);

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;

ucRfOffTime – длительность [ms] отключения поля перед включением (0..127);

ucRfOnPause – дополнительная пауза [ms] после включения поля (0..127);

ucDisableTCLSwitch – не переходить в T=CL (0x00 - переходить, 0x01 - не переходить);

ucAfi – идентификатор семейства приложений (для ISO14443B);

ucExtATQ – использование расширенного ATQ (для ISO14443B, 0x00 - не использовать, 0x01 - использовать);

ucSni – код количества временных слотов:

0 =>	SlotQuantity = 1 слот;
1 =>	SlotQuantity = 2 слота;
2 =>	SlotQuantity = 4 слота;
3 =>	SlotQuantity = 8 слотов;
4 =>	SlotQuantity = 16 слотов;

pbATQ – ссылка на массив (2 байта), в которые будет помещен ATQ карты (для ISO14443A);

pucSAK – ссылка на массив (1 байт), в который будет помещен SAK карты (для ISO14443A);

pbATS – указатель на массив данных, в который будет записан прочитанный ATS

карты (для ISO14443A);

pbUID – ссылка на массив не менее 10 байтов, в который будет помещен уникальный идентификатор карты (UID для ISO14443A, PUPID для ISO14443B);

pucUIDLength – 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);

pucMBLI – ссылка на переменную, в которую будет помещено значение MBLI CID (для ISO14443B);

pbATQB – ссылка на массив, в который будет помещен ATQ карты (для ISO14443B):

- PUPID – 4 байта;
- AppData – 4 байта;
- ProtInfo – 3 или 4 байта;

pucATQBLength – ссылка на переменную, в которую будет помещен размер считанного ATQ карты (для ISO14443B).

Возвращаемое значение:

- 0 – успешное выполнение,
- иначе – ошибка при выполнении.

4.2.22 CLSCRIF_ISO14443_4_Exchange

```
LONG CLSCRIF_ISO14443_4_Exchange(      IN LPVOID pReader,
                                         IN DWORD dwOption,
                                         IN LPCBYTE pbSendBuffer,
                                         IN DWORD dwSendLength,
                                         OUT LPBYTE pbRecvBuffer,
                                         IN OUT LPDWORD pdwRecvLength );
```

Выполняет обмен данными с картой по протоколу T=CL.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRIF_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.

4.3 Конфигурация устройств на шине RS485

4.3.1 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.

4.3.2 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.

4.3.3 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.

4.3.4 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.

4.4 Управление защищенным режимом

4.4.1 CLSCRF_Crypto_SaveAESKeysToFile

LONG CLSCRF_Crypto_SaveAESKeysToFile(IN LPSTR IsFileName,
IN LPBYTE pbIV,
IN LPBYTE pbKeys,
IN DWORD dwKeysCount,
IN LPSTR IsPassword);

IsFileName– 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;

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

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.4.2 CLSCRF_Crypto_GenerateAndSaveAESKeysToFile

LONG CLSCRF_Crypto_GenerateAndSaveAESKeysToFile(IN LPSTR IsFileName,
IN DWORD dwKeysCount,
IN LPSTR IsPassword);

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

dwKeysCount – количество генерируемых и записываемых в файл ключей;

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

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.3 CLSCRF_Crypto_LoadAESKeysFromFile

```
LONG CLSCRF_Crypto_LoadAESKeysFromFile(      IN LPVOID pReader,
                                              IN LPSTR  IsFileName,
                                              IN LPSTR  IsPassword,
                                              OUT LPDWORD lIdKeysCount);
```

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

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

IsPassword – строка с паролем, которым зашифрован файл ключей (ASCII-строка, завершающаяся нулем);

IdKeysCount – количество ключей, которое требуется загрузить и количество по факту загруженных ключей после операции.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.4.4 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.

4.4.5 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.

4.4.6 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.

4.4.7 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.

4.5 Работа со списком карт

4.5.1 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.

4.5.2 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.

4.5.3 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.

4.5.4 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.

4.5.5 CLSCRF_Cards_GetCount

LONG CLSCRF_Cards_GetCount(OUT [int](#)* piCardsCount);

Возвращает количество карт в списке.

piCardsCount – указатель на переменную, принимающую значение количества.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.5.6 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.

4.5.7 CLSCRFL_Cards_SetCurrent

LONG CLSCRFL_Cards_SetCurrent(IN [CLSCRFL_CARD](#) * pCard);

Устанавливает карту, как текущую.

pCard — указатель на объект карты.

Возвращаемое значение:

0 — успешное выполнение,
иначе — ошибка при выполнении.

4.5.8 CLSCRFL_Cards_GetCurrent

LONG CLSCRFL_Cards_GetCurrent(OUT [CLSCRFL_CARD](#) ** ppCard);

Возвращает указатель на текущую карту.

ppCard — указатель на указатель на объект карты.

Возвращаемое значение:

0 — успешное выполнение,
иначе — ошибка при выполнении.

4.5.9 CLSCRFL_Cards_Search

LONG CLSCRFL_Cards_Search(IN int UID_Len,
IN PBYTE pUID,
OUT [CLSCRFL_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.

4.5.10 CLSCRFL_Cards_CreateNew_A

LONG CLSCRFL_Cards_CreateNew_A(IN BYTE bRfMode,
IN PBYTE pbATQ,
IN PBYTE pbSAK,
IN PBYTE pbUID,
IN DWORD UID_Len,
OUT [CLSCRFL_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.

4.5.11 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.

4.5.12 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.

4.5.13 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.

4.5.14 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 – атрибуты 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.

4.6 Управление картами типа A стандарта ISO 14443

4.6.1 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.

4.6.2 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.

4.6.3 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.

4.6.4 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.

4.6.5 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.

4.6.6 CLSCRF_ISO14443A_4_Activate

```
LONG CLSCRF_ISO14443A_4_Activate(    IN LPVOID pReader,
                                      IN BYTE ucCID,
                                      IN BYTE ucBaudrate,
                                      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);

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;

pbATS – указатель на массив данных, в который будет записан прочитанный ATS карты.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.7 Управление картами типа В стандарта ISO 14443

4.7.1 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.

4.7.2 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.

4.7.3 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.

4.7.4 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.

4.8 Управление метками стандарта ISO 15693

4.8.1 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.

4.8.2 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.

4.8.3 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.

4.8.4 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.

4.8.5 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.

4.9 Обмен данными с картами Mifare Classic

4.9.1 (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.

4.9.2 (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.

4.9.3 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 pnUID[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.

4.9.4 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.

4.9.5 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.

4.9.6 CLSCRF_MifareStandard_AuthE2Ex

```
LONG CLSCRF_MifareStandard_AuthE2Ex( IN LPVOID pReader,
                                       IN BYTE  bKeyType,
                                       IN LPBYTE pbUID,
                                       IN DWORD  dwE2RecNo,
```


IN DWORD dwSector);

Производит аутентификацию сектора ключом, находящимся в EEPROM считывателя.

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]);

dwE2RecNo – номер записи в EEPROM считывателя, из которой нужно взять ключ (0..15);

dwSector – номер аутентифицируемого сектора.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.9.7 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.

4.9.8 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.

4.9.9 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.

4.9.10 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.

4.9.11 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.

4.9.12 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.

4.9.13

CLSCRF_MifareStandard_EV1_SetLoadModulationType

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


Команда изменяет 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.

4.9.14 CLSCRF_MifareStandard_CustomExchange

```
LONG CLSCRF_MifareStandard_CustomExchange( IN  LPVOID pReader,
      IN  WORD wOption,
      IN  LPCBYTE pbSendBuffer,
      IN  DWORD dwSendLength,
      IN  BYTE bTxLastBitsCount,
      IN  DWORD dwTimeout,
      OUT LPBYTE pbRecvBuffer,
      IN OUT LPDWORD pdwRecvLength,
      OUT LPBYTE pRxLastBitsCount
    );
```

Осуществляет настраиваемый обмен данных с картой типа Mifare Classic.

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

wOption – настройки обмена данными с картой:

биты 0-3 – установить в 0;

бит 4 – флаг добавления контрольной суммы к запросу на карту
(высчитывается автоматически);

бит 5 – флаг ожидания контрольной суммы в ответе от карты
(высчитывается автоматически);

бит 6 – флаг использования бита четности при обмене (высчитывается
автоматически);

биты 7, 8-15 – установить в 0;

pbSendBuffer – массив байтов, передаваемых карте;

dwSendLength – количество передаваемых в карту байтов;

bTxLastBitsCount – количество значащих бит в завершающем байте запроса
(0..7, 0 - все биты значащие);

dwTimeout – величина таймаута карты в единицах (128 / 13,56 МГц) = около
9.439528 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 байт;

pRxLastBitsCount – количество значащих бит в завершающем байте ответа
(0..7, 0 - все биты значащие).

Возвращаемое значение:

0 – успешное выполнение,

иначе — ошибка при выполнении.

4.10 Обмен данными с картами Mifare Ultralight (C)

4.10.1 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.

4.10.2 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.

4.10.3 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.

4.10.4 CLSCRF_MifareUltralightC_Authenticate

LONG CLSCRF_MifareUltralightC_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.

4.11 Обмен данными с картами Mifare Ultralight EV1

4.11.1 CLSCRF_MifareUltralightEV1_GetVersion

LONG CLSCRF_MifareUltralightEV1_GetVersion(IN LPVOID pReader,
OUT LPBYTE pbVersion);

Вычитывает версию карты.

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

pbVersion – указатель на массив из 8 байт, куда будет записана вычитанная версия;

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.11.2 CLSCRF_MifareUltralightEV1_Read

LONG CLSCRF_MifareUltralightEV1_Read(IN LPVOID pReader,
IN BYTE ucPageAddress,
OUT LPBYTE pData);

Читает 4 страницы (16 байтов), начиная с заданной страницы.

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

ucPageAddress – номер читаемой страницы;

pData – массив (не менее 16 байтов) для прочитанной страницы.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.11.3 CLSCRF_MifareUltralightEV1_ReadFast

```
LONG CLSCRF_MifareUltralightEV1_ReadFast(      IN LPVOID pReader,  
                                                IN BYTE ucPageStart,  
                                                IN BYTE ucPageEnd,  
                                                IN OUT PDWORD pdwSizeRead,  
                                                OUT LPBYTE pData );
```

Записывает 4 байта в заданную страницу.

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

ucPageStart – номер начальной читаемой страницы;

ucPageEnd – номер конечной читаемой страницы;

pdwSizeRead – при вызове функции - размер буфера pData; при выходе из функции - количество прочитанных байт;

pData – массив (размером не менее 4 * (ucPageEnd - ucPageStart + 1) байтов) для прочитанных страниц.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.11.4 CLSCRF_MifareUltralightEV1_Write

```
LONG CLSCRF_MifareUltralightEV1_Write(  IN LPVOID pReader,  
                                           IN BYTE ucPageAddress,  
                                           IN LPBYTE pData );
```

Записывает 4 байта в заданную страницу.

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

ucPageAddress – номер записываемой страницы;

pData – массив 4 байтов данных, записываемых в страницу.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.11.5 CLSCRF_MifareUltralightEV1_AuthenticatePassword

```
LONG CLSCRF_MifareUltralightEV1_AuthenticatePassword( IN LPVOID pReader,  
                                                         IN LPBYTE pPassword,  
                                                         OUT LPBYTE pPasswordAcknowledge );
```

Производит аутентификацию карты по указанному ключу.

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

pPassword – указатель на массив, содержащий пароль (4 байта);

pPasswordAcknowledge – указатель на массив из 2 байт, куда будет записано подтверждение пароля.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.11.6 CLSCRF_MifareUltralightEV1_ReadCounter

```
LONG CLSCRF_MifareUltralightEV1_ReadCounter( IN LPVOID pReader,  
                                              IN BYTE ucCounterNumber,  
                                              OUT PDWORD pdwCounterValue  
                                              );
```

Записывает 4 байта в заданную страницу.

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

ucCounterNumber – номер счетчика 0..2;

pdwCounterValue – указатель на переменную типа DWORD, в которую будет записано вычитанное значение счетчика.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.7 CLSCRF_MifareUltralightEV1_IncrementCounter

```
LONG CLSCRF_MifareUltralightEV1_IncrementCounter( IN LPVOID pReader,  
                                                  IN BYTE ucCounterNumber,  
                                                  IN DWORD dwCounterValue );
```

Записывает 4 байта в заданную страницу.

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

ucCounterNumber – номер счетчика 0..2;

dwCounterValue – добавляемое значение (только 3 наименее значимых байта будут использованы).

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.8 CLSCRF_MifareUltralightEV1_ReadSignature

```
LONG CLSCRF_MifareUltralightEV1_ReadSignature( IN LPVOID pReader,  
                                              IN BYTE ucAddr,  
                                              OUT LPBYTE pSignature );
```

Записывает 4 байта в заданную страницу.

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

ucAddr – зарезервировано, должно быть = 0x00;

pSignature – массив из 32 байт для прочитанной подписи.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.9 CLSCRF_MifareUltralightEV1_CheckTearingEvent

```
LONG CLSCRF_MifareUltralightEV1_CheckTearingEvent(    IN LPVOID pReader,  
                                                       IN BYTE ucCounterNumber,  
                                                       OUT PBYTE pucValidFlagByte );
```

Записывает 4 байта в заданную страницу.

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

ucCounterNumber – номер счетчика 0..2;

pucValidFlagBytes – указатель на переменную типа BYTE, куда будет записан valid flag (для нормальной работы должен быть 0xBD).

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.11.10 CLSCRF_MifareUltralightEV1_VirtualCardSupportLast

```
LONG CLSCRF_MifareUltralightEV1_VirtualCardSupportLast(  
    IN LPVOID pReader,  
    IN LPBYTE pIID,  
    IN LPBYTE pPCDCAPS,  
    OUT LPBYTE pucVirtualCardTypeIdentifier );
```

Записывает 4 байта в заданную страницу.

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

bIID – идентификатор инфраструктуры (16 байт);

bPCDCAPS – блок характеристик считывателя (4 байта);

pucVirtualCardTypeIdentifier – идентификатор типа виртуальной карты (1 байт).

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.12 Обмен данными с метками стандарта ISO 15693

4.12.1 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.

4.12.2 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.

4.12.3 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.

4.12.4 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.

4.12.5 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.

4.12.6 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.

4.12.7 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.

4.12.8 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.

4.12.9 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.

4.12.10 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.

4.12.11 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.

4.12.12 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.

4.12.13 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.

4.12.14 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);
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);
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.

4.12.15 CLSCRF_15693_ICode_GetRandomNumber

```
LONG CLSCRF_15693_ICode_GetRandomNumber( IN LPVOID pReader,
                                           IN BYTE bFlags,
                                           IN BYTE bICMfgCode,
                                           IN LPBYTE pbUID,
```



```

OUT LPDWORD   pdwRandomNumber,
OUT LPBYTE    pbFlags,
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);

bICMfgCode – код производителя микросхемы метки;

pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;

pdwRandomNumber – ссылка на переменную (4 байта), в которую будет помещен прочитанный случайный номер;

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.

4.12.16 CLSCRF_15693_ICode_SetPassword

```

LONG CLSCRF_15693_ICode_SetPassword(IN LPVOID pReader,
                                     IN BYTE   bFlags,
                                     IN BYTE   bICMfgCode,
                                     IN LPBYTE pbUID,
                                     IN BYTE   ucPasswordIdentifier,
                                     IN DWORD   dwPassword,
                                     IN DWORD   dwRandomNumber,
                                     OUT LPBYTE pbFlags,
                                     OUT LPBYTE pbErrorCode);

```

Передача на карту текущего пароля (только ICode SLIX/SLIX2).

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

bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);

bICMfgCode – код производителя микросхемы метки;

pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;

ucPasswordIdentifier – идентификатор пароля: 01h - Read, 02h - Write, 04h - Privacy, 08h - Destroy, 10h - EAS / AFI;

dwPassword – пароль, 32 бита;

dwRandomNumber – случайное число, 16 бит, полученное командой GetRandomNumber;

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.

4.12.17 CLSCRF_15693_ICode_WritePassword

```
LONG CLSCRF_15693_ICode_WritePassword(IN LPVOID pReader,
                                         IN BYTE bFlags,
                                         IN BYTE bICMfgCode,
                                         IN LPBYTE pbUID,
                                         IN BYTE ucPasswordIdentifier,
                                         IN DWORD dwPassword,
                                         OUT LPBYTE pbFlags,
                                         OUT LPBYTE pbErrorCode);
```

Изменить пароль на карте (только ICode SLIX/SLIX2).

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

bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);

bICMfgCode – код производителя микросхемы метки;

pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;

ucPasswordIdentifier – идентификатор пароля: 01h - Read, 02h - Write, 04h - Privacy, 08h - Destroy, 10h - EAS / AFI;

dwPassword – пароль, 32 бита;

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.

4.12.18 CLSCRF_15693_ICode_LockPassword

```
LONG CLSCRF_15693_ICode_LockPassword(IN LPVOID pReader,
                                         IN BYTE bFlags,
                                         IN BYTE bICMfgCode,
                                         IN LPBYTE pbUID,
                                         IN BYTE ucPasswordIdentifier,
                                         OUT LPBYTE pbFlags,
                                         OUT LPBYTE pbErrorCode);
```

Запретить изменять пароль на карте (только ICode SLIX/SLIX2).

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

bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);

bICMfgCode – код производителя микросхемы метки;

pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;

ucPasswordIdentifier – идентификатор пароля: 01h - Read, 02h - Write, 04h - Privacy, 08h - Destroy, 10h - EAS / 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.

4.12.19 CLSCRF_15693_ICode_PasswordProtectEAS_AFI

```
LONG CLSCRF_15693_ICode_PasswordProtectEAS_AFI(IN LPVOID pReader,
                                                IN BYTE bFlags,
                                                IN BYTE bICMfgCode,
                                                IN LPBYTE pbUID,
                                                OUT LPBYTE pbFlags,
                                                OUT LPBYTE pbErrorCode);
```

Включить парольную защиту для EAS/AFI (только ICode SLIX).

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

bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);

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.

4.12.20 CLSCRF_15693_ICode_64BitProtection

```
LONG CLSCRF_15693_ICode_64BitProtection(IN LPVOID pReader,
                                          IN BYTE bFlags,
                                          IN BYTE bICMfgCode,
                                          IN LPBYTE pbUID,
                                          OUT LPBYTE pbFlags,
                                          OUT LPBYTE pbErrorCode);
```

Включает защиту доступа по двойному паролю (на чтение и на запись) (только ICode SLIX2).

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

bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);

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.

4.12.21 CLSCRF_15693_ICode_ProtectPage

```
LONG CLSCRF_15693_ICode_ProtectPage(IN LPVOID pReader,
                                     IN BYTE bFlags,
                                     IN BYTE bICMfgCode,
                                     IN LPBYTE pbUID,
                                     IN BYTE
                                     ucProtectionPointerAddress,
                                     IN BYTE
                                     ucExtendedProtectionStatus,
                                     OUT LPBYTE pbFlags,
                                     OUT LPBYTE pbErrorCode);
```

Включает защиту доступа к странице данных (только ICode SLIX2).

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRF Create](#));

bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);

bICMfgCode – код производителя микросхемы метки;

pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;

ucProtectionPointerAddress – начальный адрес страницы H;

ucExtendedProtectionStatus – расширенные параметры защиты (см. Table 27. Extended Protection status byte);

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.

4.12.22 CLSCRF_15693_ICode_LockPageProtectionCondition

```
LONG CLSCRF_15693_ICode_LockPageProtectionCondition(
                                     IN LPVOID pReader,
                                     IN BYTE bFlags,
                                     IN BYTE bICMfgCode,
                                     IN LPBYTE pbUID,
                                     IN                                     BYTE
                                     ucProtectionPointerAddress,
```


OUT LPBYTE pbFlags,
OUT LPBYTE pbErrorCode);

Блокирует изменение параметров защиты доступа к странице данных (только ICode SLIX2).

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

bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);

bICMfgCode – код производителя микросхемы метки;

pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;

ucProtectionPointerAddress – начальный адрес страницы H;

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.

4.12.23 CLSCRF_15693_ICode_Destroy

```
LONG CLSCRF_15693_ICode_Destroy(IN    LPVOID    pReader,
                                   IN    BYTE      bFlags,
                                   IN    BYTE      bICMfgCode,
                                   IN    LPBYTE     pbUID,
                                   IN    DWORD      dwPassword,
                                   IN    DWORD      dwRandomNumber,
                                   OUT   LPBYTE     pbFlags,
                                   OUT   LPBYTE     pbErrorCode);
```

Выводит из строя карту навсегда (только ICode SLIX2).

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

bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);

bICMfgCode – код производителя микросхемы метки;

pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;

dwPassword – пароль, 32 бита;

pdwRandomNumber – ссылка на переменную (4 байта), в которую будет помещен прочитанный случайный номер;

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.

4.12.24 CLSCRF_15693_ICode_EnablePrivacy

```

LONG CLSCRF_15693_ICode_EnablePrivacy(IN    LPVOID    pReader,
                                         IN    BYTE     bFlags,
                                         IN    BYTE     bICMfgCode,
                                         IN    LPBYTE    pbUID,
                                         IN    DWORD     dwPassword,
                                         IN    DWORD     dwRandomNumber,
                                         OUT   LPBYTE    pbFlags,
                                         OUT   LPBYTE    pbErrorCode);

```

Активирует на карте приватный режим (только ICode SLIX2).

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

bFlags – флаги запроса (см. ISO 15693-3 п.7.3.1);

bICMfgCode – код производителя микросхемы метки;

pbUID – ссылка на массив (8 байтов), в котором находится уникальный идентификатор метки;

dwPassword – пароль, 32 бита;

pdwRandomNumber – ссылка на переменную (4 байта), в которую будет помещен прочитанный случайный номер;

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.

4.13 Обмен данными с картами Mifare Plus

4.13.1 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.

4.13.2 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.

4.13.3 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 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.

4.13.4 CLSCRF_MifarePlus_AuthenticateDirect

LONG CLSCRF_MifarePlus_AuthenticateDirect(IN LPVOID pReader,
IN BYTE ucAuthType,
IN BYTE ucKeyType,
IN BYTE ucSectorNumber,

IN LPBYTE pbKey,
IN BYTE ucLenCap,
IN LPBYTE pbPcdCap);

Выполняет 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;

pbKey – указатель на байтовый массив - ключ AES-128 для аутентификации (16 байт);

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.

4.13.5 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 на байтовый массив, в который будут прочитаны данные.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.13.6 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 на байтовый массив с данными для записи.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.7 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 прочесть (1..3);

pbData – указатель на байтовый массив, в который будут прочитаны данные.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.13.8 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.

4.13.9 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.

4.13.10 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.

4.13.11 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.

4.13.12 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.

4.13.13 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.

4.13.14 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.

4.13.15 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.

4.13.16 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 характеристики 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 дополнительными байтами (понадобятся в функции [CLSCRF_MifarePlus_VirtualCardSelect](#)).

Возвращаемое значение:

0 – успешное выполнение,
 иначе – ошибка при выполнении.

4.13.17 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, дополненный до 13 байт дополнительными байтами, полученными в результате выполнения функции [CLSCRF_MifarePlus_VirtualCardSupportLast](#) .

Возвращаемое значение:

0 – успешное выполнение,
 иначе – ошибка при выполнении.

4.13.18 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.

4.13.19 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.

4.14 Обмен данными с картами Mifare Plus EV1

4.14.1 CLSCRF_MifarePlusEV1_WritePerso

LONG CLSCRF_MifarePlusEV1_WritePerso(IN LPVOID pReader,
IN BYTE ucT_CL,
IN BYTE ucValueType,
IN BYTE ucSectorNumber,
IN BYTE ucBlockNumber,
IN BYTE ucBlocksCount,
IN LPBYTE pData);

Записывает данные персонализации в карту. Данные задаются явно.

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

ucT_CL – 0x00: ISO14443-3 protocol; 0x01: ISO14443-4 protocol;

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);

ucBlocksCount – количество записываемых блоков (1..15);

pData – данные блока для записи;

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.2 CLSCRF_MifarePlusEV1_CommitPerso

```
LONG CLSCRF_MifarePlusEV1_CommitPerso(      IN LPVOID pReader,
                                              IN BYTE ucOption,
                                              IN BYTE ucT_CL );
```

Завершает персонализацию карты.

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

ucOption – 0x00 or 0x01: перевести в SL1; 0x03: перевести в SL3;

ucT_CL – 0x00: ISO14443-3 protocol; 0x01: ISO14443-4 protocol.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.3 CLSCRF_MifarePlusEV1_Authenticate

```
LONG CLSCRF_MifarePlusEV1_Authenticate( IN LPVOID pReader,
                                          IN BYTE ucSecLevel,
                                          IN BYTE ucT_CL,
                                          IN BYTE ucAuthType,
                                          IN BYTE ucKeyType,
                                          IN BYTE ucSectorNumber,
                                          IN DWORD dwKeyFlashAddress,
                                          IN BYTE ucLenDIV,
                                          IN LPBYTE pbDIV,
                                          IN BYTE ucLenCap,
                                          IN OUT LPBYTE pbPcdCap,
                                          OUT LPBYTE pbRetPiccCap);
```

Выполняет первичную аутентификацию карты.

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

ucSecLevel – Security Level: 0x00, 0x01 или 0x03;

ucT_CL – 0x00: ISO14443-3 protocol; 0x01: ISO14443-4 protocol;

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;

ucLenDIV – длина вектора диверсификации;

pbDIV – вектор диверсификации;

ucLenCap – длина блока характеристик считывателя (1..6, установить минимум в 1);

pbPcdCap – указатель на байтовый массив - блок характеристик считывателя (например, задать 0x01);

pbRetPiccCap – блок характеристик карты (указатель на выделенную память 6 байт, не константную).

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.4 CLSCRF_MifarePlusEV1_AuthenticateDirect

```
LONG CLSCRF_MifarePlusEV1_AuthenticateDirect( IN LPVOID pReader,
                                              IN BYTE ucSecLevel,
                                              IN BYTE ucT_CL,
                                              IN BYTE ucAuthType,
                                              IN BYTE ucKeyType,
                                              IN BYTE ucSectorNumber,
                                              IN LPBYTE pbKey,
                                              IN BYTE ucLenDIV,
                                              IN LPBYTE pbDIV,
                                              IN BYTE ucLenCap,
                                              IN OUT LPBYTE pbPcdCap,
                                              OUT LPBYTE pbRetPiccCap);
```

Выполняет первичную аутентификацию карты.

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

ucSecLevel – Security Level: 0x00, 0x01 или 0x03;

ucT_CL – 0x00: ISO14443-3 protocol; 0x01: ISO14443-4 protocol;

ucAuthType – тип операции аутентификации (0x01 – первичная, 0x0F – последующая, 0x00 – сброс аутентификации);

ucKeyType – тип ключа для аутентификации (см. [таблицу](#));

ucSectorNumber – номер сектора для аутентификации;

pbKey – указатель на байтовый массив - ключ AES-128 для аутентификации (16 байт);

ucLenDIV – длина вектора диверсификации;

pbDIV – вектор диверсификации;

ucLenCap – длина блока характеристик считывателя (1..6, установить минимум в 1);

pbPcdCap – указатель на байтовый массив - блок характеристик считывателя (например, задать 0x01);

pbRetPiccCap – блок характеристик карты (указатель на выделенную память 6 байт, не константную).

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.5 CLSCRF_MifarePlusEV1_ReadData

```
LONG CLSCRF_MifarePlusEV1_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 прочесть (1..3);

pbData – указатель на байтовый массив, в который будут прочитаны данные.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.6 CLSCRF_MifarePlusEV1_WriteData

```
LONG CLSCRF_MifarePlusEV1_WriteData(    IN LPVOID pReader,
                                         IN BYTE ucEncryptionMode,
                                         IN BYTE ucValueType,
                                         IN BYTE ucSectorNumber,
                                         IN BYTE ucBlockNumber,
                                         IN BYTE ucBlocksCount,
                                         IN LPBYTE pbData,
                                         OUT PDWORD pdwMACCounter,
                                         OUT PBYTE pMACValue );
```

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;

pdwMACCounter – указатель на переменную типа DWORD, куда будет скопирован прочитанный счетчик MAC;

pMACValue – указатель на массив из 8 байт, куда будет скопировано значение MAC.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.7 CLSCRF_MifarePlusEV1_Increment

LONG CLSCRF_MifarePlusEV1_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.

4.14.8 CLSCRF_MifarePlusEV1_Decrement

LONG CLSCRF_MifarePlusEV1_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.

4.14.9 CLSCRF_MifarePlusEV1_Transfer

LONG CLSCRF_MifarePlusEV1_Transfer(IN LPVOID pReader,
 IN BYTE ucEncryptionMode,
 IN BYTE ucDestinationSectorNumber,
 IN BYTE ucDestinationBlockNumber,


```
OUT PDWORD pdwMACCounter,
OUT PBYTE pMACValue );
```

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;

pdwMACCounter – указатель на переменную типа DWORD, куда будет скопирован прочитанный счетчик MAC;

pMACValue – указатель на массив из 8 байт, куда будет скопировано значение MAC.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.10 CLSCRF_MifarePlusEV1_IncrementTransfer

```
LONG CLSCRF_MifarePlusEV1_IncrementTransfer( IN LPVOID pReader,
IN BYTE ucEncryptionMode,
IN BYTE ucSourceSectorNumber,
IN BYTE ucSourceBlockNumber,
IN BYTE ucDestinationSectorNumber,
IN BYTE ucDestinationBlockNumber,
IN DWORD dwValue,
OUT PDWORD pdwMACCounter,
OUT PBYTE pMACValue );
```

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;

pdwMACCounter – указатель на переменную типа DWORD, куда будет скопирован прочитанный счетчик MAC;

pMACValue – указатель на массив из 8 байт, куда будет скопировано значение MAC.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.11 CLSCRF_MifarePlusEV1_DecrementTransfer

```
LONG CLSCRF_MifarePlusEV1_DecrementTransfer( IN LPVOID pReader,  
                                              IN BYTE ucEncryptionMode,  
                                              IN BYTE ucSourceSectorNumber,  
                                              IN BYTE ucSourceBlockNumber,  
                                              IN BYTE ucDestinationSectorNumber,  
                                              IN BYTE ucDestinationBlockNumber,  
                                              IN DWORD dwValue,  
                                              OUT PDWORD pdwMACCounter,  
                                              OUT PBYTE pMACValue );
```

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;

pdwMACCounter – указатель на переменную типа DWORD, куда будет скопирован прочитанный счетчик MAC;

pMACValue – указатель на массив из 8 байт, куда будет скопировано значение MAC.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.12 CLSCRF_MifarePlusEV1_Restore

```
LONG CLSCRF_MifarePlusEV1_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.

4.14.13 CLSCRF_MifarePlusEV1_VirtualCardSupportLastISOL3

```
LONG CLSCRF_MifarePlusEV1_VirtualCardSupportLastISOL3(
    IN LPVOID pReader,
    IN LPBYTE pIID,
    IN LPBYTE pbPcdCapL3,
    OUT PBYTE pucInfo );
```

Передает карте последний идентификатор инфраструктуры, а также собственные характеристики и принимает характеристики карты.

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

pIID – указатель на идентификатор инфраструктуры (16 байт);

pbPcdCapL3 – блок характеристик считывателя;

pucInfo – указатель на байт, в который будет записана информация о карте (0x83 – 4 байт UID, 0x03 – 7 байт UID).

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.14 CLSCRF_MifarePlusEV1_ProximityCheck

```
LONG CLSCRF_MifarePlusEV1_ProximityCheck(
    IN LPVOID pReader,
    IN BYTE ucSteps,
    IN DWORD dwPcKeyNr,
    IN BYTE ucGenRndC,
    IN LPBYTE pbRndC );
```

Выполняет проверку релейной атаки.

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

ucSteps – количество циклов ProximityCheck, должно быть равно 0x01;

dwPcKeyNr – адрес ключа Proximity Check Key во flash-памяти считывателя (LSB);

ucGenRndC – 0x00: RndC включен в команду; 0x01: генерировать RndC;

pbRndC – произвольный RndC (только если GenRndC == 0x00).

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.15 CLSCRF_MifarePlusEV1_CommitReaderID

```
LONG CLSCRF_MifarePlusEV1_CommitReaderID(
    IN LPVOID pReader,
    IN BYTE ucSectorNumber,
    IN BYTE ucBlockNumber,
    IN BYTE ucTMACNumber,
    IN
        DWORD
        dwFlashKeyNumber,
    OUT LPBYTE pbPrevTMRI
);
```


Передает карте последний идентификатор инфраструктуры, а также собственные характеристики и принимает характеристики карты.

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

ucSectorNumber – номер сектора;

ucBlockNumber – номер блока в секторе;

ucTMACNumber – номер кластера TMAC в карте (1..4);

dwFlashKeyNumber – номер блока во флеш считывателя, где содержится ключ вычисления TMAC;

pbPrevTMRI – указатель на массив из 16 байт, куда будет помещен вычитанный из карты расшифрованный Transaction MAC Reader ID предыдущей операции.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.16

CLSCRF_MifarePlusEV1_ISOSelectExternalAuthenticate

```
LONG CLSCRF_MifarePlusEV1_ISOSelectExternalAuthenticate(
    IN LPVOID pReader,
    IN DWORD dwEncKeyNr,
    IN DWORD dwMacKeyNr,
    IN BYTE ucLenIID,
    IN LPBYTE pbIID,
    IN OUT LPDWORD pdwVCDDataLength,
    OUT LPBYTE pbVCDData );
```

Передает карте последний идентификатор инфраструктуры, а также собственные характеристики и принимает характеристики карты.

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

dwEncKeyNr – адрес ключа VC Polling ENC Key во flash-памяти считывателя;

dwMacKeyNr – адрес ключа VC Polling MAC Key во flash-памяти считывателя;

ucLenIID – длина IID;

pbIID – указатель на идентификатор Installation Identifier (DF Name or ISO application identifier) (16 байт);

pdwVCDDataLength – указатель на переменную типа DWORD, в которой при вызове функции должно храниться значение, соответствующее размеру буфера pbVCDData, а после успешного выполнения функции будет записано количество прочитанных из карты байт VCDData;

pbVCDData – данные виртуальной карты (16 байт).

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.17 CLSCRF_MifarePlusEV1_ISOSelect

```
LONG CLSCRF_MifarePlusEV1_ISOSelect(    IN LPVOID pReader,
                                         IN DWORD dwEncKeyNr,
                                         IN BYTE ucLenIID,
                                         IN LPBYTE pbIID,
                                         IN OUT LPDWORD pdwVCDataLength,
                                         OUT LPBYTE pbVCData );
```

Передает карте последний идентификатор инфраструктуры, а также собственные характеристики и принимает характеристики карты.

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

dwEncKeyNr – адрес ключа VC Polling ENC Key во flash-памяти считывателя;

ucLenIID – длина IID;

pbIID – указатель на идентификатор Installation Identifier (DF Name или идентификатор приложения ISO) (16 байт);

pdwVCDataLength – указатель на переменную типа DWORD, в которой при вызове функции должно храниться значение, соответствующее размеру буфера pbVCData, а после успешного выполнения функции будет записано количество прочитанных из карты байт VCData;

pbVCData – данные виртуальной карты (*pdwVCDataLength байт).

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.14.18 CLSCRF_MifarePlusEV1_AuthenticateSectorSwitch

```
LONG CLSCRF_MifarePlusEV1_AuthenticateSectorSwitch( IN LPVOID pReader,
                                                      IN BYTE ucL1L3,
                                                      IN DWORD dwSSKeyNr,
                                                      IN BYTE ucSector,
                                                      IN DWORD dwSKeyBNr,
                                                      IN BYTE ucLSS,
                                                      IN LPBYTE pbDISS,
                                                      IN BYTE ucLSK,
                                                      IN LPBYTE pbDISK );
```

Передает карте последний идентификатор инфраструктуры, а также собственные характеристики и принимает характеристики карты.

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

ucL1L3 – 0x01: L3SectorSwitch; 0x02: L1L3MixSectorSwitch;

dwSSKeyNr – номер ключа SectorSwitchKey во флеш-памяти считывателя (LSB);

ucSector – номер переключаемого сектора;

dwSKeyBNr – номер ключа SectorKeyB во флеш-памяти считывателя (LSB);

ucLSS – длина вектора диверсификации для ключа SectorSwitchKey;

pbDISS – вектор диверсификации для ключа SectorSwitchKey;

ucLSK – длина вектора диверсификации для ключа SectorKeyB;

pbDISK – вектор диверсификации для ключа SectorKeyB.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.19 CLSCRF_MifarePlusEV1_MF_PersonalizeUID

LONG CLSCRF_MifarePlusEV1_MF_PersonalizeUID(IN LPVOID pReader,
IN BYTE ucUidType
);

Передает карте последний идентификатор инфраструктуры, а также собственные характеристики и принимает характеристики карты.

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

ucUidType – (см. MIFARE Classic EV1).

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.20 CLSCRF_MifarePlusEV1_SetConfigSL1

LONG CLSCRF_MifarePlusEV1_SetConfigSL1(IN LPVOID pReader,
IN BYTE ucT_CL_Disable);

Передает карте последний идентификатор инфраструктуры, а также собственные характеристики и принимает характеристики карты.

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

ucT_CL_Disable – отключение T=CL для карты в режиме SL1: 0x00 или 0x01.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.21 CLSCRF_MifarePlusEV1_GetVersion

LONG CLSCRF_MifarePlusEV1_GetVersion(IN LPVOID pReader,
OUT LPBYTE pbVerInfo);

Передает карте последний идентификатор инфраструктуры, а также собственные характеристики и принимает характеристики карты.

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

pbVerInfo – указатель на массив из 28 байт, куда будут записаны данные версии.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.14.22 CLSCRF_MifarePlusEV1_ReadSignature

```
LONG CLSCRF_MifarePlusEV1_ReadSignature(      IN LPVOID pReader,
                                                IN BYTE ucT_CL,
                                                IN BYTE ucAddr,
                                                OUT LPBYTE pbSignature );
```

Передает карте последний идентификатор инфраструктуры, а также собственные характеристики и принимает характеристики карты.

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

ucT_CL – 0x00: ISO14443-3 protocol; 0x01: ISO14443-4 protocol;

ucAddr – 0x00 (RFU);

pbSignature – указатель на массив из 56 байт, куда будут записаны данные подписи.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.15 Обмен данными с картами Mifare DES Fire (EV1)

4.15.1 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.

4.15.2 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.

4.15.3 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.

4.15.4 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.

4.15.5 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.

4.15.6 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.

4.15.7 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.

4.15.8 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.

4.15.9 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.

4.15.10 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 структуры [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.

4.15.11 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.

4.15.12 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.

4.15.13 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.

4.15.14 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.

4.15.15 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.

4.15.16 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.

4.15.17 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 который 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.

4.15.18 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 который 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.

4.15.19 CLSCRF_MifareDesFire_GetFileSettings

```
LONG CLSCRF_MifareDesFire_GetFileSettings(      IN LPVOID pReader,
                                                IN BYTE ucFileNumber,
                                                OUT CLSCRF\_DESFIRE\_FILE\_SETTINGS * pFileSettings );
```

Считывает параметры 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.

4.15.20 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.

4.15.21 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.

4.15.22 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.

4.15.23 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.

4.15.24 CLSCRF_MifareDesFire_CreateLinearRecordFile

```
LONG CLSCRF_MifareDesFire_CreateLinearRecordFile(    IN LPVOID pReader,
ucFileNumber,                                         IN BYTE
pISOFileID,                                           IN PBYTE
ucCommSettings,                                       IN BYTE
pAccessRights,                                       IN CLSCRF\_DESFIRE\_FILE\_ACCESS\_RIGHTS *
dwRecordSize,                                         IN DWORD
dwMaxNumOfRecords );                                IN          DWORD
```

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.

4.15.25 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.

4.15.26 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.

4.15.27 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.

4.15.28 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.

4.15.29 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.

4.15.30 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 тафтологию).

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.15.31 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 тафтологию).

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.15.32 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

тафталогии).

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.15.33 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.

4.15.34 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.

4.15.35 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.

4.15.36 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.

4.15.37 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.

4.15.38 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.

4.15.39 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.

4.15.40 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.

4.15.41 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.

```

4.15.42 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.

```

4.15.43 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.

4.15.44 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 – отключить шифрацию данных в команде GetValue..

4.15.45 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.

4.15.46

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.).

```

4.15.47

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).

```

4.15.48 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.

```


4.15.49 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 flsAESKey;
} 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;

flsAESKey – true, если ключ AES.

4.15.50

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).

4.16 Работа с SAM-модулем

Важно! Все функции работы с SAM-модулем возвращают в ответе 2 байта SW1, SW2 - статусы ответа от SAM-модуля.

Эти два байта следует вычитывать в случаях:

1) если общий код ответа функции равен SCARD_F_INTERNAL_ERROR (0x80100001), а код внутренней ошибки библиотеки (получается при помощи функции GetLastError) равен UEM_SAM_APDU_ERR (0xC8);

2) если использовалась функция/команда CLSCRF_SAM_APDU, общий код ответа функции равен SCARD_S_SUCCESS (0x00000000), и код внутренней ошибки библиотеки (получается при помощи функции GetLastError) равен UEM_SUCCESS (0x00).

4.16.1 CLSCRF_SAM_GetStatusDescription

```
LONG CLSCRF_SAM_GetStatusDescription(    IN BYTE ucSW1,
                                          IN BYTE ucSW2,
                                          OUT LPSTR pError);
```

Возвращает по байтам статуса SW1, SW2 строку с описанием ошибки общего характера.

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

ucSW1 – байт статуса 1;

ucSW2 – байт статуса 2;

pError – массив для записи строки с ответом (256 байт).

Возвращаемое значение:

0 – успешное выполнение,
 иначе – ошибка при выполнении.

4.16.2 CLSCRF_SAM_GetCurrentHolder

```
LONG CLSCRF_SAM_GetCurrentHolder(    IN LPVOID pReader,
                                      OUT LPBYTE pucNumHolder);
```

Возвращает номер рабочего холдера SAM-модуля или контактной карты.

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

pucNumHolder – указатель на байтовую переменную, в которую будет записано прочитанный номер холдера (1..4, 0 – если модуль/контактная карта не найден).

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.16.3 CLSCRF_SAM_SetCurrentHolder

```
LONG      CLSCRF_SAM_SetCurrentHolder(      IN LPVOID pReader,
                                              IN BYTE ucNumHolder );
```

Меняет рабочий холдер SAM-модуля / контактной карты.

Все последующие команды будут восприниматься именно этим модулем.

Состояние предыдущего рабочего модуля при этом не изменяется.

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

ucNumHolder – номер холдера от 1 до 4.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.16.4 CLSCRF_SAM_SetCurrentHolderGetATR

```
LONG      CLSCRF_SAM_SetCurrentHolderGetATR( IN LPVOID pReader,
                                              IN BYTE ucNumHolder,
                                              IN  DWORD
                                              dwATRBufSize,
                                              OUT PBYTE pATR,
                                              OUT PDWORD
                                              pdwATRSize);
```

Меняет рабочий холдер SAM-модуля / контактной карты.

Все последующие команды будут восприниматься именно этим модулем.

Состояние предыдущего рабочего модуля при этом не изменяется.

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

ucNumHolder – номер холдера от 1 до 4;

dwATRBufSize – размер массива, выделенного запрашивающим для записи ATR;

pATR – указатель на объявленный запрашивающим массив, куда должен быть записан ATR;

pdwATRSize – размер ATR, сохранённого в массиве.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.16.5 CLSCRF_SAM_SetCurrentHolderEx

```
LONG      CLSCRF_SAM_SetCurrentHolderEx(      IN LPVOID pReader,
                                              IN BYTE ucNumHolder,
                                              IN BYTE ucWT10,
                                              IN BYTE ucPPS1,
                                              IN BYTE ucDeactivate,
```


dwATRBufSize,

IN DWORD

OUT PBYTE pATR,
OUT PDWORD

pdwATRSize);

Меняет рабочий холдер SAM-модуля / контактной карты с расширенными опциями.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRF Create](#));

ucNumHolder – номер холдера от 1 до 4 (если 0, то все модули выключаются);

ucWT10 - если не равно 0xFF, то задать нестандартное время ожидания ATR в десятках миллисекунд (ISO 7816-3, п.8.1);

ucPPS1 - если не равно 0xFF, то задать новое значение PPS1, отличное от глобального байта TA1 (ISO 7816-3, п.9.2, п.8.3);

ucDeactivate - если > 0, то выключить предыдущую рабочую контактную карту;

dwATRBufSize – размер массива, выделенного запрашивающим для записи ATR, если 0, то ATR не запрашивать;

pATR – указатель на объявленный запрашивающим массив, куда должен быть записан ATR (можно NULL, если не запрашивать).

pdwATRSize – размер ATR, сохранённого в массиве (можно NULL, если не запрашивать).

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

При подаче питания все контактные карты активируются, и одна из них является рабочей.

Если значение ucNumHolder превышает количество имеющихся холдеров,

выдается ошибка.

Иначе, Если ucNumHolder > 0

Контактная карта в холдере NumHolder становится рабочей.

Все последующие команды будут восприниматься именно этой картой.

Переключение на холдер с выключенной картой занимает не менее 50 мс.

Переключение между активированными контактными картами занимает менее 1 мс.

Попытка переключиться на холдер с отсутствующей или неисправной контактной картой занимает время WT(ожидание ATR от карты) согласно ISO 7816 - 3, п.8.1.

Для выбора нестандартного WT служат ucWT10 и FlashParam[20]:

Если ucWT10 != 0xFF

WT = ucWT10 * 10 мс(до 2540 мс).

иначе(при `ucWT10 == 0xFF`)
 Если `FlashParam[20] = 0xFF`
 WT соответствует стандарту(ISO 7816 - 3, п.8.1)(около 750 мс).

иначе(при `FlashParam[20] < 0xFF`)
 WT = `FlashParam[20] * 10 мс`(до 2540 мс).

Производители некоторых контактных карт для повышения скорости обмена допускают выбор нестандартных параметров, не совпадающих с указанными в ATR.

Например, для SAM AV2 от NXP при следовании стандарту

PPS1 = 0x18 (fKHz = 4800, Baudrate = 154838 бод),

Однако, в нарушение стандарта, допустимы следующие значения :

PPS1 = 0x96 (fKHz = 4800, Baudrate = 300000 бод),

PPS1 = 0xD7 (fKHz = 12000, Baudrate = 375000 бод).

Для выбора нестандартных параметров служат `ucPPS1` и `FlashParam[21]`.

Если `ucPPS1 != 0xFF`

PPS1 = `ucPPS1`;

иначе(при `Opt_PPS1 = 0`)

Если `FlashParam[21] = 0xFF`

PPS1 = TA1

(для SAM AV2 TA1 = 0x18 (fKHz = 4800, Baudrate = 154838));

иначе(при `FlashParam[21] < 0xFF`)

PPS1 = `FlashParam[21]`.

Если `ucDeactivate > 0`

Предыдущая рабочая контактная карта(при наличии) выключается.

иначе(при `ucDeactivate == 0`)

Предыдущая рабочая контактная карта(при наличии) остается активированной.

Иначе (при `ucNumHolder = 0`)

Рабочей контактной карты не будет(все активированные контактные карты выключаются), значения битов 7..4 игнорируются.

4.16.6 CLSCRF_SAM_APDU

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

Отправляет в SAM-модуль команду в формате APDU.

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

pbSendBuffer – массив байтов, передаваемых модулю;

dwSendLength – количество передаваемых в модуль байтов;

pbRecvBuffer – массив для ответа от модуля;

pdwRecvLength – ссылка на переменную, которая перед вызовом функции должна содержать размер массива pbRecvBuffer, а на выходе будет содержать количество принятых от модуля байтов.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.16.7 CLSCRF_SAM_Lock

```
LONG          CLSCRF_SAM_Lock(    IN LPVOID pReader,
                                   IN BYTE ucLC,
                                   IN BYTE ucKeyKind,
                                   IN BYTE ucKeyNo,
                                   IN BYTE ucKeyVer,
                                   IN PBYTE pUnlockKey,
                                   IN BYTE ucUnlockKeyNo,
                                   IN BYTE ucUnlockKeyVer,
                                   OUT LPBYTE pucSW1,
                                   OUT LPBYTE pucSW2 );
```

Блокирует SAM-модуль.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRF Create](#));

ucLC – логический канал от 0 до 3;

ucKeyKind – род ключа:

0: TDEA - DESFire;

1: TDEA - ISO 10116 (CRC16);

2: MIFARE;

3: 3 Key TDEA - ISO 10116;

4: AES 128;

5: AES 192;

6: TDEA - ISO 10116 (CRC32);

ucKeyNo – номер ключа для выполнения операции;

ucKeyVer – версия ключа для выполнения операции;

pUnlockKey[24] – ключ для разблокировки

ucUnlockKeyNo – номер ключа для разблокирования;

ucUnlockKeyVer – версия ключа для разблокирования;

(Если NULL, то модуль блокируется без ключа);

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.16.8 CLSCRF_SAM_Unlock

```

LONG          CLSCRF_SAM_Unlock(  IN LPVOID pReader,
                                   IN BYTE ucLC,
                                   IN BYTE ucKeyKind,
                                   IN BYTE ucKeyNo,
                                   IN BYTE ucKeyVer,
                                   IN PBYTE pUnlockKey,
                                   OUT LPBYTE pucSW1,
                                   OUT LPBYTE pucSW2 );

```

Разблокирует SAM-модуль.

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

ucLC – логический канал от 0 до 3;

ucKeyKind – род ключа:

- 0: TDEA - DESFire;
- 1: TDEA - ISO 10116 (CRC16);
- 2: MIFARE;
- 3: 3 Key TDEA - ISO 10116;
- 4: AES 128;
- 5: AES 192;
- 6: TDEA - ISO 10116 (CRC32);

ucKeyNo – номер ключа для разблокирования;

ucKeyVer – версия ключа для разблокирования;

pUnlockKey[24] – ключ для разблокирования
(Если NULL, то модуль блокируется без ключа);

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

- 0 – успешное выполнение,
- иначе – ошибка при выполнении.

4.16.9 CLSCRF_SAM_AuthenticateHost

```

LONG          CLSCRF_SAM_AuthenticateHost(  IN LPVOID pReader,
                                              IN BYTE ucLC,
                                              IN BYTE ucHostMode,
                                              IN BYTE ucKeyKind,
                                              IN BYTE ucKeyNo,
                                              IN BYTE ucKeyVer,
                                              IN PBYTE pKey,
                                              OUT LPBYTE pucSW1,
                                              OUT LPBYTE pucSW2 );

```

Аутентификация SAM-модуля и хоста.

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

ucLC – логический канал от 0 до 3;

ucHostMode – режим аутентификации:

0: Plain;
 1: MAC Protection; (не поддерживается)
 2: Full Protection; (не поддерживается)
 3: RFU;

ucKeyKind - род ключа:

0: TDEA - DESFire;
 1: TDEA - ISO 10116(CRC16);
 2: MIFARE;
 3: 3 Key TDEA - ISO 10116;
 4: AES 128;
 5: AES 192;
 6: TDEA - ISO 10116(CRC32);

ucKeyNo – номер ключа для аутентификации;

ucKeyVer – версия ключа для аутентификации;

pKey[24] – ключ для аутентификации;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
 иначе – ошибка при выполнении.

4.16.10 CLSCRF_SAM_GetVersion

LONG CLSCRF_SAM_GetVersion(IN LPVOID pReader,
 IN BYTE ucLC,
 OUT SAM_VERSION* pVersion,
 OUT LPBYTE pucSW1,
 OUT LPBYTE pucSW2);

Чтение информации о версии SAM-модуля.

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

ucLC – логический канал от 0 до 3;

pVersion – информация о версии SAM-модуля;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
 иначе – ошибка при выполнении.

4.16.11 CLSCRF_SAM_SwitchToAV2Mode

LONG CLSCRF_SAM_SwitchToAV2Mode(IN LPVOID pReader,
 IN BYTE ucLC,
 IN LPBYTE
 pbNewPosAkey,
 IN BYTE
 ucNewPosAkeyVer,
 IN LPBYTE

4.16.12 CLSCRF SAM Reset

4.16.13 CLSCRF SAM KillAuth

© 2021 Акционерное общество “МикроЭМ”

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.16.14 CLSCRF_SAM_Halt_A

LONG CLSCRF_SAM_Halt_A(IN LPVOID pReader,
IN BYTE ucLC);

Выполняет мягкий сброс аутентификации SAM-модуля.

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

ucLC – логический канал от 0 до 3.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.17 Работа с контактными картами стандарта ISO7816

Важно! Все функции работы с SAM-модулем возвращают в ответе 2 байта SW1, SW2 - статусы ответа от SAM-модуля.

Эти два байта следует вычитывать в случаях:

1) если общий код ответа функции равен SCARD_F_INTERNAL_ERROR (0x80100001), а код внутренней ошибки библиотеки (получается при помощи функции GetLastError) равен UEM_SAM_APDU_ERR (0xC8);

2) если использовалась функция/команда CLSCRF_SAM_APDU, общий код ответа функции равен SCARD_S_SUCCESS (0x00000000), и код внутренней ошибки библиотеки (получается при помощи функции GetLastError) равен UEM_SUCCESS (0x00).

4.17.1 CLSCRF_ISO7816_GetCurrentHolder

[Аналогична команде SAM-модуля.](#)

4.17.2 CLSCRF_ISO7816_SetCurrentHolder

[Аналогична команде SAM-модуля.](#)

4.17.3 CLSCRF_ISO7816_SetCurrentHolderGetATR

[Аналогична команде SAM-модуля.](#)

4.17.4 CLSCRIF_ISO7816_SetCurrentHolderEx

[Аналогична команде SAM-модуля.](#)

4.17.5 CLSCRIF_ISO7816_HOLDERS_INFO

```
typedef union _CLSCRIF_ISO7816_HOLDERS_INFO
{
    struct
    {
        BYTE Count;
        BYTE Active;
        BYTE CurNo;
        BYTE WTI;
    } AllHolders;
    struct
    {
        struct
        {
            BYTE isActive : 1;
            BYTE isCurrent : 1;
            BYTE RFU : 6;
        };
        BYTE Prot;
        BYTE F_D;
        WORD fKHz;
        DWORD BR;
    } SpecifiedHolder;
    BYTE BinData[9];
} CLSCRIF_ISO7816_HOLDERS_INFO ;
```

- информация о холдерах контактных карт, возвращаемая командой [CLSCRIF_ISO7816_GetHoldersInfo](#):

Если в параметрах команды был задан номер холдера 0, то вычитывается информация обо всех холдерах, и актуальна структура pHoldersInfo->SpecifiedHolder

pHoldersInfo->SpecifiedHolder.Count- количество холдеров.

pHoldersInfo->SpecifiedHolder.Active- маска активированных контактных карт :

0x01 - контактная карта 1 активирована;

0x02 - контактная карта 2 активирована;

0x04 - контактная карта 3 активирована;

0x08 - контактная карта 4 активирована.

pHoldersInfo->SpecifiedHolder.CurNo- номер холдера рабочей контактной карты(0..Count).

pHoldersInfo->SpecifiedHolder.WTI- максимальное время

ожидания ATR в десятках мс.

Если в параметрах команды был задан номер холдера от 1 до 4, то вычитывается информация об указанном холдере, и актуальна структура `pHoldersInfo->SpecifiedHolder`

`pHoldersInfo->SpecifiedHolder.isActive` - контактная карта активирована.

`pHoldersInfo->SpecifiedHolder.isCurrent` - контактная карта является рабочей.

`pHoldersInfo->SpecifiedHolder.Prot` - протокол T = (0 или 1).

`pHoldersInfo->SpecifiedHolder.F_D` - параметры обмена (значение PPS1 из ISO 7816 - 3, п.9.2, п.8.3).

`pHoldersInfo->SpecifiedHolder.fKHz` - частота тактирования (кГц).

`pHoldersInfo->SpecifiedHolder.BR` - скорость обмена (бод).

4.17.6 CLSCRF_ISO7816_GetHoldersInfo

```
LONG    CLSCRF_ISO7816_GetHoldersInfo( IN LPVOID pReader,
                                         IN BYTE ucNumHolder,
                                         OUT
```

```
CLSCRF_ISO7816_HOLDERS_INFO * pHoldersInfo);
```

Запрашивает текущее состояние холдеров.

Важно! Работает только для считывателей с детектором контакта - с холдерами на отдельной плате!!!

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

ucNumHolder – номер холдера от 1 до 4, если нужно запросить информацию по конкретному холдеру, или 0 - если нужна информация по всем холдерам;

pHoldersInfo – указатель на размещенную в памяти заранее структуру типа [CLSCRF_ISO7816_HOLDERS_INFO](#), в которую будет помещен ответ.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.17.7 CLSCRF_ISO7816_APDU

[Аналогична команде SAM-модуля.](#)

4.17.8 CLSCRF_ISO7816_Reset

[Аналогична команде SAM-модуля.](#)

4.18 Работа с ключами SAM-модуля

4.18.1 CLSCRFL_SAM_GetKeyEntry

```

LONG CLSCRFL_SAM_GetKeyEntry(    IN LPVOID pReader,
                                IN BYTE ucLC,
                                IN BYTE ucKeyKind,
                                IN BYTE ucKeyNo,
                                OUT LPBYTE pucKeyAVersion,
                                OUT LPBYTE pucKeyBVersion,
                                OUT LPBYTE pucKeyCVersion,
                                OUT LPDWORD pdwDESFireAID,
                                OUT LPBYTE pucDESFireKeyNo,
                                OUT LPBYTE pucKeyNoCEK,
                                OUT LPBYTE pucKeyVCEK,
                                OUT LPBYTE pucRefNoKUC,
                                OUT SAM\_SETTINGS * pSettings,
                                OUT SAM\_EXT\_SETTINGS * pExtSettings,
                                OUT LPBYTE pucSW1,
                                OUT LPBYTE pucSW2 );

```

Вычитывает из SAM-модуля запись о ключе.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRFL_Create](#));

ucLC – логический канал от 0 до 3;

ucKeyKind – род ключа:

- 0: TDEA - DESFire;
- 1: TDEA - ISO 10116 (CRC16);
- 2: MIFARE;
- 3: 3 Key TDEA - ISO 10116;
- 4: AES 128;
- 5: AES 192;
- 6: TDEA - ISO 10116 (CRC32);

ucKeyNo – номер Mifare-ключа в SAM-модуле от 1 до 127;

pucKeyAVersion – считанная версия ключа A;

pucKeyBVersion – считанная версия ключа B;

pucKeyCVersion – считанная версия ключа C;

pdwDESFireAID – 3 байта соответствующего AID для карт DESFire;

pucDESFireKeyNo – соответствующий AID, номер ключа DESFire;

pucKeyNoCEK – номер ссылки на ключ смены записи;

pucKeyVCEK – версия ключа смены записи;

pucRefNoKUC – номер ссылки на счётчик использования ключа;

pSettings – настройки конфигурации для записи ключа;

pExtSettings – дополнительные настройки конфигурации для записи ключа;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.18.2 CLSCRF_SAM_ChangeKeyEntry

```
LONG CLSCRF_SAM_ChangeKeyEntry( IN LPVOID pReader,
                                IN BYTE ucLC,
                                IN BYTE ucKeyNo,
                                IN BYTE ucKeyKind,
                                IN LPBYTE pbKeyA,
                                IN LPBYTE pbKeyB,
                                IN LPBYTE pbKeyC,
                                IN DWORD dwDF_AID,
                                IN BYTE ucDF_KeyNr,
                                IN BYTE ucCEKNo,
                                IN BYTE ucCEKV,
                                IN BYTE ucKUC,
                                IN SAM SETTINGS * pSettings,
                                IN BYTE ucKeyVerA,
                                IN BYTE ucKeyVerB,
                                IN BYTE ucKeyVerC,
                                IN SAM EXT SETTINGS * pExtSettings,
                                IN BOOL fPlainKeyUpdate,
                                IN BOOL fOfflineKeyChange,
                                IN LPBYTE pbSAM_UID,
                                IN SAM NV PROGRAMMING MASK * pNVProgrammingMask,
                                OUT LPBYTE pucSW1,
                                OUT LPBYTE pucSW2 );
```

Изменяет в SAM-модуле запись о ключе.

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

ucLC – логический канал от 0 до 3;

ucKeyNo – номер Mifare-ключа в SAM-модуле от 1 до 127;

ucKeyKind – род ключа:

- 0: TDEA - DESFire;
- 1: TDEA - ISO 10116 (CRC16);
- 2: MIFARE;
- 3: 3 Key TDEA - ISO 10116;
- 4: AES 128;
- 5: AES 192;
- 6: TDEA - ISO 10116 (CRC32);

pbKeyA – ключ A;

pbKeyB – ключ B;

pbKeyC – ключ C;

pbDF_AID – 3 байта соответствующего AID для карт DESFire;

ucDF_KeyNr – соответствующий AID, номер ключа DESFire;

ucCEKNo – номер ссылки на ключ смены записи;

ucCEKV – версия ключа смены записи;
ucKUC – номер ссылки на счётчик использования ключа;
pSettings – настройки конфигурации для записи ключа;
ucKeyVerA – версия ключа A;
ucKeyVerB – версия ключа B;
ucKeyVerC – версия ключа C;
pExtSettings – дополнительные настройки конфигурации для записи ключа;
fPlainKeyUpdate – устанавливает, в открытом ли виде обновляется ключ (должно быть в TRUE);
fOfflineKeyChange – признак того, в каком режиме меняется ключ (Online/Offline) (должно быть = FALSE);
pbSAM_UID – UID SAM-модуля (должно быть NULL);
pNVProgrammingMask – маска программирования из flash-памяти;
pucSW1 – указатель на байт статуса;
pucSW2 – указатель на байт статуса.
 Возвращаемое значение:
 0 – успешное выполнение,
 иначе – ошибка при выполнении.

4.18.3 SAM_SETTINGS

```

typedef union _SAM_SETTINGS
{
    struct
    {
        WORD bAllowDumpSessionKey:1;
        WORD bAllowCryptoWithSecretKey:1;
        WORD bfKeepInitVector:1;
        WORD wKeyKind:3;
        WORD RFU:2;
        WORD bAV1HigherCmdSecLevel:1;
        WORD bDisableKeyEntry:1;
        WORD bAV1EnableHostAuth:1;
        WORD bDisableChangeKey:1;
        WORD bDisableDecryption:1;
        WORD bDisableEncryption:1;
        WORD bDisableVerifyMAC:1;
        WORD bDisableGenerateMAC:1;
    };
    struct
    {
        WORD :8;
        WORD bAV2EnableHostAuthKey:1;
        WORD :1;
        WORD bAV1AV2AllowLockUnlock:1;
        WORD :5;
    };
};
  
```



```
WORD wSettings;
} SAM_SETTINGS;
```

- блок настроек для ключа (описание см. в документации на SAM-модуль в пункте 9.3.1.10 "SET configuration settings for key entry"):

bAllowDumpSessionKey – флаг разрешения вывода ключа сессии / ключа Mifare;

bAllowCryptoWithSecretKey – флаг разрешения криптоопераций с секретным ключом;

bfKeepInitVector – флаг сохранения вектора инициализации;

wKeyKind – тип ключа: 000 - TDEA DesFire 4, 001 - TDEA ISO 10116 (16-bit CRC, 4-byte MAC), 010 - MIFARE, 011 - 3TDEA ISO 10116, 100 - AES 128, 101 - AES 192, 110 - TDEA ISO 10116 (32-bit CRC, 4-byte MAC);

bAV1HigherCmdSecLevel – включение более высокого уровня защиты команд; все команды защищены CMAC, либо шифрование (только для номера ключа 0x00; для остальных записей ключей - 0x00); см. документацию на SAM-модули,

bDisableKeyEntry – флаг отключения записи ключа;

bAV1EnableHostAuth – флаг включения аутентификации хоста после сброса (только для номера ключа 0x00; для других записей ключа - 0x00);

bDisableChangeKey – флаг отключения SAM_ChangeKey_PICC;

bDisableDecryption – флаг отключения расшифровки;

bDisableEncryption – флаг отключения шифрования;

bDisableVerifyMAC – флаг отключения проверки MAC;

bDisableGenerateMAC – флаг отключения генерации MAC;

bAV2EnableHostAuthKey – флаг включения аутентификации хоста ключом в режиме AV2;

bAV1AV2AllowLockUnlock – флаг разрешения выполнения операций блокировки/разблокировки ;

wSettings – маска настроек в виде двубайтового целого.

4.18.4 SAM_EXT_SETTINGS

```
typedef union _SAM_EXT_SETTINGS
{
```

```
    struct
```

```
    {
        BYTE iKeyClass:3;
        BYTE bAllowDumpSecretKey:1;
        BYTE bRestrictToDiversifiedUse:1;
        BYTE RFU:3;
    };
    BYTE ucExtSettings;
```

```
} SAM_EXT_SETTINGS;
```

- блок расширенных настроек для ключа. (описание см. в документации на SAM-модуль)

iKeyClass – ;

bAllowDumpSecretKey – ;

bRestrictToDiversifiedUse – ;
ucExtSettings – .

4.18.5 SAM_NV_PROGRAMMING_MASK

```
typedef union _SAM_NV_PROGRAMMING_MASK
{
    struct
    {
        BYTE fUpdateKeyA:1;
        BYTE fUpdateKeyB:1;
        BYTE fUpdateKeyC:1;
        BYTE fUpdateDFKeyNoAndAID:1;
        BYTE fUpdateKeyCEK_NumberAndVersion:1;
        BYTE fUpdateRefNoKUC:1;
        BYTE fUpdateSettings:1;
        BYTE fSendKeyVersionsSeparately:1;
    };
    BYTE ucNVProgramingMask;
} SAM_NV_PROGRAMMING_MASK;
```

- блок расширенных настроек для ключа. (описание см. в документации на SAM-модуль)

fUpdateKeyA – ;
fUpdateKeyB – ;
fUpdateKeyC – ;
fUpdateDFKeyNoAndAID – ;
fUpdateKeyCEK_NumberAndVersion – ;
fUpdateRefNoKUC – ;
fUpdateSettings – ;
fSendKeyVersionsSeparately – ;
ucNVProgramingMask – .

4.18.6 SAM_VERSION

```
typedef struct _SAM_VERSION
{
    struct
    {
        BYTE ucVendorID;
        BYTE ucType;
        BYTE ucSubType;
        BYTE ucMajorVersion;
        BYTE ucMinorVersion;
        BYTE ucStorageSize;
        BYTE ucProtocolType;
    } HardwareVersionInfo;
    struct
```



```

    {
        BYTE ucVendorID;
        BYTE ucType;
        BYTE ucSubType;
        BYTE ucMajorVersion;
        BYTE ucMinorVersion;
        BYTE ucStorageSize;
        BYTE ucProtocolType;
    } SoftwareVersionInfo;
    struct
    {
        BYTE UID[7];
        BYTE BatchNumber[5];
        BYTE ucProductionDay;
        BYTE ucProductionMonth;
        BYTE ucProductionYear;
        union
        {
            struct
            {
                BYTE RFU:3;
                BYTE fDisable_DESFirePICCKeyChange:1;
                BYTE fDisable_Decryption:1;
                BYTE fDisable_Encryption:1;
                BYTE fDisable_MAC_Verification:1;
                BYTE fDisable_MAC_Generation:1;
            };
            BYTE ucBits;
        } CryptoSettings;
    } ManufacturerData;
    BYTE ucMode;
} SAM_VERSION;
- информация о версии SAM-модуля. (описание см. в документации на SAM-модуль)
HardwareVersionInfo.ucVendorID – ;
HardwareVersionInfo.ucType – ;
HardwareVersionInfo.ucSubType – ;
HardwareVersionInfo.ucMajorVersion – ;
HardwareVersionInfo.ucMinorVersion – ;
HardwareVersionInfo.ucStorageSize – ;
HardwareVersionInfo.ucProtocolType – ;
SoftwareVersionInfo.ucVendorID – ;
SoftwareVersionInfo.ucType – ;
SoftwareVersionInfo.ucSubType – ;
SoftwareVersionInfo.ucMajorVersion – ;
SoftwareVersionInfo.ucMinorVersion – ;
SoftwareVersionInfo.ucStorageSize – ;
SoftwareVersionInfo.ucProtocolType – ;

```



```

ManufacturerData.UID – ;
ManufacturerData.BatchNumber – ;
ManufacturerData.ucProductionDay – ;
ManufacturerData.ucProductionMonth – ;
ManufacturerData.ucProductionYear – ;
ManufacturerData.CryptoSettings.ucBits – ;
ManufacturerData.CryptoSettings.fDisable_DESFirePICCKeyChange – ;
ManufacturerData.CryptoSettings.fDisable_Decryption – ;
ManufacturerData.CryptoSettings.fDisable_Encryption – ;
ManufacturerData.CryptoSettings.fDisable_MAC_Verification – ;
ManufacturerData.CryptoSettings.fDisable_MAC_Generation – ;
ucMode – .

```

4.19 Работа с картами Mifare Classic при помощи SAM-модуля

4.19.1 CLSCRF_SAM_MifareAuthenticate

```

LONG CLSCRF_SAM_MifareAuthenticate(    IN LPVOID pReader,
                                       IN BYTE ucLC,
                                       IN BYTE ucAuthType,
                                       IN LPBYTE pbUID,
                                       IN BYTE ucKeyNo,
                                       IN BYTE ucKeyVer,
                                       IN BYTE ucKeyType,
                                       IN DWORD dwSector,
                                       OUT LPBYTE pucSW1,
                                       OUT LPBYTE pucSW2 );

```

Выполняет аутентификацию карты Mifare Classic.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRF Create](#));

ucLC – логический канал от 0 до 3;

ucAuthType – тип аутентификации: 1 - First Authenticate, иначе - Following Authenticate;

pbUID[4] – уникальный номер карты;

ucKeyNo – номер Mifare-ключа в SAM-модуле от 1 до 127;

ucKeyVer – версия Mifare-ключа в SAM-модуле от 0 до 255;

ucKeyType – тип Mifare-ключа: 0x0a - KeyA, иначе KeyB;

dwSector – номер аутентифицируемого сектора;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.19.2 CLSCRF_SAM_MifareRead

```
LONG CLSCRF_SAM_MifareRead(    IN LPVOID pReader,  
                                IN BYTE ucLC,  
                                IN DWORD dwSector,  
                                IN DWORD dwBlock,  
                                OUT LPBYTE pbRecvBuffer,  
                                OUT LPBYTE pucSW1,  
                                OUT LPBYTE pucSW2 );
```

Читает блок карты Mifare Classic.

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

ucLC – логический канал от 0 до 3;

dwSector – номер сектора, содержащего читаемый блок;

dwBlock – номер читаемого блока;

pbRecvBuffer – массив (не менее 16 байтов) для прочитанного блока;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.19.3 CLSCRF_SAM_MifareWrite

```
LONG CLSCRF_SAM_MifareWrite(    IN LPVOID pReader,  
                                IN BYTE ucLC,  
                                IN DWORD dwSector,  
                                IN DWORD dwBlock,  
                                IN LPBYTE pbSendBuffer,  
                                OUT LPBYTE pucSW1,  
                                OUT LPBYTE pucSW2 );
```

Записывает блок карты Mifare Classic.

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

ucLC – логический канал от 0 до 3;

dwSector – номер сектора, содержащего записываемый блок;

dwBlock – номер записываемого блока;

pbSendBuffer – массив (не менее 16 байтов) данных записываемого блока;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.19.4 CLSCRF_SAM_MifareIncrement

```
LONG CLSCRF_SAM_MifareIncrement( IN LPVOID pReader,  
                                  IN BYTE ucLC,
```



```

        IN DWORD dwSector,
        IN DWORD dwSourceBlock,
        IN DWORD dwValue,
        IN DWORD dwTargetBlock,
        OUT LPBYTE pucSW1,
        OUT LPBYTE pucSW2 );

```

Увеличить значение блока типа Value.

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

ucLC – логический канал от 0 до 3;

dwSector – номер сектора, содержащего читаемый блок;

dwSourceBlock – номер исходного блока;

dwValue – слагаемое, на которое увеличивается значение;

dwTargetBlock – номер блока-результата, может быть равен dwSourceBlock;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.19.5 CLSCRF_SAM_MifareDecrement

```

LONG CLSCRF_SAM_MifareDecrement(    IN LPVOID pReader,
                                     IN BYTE ucLC,
                                     IN DWORD dwSector,
                                     IN DWORD dwSourceBlock,
                                     IN DWORD dwValue,
                                     IN DWORD dwTargetBlock,
                                     OUT LPBYTE pucSW1,
                                     OUT LPBYTE pucSW2 );

```

Уменьшить значение блока типа Value.

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

ucLC – логический канал от 0 до 3;

dwSector – номер сектора, содержащего читаемый блок;

dwSourceBlock – номер исходного блока;

dwValue – вычитаемое, на которое уменьшается значение;

dwTargetBlock – номер блока-результата, может быть равен dwSourceBlock;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.19.6 CLSCRF_SAM_MifareRestore

```

LONG CLSCRF_SAM_MifareRestore(    IN LPVOID pReader,
                                    IN BYTE ucLC,
                                    IN DWORD dwSector,

```



```
IN DWORD dwSourceBlock,
IN DWORD dwTargetBlock,
OUT LPBYTE pucSW1,
OUT LPBYTE pucSW2 );
```

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

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRF Create](#));

ucLC – логический канал от 0 до 3;

dwSector – номер сектора, содержащего оба блока;

dwSourceBlock – номер исходного блока;

dwTargetBlock – номер блока-результата;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.19.7 CLSCRF_SAM_MifareChangeKey

```
LONG CLSCRF_SAM_MifareChangeKey(
IN LPVOID pReader,
IN BYTE ucLC,
IN BYTE ucCrypto,
IN BYTE ucKeyNo,
IN BYTE ucKeyVerA,
IN BYTE ucKeyVerB,
IN DWORD dwSector,
IN LPBYTE pbAccess,
IN LPBYTE pbUID,
OUT LPBYTE pucSW1,
OUT LPBYTE pucSW2 );
```

Сменить ключ карты Mifare Classic.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRF Create](#));

ucLC – логический канал от 0 до 3;

ucCrypto – маска метода компиляции ключей;

ucKeyNo – номер нового Mifare-ключа в SAM-модуле от 1 до 127;

ucKeyVerA – версия нового Mifare-ключа KeyA в SAM-модуле от 0 до 255;

ucKeyVerB – версия нового Mifare-ключа KeyB в SAM-модуле от 0 до 255;

dwSector – номер сектора, содержащего блок;

pbAccess[4] – новые биты доступа в трейлере;

pbUID[4] – уникальный номер карты (не требуется, если Crypto равен 0);

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.20 Работа с картами Mifare Ultralight C при помощи SAM-модуля

Функции [чтения](#) и [записи](#) смотрите в разделе стандартных команд для работы с картами Mifare Ultralight.

4.20.1 CLSCRF_SAM_MifareUltralightC_Authenticate

```
LONG CLSCRF_SAM_MifareUltralightC_Authenticate (
    IN LPVOID pReader,
    IN BYTE ucLC,
    IN LPBYTE pbUID,
    IN BYTE ucKeyNo,
    IN BYTE ucKeyVer,
    OUT LPBYTE
    pucSW1,
    OUT LPBYTE
    pucSW2 );
```

Выполняет аутентификацию карты Mifare Ultralight C.

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

ucLC – логический канал от 0 до 3;

ucKeyNo – номер нового Mifare Ultralight C ключа в SAM-модуле от 1 до 127;

ucKeyVer – версия нового Mifare Ultralight C ключа SAM-модуле от 0 до 255;

pbUID[7] – уникальный номер карты;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.20.2 CLSCRF_SAM_MifareUltralightC_WriteKey

```
LONG CLSCRF_SAM_MifareUltralightC_WriteKey ( IN LPVOID pReader,
    IN BYTE ucLC,
    IN LPBYTE pbUID,
    IN BYTE ucKeyNo,
    IN BYTE ucKeyVer,
    OUT LPBYTE pucSW1,
    OUT LPBYTE pucSW2 );
```

Сменить ключ карты Mifare Ultralight C.

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

ucLC – логический канал от 0 до 3;

ucKeyNo – номер нового Mifare Ultralight C ключа в SAM-модуле от 1 до 127;

ucKeyVer – версия нового Mifare Ultralight C ключа SAM-модуле от 0 до 255;

pbUID[7] – уникальный номер карты;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.
 Возвращаемое значение:
 0 – успешное выполнение,
 иначе – ошибка при выполнении.

4.21 Работа с картами Mifare Plus при помощи SAM-модуля

4.21.1 CLSCRF_SAM_MifarePlus_WritePerso

```
LONG CLSCRF_SAM_MifarePlus_WritePerso(      IN LPVOID pReader,
                                              IN BYTE ucLC,
                                              IN BYTE ucValueType,
                                              IN BYTE ucSectorNumber,
                                              IN BYTE ucBlockNumber,
                                              IN BYTE ucKeyNumber,
                                              IN BYTE ucKeyVersion,
                                              IN BYTE ucLenDiv,
                                              IN LPBYTE pbDiv,
                                              OUT LPBYTE pucSW1,
                                              OUT LPBYTE pucSW2 );
```

Записывает данные персонализации в карту Mifare Plus.

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

ucLC – логический канал от 0 до 3;

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);

ucKeyNumber – номер Mifare-ключа в SAM-модуле от 1 до 127;

ucKeyVersion – версия Mifare-ключа в SAM-модуле от 0 до 255;

ucLenDiv – длина входного вектора диверсификации ключа;

pbDiv – входной вектор диверсификации ключа;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
 иначе – ошибка при выполнении.

4.21.2 CLSCRF_SAM_MifarePlus_CommitPerso

```
LONG CLSCRF_SAM_MifarePlus_CommitPerso(    IN LPVOID pReader,
                                              IN BYTE ucLC,
```


OUT LPBYTE pucSW1,
OUT LPBYTE pucSW2);

Завершает персонализацию карты Mifare Plus.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRFL Create](#));

ucLC – логический канал от 0 до 3;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.21.3 CLSCRFL_SAM_MifarePlus_Authenticate

LONG CLSCRFL_SAM_MifarePlus_Authenticate(IN LPVOID pReader,
IN BYTE ucLC,
IN BYTE ucAuthType,
IN BYTE ucSecLevel,
IN BYTE ucKeyType,
IN BYTE ucSectorNumber,
IN BYTE ucKeyNumber,
IN BYTE ucKeyVersion,
IN BYTE ucLenCap,
IN LPBYTE pbPcdCap,
IN BYTE ucLenDiv,
IN LPBYTE pbDiv,
OUT LPBYTE pucSW1,
OUT LPBYTE pucSW2);

Выполняет (сброс) аутентификацию карты Mifare Plus.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRFL Create](#));

ucLC – логический канал от 0 до 3;

ucAuthType – тип аутентификации по спецификации (0x01 – первичная, 0x0F – последующая, 0x00 – сброс аутентификации);

ucSecLevel – уровень защиты карты (0x00 - SL1, 0x01 - SL2, 0x03 - SL3);

ucKeyType – тип ключа для аутентификации (см. [таблицу](#));

ucSectorNumber – номер сектора для аутентификации;

ucKeyNumber – номер Mifare-ключа в SAM-модуле от 1 до 127;

ucKeyVersion – версия Mifare-ключа в SAM-модуле от 0 до 255;

ucLenCap – длина блока характеристик считывателя (0..6, установить в 0);

pbPcdCap – блок характеристик считывателя (пока отсутствует);

ucLenDiv – длина входного вектора диверсификации ключа;

pbDiv – входной вектор диверсификации ключа;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.21.4 CLSCRF_SAM_MifarePlus_MultiBlockRead

```

LONG CLSCRF_SAM_MifarePlus_MultiBlockRead(  IN LPVOID pReader,
                                              IN BYTE ucLC,
                                              IN BYTE
ucSectorNumber,
                                              IN BYTE
ucBlockNumber,
                                              IN BYTE
ucBlocksCount,
                                              OUT LPBYTE
pbData,
                                              OUT LPBYTE
pucSW1,
                                              OUT      LPBYTE
pucSW2 );

```

Для карты в режиме SL2, производит множественное чтение блоков.

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

ucLC – логический канал от 0 до 3;

ucSectorNumber – номер сектора

ucBlockNumber – номер блока, с которого нужно начинать чтение

ucBlocksCount – количество читаемых блоков

pbData – указатель на байтовый массив, в который будут записаны данные

pucSW1 – байт статуса;

pucSW2 – байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.21.5 CLSCRF_SAM_MifarePlus_MultiBlockWrite

```

LONG CLSCRF_SAM_MifarePlus_MultiBlockWrite(  IN LPVOID pReader,
                                              IN BYTE ucLC,
                                              IN BYTE
ucSectorNumber,
                                              IN BYTE
ucBlockNumber,
                                              IN BYTE
ucBlocksCount,
                                              IN LPBYTE pbData,
                                              OUT LPBYTE
pucSW1,
                                              OUT      LPBYTE
pucSW2 );

```


Для карты в режиме SL2, производит множественную запись блоков.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRFL_Create](#))
ucLC – логический канал от 0 до 3;
ucSectorNumber – номер сектора
ucBlockNumber – номер начального блока для записи
ucBlocksCount – количество записываемых блоков
pbData – указатель на байтовый массив с данными для записи
pucSW1 – байт статуса;
pucSW2 – байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
 иначе – ошибка при выполнении.

4.21.6 CLSCRFL_SAM_MifarePlus_ReadData

```
LONG CLSCRFL_SAM_MifarePlus_ReadData( IN LPVOID pReader,
                                         IN BYTE ucLC,
                                         IN BYTE ucReadMode,
                                         IN BYTE ucValueType,
                                         IN BYTE ucSectorNumber,
                                         IN BYTE ucBlockNumber,
                                         IN BYTE ucBlocksCount,
                                         OUT LPBYTE pbData,
                                         OUT LPBYTE pucSW1,
                                         OUT LPBYTE pucSW2 );
```

Для карты в режиме SL3, производит чтение данных.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRFL_Create](#));
ucLC – логический канал от 0 до 3;
ucReadMode – режим защиты данных (см. таблицу);
ucValueType – тип читаемого значения (см. таблицу 117 в руководстве на MifarePlus);
ucSectorNumber – номер сектора с читаемым блоком;
ucBlockNumber – номер начального блока для чтения;
ucBlocksCount – количество читаемых блоков;
pbData – указатель на байтовый массив, в который будут записаны данные;
pucSW1 – указатель на байт статуса;
pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
 иначе – ошибка при выполнении.

4.21.7 CLSCRFL_SAM_MifarePlus_WriteData

```
LONG CLSCRFL_SAM_MifarePlus_WriteData( IN LPVOID pReader,
                                         IN BYTE ucLC,
                                         IN BYTE ucWriteMode,
```



```

        IN BYTE ucValueType,
        IN BYTE ucSectorNumber,
        IN BYTE ucBlockNumber,
        IN BYTE ucBlocksCount,
        IN LPBYTE pbData,
        OUT LPBYTE pucSW1,
        OUT LPBYTE pucSW2 );

```

Для карты в режиме SL3, производит запись данных.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRF Create](#));

ucLC – логический канал от 0 до 3;

ucWriteMode – режим защиты данных (см. таблицу);

ucValueType – тип записываемого значения (см. таблицу 117 в руководстве на MifarePlus);

ucSectorNumber – номер сектора для записи блока;

ucBlockNumber – номер начального блока для записи;

ucBlocksCount – количество записываемых блоков;

pbData – указатель на байтовый массив, из которого будут браться данные для записи;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.21.8 CLSCRF_SAM_MifarePlus_ChangeKey

```

LONG CLSCRF_SAM_MifarePlus_ChangeKey( IN LPVOID pReader,
                                         IN BYTE ucLC,
                                         IN BYTE
ucEncryptionMode,
                                         IN BYTE ucKeyType,
                                         IN BYTE ucSectorNumber,
                                         IN BYTE ucKeyNumber,
                                         IN BYTE ucKeyVersion,
                                         IN BYTE ucLenDiv,
                                         IN LPBYTE pbDiv,
                                         OUT LPBYTE pucSW1,
                                         OUT LPBYTE pucSW2 );

```

Для карты в режиме SL3, производит смену ключа.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRF Create](#));

ucLC – логический канал от 0 до 3;

ucEncryptionMode – режим защиты данных (0x00 - шифрование, MAC в запросе; 0x01 - шифрование, MAC в запросе, MAC в ответе);

ucKeyType – тип ключа для аутентификации (см. [таблицу](#));

ucSectorNumber – номер сектора;

ucKeyNumber – номер Mifare-ключа в SAM-модуле от 1 до 127;

ucKeyVersion – версия Mifare-ключа в SAM-модуле от 0 до 255;

ucLenDiv – длина входного вектора диверсификации ключа;

pbDiv – входной вектор диверсификации ключа;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.21.9 CLSCRF_SAM_MifarePlus_Increment

```
LONG CLSCRF_SAM_MifarePlus_Increment( IN LPVOID pReader,
                                         IN BYTE ucLC,
                                         IN BYTE
ucEncryptionMode,
                                         IN BYTE
ucSourceSectorNumber,
                                         IN BYTE
ucSourceBlockNumber,
                                         IN DWORD dwValue,
                                         OUT LPBYTE pucSW1,
                                         OUT LPBYTE pucSW2 );
```

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](#));

ucLC – логический канал от 0 до 3;

ucEncryptionMode – режим защиты данных (см. таблицу);

ucSourceSectorNumber – номер исходного сектора;

ucSourceBlockNumber – номер исходного блока;

dwValue – прибавляемое к счетчику значение;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.21.10 CLSCRF_SAM_MifarePlus_Decrement

```
LONG CLSCRF_SAM_MifarePlus_Decrement( IN LPVOID pReader,
                                         IN BYTE ucLC,
                                         IN BYTE
ucEncryptionMode,
                                         IN BYTE
ucSourceSectorNumber,
                                         IN BYTE
ucSourceBlockNumber,
                                         IN DWORD dwValue,
                                         OUT LPBYTE pucSW1,
```


OUT LPBYTE pucSW2);

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](#));

ucLC – логический канал от 0 до 3;

ucEncryptionMode – режим защиты данных (см. таблицу);

ucSourceSectorNumber – номер исходного сектора;

ucSourceBlockNumber – номер исходного блока;

dwValue – вычитаемое из счетчика значение;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.21.11 CLSCRF_SAM_MifarePlus_Transfer

```
LONG CLSCRF_SAM_MifarePlus_Transfer(
    IN LPVOID pReader,
    IN BYTE ucLC,
    IN BYTE ucEncryptionMode,
    IN BYTE
    ucDestinationSectorNumber,
    IN BYTE
    ucDestinationBlockNumber,
    OUT LPBYTE pucSW1,
    OUT LPBYTE pucSW2 );
```

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

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

ucLC – логический канал от 0 до 3;

ucEncryptionMode – режим защиты данных (см. таблицу);

ucDestinationSectorNumber – номер сектора для записи;

ucDestinationBlockNumber – номер блока для записи;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.21.12 CLSCRF_SAM_MifarePlus_IncrementTransfer

```
LONG CLSCRF_SAM_MifarePlus_IncrementTransfer( IN LPVOID pReader,
    IN BYTE ucLC,
    IN BYTE
    ucEncryptionMode,
    IN
    BYTE ucSourceSectorNumber,
```



```

ucSourceBlockNumber,          IN BYTE
ucDestinationSectorNumber,     IN BYTE
ucDestinationBlockNumber,      IN BYTE
dwValue,                      IN DWORD
pucSW1,                       OUT LPBYTE
                                OUT LPBYTE pucSW2 );

```

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](#));

ucLC – логический канал от 0 до 3;

ucEncryptionMode – режим защиты данных (см. таблицу);

ucSourceSectorNumber – номер исходного сектора;

ucSourceBlockNumber – номер исходного блока;

ucDestinationSectorNumber – номер сектора для записи;

ucDestinationBlockNumber – номер блока для записи;

dwValue – прибавляемое к счетчику значение;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.21.13 CLSCRF_SAM_MifarePlus_DecrementTransfer

```

LONG CLSCRF_SAM_MifarePlus_DecrementTransfer(    IN LPVOID pReader,
                                                    IN BYTE ucLC,
                                                    IN BYTE
ucEncryptionMode,                                IN
BYTE ucSourceSectorNumber,                        IN
                                                    IN BYTE
ucSourceBlockNumber,                             IN BYTE
                                                    IN BYTE
ucDestinationSectorNumber,                       IN BYTE
                                                    IN BYTE
ucDestinationBlockNumber,                        IN DWORD
dwValue,                                          OUT LPBYTE
pucSW1,                                          OUT LPBYTE pucSW2 );

```

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](#));

ucLC – логический канал от 0 до 3;

ucEncryptionMode – режим защиты данных (см. таблицу);

ucSourceSectorNumber – номер исходного сектора;

ucSourceBlockNumber – номер исходного блока;

ucDestinationSectorNumber – номер сектора для записи;

ucDestinationBlockNumber – номер блока для записи;

dwValue – вычитаемое из счетчика значение;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.21.14 CLSCRF_SAM_MifarePlus_Restore

```
LONG CLSCRF_SAM_MifarePlus_Restore(    IN LPVOID pReader,
                                         IN BYTE ucLC,
                                         IN BYTE ucEncryptionMode,
                                         IN BYTE ucSourceSectorNumber,
                                         IN BYTE ucSourceBlockNumber,
                                         OUT LPBYTE pucSW1,
                                         OUT LPBYTE pucSW2 );
```

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](#));

ucLC – логический канал от 0 до 3;

ucEncryptionMode – режим защиты данных (см. таблицу);

ucSourceSectorNumber – номер исходного сектора;

ucSourceBlockNumber – номер исходного блока;

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.21.15 CLSCRF_SAM_MifarePlus_VirtualCardSupport

```
LONG CLSCRF_SAM_MifarePlus_VirtualCardSupport(    IN LPVOID pReader,
                                                    IN BYTE ucLC,
                                                    IN BYTE
ucIIDCount,
                                                    IN LPBYTE IIDs,
                                                    IN BYTE ucLenCap,
                                                    IN LPBYTE
```



```

pbPcdCap,
ucDuosCount,
Duos,
pbRndQ,
pucInfo,
LPBYTE pbPiccCap,
pbUID,
pucSW1,
pucSW2 );

```

IN BYTE
IN SAM_VC_DUO*
OUT LPBYTE
OUT LPBYTE
OUT
OUT LPBYTE
OUT LPBYTE
OUT LPBYTE
OUT LPBYTE

Отправляет на SAM-модуль команду – запрос на перебор идентификаторов инсталляций, с использованием заданных ключей.

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

ucLC – логический канал от 0 до 3;

ucIIDCount – количество перебираемых идентификаторов инфраструктуры в массиве;

IIDs – массив идентификаторов инфраструктуры (16 байт * ucIIDCount);

ucLenCap – длина блока характеристик считывателя (0..3, установить в 0);

pbPcdCap – блок характеристик считывателя (пока отсутствует);

ucDuosCount – количество передаваемых для проверки карт пар ключей;

Duos – массив, состоящий из пар ключей VC Polling ENC Key и VC Polling MAC Key, представленных в виде номеров и версий ключей в SAM модуле;

pbRndQ – указатель на массив из 12 байт, в который будет записано значение RndQ, идентифицирующее карту в SAM-модуле;

pucInfo – указатель на байтовую переменную, в которую будет записана информация о карте (0x83 - 4-байтовый UID, 0x80 - 7-байтовый UID);

bpPiccCap – указатель на буфер (2 байта), в который будут записаны характеристики карты;

pbUID – указатель на буфер (7 байт), в который будет записан UID карты;

pucSW1 – байт статуса;

pucSW2 – байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.21.16 CLSCRF_SAM_MifarePlus_VirtualCardSelect

```

LONG CLSCRF_SAM_MifarePlus_VirtualCardSelect( IN LPVOID pReader,
                                                IN BYTE ucLC,
                                                IN BYTE
ucKeyNumber,

```


ucKeyVersion, IN BYTE

IN LPBYTE pbRndQ,
IN BYTE ucLenDiv,
IN LPBYTE pbDiv,
OUT LPBYTE

pucSW1, OUT LPBYTE

pucSW2);

Выбирает виртуальную карту.

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

ucLC – логический канал от 0 до 3;

ucKeyNumber – номер Mifare-ключа в SAM-модуле от 1 до 127;

ucKeyVersion – версия Mifare-ключа в SAM-модуле от 0 до 255;

pbRndQ – указатель на массив из 12 байт, в который записано значение RndQ, идентифицирующее карту в SAM-модуле

ucLenDiv – длина входного вектора диверсификации ключа;

pbDiv – входной вектор диверсификации ключа;

pucSW1 – байт статуса;

pucSW2 – байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.21.17 CLSCRF_SAM_MifarePlus_VirtualCardDeselect

LONG CLSCRF_SAM_MifarePlus_VirtualCardDeselect(IN LPVOID pReader,
IN BYTE ucLC,
OUT LPBYTE

pucSW1, OUT LPBYTE

pucSW2);

Отменяет выбор виртуальной карты.

ucLC – логический канал от 0 до 3;

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

pucSW1 – указатель на байт статуса;

pucSW2 – указатель на байт статуса.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.21.18 CLSCRF_SAM_MifarePlus_ProximityCheck

LONG CLSCRF_SAM_MifarePlus_ProximityCheck(IN LPVOID pReader,
IN BYTE ucLC,
IN BYTE ucKeyNumber,
IN BYTE ucKeyVersion,

IN BYTE ucLenDiv,
 IN LPBYTE pbDiv,
 OUT LPBYTE pucSW1,
 OUT LPBYTE pucSW2);

Выполняет проверку релейной атаки.

pReader – ссылка на объект-интерфейс (см. функцию [CLSCRFL_Create](#))

ucLC – логический канал от 0 до 3;

ucKeyNumber – номер Mifare-ключа в SAM-модуле от 1 до 127;

ucKeyVersion – версия Mifare-ключа в SAM-модуле от 0 до 255;

ucLenDiv – длина входного вектора диверсификации ключа;

pbDiv – входной вектор диверсификации ключа;

pucSW1 – байт статуса;

pucSW2 – байт статуса.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.22 Работа со считывателем NFC663

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

4.22.1 CLSCRFL_NFC663_ActivateCard

```
LONG CLSCRFL_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 [CLSCRFL_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_FRAME_SIZE_64](#),
[PHPAL_I18092MPI_FRAME_SIZE_128](#), [PHPAL_I18092MPI_FRAME_SIZE_192](#),
[PHPAL_I18092MPI_FRAME_SIZE_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.

4.22.2 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.

4.22.3 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.

4.22.4 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.

4.22.5 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_FRAME_SIZE_64](#), [PHPAL_I18092MPI_FRAME_SIZE_128](#), [PHPAL_I18092MPI_FRAME_SIZE_192](#), [PHPAL_I18092MPI_FRAME_SIZE_254](#));

GiLength – количество байт общей информации;

Gi – опционально, байты общей информации;

pbATR – буфер, куда будут записаны байты ATR (ответа с атрибутами), должен быть не меньше 64 байт;

pdwATRLength – длина считанных атрибутов в байтах;

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.22.6 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_FRAME_SIZE_64](#), [PHPAL_I18092MPI_FRAME_SIZE_128](#), [PHPAL_I18092MPI_FRAME_SIZE_192](#), [PHPAL_I18092MPI_FRAME_SIZE_254](#));

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.22.7 CLSCRF_NFC663_PresenceCheck

```
LONG CLSCRF_NFC663_PresenceCheck( IN LPVOID pReader );
```

Выполняет проверку присутствия текущей цели при работе по протоколу ISO18092.

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

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.22.8 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.

4.22.9 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.

4.22.10 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.

4.22.11 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.

4.22.12 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.

4.22.13 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.

4.22.14 PHPAL_I18092MPI_DATARATE_106

```
#define PHPAL_I18092MPI_DATARATE_106 0x00U
```

Значение DRI/DSI для 106 kBit/s.

4.22.15 PHPAL_I18092MPI_DATARATE_212

```
#define PHPAL_I18092MPI_DATARATE_212 0x01U
```

Значение DRI/DSI для 212 kBit/s.

4.22.16 PHPAL_I18092MPI_DATARATE_424

```
#define PHPAL_I18092MPI_DATARATE_424 0x02U
```

Значение DRI/DSI для 424 kBit/s.

4.22.17 PHPAL_I18092MPI_FRAME_SIZE_64

```
#define PHPAL_I18092MPI_FRAME_SIZE_64 0x00U
```

Значение FSL для макс. размера кадра в 64 байта.

4.22.18 PHPAL_I18092MPI_FRAME_SIZE_128

```
#define PHPAL_I18092MPI_FRAME_SIZE_128 0x01U
```

Значение FSL для макс. размера кадра в 128 байт.

4.22.19 PHPAL_I18092MPI_FRAME_SIZE_192

```
#define PHPAL_I18092MPI_FRAME_SIZE_192 0x02U
```

Значение FSL для макс. размера кадра в 192 байта.

4.22.20 PHPAL_I18092MPI_FRAME_SIZE_254

```
#define PHPAL_I18092MPI_FRAME_SIZE_254 0x03U
```

Значение FSL для макс. размера кадра в 254 байта.

4.22.21 PHPAL_I18092MPI_DESELECT_DSL

```
#define PHPAL_I18092MPI_DESELECT_DSL 0x08U
```

DSL отправляется для отмена выбора цели.

4.22.22 PHPAL_I18092MPI_DESELECT_RLS

```
#define PHPAL_I18092MPI_DESELECT_RLS 0x0AU
```

RLS отправляется для отмены выбора цели.

4.22.23 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.

4.22.24 PH_EXCHANGE_TXCHAINING

```
#define PH_EXCHANGE_TXCHAINING 0x0001U
```

Специфично для ISO14443-4:

Передаёт данные в карту цепочной передачей.

Комбинируется с [PH_EXCHANGE_BUFFERED_BIT](#) и

[PH_EXCHANGE_LEAVE_BUFFER_BIT](#) для выполнения буферизации.

Не принимает никаких данных.

4.22.25 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.

4.22.26 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.

4.22.27 PH_EXCHANGE_TX_CRC

```
#define PH_EXCHANGE_TX_CRC 0x0010U
```


Добавлять TX CRC.

4.22.28 PH_EXCHANGE_RX_CRC

```
#define PH_EXCHANGE_RX_CRC 0x0020U
```

Добавлять RX CRC.

4.22.29 PH_EXCHANGE_PARITY

```
#define PH_EXCHANGE_PARITY 0x0040U
```

Добавлять признак четности.

4.22.30 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.

4.22.31 PH_EXCHANGE_BUFFERED_BIT

```
#define PH_EXCHANGE_BUFFERED_BIT 0x8000U
```

Буферизует Tx-Data во внутренний буфер вместо его передачи.

4.22.32 PHPAL_I18092MPI_CONFIG_PACKETNO

```
#define PHPAL_I18092MPI_CONFIG_PACKETNO 0x0000U
```

Задать / получить номер пакета.

4.22.33 PHPAL_I18092MPI_CONFIG_DID

```
#define PHPAL_I18092MPI_CONFIG_DID 0x0001U
```

Задать / получить идентификатор устройства.

4.22.34 PHPAL_I18092MPI_CONFIG_NAD

```
#define PHPAL_I18092MPI_CONFIG_NAD 0x0002U
```


Задать / получить адрес узла.

4.22.35 PHPAL_I18092MPI_CONFIG_WT

```
#define PHPAL_I18092MPI_CONFIG_WT 0x0003U
```

Задать / получить время ожидания.

4.22.36 PHPAL_I18092MPI_CONFIG_FSL

```
#define PHPAL_I18092MPI_CONFIG_FSL 0x0004U
```

Задать / получить длину кадра.

4.22.37 PHPAL_I18092MPI_CONFIG_MAXRETRYCOUNT

```
#define PHPAL_I18092MPI_CONFIG_MAXRETRYCOUNT 0x0005U
```

Задать / получить максимальное количество повторных попыток.

4.23 Демонстрация 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](#).

4.23.1 Функции работы с NDEF сообщениями

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

4.23.1.1 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 обработке функции).

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.23.1.2 CLSCRFL_NFC_NDEF_Message_GetRecordsCount

LONG CLSCRFL_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.

4.23.1.3 CLSCRFL_NFC_NDEF_Message_GetDataSize

LONG CLSCRFL_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.

4.23.1.4 CLSCRFL_NFC_NDEF_Message_GetData

LONG CLSCRFL_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.

4.23.1.5 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.

4.23.1.6 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.

4.23.1.7 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.

4.23.1.8 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.

4.23.1.9 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.

4.23.1.10

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.

4.23.1.11 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.

4.23.1.12 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.

4.23.2 Функции работы с NDEF записями

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

4.23.2.1 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.

4.23.2.2 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.

4.23.2.3 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.

4.23.2.4 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.

4.23.2.5 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.

4.23.2.6 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.

4.23.2.7 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.

4.23.2.8 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.

4.23.2.9 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 типа в формате Unicode, которую следует установить.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.23.2.10 CLSCRFL_NFC_NDEF_Record_GetPayloadLength

LONG CLSCRFL_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.

4.23.2.11 CLSCRFL_NFC_NDEF_Record_GetPayload

LONG CLSCRFL_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.

4.23.2.12 CLSCRFL_NFC_NDEF_Record_GetPayloadText

LONG CLSCRFL_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 помещены 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.

4.23.2.13 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.

4.23.2.14 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.

4.23.2.15 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.

4.23.2.16 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.

4.23.2.17 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.

4.23.2.18 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.

4.23.2.19 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.

4.23.3 Функции работы с протоколом LLCР

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

4.23.3.1 CLSCRF_NFC_LLCР_Create

```
LONG CLSCRF_NFC_LLCР_Create(    IN LPVOID pReader,  
                                OUT LPHANDLE phLLCPLink );
```

Создаёт объект интерфейса протокола LLCР.

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

phLLCPLink – ссылка на переменную типа DWORD, в которую будет помещен указатель на созданный объект протокола LLCР.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.23.3.2 CLSCRF_NFC_LLCР_SetTimeout

```
LONG CLSCRF_NFC_LLCР_SetTimeout(    IN HANDLE hLLCPLink,  
                                     IN DWORD dwTimeoutMs );
```

Устанавливает таймаут соединения по протоколу LLCР.

Следует вызывать до открытия соединения.

hLLCPLink – указатель на объект протокола LLCР;

dwTimeoutMs – таймаут соединения в миллисекундах.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.23.3.3 CLSCRF_NFC_LLCР_SetBaudrate

```
LONG CLSCRF_NFC_LLCР_SetBaudrate(    IN HANDLE hLLCPLink,  
                                       IN BYTE ucDsiDri );
```

Устанавливает скорость обмена по протоколу LLCР.

Следует вызывать до открытия соединения.

hLLCPLink – указатель на объект протокола LLCР;

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.

4.23.3.4 CLSCRF_NFC_LLCР_SetGeneralInformation

```
LONG CLSCRF_NFC_LLCР_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;

IpGI – указатель на массив, содержащий байты общей информации;

dwGILength – размер массива байт общей информации.

Возвращаемое значение:

0 – успешное выполнение,

иначе – ошибка при выполнении.

4.23.3.5 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.

4.23.3.6 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.

4.23.3.7 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.

4.23.3.8 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.

4.23.3.9 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.

4.23.3.10 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.

4.23.3.11 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.

4.23.3.12 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.

4.23.4 Функции работы с протоколом SNEP

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

4.23.4.1 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.

4.23.4.2 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.

4.23.4.3 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.

4.23.4.4 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.

4.23.5 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.

4.23.5.1 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.

4.23.5.2 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.

4.23.5.3 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.

4.23.5.4 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.

4.23.5.5 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.

4.23.6 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 радиointерфейса 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 = CLSCRFL_Create(&pReader);  
if (Status != SCARD_S_SUCCESS)  
{  
    m_strResult = "Ошибка вызова библиотеки!";  
    return Status;  
}  
  
// Открываем соединения со считывателем по USB  
Status = CLSCRFL_OpenUSB(pReader);  
if (Status != SCARD_S_SUCCESS)  
{  
    m_strResult = "Не найден считыватель!";  
    CLSCRFL_Destroy(pReader);  
    return Status;  
}  
  
// Включаем микросхему радиообмена  
Status = CLSCRFL_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 = CLSCRFL_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 = CLSCRFL_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;
    }
}
```


4.23.6.1 Приём сообщения NDEF с текстовой записью

Инициализируем массив для принимаемого сообщения.

```
BYTE lpMsgData[1024] = { 0 };  
DWORD dwMsgSize = sizeof(lpMsgData);
```

4.23.6.1.1 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;  
}
```

4.23.6.1.2 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;
    }
}

```

4.23.6.1.3 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);

```


4.23.6.2 Отправка сообщения NDEF с текстовой записью

Инициализируем массив для отправляемого сообщения.

```
BYTE lpMsgData[1024] = { 0 };  
DWORD dwMsgSize = sizeof(lpMsgData);
```

4.23.6.2.1 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;  
}
```


4.23.6.2.2 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!";
```

4.23.6.2.3 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!";
```

4.24 Непосредственный обмен данными с картой

Внимание! Для режима ISO14443 T=CL используйте связку функций [CLSCRF_ISO14443_ActivateEx](#) и [CLSCRF_ISO14443_4_Exchange](#).

4.24.1 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.

4.25 Работа с TDA8029

4.25.1 CLSCRF_TDA8029_Exchange

```
LONG CLSCRF_TDA8029_Exchange( IN  LPVOID pReader,  
                               IN  BYTE ucCmdCode,  
                               IN  LPCBYTE pbSendBuffer,  
                               IN  DWORD  dwSendLength,  
                               OUT LPBYTE pbRecvBuffer,  
                               IN OUT LPDWORD pdwRecvLength );
```

Отправляет в TDA8029 команду в формате ALPARI.

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

ucCmdCode – код команды;

pbSendBuffer – массив байтов, содержащих команду (не включает заголовок и байт контрольной суммы LRC);

dwSendLength – количество байтов (не включая заголовок и байт контрольной суммы LRC);

pbRecvBuffer – массив для ответа (не включает заголовок и байт контрольной суммы LRC);

pdwRecvLength – ссылка на переменную, которая перед вызовом функции

должна содержать размер массива **pbRecvBuffer**, а на выходе будет содержать количество принятых байтов.

Если общий код ответа функции равен **SCARD_F_INTERNAL_ERROR** (0x80100001), а код внутренней ошибки библиотеки (получается при помощи функции **GetLastError**) равен **UEM_UNKNOWN_ERR** (0x80), то TDA8029 вернула ошибку. Код этой ошибки будет сохранен в **pbRecvBuffer[1]**, а **pdwRecvLength** будет = 0x00000001.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.25.2 CLSCRF_TDA8029_APDU

```
LONG           CLSCRF_TDA8029_APDU( IN   LPVOID pReader,
                                     IN   LPCBYTE pbSendBuffer,
                                     IN   DWORD dwSendLength,
                                     OUT LPBYTE pbRecvBuffer,
                                     IN OUT LPDWORD pdwRecvLength );
```

Отправляет через TDA8029 на карту команду в формате APDU.

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

pbSendBuffer – массив байтов, передаваемых карте;

dwSendLength – количество передаваемых карте байтов;

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.

Если общий код ответа функции равен **SCARD_F_INTERNAL_ERROR** (0x80100001), а код внутренней ошибки библиотеки (получается при помощи функции **GetLastError**) равен **UEM_UNKNOWN_ERR** (0x80), то TDA8029 вернула ошибку. Код этой ошибки будет сохранен в **pbRecvBuffer[1]**, а **pdwRecvLength** будет = 0x00000001.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.25.3 CLSCRF_TDA8029_Reset

```
LONG           CLSCRF_TDA8029_Reset(IN LPVOID pReader );
```

Выполняет сброс (перезапуск) TDA8029.

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

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.25.4 CLSCRF_TDA8029_GetStatus

LONG CLSCRF_TDA8029_GetStatus(IN LPVOID pReader,
OUT LPBYTE pucStatus);

Вычитывает из TDA его текущее состояние.

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

pucStatus – указатель на байт, куда будут записаны флаги текущего состояния TDA (см. описание на команду TDA8029 get_reader_status).

Если общий код ответа функции равен SCARD_F_INTERNAL_ERROR (0x80100001), а код внутренней ошибки библиотеки (получается при помощи функции GetLastError) равен UEM_UNKNOWN_ERR (0x80), то TDA8029 вернула ошибку. Код этой ошибки будет сохранен в ***pucStatus**.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.25.5 CLSCRF_TDA8029_GetATR

LONG CLSCRF_TDA8029_GetATR(IN LPVOID pReader,
OUT LPBYTE pbRecvBuffer,
IN OUT LPDWORD pdwRecvLength);

Возвращает ATR текущей карты.

pbATRBuffer – массив для хранения ATR, полученного от карты;

pdwATRLength – ссылка на переменную, которая перед вызовом функции должна содержать размер массива pbATRBuffer, а на выходе будет содержать размер принятого от карты ATR.

Если общий код ответа функции равен SCARD_F_INTERNAL_ERROR (0x80100001), а код внутренней ошибки библиотеки (получается при помощи функции GetLastError) равен UEM_UNKNOWN_ERR (0x80), то TDA8029 вернула ошибку. Код этой ошибки будет сохранен в **pbATRBuffer[1]**, а **pdwATRLength** будет = 0x00000001.

Возвращаемое значение:

0 – успешное выполнение,
иначе – ошибка при выполнении.

4.26 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_GetLastError](#). 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



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 (А и В).

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 А и для карт типа В.

5.1 Активация карты типа А из состояния IDLE

5.1.1 Переключить режим Rf на тип А скорость 106.

Команда	5 1 00
Ответ	00

5.1.2 Активировать карту типа А 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 послышки 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 5 1 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 послышки 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 (PUP1+ Param1+ Param2+ Param3+ Param4)

9C 01 F5 B8 – PUP1

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 5 1 10

Ответ 00

5.4 Активация карты типа B из состояния HALT

5.4.1 Переключить режим Rf на тип B скорость 106.

Команда 5 1 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 послышки 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.