

# TWN4

## Simple Protocol

DocRev25, April 29, 2022



ELATEC GmbH

# Contents

1. Simple Protocol	3
1.1. Command	3
1.2. Response	3
1.3. Data Transmission	4
1.3.1. ASCII	4
1.3.2. Binary	4
1.3.3. CRC	4
1.3.4. Reference messages	4
1.4. Data Types	5
1.5. Commands	5
1.5.1. API SYS	5
1.5.1.1. Reset	5
1.5.1.2. StartBootloader	6
1.5.1.3. GetSysTicks	6
1.5.1.4. GetVersionString	6
1.5.1.5. GetUSBType	6
1.5.1.6. GetDeviceType	7
1.5.1.7. Sleep	7
1.5.1.8. GetDeviceUID	7
1.5.1.9. SetParameters	7
1.5.1.10. GetLastError	8
1.5.1.11. GetProdSerNo	8
1.5.2. API IO	8
1.5.2.1. WriteByte	8
1.5.2.2. ReadByte	8
1.5.2.3. TestEmpty	9
1.5.2.4. TestFull	9
1.5.2.5. GetBufferSize	9
1.5.2.6. GetByteCount	9
1.5.2.7. SetCOMParameters	10
1.5.2.8. GetUSBDeviceState	10
1.5.2.9. GetHostChannel	10
1.5.2.10. USBRemoteWakeup	10
1.5.2.11. WriteBytes	11
1.5.2.12. ReadBytes	11
1.5.3. API PERIPH	11
1.5.3.1. GPIOConfigureOutputs	11
1.5.3.2. GPIOConfigureInputs	11
1.5.3.3. GPIOSetBits	12
1.5.3.4. GPIOClearBits	12
1.5.3.5. GPIONToggleBits	12
1.5.3.6. GPIOBlinkBits	12
1.5.3.7. GPIOTestBit	13
1.5.3.8. Beep	13

1.5.3.9.	DiagLEDOn . . . . .	13
1.5.3.10.	DiagLEDOff . . . . .	13
1.5.3.11.	DiagLEDToggle . . . . .	14
1.5.3.12.	DiagLEDIsOn . . . . .	14
1.5.3.13.	SendWiegand . . . . .	14
1.5.3.14.	SendOmron . . . . .	14
1.5.3.15.	LEDInit . . . . .	15
1.5.3.16.	LEDOn . . . . .	15
1.5.3.17.	LEDOff . . . . .	15
1.5.3.18.	LEDToggle . . . . .	15
1.5.3.19.	LEDBlink . . . . .	16
1.5.3.20.	BeepOn . . . . .	16
1.5.3.21.	BeepOff . . . . .	16
1.5.4.	API RF . . . . .	16
1.5.4.1.	SearchTag . . . . .	16
1.5.4.2.	SetRFOff . . . . .	17
1.5.4.3.	SetTagTypes . . . . .	17
1.5.4.4.	GetTagTypes . . . . .	17
1.5.4.5.	GetSupportedTagTypes . . . . .	17
1.5.5.	API TILF . . . . .	18
1.5.5.1.	TILF_SearchTag . . . . .	18
1.5.5.2.	TILF_ChargeOnlyRead . . . . .	18
1.5.5.3.	TILF_ChargeOnlyReadLo . . . . .	18
1.5.5.4.	TILF_SPPProgramPage . . . . .	18
1.5.5.5.	TILF_SPPProgramPageLo . . . . .	19
1.5.5.6.	TILF_MPGeneralReadPage . . . . .	19
1.5.5.7.	TILF_MPSelectiveReadPage . . . . .	19
1.5.5.8.	TILF_MPPProgramPage . . . . .	19
1.5.5.9.	TILF_MPSelectiveProgramPage . . . . .	20
1.5.5.10.	TILF_MPLockPage . . . . .	20
1.5.5.11.	TILF_MPSelectiveLockPage . . . . .	20
1.5.5.12.	TILF_MPGeneralReadPageLo . . . . .	20
1.5.5.13.	TILF_MPSelectiveReadPageLo . . . . .	21
1.5.5.14.	TILF_MPPProgramPageLo . . . . .	21
1.5.5.15.	TILF_MPSelectiveProgramPageLo . . . . .	21
1.5.5.16.	TILF_MPLockPageLo . . . . .	21
1.5.5.17.	TILF_MPSelectiveLockPageLo . . . . .	22
1.5.5.18.	TILF_MUGeneralReadPage . . . . .	22
1.5.5.19.	TILF_MUSelectiveReadPage . . . . .	22
1.5.5.20.	TILF_MUSpecialReadPage . . . . .	22
1.5.5.21.	TILF_MUProgramPage . . . . .	23
1.5.5.22.	TILF_MUSelectiveProgramPage . . . . .	23
1.5.5.23.	TILF_MUSpecialProgramPage . . . . .	23
1.5.5.24.	TILF_MULockPage . . . . .	24
1.5.5.25.	TILF_MUSelectiveLockPage . . . . .	24
1.5.5.26.	TILF_MUSpecialLockPage . . . . .	24
1.5.6.	API HITAG1S . . . . .	25
1.5.6.1.	Hitag1S_ReadPage . . . . .	25
1.5.6.2.	Hitag1S_ReadBlock . . . . .	25
1.5.6.3.	Hitag1S_WritePage . . . . .	25
1.5.6.4.	Hitag1S_WriteBlock . . . . .	26

1.5.6.5. Hitag1S_Halt . . . . .	26
1.5.7. API HITAG2 . . . . .	26
1.5.7.1. Hitag2_ReadPage . . . . .	26
1.5.7.2. Hitag2_WritePage . . . . .	27
1.5.7.3. Hitag2_Halt . . . . .	27
1.5.7.4. Hitag2_SetPassword . . . . .	27
1.5.8. API SM4X00 . . . . .	27
1.5.8.1. SM4X00_GenericRaw . . . . .	27
1.5.8.2. SM4X00_Generic . . . . .	28
1.5.9. API I2C . . . . .	28
1.5.9.1. I2CInit . . . . .	28
1.5.9.2. I2CDeInit . . . . .	28
1.5.9.3. I2CMasterStart . . . . .	28
1.5.9.4. I2CMasterStop . . . . .	29
1.5.9.5. I2CMasterTransmitByte . . . . .	29
1.5.9.6. I2CMasterReceiveByte . . . . .	29
1.5.9.7. I2CMasterBeginWrite . . . . .	29
1.5.9.8. I2CMasterBeginRead . . . . .	30
1.5.9.9. I2CMasterSetAck . . . . .	30
1.5.10. API MIFARECLASSIC . . . . .	30
1.5.10.1. MifareClassic_Login . . . . .	30
1.5.10.2. MifareClassic_ReadBlock . . . . .	30
1.5.10.3. MifareClassic_WriteBlock . . . . .	31
1.5.10.4. MifareClassic_ReadValueBlock . . . . .	31
1.5.10.5. MifareClassic_WriteValueBlock . . . . .	31
1.5.10.6. MifareClassic_IncrementValueBlock . . . . .	31
1.5.10.7. MifareClassic_DecrementValueBlock . . . . .	32
1.5.10.8. MifareClassic_CopyValueBlock . . . . .	32
1.5.11. API MIFAREULTRALIGHT . . . . .	32
1.5.11.1. MifareUltralight_ReadPage . . . . .	32
1.5.11.2. MifareUltralight_WritePage . . . . .	33
1.5.11.3. MifareUltralightC_Authenticate . . . . .	33
1.5.11.4. MifareUltralightC_SAMAuthenticate . . . . .	33
1.5.11.5. MifareUltralightC_WriteKeyFromSAM . . . . .	33
1.5.11.6. MifareUltralightEV1_FastRead . . . . .	34
1.5.11.7. MifareUltralightEV1_IncCounter . . . . .	34
1.5.11.8. MifareUltralightEV1_ReadCounter . . . . .	34
1.5.11.9. MifareUltralightEV1_ReadSig . . . . .	35
1.5.11.10. MifareUltralightEV1_GetVersion . . . . .	35
1.5.11.11. MifareUltralightEV1_PwdAuth . . . . .	35
1.5.11.12. MifareUltralightEV1_CheckTearingEvent . . . . .	35
1.5.12. API ISO15693 . . . . .	36
1.5.12.1. ISO15693_GenericCommand . . . . .	36
1.5.12.2. ISO15693_GetSystemInformation . . . . .	36
1.5.12.3. ISO15693_GetSystemInformationExt . . . . .	36
1.5.12.4. ISO15693_GetTagTypeFromUID . . . . .	36
1.5.12.5. ISO15693_GetTagTypeFromSystemInfo . . . . .	37
1.5.12.6. ISO15693_ReadSingleBlock . . . . .	37
1.5.12.7. ISO15693_ReadSingleBlockExt . . . . .	37
1.5.12.8. ISO15693_WriteSingleBlock . . . . .	37
1.5.12.9. ISO15693_WriteSingleBlockExt . . . . .	38

1.5.13. API CRYPTO . . . . .	38
1.5.13.1. Crypto_Init . . . . .	38
1.5.13.2. Encrypt . . . . .	38
1.5.13.3. Decrypt . . . . .	39
1.5.13.4. CBC_ResetInitVector . . . . .	39
1.5.14. API DESFIRE . . . . .	39
1.5.14.1. DESFire_GetApplicationIDs . . . . .	39
1.5.14.2. DESFire_CreateApplication . . . . .	40
1.5.14.3. DESFire_DeleteApplication . . . . .	40
1.5.14.4. DESFire_SelectApplication . . . . .	40
1.5.14.5. DESFire_Authenticate . . . . .	41
1.5.14.6. DESFire_GetKeySettings . . . . .	41
1.5.14.7. DESFire_GetFileIDs . . . . .	41
1.5.14.8. DESFire_GetFileSettings . . . . .	42
1.5.14.9. DESFire_ReadData . . . . .	42
1.5.14.10. DESFire_WriteData . . . . .	42
1.5.14.11. DESFire_GetValue . . . . .	43
1.5.14.12. DESFire_Credit . . . . .	43
1.5.14.13. DESFire_Debit . . . . .	43
1.5.14.14. DESFire_LimitedCredit . . . . .	43
1.5.14.15. DESFire_FreeMem . . . . .	44
1.5.14.16. DESFire_FormatTag . . . . .	44
1.5.14.17. DESFire_CreateDataFile . . . . .	44
1.5.14.18. DESFire_CreateValueFile . . . . .	45
1.5.14.19. DESFire_GetVersion . . . . .	45
1.5.14.20. DESFire_DeleteFile . . . . .	45
1.5.14.21. DESFire_CommitTransaction . . . . .	46
1.5.14.22. DESFire_AbortTransaction . . . . .	46
1.5.14.23. DESFire_GetUID . . . . .	46
1.5.14.24. DESFire_GetKeyVersion . . . . .	46
1.5.14.25. DESFire_ChangeKeySettings . . . . .	47
1.5.14.26. DESFire_ChangeKey . . . . .	47
1.5.14.27. DESFire_ChangeFileSettings . . . . .	47
1.5.14.28. DESFire_DisableFormatCard . . . . .	48
1.5.14.29. DESFire_EnableRandomID . . . . .	48
1.5.14.30. DESFire_SetDefaultKey . . . . .	48
1.5.14.31. DESFire_SetATS . . . . .	48
1.5.14.32. DESFire_CreateRecordFile . . . . .	49
1.5.14.33. DESFire_ReadRecords . . . . .	49
1.5.14.34. DESFire_WriteRecord . . . . .	49
1.5.14.35. DESFire_ClearRecordFile . . . . .	50
1.5.15. API ISO7816 . . . . .	50
1.5.15.1. ISO7816_GetSlotStatus . . . . .	50
1.5.15.2. ISO7816_IccPowerOn . . . . .	50
1.5.15.3. ISO7816_IccPowerOff . . . . .	51
1.5.15.4. ISO7816_SetCommSettings . . . . .	51
1.5.15.5. ISO7816_Transceive . . . . .	51
1.5.15.6. ISO7816_ExchangeAPDU . . . . .	52
1.5.15.7. ISO7816_T0_TPDU . . . . .	52
1.5.15.8. ISO7816_CheckWellKnownCards . . . . .	52

1.5.16. API ICLASS . . . . .	53
1.5.16.1. ICLASS_GetPACBits . . . . .	53
1.5.16.2. ICLASS_SelectPage . . . . .	53
1.5.16.3. ICLASS_Authenticate . . . . .	53
1.5.16.4. ICLASS_ReadBlock . . . . .	54
1.5.16.5. ICLASS_WriteBlock . . . . .	54
1.5.17. API ISO14443 . . . . .	54
1.5.17.1. ISO14443A_GetATS . . . . .	54
1.5.17.2. ISO14443B_GetATQB . . . . .	55
1.5.17.3. ISO14443_4_CheckPresence . . . . .	55
1.5.17.4. ISO14443_4_TDX . . . . .	55
1.5.17.5. ISO14443A_GetATQA . . . . .	55
1.5.17.6. ISO14443A_GetSAK . . . . .	56
1.5.17.7. ISO14443B_GetAnswerToATTRIB . . . . .	56
1.5.17.8. ISO14443_3_TDX . . . . .	56
1.5.17.9. ISO14443A_SearchMultiTag . . . . .	56
1.5.17.10. ISO14443A_SelectTag . . . . .	57
1.5.18. API AT55 . . . . .	57
1.5.18.1. AT55_Begin . . . . .	57
1.5.18.2. AT55_ReadBlock . . . . .	57
1.5.18.3. AT55_ReadBlockProtected . . . . .	57
1.5.18.4. AT55_WriteBlock . . . . .	58
1.5.18.5. AT55_WriteBlockProtected . . . . .	58
1.5.18.6. AT55_WriteBlockAndLock . . . . .	58
1.5.18.7. AT55_WriteBlockProtectedAndLock . . . . .	58
1.5.19. API NFCSNEP . . . . .	59
1.5.19.1. SNEP_Init . . . . .	59
1.5.19.2. SNEP_GetConnectionState . . . . .	59
1.5.19.3. SNEP_GetFragmentByteCount . . . . .	59
1.5.19.4. SNEP_BeginMessage . . . . .	59
1.5.19.5. SNEP_SendMessageFragment . . . . .	60
1.5.19.6. SNEP_TestMessage . . . . .	60
1.5.19.7. SNEP_ReceiveMessageFragment . . . . .	60
1.5.19.8. SNEP_RequestMessage . . . . .	60
1.5.20. API EM4150 . . . . .	61
1.5.20.1. EM4150_Login . . . . .	61
1.5.20.2. EM4150_ReadWord . . . . .	61
1.5.20.3. EM4150_WriteWord . . . . .	61
1.5.20.4. EM4150_WritePassword . . . . .	62
1.5.20.5. EM4150_GetTagInfo . . . . .	62
1.5.21. API FILESYS . . . . .	62
1.5.21.1. FSMount . . . . .	62
1.5.21.2. FSFormat . . . . .	63
1.5.21.3. FSOpen . . . . .	63
1.5.21.4. FSClose . . . . .	63
1.5.21.5. FSCloseAll . . . . .	63
1.5.21.6. FSSeek . . . . .	64
1.5.21.7. FSTell . . . . .	64
1.5.21.8. FSReadBytes . . . . .	64
1.5.21.9. FSWriteBytes . . . . .	64
1.5.21.10. FSFindFirst . . . . .	65

1.5.21.11FSFindNext . . . . .	65
1.5.21.12FSDelete . . . . .	65
1.5.21.13FSRename . . . . .	65
1.5.21.14FSGetStorageInfo . . . . .	66
1.5.22. API MIFAREPLUS . . . . .	66
1.5.22.1. MFP_WritePerso . . . . .	66
1.5.22.2. MFP_CommitPerso . . . . .	66
1.5.22.3. MFP_Authenticate . . . . .	67
1.5.22.4. MFP_ReadBlock . . . . .	67
1.5.22.5. MFP_WriteBlock . . . . .	67
1.5.22.6. MFP_ReadValueBlock . . . . .	67
1.5.22.7. MFP_WriteValueBlock . . . . .	68
1.5.22.8. MFP_IncrementValueBlock . . . . .	68
1.5.22.9. MFP_DecrementValueBlock . . . . .	68
1.5.22.10MFP_CopyValueBlock . . . . .	68
1.5.23. API ADC . . . . .	69
1.5.23.1. ADCInitChannel . . . . .	69
1.5.23.2. ADCGetConversionValue . . . . .	69
1.5.24. API FELICA . . . . .	69
1.5.24.1. FeliCa_TDX . . . . .	69
1.5.24.2. FeliCa_ReadWithoutEncryption . . . . .	70
1.5.24.3. FeliCa_WriteWithoutEncryption . . . . .	70
1.5.24.4. FeliCa_RequestSystemCode . . . . .	70
1.5.24.5. FeliCa_Poll . . . . .	71
1.5.24.6. FeliCa_RequestService . . . . .	71
1.5.25. API SLE44XX . . . . .	71
1.5.25.1. SLE44XX_GetATR . . . . .	71
1.5.25.2. SLE444X_ReadMainMemory . . . . .	72
1.5.25.3. SLE444X_UpdateMainMemory . . . . .	72
1.5.25.4. SLE444X_ReadSecurityMemory . . . . .	72
1.5.25.5. SLE444X_UpdateSecurityMemory . . . . .	72
1.5.25.6. SLE444X_ReadProtectionMemory . . . . .	73
1.5.25.7. SLE444X_WriteProtectionMemory . . . . .	73
1.5.25.8. SLE444X_CompareVerificationData . . . . .	73
1.5.25.9. SLE44X8_ReadMainMemory . . . . .	73
1.5.25.10SLE44X8_WriteErrorCounter . . . . .	74
1.5.25.11SLE44X8_VerifyPSCByte . . . . .	74
1.5.25.12SLE44X8_UpdateMainMemory . . . . .	74
1.5.26. API NTAG . . . . .	75
1.5.26.1. NTAG_Read . . . . .	75
1.5.26.2. NTAG_Write . . . . .	75
1.5.26.3. NTAG_FastRead . . . . .	75
1.5.26.4. NTAG_ReadCounter . . . . .	76
1.5.26.5. NTAG_ReadSig . . . . .	76
1.5.26.6. NTAG_GetVersion . . . . .	76
1.5.26.7. NTAG_PwdAuth . . . . .	76
1.5.26.8. NTAG_SectorSelect . . . . .	77
1.5.27. API SRX . . . . .	77
1.5.27.1. SRX_ReadBlock . . . . .	77
1.5.27.2. SRX_WriteBlock . . . . .	77

1.5.28. API SAMAVX . . . . .	78
1.5.28.1. SAMAVx_AuthenticateHost . . . . .	78
1.5.28.2. SAMAVx_GetKeyEntry . . . . .	78
1.5.29. API EM4102 . . . . .	78
1.5.29.1. EM4102_GetTagInfo . . . . .	78
1.5.30. API SPI . . . . .	79
1.5.30.1. SPIInit . . . . .	79
1.5.30.2. SPIDeInit . . . . .	79
1.5.30.3. SPIMasterBeginTransfer . . . . .	79
1.5.30.4. SPIMasterEndTransfer . . . . .	79
1.5.30.5. SPITransmit . . . . .	80
1.5.30.6. SPIReceive . . . . .	80
1.5.30.7. SPITransceive . . . . .	80
1.5.31. API BLE . . . . .	81
1.5.31.1. BLEPresetConfig . . . . .	81
1.5.31.2. BLEPresetUserData . . . . .	81
1.5.31.3. BLEInit . . . . .	81
1.5.31.4. BLECheckEvent . . . . .	82
1.5.31.5. BLEGetAddress . . . . .	82
1.5.31.6. BLEGetVersion . . . . .	82
1.5.31.7. BLEGetEnvironment . . . . .	82
1.5.31.8. BLEGetGattServerAttributeValue . . . . .	83
1.5.31.9. BLESetGattServerAttributeValue . . . . .	83
1.5.31.10 BLERequestRssi . . . . .	83
1.5.31.11 BLERequestEndpointClose . . . . .	83
1.5.31.12 BLEGetGattServerCharacteristicStatus . . . . .	84
1.5.31.13 BLEFindGattServerAttribute . . . . .	84
1.5.31.14 BLEDiscover . . . . .	84
1.5.31.15 BLECheckDiscoveredString . . . . .	84
1.5.31.16 BLEConnectToDevice . . . . .	85
1.5.31.17 BLEDisconnectFromDevice . . . . .	85
1.5.31.18 BLEGattGetAttribute . . . . .	85
1.5.31.19 BLEGattGetValue . . . . .	85
1.5.31.20 BLEGattSetValue . . . . .	86
1.5.31.21 BLECommand . . . . .	86
1.5.31.22 BLESecurity . . . . .	86
1.5.31.23 BLESecuritySetOob . . . . .	87
1.5.31.24 BLESecurityUseScOob . . . . .	87
1.5.31.25 BLESetStreamingUUID . . . . .	87
1.5.31.26 BLESetStreamingMode . . . . .	88
1.5.31.27 BLEGetDiscoveredData . . . . .	88
1.5.32. API I2CCARD . . . . .	88
1.5.32.1. I2CCard_Read . . . . .	88
1.5.32.2. I2CCard_Write . . . . .	89
1.5.33. API TOPAZ . . . . .	89
1.5.33.1. TopazRID . . . . .	89
1.5.33.2. TopazReadByte . . . . .	89
1.5.33.3. TopazReadAllBlocks . . . . .	90
1.5.33.4. TopazWriteByteWithErase . . . . .	90
1.5.33.5. TopazWriteByteNoErase . . . . .	90



1.5.34. API CTS . . . . .	91
1.5.34.1. CTS_ReadBlock . . . . .	91
1.5.34.2. CTS_WriteBlock . . . . .	91
1.5.34.3. CTS_UpdateBlock . . . . .	91
1.5.35. API EM4305 . . . . .	92
1.5.35.1. EM4305_Begin . . . . .	92
1.5.35.2. EM4305_Read . . . . .	92
1.5.35.3. EM4305_Write . . . . .	92
1.5.35.4. EM4305_Login . . . . .	92
1.5.35.5. EM4305_Protect . . . . .	93
1.5.35.6. EM4305_Disable . . . . .	93
A. How to Set Specific Tags in Simple Protocol . . . . .	94
A.1. Example with Enabling Only MIFARE . . . . .	94
A.2. Example with Felica and HID Prox Only . . . . .	94
B. Disclaimer . . . . .	97

# 1. Simple Protocol

This document describes the serial protocol of TWN4.

In order to operate this protocol, a firmware type TWN4\_Cxvvv\_PRSwww.bix is required, where vvv and www are the version numbers.

A firmware as mentioned above combines virtual USB (CDC) or true serial communication with a TWN4 app, which implements the simple protocol (PRS = PRotocol Simple).

This protocol is called simple because it is based on a communication with ASCII characters which can also be tested manually by using a terminal program. There is no additional overhead for things like packet repetition, address bytes...

The simple protocol is also available in binary mode. This means, that the data is not transmitted via ASCII characters but as single bytes.

Moreover it is possible to add a CRC at the end of every transmission. This lets you detect transmission errors.

The communication is based on a command/response structure: TWN4 will only send data to the host as a response of a command. Command and response are lines of bytes terminated by a carriage return. Carriage return is not shown explicitly anymore in the following documentation. A byte is always represented and transmitted by two hexadecimal ASCII characters.

## 1.1. Command

A command always starts with two bytes which reflect the API and function number to be executed.

## 1.2. Response

A response always starts with a byte, which reflects execution of the command on protocol level. Following possible error values:

ERR_NONE	0
ERR_UNKNOWN_FUNCTION	1
ERR_MISSING_PARAMETER	2
ERR_UNUSED_PARAMETERS	3
ERR_INVALID_FUNCTION	4
ERR_PARSER	5

## 1.3. Data Transmission

Data can be transmitted in two ways:

- by sending ASCII characters
- by sending binary values

Standard communication setting is **ASCII**, **CRC off** with **9600 baud**.

Communication settings can be done in the app **App\_PRS104\_Simple\_Protocol.c** in the folder \Apps\Samples\Simple Protocol in the TWN4DevPack.

### 1.3.1. ASCII

To transmit a value of e.g. 0x1F, it is necessary to split this into two ASCII characters '1' and 'F'. These characters has to be sent sequentially.

### 1.3.2. Binary

To transmit a value of e.g. 0x1F, it can be sent directly in binary format. The first two bytes (LSB first) indicate the number of the following bytes.

<Length of command bytes (2 bytes, LSB first)> <command bytes>

See 1.3.4 for an example.

### 1.3.3. CRC

On both ASCII and binary format, a CRC can be added at the end of each transmission. In case of binary format, the length bytes are not part of the CRC calculation. The CRC is calculated as follows:

```
uint16_t UpdateCRC(uint16_t CRC,byte Byte)
{
    // Update CCITT CRC (reverse polynom 0x8408)
    Byte ^= (byte)CRC;
    Byte ^= (byte)(Byte << 4);
    return (uint16_t)((((Byte << 8) | (CRC >> 8)) ^ (Byte >> 4) ^ (Byte << 3)));
}
```

The CRC calculation starts with CRC = 0xFFFF

### 1.3.4. Reference messages

The following table shows reference messages for function GetUSBType

Mode	CRC	Command (Host -> TWN4)	Response (TWN4 -> Host)
ASCII	Off	""0005\r""	""0001\r""
	On	""000515A7\r""	""000131E1\r""
Binary	Off	0x02 0x00 0x00 0x05	0x02 0x00 0x00 0x01
	On	0x04 0x00 0x00 0x05 0x15 0xA7	0x04 0x00 0x00 0x01 0x31 0xE1

## 1.4. Data Types

The description of the commands is using data types, which have to be built-up as follows:

Data Type	Description
[Byte]:	One single byte (sent as two hex digits)
[UInt16]:	Two bytes (LSB first)
[UInt32]:	Four bytes (LSB first)
[Bool]:	One single byte which can hold two values: 0 or 1
[Byte Array(n)]:	A sequence of bytes with known and fixed number of bytes. The number of bytes is not transferred explicitly, because both host and TWN4 do know this number.
[Byte Array(Var)]:	A sequence of bytes, where the first byte holds the number of following bytes
[Byte Array(Var), x LB]:	A sequence of bytes, where the first x bytes hold the number of following bytes
[ASCII string]:	A sequence of bytes which contain ASCII characters, except the first byte which holds the number of following bytes

In Simple Protocol, all numbers are sent with LSB first. For example, the number 0x1234 has to be sent as 3412.

## 1.5. Commands

### 1.5.1. API SYS

#### 1.5.1.1. Reset

Command:	[0001]
Response:	[00]
Example Command: Response:	0001

**1.5.1.2. StartBootloader**

Command:	[0002]
Response:	[00]
Example	
Command:	0002
Response:	

**1.5.1.3. GetSysTicks**

Command:	[0003]
Response:	[00][UInt32: <i>Ticks</i> ]
Example	
Command:	0003
Response:	00D3480700 (Ticks: 477395)

**1.5.1.4. GetVersionString**

Command:	[0004][Byte: <i>MaxLen</i> ]
Response:	[00][ASCII string: <i>Version</i> ]
Example	
Command:	0004FF (MaxLen: FF)
Response:	001D54574E342F42312E30332F434346312E35372F505253312E3033-2F5049 (Version: TWN4/B1.03/CCF1.57/PRS1.03/PI)

**1.5.1.5. GetUSBType**

Command:	[0005]
Response:	[00][Byte: <i>Type</i> ]
Example	
Command:	0005
Response:	0001 (Type: 1)

**1.5.1.6. GetDeviceType**

Command:	[0006]
Response:	[00][Byte: <i>Type</i> ]
Example	
Command:	0006
Response:	000B (Type: 11)

**1.5.1.7. Sleep**

Command:	[0007][UInt32: <i>Ticks</i> ][UInt32: <i>Flags</i> ]
Response:	[00][Byte: <i>Result</i> ]
Example	
Command:	0007E803000001000000 (Ticks: E8030000, Flags: 01000000)
Response:	0000 (Result: 0)

**1.5.1.8. GetDeviceUID**

Command:	[0008]
Response:	[00][Byte Array(12): <i>UID</i> ]
Example	
Command:	0008
Response:	002D002F000B47303531353233 (UID: 2D002F000B47303531353233)

**1.5.1.9. SetParameters**

Command:	[0009][Byte Array(Var): <i>TLV</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	00090707010103010200 (TLV: 07010103010200)
Response:	0001 (Result: true)

**1.5.1.10. GetLastError**

Command:	[000A]
Response:	[00][UInt32: <i>LastError</i> ]
Example	
Command:	000A
Response:	00CB000000 (LastError: 203)

**1.5.1.11. GetProdSerNo**

Command:	[000D][Byte: <i>MaxLen</i> ]
Response:	[00][ASCII string: <i>SerNo</i> ]
Example	
Command:	000DFF (MaxLen: FF)
Response:	001031323334353637383930313233343536 (SerNo: 1234567890123456)

**1.5.2. API IO****1.5.2.1. WriteByte**

Command:	[0100][Byte: <i>Channel</i> ][Byte: <i>Byte</i> ]
Response:	[00]
Example	
Command:	01000041 (Channel: 00, Byte: 41)
Response:	00

**1.5.2.2. ReadByte**

Command:	[0101][Byte: <i>Channel</i> ]
Response:	[00][Byte: <i>Byte</i> ]
Example	
Command:	010100 (Channel: 00)
Response:	0000 (Byte: 0)

**1.5.2.3. TestEmpty**

Command:	[0102][Byte: <i>Channel</i> ][Byte: <i>Dir</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	01020001 (Channel: 00, Dir: 01)
Response:	0001 (Result: Yes)

**1.5.2.4. TestFull**

Command:	[0103][Byte: <i>Channel</i> ][Byte: <i>Dir</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	01030001 (Channel: 00, Dir: 01)
Response:	0000 (Result: No)

**1.5.2.5. GetBufferSize**

Command:	[0104][Byte: <i>Channel</i> ][Byte: <i>Dir</i> ]
Response:	[00][UInt16: <i>BufferSize</i> ]
Example	
Command:	01040001 (Channel: 00, Dir: 01)
Response:	000000 (BufferSize: 0)

**1.5.2.6. GetByteCount**

Command:	[0105][Byte: <i>Channel</i> ][Byte: <i>Dir</i> ]
Response:	[00][UInt16: <i>ByteCount</i> ]
Example	
Command:	01050001 (Channel: 00, Dir: 01)
Response:	000000 (ByteCount: 0)



**1.5.2.7. SetCOMParameters**

Command:	[0109][Byte: <i>Channel</i> ][UInt32: <i>Baudrate</i> ][Byte: <i>WordLength</i> ][Byte: <i>Parity</i> ][Byte: <i>StopBits</i> ][Byte: <i>FlowControl</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0109028025000008000100 (Channel: 02, Baudrate: 80250000, WordLength: 08, Parity: 00, StopBits: 01, FlowControl: 00)
Response:	0001 (Result: true)

**1.5.2.8. GetUSBDeviceState**

Command:	[010A]
Response:	[00][Byte: <i>State</i> ]
Example	
Command:	010A
Response:	0003 (State: USB_DEVICE_STATE_CONFIGURED)

**1.5.2.9. GetHostChannel**

Command:	[010B]
Response:	[00][Byte: <i>Channel</i> ]
Example	
Command:	010B
Response:	0001 (Channel: CHANNEL_USB)

**1.5.2.10. USBRemoteWakeup**

Command:	[010C]
Response:	[00]
Example	
Command:	010C
Response:	00

**1.5.2.11. WriteBytes**

Command:	[010D][Byte: <i>Channel</i> ][Byte Array(Var), 2 LB: <i>Bytes</i> ]
Response:	[00][UInt16: <i>BytesWritten</i> ]
Example	
Command:	010D020300000815 (Channel: 02, Bytes: 000815)
Response:	000300 (BytesWritten: 3)

**1.5.2.12. ReadBytes**

Command:	[010E][Byte: <i>Channel</i> ][UInt16: <i>MaxBytes</i> ]
Response:	[00][Byte Array(Var), 2 LB: <i>Bytes</i> ]
Example	
Command:	010E020F00 (Channel: 02, MaxBytes: 0F00)
Response:	000300000815 (Bytes: 000815)

**1.5.3. API PERIPH****1.5.3.1. GPIOConfigureOutputs**

Command:	[0400][Byte: <i>Bits</i> ][Byte: <i>PullUpDown</i> ][Byte: <i>OutputType</i> ]
Response:	[00]
Example	
Command:	0400010000 (Bits: 01, PullUpDown: 00, OutputType: 00)
Response:	00

**1.5.3.2. GPIOConfigureInputs**

Command:	[0401][Byte: <i>Bits</i> ][Byte: <i>PullUpDown</i> ]
Response:	[00]
Example	
Command:	04010100 (Bits: 01, PullUpDown: 00)
Response:	00

**1.5.3.3. GPIOSetBits**

Command:	[0402][Byte: <i>Bits</i> ]
Response:	[00]
Example	
Command:	040201 (Bits: 01)
Response:	00

**1.5.3.4. GPIOClearBits**

Command:	[0403][Byte: <i>Bits</i> ]
Response:	[00]
Example	
Command:	040301 (Bits: 01)
Response:	00

**1.5.3.5. GPIToggleBits**

Command:	[0404][Byte: <i>Bits</i> ]
Response:	[00]
Example	
Command:	040401 (Bits: 01)
Response:	00

**1.5.3.6. GPIOBlinkBits**

Command:	[0405][Byte: <i>Bits</i> ][UInt16: <i>TimeHi</i> ][UInt16: <i>TimeLo</i> ]
Response:	[00]
Example	
Command:	04050164006400 (Bits: 01, TimeHi: 6400, TimeLo: 6400)
Response:	00

**1.5.3.7. GPIOTestBit**

Command:	[0406][Byte: <i>Bit</i> ]
Response:	[00][Byte: <i>Result</i> ]
Example	
Command:	040601 (Bit: 01)
Response:	0000 (Result: 0)

**1.5.3.8. Beep**

Command:	[0407][Byte: <i>Volume</i> ][UInt16: <i>Frequency</i> ][UInt16: <i>OnTime</i> ][UInt16: <i>OffTime</i> ]
Response:	[00]
Example	
Command:	0407646009F401F401 (Volume: 64, Frequency: 6009, OnTime: F401, OffTime: F401)
Response:	00

**1.5.3.9. DiagLEDOn**

Command:	[0408]
Response:	[00]
Example	
Command:	0408
Response:	00

**1.5.3.10. DiagLEDOff**

Command:	[0409]
Response:	[00]
Example	
Command:	0409
Response:	00

**1.5.3.11. DiagLEDToggle**

Command:	[040A]
Response:	[00]
Example	
Command:	040A
Response:	00

**1.5.3.12. DiagLEDIsOn**

Command:	[040B]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	040B
Response:	0000 (Result: No)

**1.5.3.13. SendWiegand**

Command:	[040C][Byte: <i>GPIOData0</i> ][Byte: <i>GPIOData1</i> ][UInt16: <i>PulseTime</i> ][UInt16: <i>IntervalTime</i> ][Byte Array(Var): <i>Bits</i> ][Byte: <i>BitCount</i> ]
Response:	[00]
Example	
Command:	040C08106400E80301AA08 (GPIOData0: 08, GPIOData1: 10, PulseTime: 6400, IntervalTime: E803, Bits: AA, BitCount: 08)
Response:	00

**1.5.3.14. SendOmron**

Command:	[040D][Byte: <i>GPIOClock</i> ][Byte: <i>GPIOData</i> ][UInt16: <i>T1</i> ][UInt16: <i>T2</i> ][UInt16: <i>T3</i> ][Byte Array(Var): <i>Bits</i> ][Byte: <i>BitCount</i> ]
Response:	[00]
Example	
Command:	040D0810F401F401F40101AA08 (GPIOClock: 08, GPIOData: 10, T1: F401, T2: F401, T3: F401, Bits: AA, BitCount: 08)
Response:	00

**1.5.3.15. LEDInit**

Command:	[0410][Byte: <i>LEDs</i> ]
Response:	[00]
Example	
Command:	041007 (LEDs: 07)
Response:	00

**1.5.3.16. LEDOn**

Command:	[0411][Byte: <i>LEDs</i> ]
Response:	[00]
Example	
Command:	041107 (LEDs: 07)
Response:	00

**1.5.3.17. LEDOff**

Command:	[0412][Byte: <i>LEDs</i> ]
Response:	[00]
Example	
Command:	041207 (LEDs: 07)
Response:	00

**1.5.3.18. LEDToggle**

Command:	[0413][Byte: <i>LEDs</i> ]
Response:	[00]
Example	
Command:	041307 (LEDs: 07)
Response:	00

**1.5.3.19. LEDBlink**

Command:	[0414][Byte: <i>LEDs</i> ][UInt16: <i>TimeOn</i> ][UInt16: <i>TimeOff</i> ]
Response:	[00]
Example	
Command:	041407F401F401 (LEDs: 07, TimeOn: F401, TimeOff: F401)
Response:	00

**1.5.3.20. BeepOn**

Command:	[0416][Byte: <i>Volume</i> ][UInt16: <i>Frequency</i> ]
Response:	[00]
Example	
Command:	0416646009 (Volume: 64, Frequency: 6009)
Response:	00

**1.5.3.21. BeepOff**

Command:	[0417]
Response:	[00]
Example	
Command:	0417
Response:	00

**1.5.4. API RF****1.5.4.1. SearchTag**

Command:	[0500][Byte: <i>MaxIDBytes</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>TagType</i> ][Byte: <i>IDBitCount</i> ][Byte Array(Var): <i>ID</i> ]
Example	
Command:	050010 (MaxIDBytes: 10)
Response:	000180200466CF4DC2 (Result: true, TagType: ISO14443A/MIFARE, IDBitCount: 32, ID: 66CF4DC2)

**1.5.4.2. SetRFOff**

Command:	[0501]
Response:	[00]
Example	
Command:	0501
Response:	00

**1.5.4.3. SetTagTypes**

Command:	[0502][UInt32: <i>TagTypesLF</i> ][UInt32: <i>TagTypesHF</i> ]
Response:	[00]
Example	
Command:	0502FFFFFFFFFFFFFFFFFFFF ( <i>TagTypesLF</i> : FFFFFFFF, <i>TagTypesHF</i> : FFFFFFFF)
Response:	00

**1.5.4.4. GetTagTypes**

Command:	[0503]
Response:	[00][UInt32: <i>LFTagTypes</i> ][UInt32: <i>HFTagTypes</i> ]
Example	
Command:	0503
Response:	002FFE0700F7000000 ( <i>LFTagTypes</i> : 523823, <i>HFTagTypes</i> : 247)

**1.5.4.5. GetSupportedTagTypes**

Command:	[0504]
Response:	[00][UInt32: <i>LFTagTypes</i> ][UInt32: <i>HFTagTypes</i> ]
Example	
Command:	0504
Response:	002FFE0700F7000000 ( <i>LFTagTypes</i> : 523823, <i>HFTagTypes</i> : 247)



**1.5.5. API TILF****1.5.5.1. TILF\_SearchTag**

Command:	[0600][Byte: <i>MaxIDBytes</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>IDBitCount</i> ][Byte Array(Var): <i>ID</i> ]
Example	
Command:	060010 (MaxIDBytes: 10)
Response:	0001400800000000042E8653 (Result: true, IDBitCount: 64, ID: 00000000042E8653)

**1.5.5.2. TILF\_ChargeOnlyRead**

Command:	[0601]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>Data</i> ]
Example	
Command:	0601
Response:	000100000000042E8653 (Result: true, Data: 00000000042E8653)

**1.5.5.3. TILF\_ChargeOnlyReadLo**

Command:	[0602]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>ReadData</i> ]
Example	
Command:	0602
Response:	000100007F7E7EFFFFDFFFFFFFFFFFFFFFFFFFFD (Result: true, ReadData: 00007F7E7EFFFFDFFFFFFFFFFFFFFFFFFFFD)

**1.5.5.4. TILF\_SPProgramPage**

Command:	[0603][Byte Array(8): <i>WriteData</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>ReadData</i> ]
Example	
Command:	06030001020304050607 (WriteData: 0001020304050607)
Response:	000100007ECA61742000000000DADF7E0000 (Result: true, ReadData: 00007ECA61742000000000DADF7E0000)

**1.5.5.5. TILF\_SPProgramPageLo**

Command:	[0604][Byte Array(10): <i>WriteData</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>ReadData</i> ]
Example	
Command:	060400010203040506070809 (WriteData: 00010203040506070809)
Response:	000100007ECA61742000000000DADF7E0000 (Result: true, ReadData: 00007ECA61742000000000DADF7E0000)

**1.5.5.6. TILF\_MPGeneralReadPage**

Command:	[0605][Byte: <i>Address</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>ReadData</i> ]
Example	
Command:	060500 (Address: 00)
Response:	0001000000000042E8653 (Result: true, ReadData: 000000000042E8653)

**1.5.5.7. TILF\_MPSelectiveReadPage**

Command:	[0606][Byte: <i>Address</i> ][Byte Array(3): <i>SelectiveAddress</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>ReadData</i> ]
Example	
Command:	060600000102 (Address: 00, SelectiveAddress: 000102)
Response:	0001000000000042E8653 (Result: true, ReadData: 000000000042E8653)

**1.5.5.8. TILF\_MPProgramPage**

Command:	[0607][Byte: <i>Address</i> ][Byte Array(8): <i>WriteData</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>ReadData</i> ]
Example	
Command:	0607004469726563746F72 (Address: 00, WriteData: 4469726563746F72)
Response:	0001000000000042E8653 (Result: true, ReadData: 000000000042E8653)

**1.5.5.9. TILF\_MPSelectiveProgramPage**

Command:	[0608][Byte: <i>Address</i> ][Byte Array(3): <i>SelectiveAddress</i> ][Byte Array(8): <i>WriteData</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>ReadData</i> ]
Example	
Command:	0608000001024469726563746F72 (Address: 00, SelectiveAddress: 000102, WriteData: 4469726563746F72)
Response:	000100000000042E8653 (Result: true, ReadData: 00000000042E8653)

**1.5.5.10. TILF\_MPLockPage**

Command:	[0609][Byte: <i>Address</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>ReadData</i> ]
Example	
Command:	060900 (Address: 00)
Response:	0000 (Result: fail, ReadData: )

**1.5.5.11. TILF\_MPSelectiveLockPage**

Command:	[060A][Byte: <i>Address</i> ][Byte Array(3): <i>SelectiveAddress</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>ReadData</i> ]
Example	
Command:	060A00000102 (Address: 00, SelectiveAddress: 000102)
Response:	0000 (Result: fail, ReadData: )

**1.5.5.12. TILF\_MPGeneralReadPageLo**

Command:	[060B][Byte: <i>Address</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>ReadData</i> ]
Example	
Command:	060B00 (Address: 00)
Response:	000100007ECA61742000000000DADF7E0000 (Result: true, ReadData: 00007ECA61742000000000DADF7E0000)

**1.5.5.13. TILF\_MPSelectiveReadPageLo**

Command:	[060C][Byte: <i>Address</i> ][Byte Array(3): <i>SelectiveAddress</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>ReadData</i> ]
Example	
Command:	060C00000102 (Address: 00, SelectiveAddress: 000102)
Response:	000100007ECA61742000000000DADF7E0000 (Result: true, ReadData: 00007ECA61742000000000DADF7E0000)

**1.5.5.14. TILF\_MPProgramPageLo**

Command:	[060D][Byte: <i>Address</i> ][Byte Array(10): <i>WriteData</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>ReadData</i> ]
Example	
Command:	060D00536F6D6520746578742E (Address: 00, WriteData: 536F6D6520746578742E)
Response:	000100007ECA61742000000000DADF7E0000 (Result: true, ReadData: 00007ECA61742000000000DADF7E0000)

**1.5.5.15. TILF\_MPSelectiveProgramPageLo**

Command:	[060E][Byte: <i>Address</i> ][Byte Array(3): <i>SelectiveAddress</i> ][Byte Array(10): <i>WriteData</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>ReadData</i> ]
Example	
Command:	060E00000102536F6D6520746578742E (Address: 00, SelectiveAddress: 000102, WriteData: 536F6D6520746578742E)
Response:	000100007ECA61742000000000DADF7E0000 (Result: true, ReadData: 00007ECA61742000000000DADF7E0000)

**1.5.5.16. TILF\_MPLockPageLo**

Command:	[060F][Byte: <i>Address</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>ReadData</i> ]
Example	
Command:	060F00 (Address: 00)
Response:	0000 (Result: fail, ReadData: )

**1.5.5.17. TILF\_MPSelectiveLockPageLo**

Command:	[0610][Byte: <i>Address</i> ][Byte Array(3): <i>SelectiveAddress</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>ReadData</i> ]
Example	
Command:	061000000102 (Address: 00, SelectiveAddress: 000102)
Response:	000100007FEFFFFFFFBFF7FFFAFFFFFFFFF7 (Result: true, ReadData: 00007FEFFFFFFFBFF7FFFAFFFFFFFFF7)

**1.5.5.18. TILF\_MUGeneralReadPage**

Command:	[0611][Byte: <i>Address</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(7): <i>Data</i> ]
Example	
Command:	061100 (Address: 00)
Response:	0000 (Result: fail, Data: )

**1.5.5.19. TILF\_MUSelectiveReadPage**

Command:	[0612][Byte: <i>Address</i> ][Byte: <i>SelectiveAddress</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(7): <i>Data</i> ]
Example	
Command:	06120000 (Address: 00, SelectiveAddress: 00)
Response:	0000 (Result: fail, Data: )

**1.5.5.20. TILF\_MUSpecialReadPage**

Command:	[0613][Byte: <i>Address</i> ][Byte Array(5): <i>SpecialAddress1</i> ][Byte Array(3): <i>SpecialAddress2</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(7): <i>Data</i> ]
Example	
Command:	0613000001020304000102 (Address: 00, SpecialAddress1: 0001020304, SpecialAddress2: 000102)
Response:	0000 (Result: fail, Data: )

**1.5.5.21. TILF\_MUProgramPage**

Command:	[0614][Byte: <i>Address</i> ][Byte Array(5): <i>WriteData</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(7): <i>ReadData</i> ]
Example	
Command:	06140048656C6C6F (Address: 00, WriteData: 48656C6C6F)
Response:	0000 (Result: fail, ReadData: )

**1.5.5.22. TILF\_MUSelectiveProgramPage**

Command:	[0615][Byte: <i>Address</i> ][Byte: <i>SelectiveAddress</i> ][Byte Array(5): <i>WriteData</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(7): <i>ReadData</i> ]
Example	
Command:	0615000048656C6C6F (Address: 00, SelectiveAddress: 00, WriteData: 48656C6C6F)
Response:	0000 (Result: fail, ReadData: )

**1.5.5.23. TILF\_MUSpecialProgramPage**

Command:	[0616][Byte: <i>Address</i> ][Byte Array(5): <i>SpecialAddress1</i> ][Byte Array(3): <i>SpecialAddress2</i> ][Byte Array(5): <i>WriteData</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(7): <i>ReadData</i> ]
Example	
Command:	061600000102030400010248656C6C6F (Address: 00, SpecialAddress1: 0001020304, SpecialAddress2: 000102, WriteData: 48656C6C6F)
Response:	0000 (Result: fail, ReadData: )

**1.5.5.24. TILF\_MULockPage**

Command:	[0617][Byte: <i>Address</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(7): <i>ReadData</i> ]
Example	
Command:	061700 (Address: 00)
Response:	0000 (Result: fail, ReadData: )

**1.5.5.25. TILF\_MUSelectiveLockPage**

Command:	[0618][Byte: <i>Address</i> ][Byte: <i>SelectiveAddress</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(7): <i>ReadData</i> ]
Example	
Command:	06180000 (Address: 00, SelectiveAddress: 00)
Response:	0000 (Result: fail, ReadData: )

**1.5.5.26. TILF\_MUSpecialLockPage**

Command:	[0619][Byte: <i>Address</i> ][Byte Array(5): <i>SpecialAddress1</i> ][Byte Array(3): <i>SpecialAddress2</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(7): <i>ReadData</i> ]
Example	
Command:	0619000001020304000102 (Address: 00, SpecialAddress1: 0001020304, SpecialAddress2: 000102)
Response:	0000 (Result: fail, ReadData: )

**1.5.6. API HITAG1S****1.5.6.1. Hitag1S\_ReadPage**

Command:	[0701][Byte: <i>PageAddress</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>Data</i> ]
Example	
Command:	070104 (PageAddress: 04)
Response:	0001FF8CA64A (Result: true, Data: FF8CA64A)

**1.5.6.2. Hitag1S\_ReadBlock**

Command:	[0702][Byte: <i>BlockAddress</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>Data</i> ]
Example	
Command:	070204 (BlockAddress: 04)
Response:	0001100001020398F8C802FFFFFFFFFFFFFFFFFFFF (Result: true, Data: 0001020398F8C802FFFFFFFFFFFFFFFFFFFF)

**1.5.6.3. Hitag1S\_WritePage**

Command:	[0703][Byte: <i>PageAddress</i> ][Byte Array(4): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	07030407040400 (PageAddress: 04, Data: 07040400)
Response:	0001 (Result: true)



#### 1.5.6.4. Hitag1S\_WriteBlock

Command:	[0704][Byte: <i>BlockAddress</i> ][Byte Array(16): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>BytesWritten</i> ]
Example	
Command:	070404000 (BlockAddress: 04, Data: 0000000000000000000000000000000000)
Response:	000110 (Result: true, BytesWritten: 16)

#### 1.5.6.5. Hitag1S\_Halt

Command:	[0705]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0705
Response:	0001 (Result: true)

### 1.5.7. API HITAG2

#### 1.5.7.1. Hitag2\_ReadPage

Command:	[0801][Byte: <i>PageAddress</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>Data</i> ]
Example	
Command:	080104 (PageAddress: 04)
Response:	0001FF800000 (Result: true, Data: FF800000)

**1.5.7.2. Hitag2\_WritePage**

Command:	[0802][Byte: <i>PageAddress</i> ][Byte Array(4): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	080204FF800000 (PageAddress: 04, Data: FF800000)
Response:	0001 (Result: true)

**1.5.7.3. Hitag2\_Halt**

Command:	[0803]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0803
Response:	0001 (Result: true)

**1.5.7.4. Hitag2\_SetPassword**

Command:	[0804][Byte Array(4): <i>Password</i> ]
Response:	[00]
Example	
Command:	080400010203 (Password: 00010203)
Response:	00

**1.5.8. API SM4X00****1.5.8.1. SM4X00\_GenericRaw**

Command:	[0900][Byte Array(Var): <i>TXData</i> ][Byte: <i>MaxRXDataLength</i> ][UInt16: <i>Timeout</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>RXData</i> ]
Example	
Command:	090005040A00000040B80B (TXData: 040A000000, MaxRXDataLength: 40, Timeout: B80B)
Response:	00010D0A0000009010501001801030100 (Result: true, RXData: 0A0000009010501001801030100)

**1.5.8.2. SM4X00\_Generic**

Command:	[0901][Byte Array(Var): <i>TXData</i> ][Byte: <i>MaxRXDataLength</i> ][UInt16: <i>Timeout</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>RXData</i> ]
Example	
Command:	0901020A0040B80B (TXData: 0A00, MaxRXDataLength: 40, Timeout: B80B)
Response:	0001100F0A000009010501001801030100EB63 (Result: true, RXData: 0F0A000009010501001801030100EB63)

**1.5.9. API I2C****1.5.9.1. I2CInit**

Command:	[0A00][UInt16: <i>Mode</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0A000000 (Mode: 0000)
Response:	0001 (Result: true)

**1.5.9.2. I2CDeInit**

Command:	[0A01]
Response:	[00]
Example	
Command:	0A01
Response:	00

**1.5.9.3. I2CMasterStart**

Command:	[0A02]
Response:	[00]
Example	
Command:	0A02
Response:	00

**1.5.9.4. I2CMasterStop**

Command:	[0A03]
Response:	[00]
Example	
Command:	0A03
Response:	00

**1.5.9.5. I2CMasterTransmitByte**

Command:	[0A04][Byte: <i>Data</i> ]
Response:	[00]
Example	
Command:	0A0400 (Data: 00)
Response:	00

**1.5.9.6. I2CMasterReceiveByte**

Command:	[0A05]
Response:	[00][Byte: <i>Data</i> ]
Example	
Command:	0A05
Response:	0000 (Data: 0)

**1.5.9.7. I2CMasterBeginWrite**

Command:	[0A06][Byte: <i>Address</i> ]
Response:	[00]
Example	
Command:	0A0630 (Address: 30)
Response:	00

**1.5.9.8. I2CMasterBeginRead**

Command:	[0A07][Byte: <i>Address</i> ]
Response:	[00]
Example	
Command:	0A0730 (Address: 30)
Response:	00

**1.5.9.9. I2CMasterSetAck**

Command:	[0A08][Byte: <i>SetOn</i> ]
Response:	[00]
Example	
Command:	0A0801 (SetOn: 01)
Response:	00

**1.5.10. API MIFARECLASSIC****1.5.10.1. MifareClassic\_Login**

Command:	[0B00][Byte Array(6): <i>Key</i> ][Byte: <i>KeyType</i> ][Byte: <i>Sector</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0B00A0A1A2A3A4A50000 (Key: A0A1A2A3A4A5, KeyType: 00, Sector: 00)
Response:	0001 (Result: true)

**1.5.10.2. MifareClassic\_ReadBlock**

Command:	[0B01][Byte: <i>Block</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>Data</i> ]
Example	
Command:	0B0102 (Block: 02)
Response:	00010000000000000000000000000000 (Result: true, Data: 00000000000000000000000000000000)

### 1.5.10.3. MifareClassic\_WriteBlock

Command:	[0B02][Byte: <i>Block</i> ][Byte Array(16): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0B020200 (Block: 02, Data: 00)
Response:	0001 (Result: true)

#### 1.5.10.4. MifareClassic\_ReadValueBlock

Command:	[0B03][Byte: <i>Block</i> ]
Response:	[00][Bool: <i>Result</i> ][UInt32: <i>Value</i> ]
Example	
Command:	0B0302 (Block: 02)
Response:	000101000000 (Result: true, Value: 1)

#### 1.5.10.5. MifareClassic\_WriteValueBlock

Command:	[0B04][Byte: <i>Block</i> ][UInt32: <i>Value</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0B040201000000 (Block: 02, Value: 01000000)
Response:	0001 (Result: true)

#### 1.5.10.6. MifareClassic\_IncrementValueBlock

Command:	[0B05][Byte: <i>Block</i> ][UInt32: <i>Value</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0B050201000000 (Block: 02, Value: 01000000)
Response:	0001 (Result: true)

**1.5.10.7. MifareClassic\_DecrementValueBlock**

Command:	[0B06][Byte: <i>Block</i> ][UInt32: <i>Value</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0B060201000000 (Block: 02, Value: 01000000)
Response:	0001 (Result: true)

**1.5.10.8. MifareClassic\_CopyValueBlock**

Command:	[0B07][Byte: <i>SourceBlock</i> ][Byte: <i>DestBlock</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0B07090A (SourceBlock: 09, DestBlock: 0A)
Response:	0001 (Result: true)

**1.5.11. API MIFAREULTRALIGHT****1.5.11.1. MifareUltralight\_ReadPage**

Command:	[0C00][Byte: <i>Page</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>Data</i> ]
Example	
Command:	0C0004 (Page: 04)
Response:	000100010203147870672E636F6D3A636172 (Result: true, Data: 00010203147870672E636F6D3A636172)

**1.5.11.2. MifareUltralight\_WritePage**

Command:	[0C01][Byte: <i>Page</i> ][Byte Array(4): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0C010400010203 (Page: 04, Data: 00010203)
Response:	0001 (Result: true)

**1.5.11.3. MifareUltralightC\_Authenticate**

Command:	[0C02][Byte Array(16): <i>Key</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0C0249454D4B41455242214E4143554F5946 (Key: 49454D4B41455242214E4143554F5946)
Response:	0001 (Result: true)

**1.5.11.4. MifareUltralightC\_SAMAuthenticate**

Command:	[0C03][Byte: <i>KeyNo</i> ][Byte: <i>KeyVersion</i> ][Byte Array(Var): <i>DIVInput</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0C03010000 (KeyNo: 01, KeyVersion: 00, DIVInput: )
Response:	0001 (Result: true)

**1.5.11.5. MifareUltralightC\_WriteKeyFromSAM**

Command:	[0C04][Byte: <i>KeyNo</i> ][Byte: <i>KeyVersion</i> ][Byte Array(Var): <i>DIVInput</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0C04010000 (KeyNo: 01, KeyVersion: 00, DIVInput: )
Response:	0000 (Result: fail)



**1.5.11.6. MifareUltralightEV1\_FastRead**

Command:	[0C05][Byte: <i>StartPage</i> ][Byte: <i>NumberOfPages</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>Data</i> ]
Example	
Command:	0C050401 (StartPage: 04, NumberOfPages: 01)
Response:	0001040000000000 (Result: true, Data: 00000000)

**1.5.11.7. MifareUltralightEV1\_IncCounter**

Command:	[0C06][Byte: <i>CounterAddr</i> ][UInt32: <i>IncrValue</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0C06000000000000 (CounterAddr: 00, IncrValue: 00000000)
Response:	0001 (Result: true)

**1.5.11.8. MifareUltralightEV1\_ReadCounter**

Command:	[0C07][Byte: <i>CounterAddr</i> ]
Response:	[00][Bool: <i>Result</i> ][UInt32: <i>CounterValue</i> ]
Example	
Command:	0C0700 (CounterAddr: 00)
Response:	000102000000 (Result: true, CounterValue: 2)

**1.5.11.9. MifareUltralightEV1\_ReadSig**

Command:	[0C08]
Response:	[00][Bool: <i>Result</i> ][Byte Array(32): <i>ECCSig</i> ]
Example	
Command:	0C08
Response:	00013A4F2622AF2039E47F8AA1BF84C52EE949860DD07125BEF75EC4- 17833B80C105 (Result: true, ECCSig: 3A4F2622AF2039E47F8AA1BF84C52EE949860DD07125BEF75EC417833B80C105)

**1.5.11.10. MifareUltralightEV1\_GetVersion**

Command:	[0C09]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>Version</i> ]
Example	
Command:	0C09
Response:	00010004030101000E03 (Result: true, Version: 0004030101000E03)

**1.5.11.11. MifareUltralightEV1\_PwdAuth**

Command:	[0C0A][Byte Array(4): <i>Password</i> ][Byte Array(2): <i>PwdAck</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0C0AFFFFFFFFF0000 (Password: FFFFFFFF, PwdAck: 0000)
Response:	0001 (Result: true)

**1.5.11.12. MifareUltralightEV1\_CheckTearingEvent**

Command:	[0C0B][Byte: <i>CounterAddr</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>ValidFlag</i> ]
Example	
Command:	0C0B00 (CounterAddr: 00)
Response:	0001BD (Result: true, ValidFlag: 189)

**1.5.12. API ISO15693****1.5.12.1. ISO15693\_GenericCommand**

Command:	[0D00][Byte: <i>Flags</i> ][Byte: <i>Command</i> ][Byte Array(Var): <i>Data</i> ][Byte: <i>BufferSize</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>Data</i> ]
Example	
Command:	0D001020010020 (Flags: 10, Command: 20, Data: 00, BufferSize: 20)
Response:	00010400000000 (Result: true, Data: 00000000)

**1.5.12.2. ISO15693\_GetSystemInformation**

Command:	[0D01]
Response:	[00][Bool: <i>Result</i> ][Byte Array(15): <i>SystemInfo</i> ]
Example	
Command:	0D01
Response:	0001EF50781B06013C16E002000442000F (Result: true, SystemInfo: EF50781B06013C16E002000442000F)

**1.5.12.3. ISO15693\_GetSystemInformationExt**

Command:	[0D02]
Response:	[00][Bool: <i>Result</i> ][Byte Array(15): <i>SystemInfo</i> ]
Example	
Command:	0D02
Response:	0001EF7D50C3ED084402E0000004000844 (Result: true, SystemInfo: EF7D50C3ED084402E0000004000844)

**1.5.12.4. ISO15693\_GetTagTypeFromUID**

Command:	[0D03][Byte Array(8): <i>UID</i> ]
Response:	[00][Byte: <i>TagType</i> ]
Example	
Command:	0D03E0163C01061B7850 (UID: E0163C01061B7850)
Response:	00FF (TagType: 255)

**1.5.12.5. ISO15693\_GetTagTypeFromSystemInfo**

Command:	[0D04][Byte Array(15): <i>SystemInfo</i> ]
Response:	[00][Byte: <i>TagType</i> ]
Example	
Command:	0D04EF7D50C3ED084402E0000004000844 (SystemInfo: EF7D50C3ED084402E0000004000844)
Response:	0043 (TagType: 67)

**1.5.12.6. ISO15693\_ReadSingleBlock**

Command:	[0D05][UInt16: <i>BlockNumber</i> ][Byte: <i>BufferSize</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>BlockData</i> ]
Example	
Command:	0D050500FF (BlockNumber: 0500, BufferSize: FF)
Response:	00010400000000 (Result: true, BlockData: 00000000)

**1.5.12.7. ISO15693\_ReadSingleBlockExt**

Command:	[0D06][UInt16: <i>BlockNumber</i> ][Byte: <i>BufferSize</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>BlockData</i> ]
Example	
Command:	0D060000FF (BlockNumber: 0000, BufferSize: FF)
Response:	00010401020304 (Result: true, BlockData: 01020304)

**1.5.12.8. ISO15693\_WriteSingleBlock**

Command:	[0D07][UInt16: <i>BlockNumber</i> ][Byte Array(Var): <i>BlockData</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0D0705000411223344 (BlockNumber: 0500, BlockData: 11223344)
Response:	0001 (Result: true)

#### 1.5.12.9. ISO15693\_WriteSingleBlockExt

Command:	[0D08][UInt16: <i>BlockNumber</i> ][Byte Array(Var): <i>BlockData</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0D08000004426C612E (BlockNumber: 0000, BlockData: 426C612E)
Response:	0001 (Result: true)

### 1.5.13. API CRYPTO

#### 1.5.13.1. Crypto\_Init

Command:	[0E00][Byte: <i>CryptoEnv</i> ][Byte: <i>CryptoMode</i> ][Byte Array(Var): Key]
Response:	[00]
Example	
Command:	0E0000001000000000000000000000000000000000000000 (CryptoEnv: 00, CryptoMode: 00, Key: 00000000000000000000000000000000)
Response:	00

### 1.5.13.2. Encrypt

Command:	[0E01][Byte: <i>CryptoEnv</i> ][Byte Array(Var): <i>PlainBlock</i> ]
Response:	[00][Byte Array(Var): <i>CipheredBlock</i> ]
Example	
Command:	0E01001080000000000000000000000000 (CryptoEnv: 00, PlainBlock: 80000000000000000000000000000000)
Response:	00103AD78E726C1EC02B7EBFE92B23D9EC34 (CipheredBlock: 3AD78E726C1EC02B7EBFE92B23D9EC34)

**1.5.13.3. Decrypt**

Command:	[0E02][Byte: <i>CryptoEnv</i> ][Byte Array(Var): <i>CipheredBlock</i> ]
Response:	[00][Byte Array(Var): <i>PlainBlock</i> ]
Example	
Command:	0E0200103AD78E726C1EC02B7EBFE92B23D9EC34 (CryptoEnv: 00, CipheredBlock: 3AD78E726C1EC02B7EBFE92B23D9EC34)
Response:	0010800000000000000000000000000000000000 (PlainBlock: 8000000000000000000000000000000000000000)

**1.5.13.4. CBC\_ResetInitVector**

Command:	[0E03][Byte: <i>CryptoEnv</i> ]
Response:	[00]
Example	
Command:	0E0300 (CryptoEnv: 00)
Response:	00

**1.5.14. API DESFIRE****1.5.14.1. DESFire\_GetApplicationIDs**

Command:	[0F00][Byte: <i>CryptoEnv</i> ][Byte: <i>MaxAIDCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][variable number of UInt32: <i>AIDs</i> ]
Example	
Command:	0F00001C (CryptoEnv: 00, MaxAIDCnt: 1C)
Response:	00010133221100 (Result: true, AIDs: 00112233)

**1.5.14.2. DESFire\_CreateApplication**

Command:	[0F01][Byte: <i>CryptoEnv</i> ][UInt32: <i>AID</i> ][4 Bit: <i>ChangeKeyAccessRights</i> ][1 Bit: <i>ConfigurationChangeable</i> ][1 Bit: <i>FreeCreateDelete</i> ][1 Bit: <i>FreeDirectoryList</i> ][1 Bit: <i>AllowChangeMasterKey</i> ][UInt32: <i>NumberOfKeys</i> ][UInt32: <i>KeyType</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F0100907856000F0100000000000000 (CryptoEnv: 00, AID: 90785600, ChangeKeyAccessRights: 15, ConfigurationChangeable: 1, FreeCreateDelete: 1, FreeDirectoryList: 1, AllowChangeMasterKey: 1, NumberOfKeys: 01000000, KeyType: 00000000)
Response:	0001 (Result: true)

**1.5.14.3. DESFire\_DeleteApplication**

Command:	[0F02][Byte: <i>CryptoEnv</i> ][UInt32: <i>AID</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F020090785600 (CryptoEnv: 00, AID: 90785600)
Response:	0001 (Result: true)

**1.5.14.4. DESFire\_SelectApplication**

Command:	[0F03][Byte: <i>CryptoEnv</i> ][UInt32: <i>AID</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F030033221100 (CryptoEnv: 00, AID: 33221100)
Response:	0001 (Result: true)

#### 1.5.14.5. DESFire\_Authenticate

Command:	[0F04][Byte: <i>CryptoEnv</i> ][Byte: <i>KeyNoTag</i> ][Byte Array(Var): <i>Key</i> ][Byte: <i>KeyType</i> ][Byte: <i>Mode</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F0400001000 (CryptoEnv: 00, KeyNoTag: 00, Key: 00000000000000000000000000000000, KeyType: 00, Mode: 00)
Response:	0001 (Result: true)

#### 1.5.14.6. DESFire\_GetKeySettings

Command:	[0F05][Byte: <i>CryptoEnv</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>KeySettings</i> ][UInt32: <i>NumberOfKeys</i> ][UInt32: <i>KeyType</i> ]
Example	
Command:	0F0500 (CryptoEnv: 00)
Response:	00010F010000000000000000 (Result: true, KeySettings: 15, NumberOfKeys: 1, KeyType: 0)

#### 1.5.14.7. DESFire\_GetFileIDs

Command:	[0F06][Byte: <i>CryptoEnv</i> ][Byte: <i>MaxFileIDCount</i> ]
Response:	[00][Bool: <i>Result</i> ][variable number of Bytes: <i>FileIDList</i> ]
Example	
Command:	0F0600FF (CryptoEnv: 00, MaxFileIDCount: FF)
Response:	00010400010203 (Result: true, FileIDList: 00, 01, 02, 03)



#### 1.5.14.8. DESFire\_GetFileSettings

[illegible]

#### 1.5.14.9. DESFire\_ReadData

Command:	[0F08][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ][UInt16: <i>Offset</i> ][Byte: <i>Length</i> ][Byte: <i>CommSet</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>Data</i> ]
Example	
Command:	0F080000000000300 (CryptoEnv: 00, FileNo: 00, Offset: 0000, Length: 03, CommSet: 00)
Response:	000103001122 (Result: true, Data: 001122)

#### 1.5.14.10. DESFire\_WriteData

Command:	[0F09][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ][UInt16: <i>Offset</i> ][Byte Array(Var): <i>Data</i> ][Byte: <i>CommSet</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F090000000000300112200 (CryptoEnv: 00, FileNo: 00, Offset: 0000, Data: 001122, CommSet: 00)
Response:	0001 (Result: true)

**1.5.14.11. DESFire\_GetValue**

Command:	[0F0A][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ][Byte: <i>CommSet</i> ]
Response:	[00][Bool: <i>Result</i> ][UInt32: <i>Value</i> ]
Example	
Command:	0F0A000000 (CryptoEnv: 00, FileNo: 00, CommSet: 00)
Response:	000100000000 (Result: true, Value: 0)

**1.5.14.12. DESFire\_Credit**

Command:	[0F0B][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ][UInt32: <i>Value</i> ][Byte: <i>CommSet</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F0B00040000000000 (CryptoEnv: 00, FileNo: 04, Value: 00000000, CommSet: 00)
Response:	0001 (Result: true)

**1.5.14.13. DESFire\_Debit**

Command:	[0F0C][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ][UInt32: <i>Value</i> ][Byte: <i>CommSet</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F0C00040000000000 (CryptoEnv: 00, FileNo: 04, Value: 00000000, CommSet: 00)
Response:	0001 (Result: true)

**1.5.14.14. DESFire\_LimitedCredit**

Command:	[0F0D][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ][UInt32: <i>Value</i> ][Byte: <i>CommSet</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F0D00040000000000 (CryptoEnv: 00, FileNo: 04, Value: 00000000, CommSet: 00)
Response:	0001 (Result: true)

**1.5.14.15. DESFire\_FreeMem**

Command:	[0F0E][Byte: <i>CryptoEnv</i> ]
Response:	[00][Bool: <i>Result</i> ][UInt16: <i>FreeMemory</i> ]
Example	
Command:	0F0E00 (CryptoEnv: 00)
Response:	00016011 (Result: true, FreeMemory: 4448)

**1.5.14.16. DESFire\_FormatTag**

Command:	[0F0F][Byte: <i>CryptoEnv</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F0F00 (CryptoEnv: 00)
Response:	0001 (Result: true)

**1.5.14.17. DESFire\_CreateDataFile**

Command:	[0F10][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ][Byte: <i>FileType</i> ][Byte: <i>CommSet</i> ][UInt16: <i>AccessRights</i> ][UInt32: <i>FileSize</i> ]appending 0's]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F1000050000EEEE0F0000000000000000000000000000000000 (CryptoEnv: 00, FileNo: 05, FileType: 00, CommSet: 00, AccessRights: EEEE, FileSize: 0F000000, appending 0's: 00000000000000000000000000000000)
Response:	0001 (Result: true)

**1.5.14.18. DESFire\_CreateValueFile**

Command:	[0F11][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ][Byte: <i>FileType</i> ][Byte: <i>CommSet</i> ][UInt16: <i>AccessRights</i> ][UInt32: <i>LowerLimit</i> ][UInt32: <i>UpperLimit</i> ][UInt32: <i>LimitedCreditValue</i> ][1 Bit: <i>FreeGetValue</i> ][1 Bit: <i>LimitedCreditEnabled</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example Command:	0F1100040200EEEE0000000000F0000000F00000001000000 (CryptoEnv: 00, FileNo: 04, FileType: 02, CommSet: 00, AccessRights: EEEE, LowerLimit: 00000000, UpperLimit: 0F000000, LimitedCreditValue: 0F000000, FreeGetValue: 1, LimitedCreditEnabled: 1)
Response:	0001 (Result: true)

**1.5.14.19. DESFire\_GetVersion**

Command:	[0F12][Byte: <i>CryptoEnv</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(34): <i>Version</i> ]
Example Command:	0F1200 (CryptoEnv: 00)
Response:	00010401010100001000000504010101030010000005000000000000- 00BA14D0A7103110 (Result: true, Version: 040101010000100000050401010103001000000500000000000000BA14D0A7103110)

**1.5.14.20. DESFire\_DeleteFile**

Command:	[0F13][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example Command:	0F130005 (CryptoEnv: 00, FileNo: 05)
Response:	0001 (Result: true)

**1.5.14.21. DESFire\_CommitTransaction**

Command:	[0F14][Byte: <i>CryptoEnv</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F1400 (CryptoEnv: 00)
Response:	0001 (Result: true)

**1.5.14.22. DESFire\_AbortTransaction**

Command:	[0F15][Byte: <i>CryptoEnv</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F1500 (CryptoEnv: 00)
Response:	0001 (Result: true)

**1.5.14.23. DESFire\_GetUID**

Command:	[0F16][Byte: <i>CryptoEnv</i> ][Byte: <i>BufferSize</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>UID</i> ]
Example	
Command:	0F1600FF (CryptoEnv: 00, BufferSize: FF)
Response:	000107045243523D2480 (Result: true, UID: 045243523D2480)

**1.5.14.24. DESFire\_GetKeyVersion**

Command:	[0F17][Byte: <i>CryptoEnv</i> ][Byte: <i>KeyNo</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(1): <i>KeyVersion</i> ]
Example	
Command:	0F170000 (CryptoEnv: 00, KeyNo: 00)
Response:	0001FF (Result: true, KeyVersion: FF)

#### 1.5.14.25. DESFire\_ChangeKeySettings

Command:	[0F18][Byte: <i>CryptoEnv</i> ][4 Bit: <i>ChangeKeyAccessRights</i> ][1 Bit: <i>ConfigurationChangeable</i> ][1 Bit: <i>FreeCreateDelete</i> ][1 Bit: <i>FreeDirectoryList</i> ][1 Bit: <i>AllowChangeMasterKey</i> ][UInt32: <i>NumberOfKeys</i> ][UInt32: <i>KeyType</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F18000F000000000000000000000000 (CryptoEnv: 00, ChangeKeyAccessRights: 15, ConfigurationChangeable: 1, FreeCreateDelete: 1, FreeDirectoryList: 1, AllowChangeMasterKey: 1, NumberOfKeys: 00000000, KeyType: 00000000)
Response:	0001 (Result: true)

#### 1.5.14.26. DESFire\_ChangeKey

[illegible]

#### 1.5.14.27. DESFire ChangeFileSettings

Command:	[0F1A][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ][Byte: <i>NewCommSet</i> ][UInt16: <i>OldAccessRights</i> ][UInt16: <i>NewAccessRights</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F1A000000EEEEEEEE (CryptoEnv: 00, FileNo: 00, NewCommSet: 00, OldAccessRights: EEEE, NewAccessRights: EEEE)
Response:	0001 (Result: true)

#### 1.5.14.28. DESFire\_DisableFormatCard

Command:	[0F1B][Byte: <i>CryptoEnv</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F1B00 (CryptoEnv: 00)
Response:	0001 (Result: true)

#### 1.5.14.29. DESFire EnableRandomID

Command:	[0F1C][Byte: <i>CryptoEnv</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F1C00 (CryptoEnv: 00)
Response:	0001 (Result: true)

#### 1.5.14.30. DESFire SetDefaultKey

Command:	[0F1D][Byte: <i>CryptoEnv</i> ][Byte Array(Var): <i>Key</i> ][Byte: <i>KeyVersion</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example Command:	0F1D001000000000000000000000000000000000000000FF (CryptoEnv: 00, Key: 00000000000000000000000000000000, KeyVersion: FF)
Response:	0001 (Result: true)

#### 1.5.14.31. DESFire SetATS

Command:	[0F1E][Byte: <i>CryptoEnv</i> ][Byte Array(Var): <i>ATS</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F1E0008087577810280CAFE (CryptoEnv: 00, ATS: 087577810280CAFE)
Response:	0001 (Result: true)





**1.5.14.35. DESFire\_ClearRecordFile**

Command:	[0F22][Byte: <i>CryptoEnv</i> ][Byte: <i>FileNo</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	0F220005 (CryptoEnv: 00, FileNo: 05)
Response:	0001 (Result: true)

**1.5.15. API ISO7816****1.5.15.1. ISO7816\_GetSlotStatus**

Command:	[1000][Byte: <i>Channel</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(3): <i>SlotStatus</i> ]
Example	
Command:	100020 (Channel: 20)
Response:	0001000000 (Result: true, SlotStatus: 000000)

**1.5.15.2. ISO7816\_IccPowerOn**

Command:	[1001][Byte: <i>Channel</i> ][Byte: <i>MaxATRByteCnt</i> ][Byte: <i>bPowerSelect</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>ATR</i> ][Byte: <i>bStatus</i> ][Byte: <i>bError</i> ]
Example	
Command:	100120FF00 (Channel: 20, MaxATRByteCnt: FF, bPowerSelect: 00)
Response:	00010F3B959680B1FE551FC74772616365130000 (Result: true, ATR: 3B959680B1FE551FC7477261636513, bStatus: 0, bError: 0)

**1.5.15.3. ISO7816\_IccPowerOff**

Command:	[1002][Byte: <i>Channel</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(3): <i>SlotStatus</i> ]
Example	
Command:	100220 (Channel: 20)
Response:	0001010000 (Result: true, SlotStatus: 010000)

**1.5.15.4. ISO7816\_SetCommSettings**

Command:	[1003][Byte: <i>Channel</i> ][Byte Array(14): <i>CommSettings</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1003200100740101000000FF5500FE0000 (Channel: 20, CommSettings: 0100740101000000FF5500FE0000)
Response:	0001 (Result: true)

**1.5.15.5. ISO7816\_Transceive**

Command:	[1004][Byte: <i>Channel</i> ][Byte Array(Var), 2 LB: <i>TX</i> ][Byte: <i>MaxRXByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var), 2 LB: <i>RX</i> ]
Example	
Command:	100420050000C10120E0FF (Channel: 20, TX: 00C10120E0, MaxRXByteCnt: FF)
Response:	000102006E00 (Result: true, RX: 6E00)

**1.5.15.6. ISO7816\_ExchangeAPDU**

Command:	[1005][Byte: <i>Channel</i> ][Byte Array(9): <i>Header</i> ][Byte Array(Var), 2 LB: <i>TXData</i> ][UInt16: <i>MaxRXByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var), 2 LB: <i>RXData</i> ][UInt16: <i>StatusWord</i> ]
Example	
Command:	10052000A40004020000000102003F008000 (Channel: 20, Header: 00A400040200000001, TXData: 3F00, MaxRXByteCnt: 8000)
Response:	00010000006E (Result: true, RXData: , StatusWord: 28160)

**1.5.15.7. ISO7816\_T0\_TPDU**

Command:	[1006][Byte: <i>Channel</i> ][Byte Array(5): <i>Header</i> ][Byte Array(Var), 2 LB: <i>TXData</i> ][UInt16: <i>MaxRXByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var), 2 LB: <i>RXData</i> ][UInt16: <i>StatusWord</i> ]
Example	
Command:	10062000A400040202003F008000 (Channel: 20, Header: 00A4000402, TXData: 3F00, MaxRXByteCnt: 8000)
Response:	00010000006E (Result: true, RXData: , StatusWord: 28160)

**1.5.15.8. ISO7816\_CheckWellKnownCards**

Command:	[1007][Byte: <i>Channel</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>CardType</i> ]
Example	
Command:	100720 (Channel: 20)
Response:	000110000000 (Result: true, CardType: 10000000)

**1.5.16. API ICLASS****1.5.16.1. ICLASS\_GetPACBits**

Command:	[1100][Byte: <i>MaxPACBytes</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>PACBitCnt</i> ][Byte Array(Var): <i>PAC</i> ]
Example	
Command:	1100FF (MaxPACBytes: FF)
Response:	00011A0405000980 (Result: true, PACBitCnt: 26, PAC: 00140026)

**1.5.16.2. ICLASS\_SelectPage**

Command:	[1101][Byte: <i>Book</i> ][Byte: <i>Page</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>ConfigBlock</i> ]
Example	
Command:	11010000 (Book: 00, Page: 00)
Response:	000112FFFFFFE91FFF3C (Result: true, ConfigBlock: 12FFFFFFE91FFF3C)

**1.5.16.3. ICLASS\_Authenticate**

Command:	[1102][Byte Array(3): <i>KeyReferenceOID</i> ][Byte: <i>KeyType</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	110203002300 (KeyReferenceOID: 030023, KeyType: 00)
Response:	0001 (Result: true)

**1.5.16.4. ICLASS\_ReadBlock**

Command:	[1103][Byte: <i>Block</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>BlockData</i> ]
Example	
Command:	110313 (Block: 13)
Response:	00010000000000000000 (Result: true, BlockData: 0000000000000000)

**1.5.16.5. ICLASS\_WriteBlock**

Command:	[1104][Byte: <i>Block</i> ][Byte Array(8): <i>BlockData</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1104130000000000000000 (Block: 13, BlockData: 0000000000000000)
Response:	0001 (Result: true)

**1.5.17. API ISO14443****1.5.17.1. ISO14443A\_GetATS**

Command:	[1200][Byte: <i>MaxATSByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>ATS</i> ]
Example	
Command:	120020 (MaxATSByteCnt: 20)
Response:	000106067577810280 (Result: true, ATS: 067577810280)

**1.5.17.2. ISO14443B\_GetATQB**

Command:	[1201][Byte: <i>MaxATQBByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>ATQB</i> ]
Example	
Command:	1201FF (MaxATQBByteCnt: FF)
Response:	00010C5077FB135400000000B37171 (Result: true, ATQB: 5077FB135400000000B37171)

**1.5.17.3. ISO14443\_4\_CheckPresence**

Command:	[1202]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1202
Response:	0001 (Result: true)

**1.5.17.4. ISO14443\_4\_TDX**

Command:	[1203][Byte Array(Var): <i>TX</i> ][Byte: <i>MaxRXByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>RX</i> ]
Example	
Command:	1203016020 (TX: 60, MaxRXByteCnt: 20)
Response:	0001026F00 (Result: true, RX: 6F00)

**1.5.17.5. ISO14443A\_GetATQA**

Command:	[1204]
Response:	[00][Bool: <i>Result</i> ][Byte Array(2): <i>ATQA</i> ]
Example	
Command:	1204
Response:	00010403 (Result: true, ATQA: 0403)

**1.5.17.6. ISO14443A\_GetSAK**

Command:	[1205]
Response:	[00][Bool: <i>Result</i> ][Byte Array(1): <i>SAK</i> ]
Example	
Command:	1205
Response:	000120 (Result: true, SAK: 20)

**1.5.17.7. ISO14443B\_GetAnswerToATTRIB**

Command:	[1206][Byte: <i>MaxAnswerToATTRIBByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>AnswerToATTRIB</i> ]
Example	
Command:	1206FF (MaxAnswerToATTRIBByteCnt: FF)
Response:	00010100 (Result: true, AnswerToATTRIB: 00)

**1.5.17.8. ISO14443\_3\_TDX**

Command:	[1207][Byte Array(Var): <i>TX</i> ][Byte: <i>MaxRXByteCnt</i> ][UInt16: <i>Timeout</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>RX</i> ]
Example	
Command:	1207041A004176FFFF00 (TX: 1A004176, MaxRXByteCnt: FF, Timeout: FF00)
Response:	00010104 (Result: true, RX: 04)

**1.5.17.9. ISO14443A\_SearchMultiTag**

Command:	[1208][Byte: <i>MaxUIDListByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>UIDCnt</i> ][variable number of Bytes: <i>UIDList</i> ]
Example	
Command:	1208FF (MaxUIDListByteCnt: FF)
Response:	000103180704D7A79A97378007042DA79A973780070450A79A973780 (Result: true, UIDCnt: 3, UIDList: 04D7A79A973780, 042DA79A973780, 0450A79A973780)

**1.5.17.10. ISO14443A\_SelectTag**

Command:	[1209][Byte Array(Var): <i>UID</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	12090704D7A79A973780 (UID: 04D7A79A973780)
Response:	0001 (Result: true)

**1.5.18. API AT55****1.5.18.1. AT55\_Begin**

Command:	[1500]
Response:	[00]
Example	
Command:	1500
Response:	00

**1.5.18.2. AT55\_ReadBlock**

Command:	[1501][Byte: <i>Address</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>Data</i> ]
Example	
Command:	150100 (Address: 00)
Response:	0001F0148040 (Result: true, Data: F0148040)

**1.5.18.3. AT55\_ReadBlockProtected**

Command:	[1502][Byte: <i>Address</i> ][Byte Array(4): <i>Password</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>Data</i> ]
Example	
Command:	1502000000000000 (Address: 00, Password: 00000000)
Response:	0001B8A31C02 (Result: true, Data: B8A31C02)



**1.5.18.4. AT55\_WriteBlock**

Command:	[1503][Byte: <i>Address</i> ][Byte Array(4): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	15030000010203 (Address: 00, Data: 00010203)
Response:	0001 (Result: true)

**1.5.18.5. AT55\_WriteBlockProtected**

Command:	[1504][Byte: <i>Address</i> ][Byte Array(4): <i>Data</i> ][Byte Array(4): <i>Password</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1504000001020300000000 (Address: 00, Data: 00010203, Password: 00000000)
Response:	0001 (Result: true)

**1.5.18.6. AT55\_WriteBlockAndLock**

Command:	[1505][Byte: <i>Address</i> ][Byte Array(4): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	15050000010203 (Address: 00, Data: 00010203)
Response:	0001 (Result: true)

**1.5.18.7. AT55\_WriteBlockProtectedAndLock**

Command:	[1506][Byte: <i>Address</i> ][Byte Array(4): <i>Data</i> ][Byte Array(4): <i>Password</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1506000001020300000000 (Address: 00, Data: 00010203, Password: 00000000)
Response:	0001 (Result: true)

**1.5.19. API NFCSNEP****1.5.19.1. SNEP\_Init**

Command:	[1800]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1800
Response:	0001 (Result: true)

**1.5.19.2. SNEP\_GetConnectionState**

Command:	[1801]
Response:	[00][Byte: <i>ConnectionState</i> ]
Example	
Command:	1801
Response:	0002 (ConnectionState: 2)

**1.5.19.3. SNEP\_GetFragmentByteCount**

Command:	[1802][Byte: <i>Direction</i> ]
Response:	[00][UInt16: <i>ByteCount</i> ]
Example	
Command:	180201 (Direction: 01)
Response:	000000 (ByteCount: 0)

**1.5.19.4. SNEP\_BeginMessage**

Command:	[1803][UInt32: <i>MsgByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1803FF000000 (MsgByteCnt: FF000000)
Response:	0001 (Result: true)

**1.5.19.5. SNEP\_SendMessageFragment**

Command:	[1804][Byte Array(Var), 2 LB: <i>MsgFrag</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	18041500D101115501656C617465632D726669642E636F6D2F (MsgFrag: D101115501656C617465632D726669642E636F6D2F)
Response:	0001 (Result: true)

**1.5.19.6. SNEP\_TestMessage**

Command:	[1805]
Response:	[00][Bool: <i>Result</i> ][UInt32: <i>MsgByteCnt</i> ]
Example	
Command:	1805
Response:	0000 (Result: fail, MsgByteCnt: )

**1.5.19.7. SNEP\_ReceiveMessageFragment**

Command:	[1806][UInt16: <i>FragByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var), 2 LB: <i>MsgFrag</i> ]
Example	
Command:	1806FF00 (FragByteCnt: FF00)
Response:	0000 (Result: fail, MsgFrag: )

**1.5.19.8. SNEP\_RequestMessage**

Command:	[1807][UInt32: <i>MsgByteCnt</i> ][UInt32: <i>AcceptableLength</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1807FF000000FF000000 (MsgByteCnt: FF000000, AcceptableLength: FF000000)
Response:	0001 (Result: true)

**1.5.20. API EM4150****1.5.20.1. EM4150\_Login**

Command:	[1900][Byte Array(4): <i>Password</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	190000000000 (Password: 00000000)
Response:	0001 (Result: true)

**1.5.20.2. EM4150\_ReadWord**

Command:	[1901][Byte: <i>Address</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>Word</i> ]
Example	
Command:	190101 (Address: 01)
Response:	000100010203 (Result: true, Word: 00010203)

**1.5.20.3. EM4150\_WriteWord**

Command:	[1902][Byte: <i>Address</i> ][Byte Array(4): <i>Word</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	19020100010203 (Address: 01, Word: 00010203)
Response:	0001 (Result: true)

**1.5.20.4. EM4150\_WritePassword**

Command:	[1903][Byte Array(4): <i>ActualPassword</i> ][Byte Array(4): <i>NewPassword</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	190300000000001010101 (ActualPassword: 00000000, NewPassword: 01010101)
Response:	0001 (Result: true)

**1.5.20.5. EM4150\_GetTagInfo**

Command:	[1904]
Response:	[00][UInt32: <i>TagInfo</i> ]
Example	
Command:	1904
Response:	0001000000 (TagInfo: 1)

**1.5.21. API FILESYS****1.5.21.1. FSMount**

Command:	[1A00][Byte: <i>StorageID</i> ][UInt32: <i>Mode</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1A000102000000 (StorageID: 01, Mode: 02000000)
Response:	0001 (Result: true)

**1.5.21.2. FSFormat**

Command:	[1A01][Byte: <i>StorageID</i> ][UInt32: <i>MagicValue</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1A0101446F4974 (StorageID: 01, MagicValue: 446F4974)
Response:	0001 (Result: true)

**1.5.21.3. FSOpen**

Command:	[1A02][Byte: <i>FileEnv</i> ][Byte: <i>StorageID</i> ][UInt32: <i>FileID</i> ][Byte: <i>Mode</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1A0200013322110000 (FileEnv: 00, StorageID: 01, FileID: 33221100, Mode: 00)
Response:	0001 (Result: true)

**1.5.21.4. FSClose**

Command:	[1A03][Byte: <i>FileEnv</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1A0300 (FileEnv: 00)
Response:	0001 (Result: true)

**1.5.21.5. FSCloseAll**

Command:	[1A04]
Response:	[00]
Example	
Command:	1A04
Response:	00

**1.5.21.6. FSSeek**

Command:	[1A05][Byte: <i>FileEnv</i> ][Byte: <i>Origin</i> ][UInt32: <i>Pos</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1A05000001000000 (FileEnv: 00, Origin: 00, Pos: 01000000)
Response:	0001 (Result: true)

**1.5.21.7. FSTell**

Command:	[1A06][Byte: <i>FileEnv</i> ][Byte: <i>Origin</i> ]
Response:	[00][Bool: <i>Result</i> ][UInt32: <i>Pos</i> ]
Example	
Command:	1A060000 (FileEnv: 00, Origin: 00)
Response:	000101000000 (Result: true, Pos: 1)

**1.5.21.8. FSReadBytes**

Command:	[1A07][Byte: <i>FileEnv</i> ][UInt16: <i>ByteCount</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var), 2 LB: <i>Data</i> ]
Example	
Command:	1A07001E00 (FileEnv: 00, ByteCount: 1E00)
Response:	000107004D792064617461 (Result: true, Data: 4D792064617461)

**1.5.21.9. FSWriteBytes**

Command:	[1A08][Byte: <i>FileEnv</i> ][Byte Array(Var), 2 LB: <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ][UInt16: <i>BytesWritten</i> ]
Example	
Command:	1A080007004D792064617461 (FileEnv: 00, Data: 4D792064617461)
Response:	00010700 (Result: true, BytesWritten: 7)

**1.5.21.10. FSFindFirst**

Command:	[1A09][Byte: <i>StorageID</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>FileInfo</i> ]
Example	
Command:	1A0901 (StorageID: 01)
Response:	00013322110002000000 (Result: true, FileInfo: 3322110002000000)

**1.5.21.11. FSFindNext**

Command:	[1A0A]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>FileInfo</i> ]
Example	
Command:	1A0A
Response:	00013422110002000000 (Result: true, FileInfo: 3422110002000000)

**1.5.21.12. FSDelete**

Command:	[1A0B][Byte: <i>StorageID</i> ][UInt32: <i>FileID</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1A0B0133221100 (StorageID: 01, FileID: 33221100)
Response:	0001 (Result: true)

**1.5.21.13. FSRename**

Command:	[1A0C][Byte: <i>StorageID</i> ][UInt32: <i>OldFileID</i> ][UInt32: <i>NewFileID</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1A0C017766554433221100 (StorageID: 01, OldFileID: 77665544, NewFileID: 33221100)
Response:	0001 (Result: true)





### 1.5.22.3. MFP\_Authenticate

Command:	[1B02][Byte: <i>CryptoEnv</i> ][UInt16: <i>KeyBNr</i> ][Byte Array(16): <i>Key</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1B0200004000000000000000000000000000000000000000 (CryptoEnv: 00, KeyBNr: 0040, Key: 00000000000000000000000000000000)
Response:	0001 (Result: true)

#### 1.5.22.4. MFP ReadBlock

Command:	[1B03][Byte: <i>CryptoEnv</i> ][UInt16: <i>Block</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>Data</i> ]
Example	
Command:	1B03000400 (CryptoEnv: 00, Block: 0400)
Response:	000101020304050607080900010203040506 (Result: true, Data: 01020304050607080900010203040506)

#### 1.5.22.5. MFP WriteBlock

Command:	[1B04][Byte: <i>CryptoEnv</i> ][UInt16: <i>Block</i> ][Byte Array(16): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1B0400040001020304050607080900010203040506 (CryptoEnv: 00, Block: 0400, Data: 01020304050607080900010203040506)
Response:	0001 (Result: true)

#### 1.5.22.6. MFP ReadValueBlock

Command:	[1B05][Byte: <i>CryptoEnv</i> ][UInt16: <i>Block</i> ]
Response:	[00][Bool: <i>Result</i> ][UInt32: <i>Value</i> ]
Example	
Command:	1B05000400 (CryptoEnv: 00, Block: 0400)
Response:	000100000000 (Result: true, Value: 0)

**1.5.22.7. MFP\_WriteValueBlock**

Command:	[1B06][Byte: <i>CryptoEnv</i> ][UInt16: <i>Block</i> ][UInt32: <i>Value</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1B060004000000000000 (CryptoEnv: 00, Block: 0400, Value: 00000000)
Response:	0001 (Result: true)

**1.5.22.8. MFP\_IncrementValueBlock**

Command:	[1B07][Byte: <i>CryptoEnv</i> ][UInt16: <i>Block</i> ][UInt32: <i>Value</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1B0700040001000000 (CryptoEnv: 00, Block: 0400, Value: 01000000)
Response:	0001 (Result: true)

**1.5.22.9. MFP\_DecrementValueBlock**

Command:	[1B08][Byte: <i>CryptoEnv</i> ][UInt16: <i>Block</i> ][UInt32: <i>Value</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1B0800040001000000 (CryptoEnv: 00, Block: 0400, Value: 01000000)
Response:	0001 (Result: true)

**1.5.22.10. MFP\_CopyValueBlock**

Command:	[1B09][Byte: <i>CryptoEnv</i> ][UInt16: <i>SourceBlock</i> ][UInt16: <i>DestBlock</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1B090004000500 (CryptoEnv: 00, SourceBlock: 0400, DestBlock: 0500)
Response:	0001 (Result: true)

**1.5.23. API ADC****1.5.23.1. ADCInitChannel**

Command:	[1C00][Byte: <i>ADCChannel</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1C0001 (ADCChannel: 01)
Response:	0001 (Result: true)

**1.5.23.2. ADCGetConversionValue**

Command:	[1C01][Byte: <i>ADCChannel</i> ]
Response:	[00][UInt16: <i>Value</i> ]
Example	
Command:	1C0101 (ADCChannel: 01)
Response:	003700 (Value: 55)

**1.5.24. API FELICA****1.5.24.1. FeliCa\_TDX**

Command:	[1D00][Byte Array(Var): <i>TX</i> ][Byte: <i>MaxRXByteCnt</i> ][Byte: <i>MaximumResponseTime</i> ][Byte: <i>NumberOfBlocks</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>RX</i> ]
Example	
Command:	1D00060600FFFF0000FFFF04 (TX: 0600FFFF0000, MaxRXByteCnt: FF, MaximumResponseTime: FF, NumberOfBlocks: 04)
Response:	000112120101010701450F16000120220427674EFF (Result: true, RX: 120101010701450F16000120220427674EFF)

#### 1.5.24.2. FeliCa\_ReadWithoutEncryption

[illegible]

### 1.5.24.3. FeliCa\_WriteWithoutEncryption

[illegible]

#### 1.5.24.4. FeliCa\_RequestSystemCode

Command:	[1D03][Byte: <i>MaxNumberOfSystemCodes</i> ]
Response:	[00][Bool: <i>Result</i> ][variable number of UInt16: <i>SystemCodeList</i> ]
Example	
Command:	1D0308 (MaxNumberOfSystemCodes: 08)
Response:	000103030000FEA786 (Result: true, SystemCodeList: 0003, FE00, 86A7)

**1.5.24.5. FeliCa\_Poll**

Command:	[1D04][UInt16: <i>SystemCode</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>IDm</i> ][Byte Array(8): <i>PMm</i> ]
Example	
Command:	1D04FFFF (SystemCode: FFFF)
Response:	0001011603002D0CA50B03014B024F4993FF (Result: true, IDm: 011603002D0CA50B, PMm: 03014B024F4993FF)

**1.5.24.6. FeliCa\_RequestService**

Command:	[1D05][variable number of UInt16: <i>ServiceCodeList</i> ]
Response:	[00][Bool: <i>Result</i> ][variable number of UInt16: <i>KeyVersionList</i> ]
Example	
Command:	1D05010000 (ServiceCodeList: 0000)
Response:	0001010100 (Result: true, KeyVersionList: 0001)

**1.5.25. API SLE44XX****1.5.25.1. SLE44XX\_GetATR**

Command:	[1F00][Byte: <i>Channel</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>ATR</i> ]
Example	
Command:	1F0028 (Channel: 28)
Response:	0001FFFFFFFF (Result: true, ATR: FFFFFFFFFF)

**1.5.25.2. SLE444X\_ReadMainMemory**

Command:	[1F01][Byte: <i>Channel</i> ][UInt16: <i>Address</i> ][UInt16: <i>ByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var), 2 LB: <i>Data</i> ]
Example	
Command:	1F012800000100 (Channel: 28, Address: 0000, ByteCnt: 0100)
Response:	00010100FF (Result: true, Data: FF)

**1.5.25.3. SLE444X\_UpdateMainMemory**

Command:	[1F02][Byte: <i>Channel</i> ][UInt16: <i>Address</i> ][Byte: <i>Value</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1F0228000000 (Channel: 28, Address: 0000, Value: 00)
Response:	0001 (Result: true)

**1.5.25.4. SLE444X\_ReadSecurityMemory**

Command:	[1F03][Byte: <i>Channel</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>SecMemData</i> ]
Example	
Command:	1F0328 (Channel: 28)
Response:	0001FFFFFFFF (Result: true, SecMemData: FFFFFFFFFF)

**1.5.25.5. SLE444X\_UpdateSecurityMemory**

Command:	[1F04][Byte: <i>Channel</i> ][Byte: <i>Address</i> ][Byte: <i>SecMemData</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1F042800FF (Channel: 28, Address: 00, SecMemData: FF)
Response:	0001 (Result: true)

**1.5.25.6. SLE444X\_ReadProtectionMemory**

Command:	[1F05][Byte: <i>Channel</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>ProtMemData</i> ]
Example	
Command:	1F0528 (Channel: 28)
Response:	0001FFFFFFFF (Result: true, ProtMemData: FFFFFFFF)

**1.5.25.7. SLE444X\_WriteProtectionMemory**

Command:	[1F06][Byte: <i>Channel</i> ][Byte: <i>Address</i> ][Byte: <i>ProtMemData</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1F062800FF (Channel: 28, Address: 00, ProtMemData: FF)
Response:	0001 (Result: true)

**1.5.25.8. SLE444X\_CompareVerificationData**

Command:	[1F07][Byte: <i>Channel</i> ][Byte: <i>Address</i> ][Byte: <i>VerificationData</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1F072800FF (Channel: 28, Address: 00, VerificationData: FF)
Response:	0001 (Result: true)

**1.5.25.9. SLE44X8\_ReadMainMemory**

Command:	[1F08][Byte: <i>Channel</i> ][UInt16: <i>Address</i> ][UInt16: <i>ByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var), 2 LB: <i>Data</i> ]
Example	
Command:	1F0828FD030300 (Channel: 28, Address: FD03, ByteCnt: 0300)
Response:	00010300FFFFFF (Result: true, Data: FFFFFFFF)



**1.5.25.10. SLE44X8\_WriteErrorCounter**

Command:	[1F09][Byte: <i>Channel</i> ][UInt16: <i>Address</i> ][Byte: <i>ErrorCounter</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1F0928FD03FE (Channel: 28, Address: FD03, ErrorCounter: FE)
Response:	0001 (Result: true)

**1.5.25.11. SLE44X8\_VerifyPSCByte**

Command:	[1F0A][Byte: <i>Channel</i> ][UInt16: <i>Address</i> ][Byte: <i>PSCByte</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1F0A28FE03FF (Channel: 28, Address: FE03, PSCByte: FF)
Response:	0001 (Result: true)

**1.5.25.12. SLE44X8\_UpdateMainMemory**

Command:	[1F0B][Byte: <i>Channel</i> ][UInt16: <i>Address</i> ][Byte: <i>Value</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	1F0B28FD03FF (Channel: 28, Address: FD03, Value: FF)
Response:	0001 (Result: true)

## 1.5.26. API NTAG

### 1.5.26.1. NTAG\_Read

Command:	[2000][Byte: <i>Page</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>Page</i> ]
Example	
Command:	200004 (Page: 04)
Response:	000103B691028C537091016855016E78702E (Result: true, Page: 03B691028C537091016855016E78702E)

### 1.5.26.2. NTAG\_Write

Command:	[2001][Byte: <i>Page</i> ][Byte Array(4): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	20010400000000 (Page: 04, Data: 00000000)
Response:	0001 (Result: true)

### 1.5.26.3. NTAG\_FastRead

Command:	[2002][Byte: <i>StartPage</i> ][Byte: <i>NumberOfPages</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>Data</i> ]
Example	
Command:	20020401 (StartPage: 04, NumberOfPages: 01)
Response:	00010403B69102 (Result: true, Data: 03B69102)

**1.5.26.4. NTAG\_ReadCounter**

Command:	[2003]
Response:	[00][Bool: <i>Result</i> ][UInt32: <i>CounterValue</i> ]
Example	
Command:	2003
Response:	000101000000 (Result: true, CounterValue: 1)

**1.5.26.5. NTAG\_ReadSig**

Command:	[2004]
Response:	[00][Bool: <i>Result</i> ][Byte Array(32): <i>ECCSig</i> ]
Example	
Command:	2004
Response:	0001A9AC15AFB52080BA26A45B1DA442F363E31B41271AB12B3E6F67- 864615B05321 (Result: true, ECCSig: A9AC15AFB52080BA26A45B1DA442F363E31B41271AB12B3E6F67864615B05321)

**1.5.26.6. NTAG\_GetVersion**

Command:	[2005]
Response:	[00][Bool: <i>Result</i> ][Byte Array(8): <i>Version</i> ]
Example	
Command:	2005
Response:	00010004040502011503 (Result: true, Version: 0004040502011503)

**1.5.26.7. NTAG\_PwdAuth**

Command:	[2006][Byte Array(4): <i>Password</i> ][Byte Array(2): <i>PwdAck</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2006FFFFFFFFF0000 (Password: FFFFFFFF, PwdAck: 0000)
Response:	0001 (Result: true)

**1.5.26.8. NTAG\_SectorSelect**

Command:	[2007][Byte: <i>Sector</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	200700 (Sector: 00)
Response:	0001 (Result: true)

**1.5.27. API SRX****1.5.27.1. SRX\_ReadBlock**

Command:	[2100][Byte: <i>Block</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>Data</i> ]
Example	
Command:	210000 (Block: 00)
Response:	000100000000 (Result: true, Data: 00000000)

**1.5.27.2. SRX\_WriteBlock**

Command:	[2101][Byte: <i>Block</i> ][Byte Array(4): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	21010000000000 (Block: 00, Data: 00000000)
Response:	0001 (Result: true)

### 1.5.28. API SAMAVX

#### 1.5.28.1. SAMAVx AuthenticateHost

[illegible]

#### 1.5.28.2. SAMAVx\_GetKeyEntry

Command:	[2201][Byte: <i>KeyNo</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(13): <i>TSAMAVxKeyEntryData</i> ]
Example	
Command:	220101 (KeyNo: 01)
Response:	000100010200000000000000FF0C00 (Result: true, TSAMAVxKeyEntryData: 00010200000000000000FF0C00)

### 1.5.29. API EM4102

### 1.5.29.1. EM4102 GetTagInfo

Command:	[2300]
Response:	[00][UInt32: <i>TagInfo</i> ]
Example	
Command:	2300
Response:	0001000000 (TagInfo: 1)

**1.5.30. API SPI****1.5.30.1. SPIInit**

Command:	[2400][Byte: <i>Mode</i> ][Byte: <i>CPOL</i> ][Byte: <i>CPHA</i> ][Byte: <i>ClockRate</i> ][Byte: <i>BitOrder</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	240001000000000 (Mode: 01, CPOL: 00, CPHA: 00, ClockRate: 00, BitOrder: 00)
Response:	0001 (Result: true)

**1.5.30.2. SPIDeInit**

Command:	[2401]
Response:	[00]
Example	
Command:	2401
Response:	00

**1.5.30.3. SPIMasterBeginTransfer**

Command:	[2402]
Response:	[00]
Example	
Command:	2402
Response:	00

**1.5.30.4. SPIMasterEndTransfer**

Command:	[2403]
Response:	[00]
Example	
Command:	2403
Response:	00

**1.5.30.5. SPITransmit**

Command:	[2404][Byte Array(Var), 2 LB: <i>TXData</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2404010000 (TXData: 00)
Response:	0001 (Result: true)

**1.5.30.6. SPIReceive**

Command:	[2405][UInt16: <i>ByteCount</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var), 2 LB: <i>RXData</i> ]
Example	
Command:	24050100 (ByteCount: 0100)
Response:	000101005A (Result: true, RXData: 5A)

**1.5.30.7. SPITransceive**

Command:	[2406][Byte Array(Var), 2 LB: <i>TXData</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var), 2 LB: <i>RXData</i> ]
Example	
Command:	2406010000 (TXData: 00)
Response:	000101005A (Result: true, RXData: 5A)

**1.5.31. API BLE****1.5.31.1. BLEPresetConfig**

Command:	[2500][Byte Array(17): <i>BLEConfig</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2500881300000A01A0000702020000D2040000 (BLEConfig: 881300000A01A0000702020000D2040000)
Response:	0001 (Result: true)

**1.5.31.2. BLEPresetUserData**

Command:	[2501][Byte: <i>ScanResp</i> ][Byte Array(Var): <i>UserData</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2501001E0201061AFF4C000215E2C56DB5DFFB48D2B060D0F5A71096- E000000000C3 (ScanResp: 00, UserData: 0201061AFF4C000215E2C56DB5DFFB48D2B060D0F5A71096E000000000C3)
Response:	0001 (Result: true)

**1.5.31.3. BLEInit**

Command:	[2502][Byte: <i>Mode</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	250201 (Mode: 01)
Response:	0001 (Result: true)



**1.5.31.4. BLECheckEvent**

Command:	[2503]
Response:	[00][Byte: <i>Event</i> ]
Example	
Command:	2503
Response:	0081000000 (Event: BLE_EVENT_LE_GAP_SCAN_RESPONSE)

**1.5.31.5. BLEGetAddress**

Command:	[2504]
Response:	[00][Bool: <i>Result</i> ][Byte Array(6): <i>DeviceAddress</i> ][Byte Array(6): <i>RemoteAddress</i> ][Byte Array(1): <i>RemoteType</i> ]
Example	
Command:	2504
Response:	000149D702570B009872F9F36D4601 (Result: true, DeviceAddress: 49D702570B00, RemoteAddress: 9872F9F36D46, RemoteType: 01)

**1.5.31.6. BLEGetVersion**

Command:	[2505]
Response:	[00][Bool: <i>Result</i> ][Byte Array(16): <i>HWVersion</i> ][Byte Array(12): <i>BootString</i> ]
Example	
Command:	2505
Response:	000156312E30342C32382E30362E3230313702000400000018090000-0101 (Result: true, HWVersion: 56312E30342C32382E30362E32303137, BootString: 020004000000180900000101)

**1.5.31.7. BLEGetEnvironment**

Command:	[2506]
Response:	[00][Bool: <i>Result</i> ][Byte Array(1): <i>DeviceRole</i> ][Byte Array(1): <i>SecurityMode</i> ][Byte Array(1): <i>Rssi</i> ]
Example	
Command:	2506
Response:	0001000000 (Result: true, DeviceRole: 00, SecurityMode: 00, Rssi: 00)

**1.5.31.8. BLEGetGattServerAttributeValue**

Command:	[2507][UInt16: <i>AttrHandle</i> ][Byte: <i>MaxLen</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>Data</i> ]
Example	
Command:	2507120014 (AttrHandle: 1200, MaxLen: 14)
Response:	0001104254312E3035454C2020202020202020 (Result: true, Data: 4254312E3035454C2020202020202020)

**1.5.31.9. BLESetGattServerAttributeValue**

Command:	[2508][UInt16: <i>AttrHandle</i> ][UInt16: <i>Offset</i> ][Byte Array(Var): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	25081500000000500000000000 (AttrHandle: 1500, Offset: 0000, Data: 000000000000)
Response:	0001 (Result: true)

**1.5.31.10. BLERequestRssi**

Command:	[2509]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2509
Response:	0001 (Result: true)

**1.5.31.11. BLERequestEndpointClose**

Command:	[250A]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	250A
Response:	0001 (Result: true)

**1.5.31.12. BLEGetGattServerCharacteristicStatus**

Command:	[250B]
Response:	[00][Bool: <i>Result</i> ][UInt16: <i>AttrHandle</i> ][Byte: <i>AttrStatusFlag</i> ][UInt16: <i>AttrConfigFlag</i> ]
Example	
Command:	250B
Response:	0001000000000000 (Result: true, AttrHandle: 0, AttrStatusFlag: 0, AttrConfigFlag: 0)

**1.5.31.13. BLEFindGattServerAttribute**

Command:	[250C][Byte Array(Var): <i>UUID</i> ]
Response:	[00][Bool: <i>Result</i> ][UInt16: <i>AttrHandle</i> ]
Example	
Command:	250C02262A (UUID: 262A)
Response:	00011200 (Result: true, AttrHandle: 18)

**1.5.31.14. BLEDiscover**

Command:	[250D][Byte: <i>DiscoverMode</i> ][UInt32: <i>GattHandle</i> ][Byte Array(17): <i>BLEUUID</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	250D00FFFF280010FA349B5F80000080001000001DB80000 (DiscoverMode: 00, GattHandle: FFFF2800, BLEUUID: 10FA349B5F80000080001000001DB80000)
Response:	0001 (Result: true)

**1.5.31.15. BLECheckDiscoveredString**

Command:	[250E][Byte: <i>CheckMode</i> ][Byte Array(Var): <i>CompareString</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	250E0006454C41544543 (CheckMode: 00, CompareString: 454C41544543)
Response:	0001 (Result: true)

**1.5.31.16. BLEConnectToDevice**

Command:	[250F][Byte Array(6): <i>Address</i> ][Byte: <i>AddressType</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	250F9872F9F36D4601 (Address: 9872F9F36D46, AddressType: 01)
Response:	0001 (Result: true)

**1.5.31.17. BLEDisconnectFromDevice**

Command:	[2510]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2510
Response:	0001 (Result: true)

**1.5.31.18. BLEGattGetAttribute**

Command:	[2511]
Response:	[00][Bool: <i>Result</i> ][Byte Array(17): <i>BLEUUID</i> ][UInt32: <i>GattHandle</i> ]
Example	
Command:	2511
Response:	000100574E340042312E30382F4E4346332E3100000000 (Result: true, BLEUUID: 00574E340042312E30382F4E4346332E31, GattHandle: 0)

**1.5.31.19. BLEGattGetValue**

Command:	[2512][Byte: <i>ReadMode</i> ][UInt32: <i>GattHandle</i> ][Byte Array(17): <i>BLEUUID</i> ][Byte: <i>MaxLen</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>AttrOpcode</i> ][Byte Array(Var): <i>Data</i> ]
Example	
Command:	2512002A00000010FA349B5F80000080001000001DB8000010 (ReadMode: 00, GattHandle: 2A000000, BLEUUID: 10FA349B5F80000080001000001DB80000, MaxLen: 10)
Response:	00010B10CD7CBE4FB6264731587303F12FB369FE (Result: true, AttrOpcode: gatt_read_response, Data: CD7CBE4FB6264731587303F12FB369FE)

**1.5.31.20. BLEGattSetValue**

Command:	[2513][Byte: <i>WriteMode</i> ][UInt32: <i>GattHandle</i> ][UInt16: <i>Offset</i> ][Byte Array(Var): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2513002C000000000000F0102030405060708090A0B0C0D0E0F (WriteMode: 00, GattHandle: 2C000000, Offset: 0000, Data: 0102030405060708090A0B0C0D0E0F)
Response:	0001 (Result: true)

**1.5.31.21. BLECommand**

Command:	[2514][Byte: <i>ConnMode</i> ][UInt32: <i>Parameter</i> ]
Response:	[00][UInt16: <i>Status</i> ]
Example	
Command:	251400FA000000 (ConnMode: 00, Parameter: FA000000)
Response:	0001000000 (Status: 1)

**1.5.31.22. BLESecurity**

Command:	[2515][Byte: <i>SMMode</i> ][UInt32: <i>Flag1</i> ][UInt32: <i>Flag2</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	25150B40E2010000000000 (SMMode: 0B, Flag1: 40E20100, Flag2: 00000000)
Response:	0001 (Result: true)

**1.5.31.23. BLESecuritySetOob**

Command:	[2516][Byte: <i>SMOOBMode</i> ][Byte Array(Var): <i>OobData</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	25170010000102030405060708090A0B0C0D0E0F (SMOOBMode: 00, OobData: 000102030405060708090A0B0C0D0E0F)
Response:	0001 (Result: true)

**1.5.31.24. BLESecurityUseScOob**

Command:	[2517][Byte: <i>Enable</i> ][Byte: <i>MaxLength</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>OobData</i> ]
Example	
Command:	25180120 (Enable: 01, MaxLength: 20)
Response:	000120209678F5BF6EE4EA4F49FA2D22163C57B9A87F40D20183C187- 7A93B010A6F2F5 (Result: true, OobData: 209678F5BF6EE4EA4F49FA2D22163C57B9A87F40D20183C1877A93B010A6F2F5)

**1.5.31.25. BLESetStreamingUUID**

Command:	[2518][Byte Array(Var): <i>ServiceUUID</i> ][Byte Array(Var): <i>CharacUUID</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2518108EDFAE3D9BCD0E887442124104C0445A1052C79E169D4822AA- 434C0A2FDF9EC243 (ServiceUUID: 8EDFAE3D9BCD0E887442124104C0445A, CharacUUID: 52C79E169D4822AA434C0A2FDF9EC243)
Response:	0001 (Result: true)

**1.5.31.26. BLESetStreamingMode**

Command:	[2519][Byte: <i>ConnMode</i> ][Byte: <i>GattMode</i> ][Byte: <i>TransferMode</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2519010000 (ConnMode: 01, GattMode: 00, TransferMode: 00)
Response:	0001 (Result: true)

**1.5.31.27. BLEGetDiscoveredData**

Command:	[251A][Byte: <i>MaxLen</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>Data</i> ]
Example	
Command:	251A40 (MaxLen: 40)
Response:	00011F02011A03036FFD17166FFD2EACD0A563838173CB517FFD702A- C8D6B68B7AAD (Result: true, Data: 02011A03036FFD17166FFD2EACD0A563838173CB517FFD702AC8D6B68B7AAD)

**1.5.32. API I2CCARD****1.5.32.1. I2CCard\_Read**

Command:	[2800][Byte: <i>Channel</i> ][UInt16: <i>Addr</i> ][Byte: <i>ByteCnt</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(Var): <i>Data</i> ]
Example	
Command:	28002800000A (Channel: 28, Addr: 0000, ByteCnt: 0A)
Response:	00010A001122849A2789DFD54342 (Result: true, Data: 001122849A2789DFD543)

**1.5.32.2. I2CCard\_Write**

Command:	[2801][Byte: <i>Channel</i> ][UInt16: <i>Addr</i> ][Byte Array(Var): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	280128000000401020304 (Channel: 28, Addr: 0000, Data: 01020304)
Response:	0001 (Result: true)

**1.5.33. API TOPAZ****1.5.33.1. TopazRID**

Command:	[2900]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>HR0</i> ][Byte: <i>HR1</i> ][Byte Array(4): <i>UID</i> ]
Example	
Command:	2900
Response:	0001124CA9747300 (Result: true, HR0: 18, HR1: 76, UID: A9747300)

**1.5.33.2. TopazReadByte**

Command:	[2901][Byte Array(4): <i>UID</i> ][Byte: <i>ADD</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>Data</i> ]
Example	
Command:	2901A97473000A (UID: A9747300, ADD: 0A)
Response:	000133 (Result: true, Data: 51)



**1.5.33.3. TopazReadAllBlocks**

Command:	[2902][Byte Array(4): <i>UID</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte: <i>HR0</i> ][Byte: <i>HR1</i> ][Byte Array(120): <i>Data</i> ]
Example	
Command:	2902A9747300 (UID: A9747300)
Response:	0001124CA974730000102500E11033000103F230330203F002030319- D1011555036A7562617465632E65752F6E66632D746167732F2D7461- 67732F00AB001100- 0005555AAAA- 124C060001E0000000000000 (Result: true, HR0: 18, HR1: 76, Data: A974730000102500E11033000103F230330203F002030319D101155- 5036A7562617465632E65752F6E66632D746167732F2D746167732F- 00AB0011000- 0005555AAAA124C- 060001E000000000000000)

**1.5.33.4. TopazWriteByteWithErase**

Command:	[2903][Byte Array(4): <i>UID</i> ][Byte: <i>ADD</i> ][Byte: <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2903A97473000A11 (UID: A9747300, ADD: 0A, Data: 11)
Response:	0001 (Result: true)

**1.5.33.5. TopazWriteByteNoErase**

Command:	[2904][Byte Array(4): <i>UID</i> ][Byte: <i>ADD</i> ][Byte: <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2904A97473000A22 (UID: A9747300, ADD: 0A, Data: 22)
Response:	0001 (Result: true)

**1.5.34. API CTS****1.5.34.1. CTS\_ReadBlock**

Command:	[2A00][Byte: <i>Block</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(2): <i>Data</i> ]
Example	
Command:	2A0000 (Block: 00)
Response:	00016002 (Result: true, Data: 6002)

**1.5.34.2. CTS\_WriteBlock**

Command:	[2A01][Byte: <i>Block</i> ][Byte Array(2): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2A01000000 (Block: 00, Data: 0000)
Response:	0001 (Result: true)

**1.5.34.3. CTS\_UpdateBlock**

Command:	[2A02][Byte: <i>Block</i> ][Byte Array(2): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2A02000000 (Block: 00, Data: 0000)
Response:	0001 (Result: true)

**1.5.35. API EM4305****1.5.35.1. EM4305\_Begin**

Command:	[2F00]
Response:	[00]
Example	
Command:	2F00
Response:	00

**1.5.35.2. EM4305\_Read**

Command:	[2F01][Byte: <i>Address</i> ]
Response:	[00][Bool: <i>Result</i> ][Byte Array(4): <i>Data</i> ]
Example	
Command:	2F0100 (Address: 00)
Response:	000100000001 (Result: true, Data: 00000001)

**1.5.35.3. EM4305\_Write**

Command:	[2F02][Byte: <i>Address</i> ][Byte Array(4): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2F0205003A36BA (Address: 05, Data: 003A36BA)
Response:	0001 (Result: true)

**1.5.35.4. EM4305\_Login**

Command:	[2F03][Byte Array(4): <i>Password</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2F0300000000 (Password: 00000000)
Response:	0001 (Result: true)

**1.5.35.5. EM4305\_Protect**

Command:	[2F04][Byte Array(4): <i>Data</i> ]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2F0400000000 (Data: 00000000)
Response:	0001 (Result: true)

**1.5.35.6. EM4305\_Disable**

Command:	[2F05]
Response:	[00][Bool: <i>Result</i> ]
Example	
Command:	2F05
Response:	0001 (Result: true)

## A. How to Set Specific Tags in Simple Protocol

The firmware function `SetTagTypes()` enables you to set specific transponder types to search for these types only. To set a specific transponder type, you need the definition for this type, which can be found in the table below.

### A.1. Example with Enabling Only MIFARE

For a MIFARE transponder, the definition is as following:

HFTAG_MIFARE	0x80
--------------	------

With the TAGMASK definition ( $1 \ll (\text{Tagtype} \& 0x1F)$ ) we can calculate:

$$1 \ll (0x80 \& 0x1F)$$

Result from  $(0x80 \& 0x1F)$  is 0, so we shift 1 to the left for 0 places and we get 00000001 in binary, which is 00000001h.

Now we take this result and put it in `SetTagTypes` command from Simple Protocol.

```
[0502] [UInt32: TagTypesLF] [UInt32: TagTypesHF]
```

In this command it is stated that `HFTagTypes` is `UInt32`. Simple Protocol works with little endian, so instead of 00000001h it needs to be 01000000h.

Therefore to enable only Mifare following command should be sent (without the spaces):

```
0502 0000000000 01000000
```

where

0502 - command code

00000000 - means that none low frequency technology is enabled

01000000 - means that only Mifare from high frequency technologies is enabled

### A.2. Example with Felica and HID Prox Only

Felica and HID Prox have the following definitions:

HFTAG_FELICA	0x85
LFTAG_HIDPROX	0x49

High Frequency:

With the TAGMASK definition ( $1 \ll (\text{Tagtype} \& 0x1F)$ ) we can calculate:

$1 \ll (0x85 \& 0x1F)$

Result from  $(0x85 \& 0x1F)$  is 5, so we shift 1 to the left for 5 places and we get 00100000 in binary, which is 00000020h.

HFTagTypes [UInt32] in little endian is now 20000000.

Low Frequency:

With the TAGMASK definition ( $1 \ll (\text{Tagtype} \& 0x1F)$ ) we can calculate:

$1 \ll (0x49 \& 0x1F)$

Result from  $(0x49 \& 0x1F)$  is 9, so we shift 1 to the left for 9 places and we get 0000001000000000 in binary, which is 00000200h.

LFTagTypes [UInt32] in little endian is now 00020000.

Correct command is (without the spaces):

0502 00020000 20000000

The following definitions can also be found in `twn4.sys.h`, which is part of the TWN4 Development Pack.

Technology	Definition	Tagtype
LF	LFTAG_EM4102	0x40
	LFTAG_HITAG1S	0x41
	LFTAG_HITAG2	0x42
	LFTAG_EM4150	0x43
	LFTAG_AT5555	0x44
	LFTAG_ISOFDX	0x45
	LFTAG_EM4026	0x46
	LFTAG_HITAGU	0x47
	LFTAG_EM4305	0x48
	LFTAG_HIDPROX	0x49
	LFTAG_TIRIS	0x4A
	LFTAG_COTAG	0x4B
	LFTAG_IOPROX	0x4C
	LFTAG_INDITAG	0x4D
	LFTAG_HONEYTAG	0x4E
	LFTAG_AWID	0x4F
	LFTAG_GPROX	0x50
	LFTAG_PYRAMID	0x51
	LFTAG_KERI	0x52
	LFTAG_DEISTER	0x53
	LFTAG_CARDAX	0x54
	LFTAG_NEDAP	0x55
	LFTAG_PAC	0x56
	LFTAG_IDTECK	0x57
	LFTAG_ULTRAPROX	0x58
	LFTAG_ICT	0x59
	LFTAG_ISONAS	0x5A
HF	HFTAG_MIFARE	0x80
	HFTAG_ISO14443B	0x81
	HFTAG_ISO15693	0x82
	HFTAG_LEGIC	0x83
	HFTAG_HIDICLASS	0x84
	HFTAG_FELICA	0x85
	HFTAG_SRX	0x86
	HFTAG_NFCP2P	0x87
	HFTAG_BLE	0x88
	HFTAG_TOPAZ	0x89
	HFTAG_CTS	0x8A
	HFTAG_BLELC	0x8B

## **B. Disclaimer**

ELATEC reserves the right to change any information or data in this document without prior notice. The distribution and the update of this document is not controlled. ELATEC declines all responsibility for the use of product with any other specifications but the ones mentioned above. Any additional requirement for a specific custom application has to be validated by the customer himself at his own responsibility. Where application information is given, it is only advisory and does not form part of the specification.

All referenced brands, product names, service names and trademarks mentioned in this document are the property of their respective owners.