H22 OEMSDK

Deng Elitward
V03
20110511

Table of Contents

Table of contents

Module Index

Modules

Here is a list of all modules:

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

Module Documentation

Platform / Device Information

Enumerations

- enum **DEVICE_INFO_TYPE** { **DEVICE_INFO_TYPE_OS_VERSION**, **DEVICE_INFO_TYPE_BOOTLOADER_VERSION**, **DEVICE_INFO_TYPE_IMEI**, **DEVICE_INFO_TYPE_CUSTOMER_SERIAL_NUMBER**, **DEVICE_INFO_TYPE_WIFI_ADDRESS**, **DEVICE_INFO_TYPE_BLUETOOTH_ADDRESS**, **DEVICE_INFO_TYPE_COMPANY_SERIAL_NUMBER**, **DEVICE_INFO_TYPE_PCV_VERSION** }

Functions

- BOOL **apiGetDeviceInformation** (DWORD dwInfoType, LPTSTR lpszValue, DWORD *lpdwLen, DWORD *lpdwError)

  This function gets the information of different parts.

  Including H22 OS Version, Boot loader Version, IMEI, Customer Serial Number, WiFi MAC Address, Bluetooth MAC Address, Company Serial Number and PCV Production Version.
- BOOL **apiSetDeviceInformation** (DWORD dwInfoType, LPCTSTR lpszValue, DWORD *lpdwError)

  This function set the information of different parts.

  Including Customer Serial Number, Bluetooth MAC Address and Company Serial Number only.

Detailed Description

Additional documentation for group 'Platform / Device Information'

Enumeration Type Documentation

enum DEVICE_INFO_TYPE

Used in function **apiGetDeviceInformation()** & **apiSetDeviceInformation()**

Enumerator:

DEVICE_INFO_TYPE_OS_VERSION	H22 OS Version
DEVICE_INFO_TYPE_BOOTLOADER_VERSION	Boot loader Version
DEVICE_INFO_TYPE_IMEI	IMEI
DEVICE_INFO_TYPE_CUSTOMER_SERIAL_NUMBER	Customer Serial Number
DEVICE_INFO_TYPE_WIFI_ADDRESS	WiFi MAC Address
DEVICE_INFO_TYPE_BLUETOOTH_ADDRESS	Bluetooth MAC Address
DEVICE_INFO_TYPE_COMPANY_SERIAL_NUMBER	Company Serial Number
DEVICE_INFO_TYPE_PCV_VERSION	PCV Production Version

Function Documentation

BOOL apiGetDeviceInformation (DWORD*dwInfoType*, LPTSTR*lpszValue*, DWORD **lpdwLen*, DWORD **lpdwError*)

This function gets the information of different parts.

Including H22 OS Version, Boot loader Version, IMEI, Customer Serial Number, WiFi MAC Address, Bluetooth MAC Address, Company Serial Number and PCV Production Version.

Parameters:

in	*dwInfoType*	A DWORD specifying which type of Information is being requested. Could be any member defined in enum DEVICE_INFO_TYPE.
out	*lpszValue*	A pointer to a sting buff, which will be used to retrieve requested information in the form of a null-terminated string. The length of the information string depends on dwInfoType parameter. If buff size is too small, it will not be set, the function should return FALSE and lpdwError be set accordingly.
in,out	*lpdwLen*	A pointer to a DWORD variable indicating the size of retrieving buff size in character. The length of buff being set will be also stored after return. If buff size is too small, this returned value tells the required buff size, , the function should return FALSE and lpdwError be set accordingly.
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Remarks:

The minimum length of the buff (lpszValue) depends on dwInfoType parameter according, reference the table below:

DEVICE_INFO_TYPE	Minimum length of lpszValue	Example
DEVICE_INFO_TYPE_OS_VERSION	12+1	0.17.03A
DEVICE_INFO_TYPE_BOOTLOADER_VERSION	12+1	0.05.03
DEVICE_INFO_TYPE_IMEI	15+1	123455432112345
DEVICE_INFO_TYPE_CUSTOMER_SERIAL_NUMBER	12+1	NA
DEVICE_INFO_TYPE_WIFI_ADDRESS	12+1	001122334455
DEVICE_INFO_TYPE_BLUETOOTH_ADDRESS	12+1	112233445566
DEVICE_INFO_TYPE_COMPANY_SERIAL_NUMBER	16+1	NA
DEVICE_INFO_TYPE_PCV_VERSION	8+1	NA

The +1 is reserved for null-terminator.

Returns:

TRUE - Success to get the device information

FALSE - Fail to get the device information. To get more error information, see parameter lpdwError.

BOOL apiSetDeviceInformation (DWORD*dwInfoType*, LPCTSTR*lpszValue*, DWORD **lpdwError*)

This function set the information of different parts.

Including Customer Serial Number, Bluetooth MAC Address and Company Serial Number only.

Parameters:

in	*dwInfoType*	A DWORD specifying which type of Information is being requested. Could be DEVICE_INFO_TYPE_CUSTOMER_SERIAL_NUMBER, DEVICE_INFO_TYPE_BLUETOOTH_ADDRESS, DEVICE_INFO_TYPE_COMPANY_SERIAL_NUMBER (defined in enum DEVICE_INFO_TYPE).
in	*lpszValue*	A pointer to a null-terminated string, which contains the information to set. The length of the information string depends on dwInfoType parameter. Reference **apiGetDeviceInformation()** for examples. If containing information is not accepted, such as inappropriate length, wrong format or containing illegal characters, the function will return FALSE. Additional error code will be set in lpdwError.
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE **-** Success to set the device information
FALSE **-** Fail to set the device information. To get more error information, see parameter lpdwError.

Accelerometer

Data Structures

- struct **GSENSOR_ACCELERATION**

Enumerations

- enum **GSENSOR_POWER** { **GSENSOR_ON**, **GSENSOR_OFF** }
- enum **GSENSOR_ROTATION** { **GSENSOR_ROTATE_0**, **GSENSOR_ROTATE_90**, **GSENSOR_ROTATE_180**, **GSENSOR_ROTATE_270** }

Functions

- BOOL **apiSetGSensorPower** (DWORD dwValue, DWORD *lpdwError)
  This function will set the gravity sensor's power status (ON/OFF).
- BOOL **apiGetGSensorPower** (DWORD *lpdwValue, DWORD *lpdwError)
  This function will get gravity sensor's power status (ON/OFF).
- BOOL **apiGetGSensorScreenRotation** (DWORD *lpdwValue, DWORD *lpdwError)
  This function will get gravity sensor rotate direction value.
- BOOL **apiGetGSensorAcceleration** (**GSENSOR_ACCELERATION** *lpAcceleration, DWORD *lpdwError)

This function will get gravity sensor's acceleration value in 3-dimensions.

- BOOL **apiGSensorOpenDevice** (HANDLE hNewGSensorReading, HANDLE *lphGSensorDevice, DWORD *lpdwError)
  This function creates a connection to the gravity sensor driver to enable real-time monitoring of the device spatial movement.
- BOOL **apiGSensorCloseDevice** (HANDLE hGSensorDevice, DWORD *lpdwError)
  This function closes a connection to the gravity sensor driver.

Detailed Description

The following definition and functions are for 'Accelerometer'

Enumeration Type Documentation

enum GSENSOR_POWER

Used in function **apiSetGSensorPower()** & **apiGetGSensorPower()**

Enumerator:

GSENSOR_ON	The gravity sensor is On
GSENSOR_OFF	The gravity sensor is Off

enum GSENSOR_ROTATION

Used in function **apiGetGSensorScreenRotation()**

Enumerator:

GSENSOR_ROTATE_0	0 degree
GSENSOR_ROTATE_90	90 degrees in a clockwise direction
GSENSOR_ROTATE_180	180 degrees
GSENSOR_ROTATE_270	270 degrees in a clockwise direction

Function Documentation

BOOL apiGetGSensorAcceleration (GSENSOR_ACCELERATION **lpAcceleration*, DWORD **lpdwError*)

This function will get gravity sensor's acceleration value in 3-dimensions.

Parameters:

out	*lpAcceleration*	A pointer to a struct of **GSENSOR_ACCELERATION** to retrieve the acceleration.
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to get gravity sensor's acceleration.

FALSE - Fail to get gravity sensor's acceleration. To get more error information, see parameter lpdwError.

BOOL apiGetGSensorPower (DWORD **lpdwValue*, DWORD **lpdwError*)

This function will get gravity sensor's power status (ON/OFF).

Parameters:

out	*lpdwValue*	A pointer to a DWORD to retrieve the status of gravity sensor (defined in GSENSOR_POWER).
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to get gravity sensor's power status.
FALSE - Fail to get gravity sensor's power status. To get more error information, see parameter lpdwError.

BOOL apiGetGSensorScreenRotation (DWORD **lpdwValue*, DWORD **lpdwError*)

This function will get gravity sensor rotate direction value.

Parameters:

out	*lpdwValue*	A pointer to a DWORD to retrieve rotate angle value (defined in enum GSENSOR_ROTATION).
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to get gravity sensor rotate direction value.
FALSE - Fail to get gravity sensor rotate direction value. To get more error information, see parameter lpdwError.

BOOL apiGSensorCloseDevice (HANDLE*hGSensorDevice*, DWORD **lpdwError*)

This function closes a connection to the gravity sensor driver.

Parameters:

in	*hGSensorDevice*	A HANDLE returned by a call to apiGSensorOpenDevice.
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to close a connection to the gravity sensor driver.
FALSE - Fail to close a connection to the gravity sensor driver. To get more error information, see parameter lpdwError.

BOOL apiGSensorOpenDevice (HANDLE*hNewGSensorReading*, HANDLE **lphGSensorDevice*, DWORD **lpdwError*)

This function creates a connection to the gravity sensor driver to enable real-time monitoring of the device spatial movement.

Parameters:

in	*hNewGSensorReading*	A HANDLE to a Windows Embedded CE event created using CreateEvent, or NULL. The gravity sensor driver signals the past event whenever there is a new gravity sensor reading.
out	*lphGSensorDevice*	A pointer to a HANDLE. If successful, returns a HANDLE to the gravity sensor driver. If unsuccessful, returns NULL.
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE **-** Success to create a connection to the gravity sensor driver.
FALSE **-** Fail to create a connection to the gravity sensor driver. To get more error information, see parameter lpdwError.

BOOL apiSetGSensorPower (DWORD*dwValue*, DWORD **lpdwError*)

This function will set the gravity sensor's power status (ON/OFF).

Parameters:

in	*dwValue*	A DWORD specifying whether to turn on or off the gravity sensor (defined in GSENSOR_POWER).
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE **-** Success to set gravity sensor's power status.
FALSE **-** Fail to set gravity sensor's power status. To get more error information, see parameter lpdwError.

Ambient Light Sensor

Enumerations

- enum **LIGHT_SENSOR_POWER** { **LIGHT_SENSOR_ON**, **LIGHT_SENSOR_OFF** }

Functions

- BOOL **apiGetLightSensorInfo** (DWORD *lpdwValue, DWORD *lpdwError)
  This function will get light sensor's current brightness level value.
- BOOL **apiSetLightSensorPower** (DWORD dwValue, DWORD *lpdwError)
  This function will turn light sensor on or off.
- BOOL **apiLightSensorOpenDevice** (HANDLE hNewLightSensorReading, HANDLE *lphLightSensorDevice, DWORD *lpdwError)
  This function creates a connection to the light sensor driver to enable real-time monitoring of brightness in the environment.
- BOOL **apiLightSensorCloseDevice** (HANDLE hLightSensorDevice, DWORD *lpdwError)
  This function closes the connection to the light sensor driver.

Detailed Description

The following definition and functions are for 'Ambient Light Sensor'

Enumeration Type Documentation

enum LIGHT_SENSOR_POWER

Used in function **apiSetLightSensorPower()**

Enumerator:

LIGHT_SENSOR_ON The Light sensor is On

LIGHT_SENSOR_OFF The Light sensor is Off

Function Documentation

BOOL apiGetLightSensorInfo (DWORD **lpdwValue*, DWORD **lpdwError*)

This function will get light sensor's current brightness level value.

Parameters:

out	*lpdwValue*	A pointer to a DWORD to retrieve the value of Bright Level, range from 0 to 5. If Light sensor is Off, this value will be 0. When Light sensor is On, the value will be a positive number. (The larger number means the brighter.)
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to get light sensor's brightness level.
FALSE - Fail to get light sensor's brightness level. To get more error information, see parameter lpdwError.

BOOL apiLightSensorCloseDevice (HANDLE*hLightSensorDevice*, DWORD **lpdwError*)

This function closes the connection to the light sensor driver.

Parameters:

in	*hLightSensorDevice*	A HANDLE returned by a call to apiLightSensorOpenDevice.
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to close the connection to the light sensor driver.

FALSE - Fail to close the connection to the light sensor driver. To get more error information, see parameter lpdwError.

BOOL apiLightSensorOpenDevice (HANDLE*hNewLightSensorReading*, HANDLE **lphLightSensorDevice*, DWORD **lpdwError*)

This function creates a connection to the light sensor driver to enable real-time monitoring of brightness in the environment.

Parameters:

in	*hNewLightSensorReading*	A HANDLE to a Windows Embedded CE event created using CreateEvent, or NULL. The light sensor driver signals the past event whenever there is a new Light Sensor reading.
out	*lphLightSensorDevice*	A pointer to a HANDLE. If successful, returns a HANDLE to the light sensor driver. If unsuccessful, returns NULL.
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to create a connection to the light sensor driver.
FALSE - Fail to create a connection to the light sensor driver. To get more error information, see parameter lpdwError.

BOOL apiSetLightSensorPower (DWORD*dwValue*, DWORD **lpdwError*)

This function will turn light sensor on or off.

Parameters:

in	*dwValue*	A DWORD specifying whether to turn on or off the light sensor (defined in enum LIGHT_SENSOR_POWER).
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to the set light sensor's power.
FALSE - Fail to the set light sensor's power. To get more error information, see parameter lpdwError.

Barcode Sensor

The following definition and functions are for 'Barcode Sensor'

CCX Supplicant

The following definition and functions are for 'CCX Supplicant'

Near Field Communication (RFID)

Enumerations

- enum **NFC_POWER** { **NFC_POWER_OFF** = 0x00, **NFC_POWER_ON** = 0x01 }
- enum **NFC_POLLING_TYPE** { **NFC_POLLING_TYPE_ISO14443A** = 0x01, **NFC_POLLING_TYPE_ISO14443B** = 0x02, **NFC_POLLING_TYPE_FELICA212** = 0x04, **NFC_POLLING_TYPE_FELICA424** = 0x08, **NFC_POLLING_TYPE_ISO15693** = 0X10 }
- enum **NFC_DATA_FORMAT_TYPE** { **NFC_DATA_FORMAT_TYPE_A** = 6, **NFC_DATA_FORMAT_TYPE_B** = 7, **NFC_DATA_FORMAT_FELICA** = 8, **NFC_DATA_FORMAT_JEWEL** = 9, **NFC_DATA_FORMAT_MIFARE_1K** = 10, **NFC_DATA_FORMAT_MIFARE_UL** = 11, **NFC_DATA_FORMAT_MIFARE_DESFIRE** = 12, **NFC_DATA_FORMAT_ISO15693** = 14, **NFC_DATA_FORMAT_MIFARE_MINI** = 15 }
- enum **NFC_MifareCmd** { **NFC_MifareRaw** = 0x00U, **NFC_MifareAuthentA** = 0x60U, **NFC_MifareAuthentB** = 0x61U, **NFC_MifareRead16** = 0x30U, **NFC_MifareRead** = 0x30U, **NFC_MifareWrite16** = 0xA0U, **NFC_MifareWrite4** = 0xA2U, **NFC_MifareInc** = 0xC1U, **NFC_MifareDec** = 0xC0U, **NFC_MifareTransfer** = 0xB0U, **NFC_MifareRestore** = 0xC2U, **NFC_MifareReadSector** = 0x38U, **NFC_MifareWriteSector** = 0xA8U, **NFC_MifareInvalidCmd** = 0xFFU }
- enum **NFC_Iso14443_4_Cmd** { **NFC_Iso14443_4_Raw** = 0x00U }
- enum **NFC_Iso15693_Cmd** { **NFC_Iso15693_Cmd** = 0x20U, **NFC_Iso15693_Inventory** = 0x01U, **NFC_Iso15693_Stay_Quite** = 0x02U, **NFC_Iso15693_Read_Single_Block** = 0x20U, **NFC_Iso15693_Write_Single_Block** = 0x21U, **NFC_Iso15693_Lock_Block** = 0x22U, **NFC_Iso15693_Read_Multiple_Blocks** = 0x23U, **NFC_Iso15693_Write_Multiple_Blocks** = 0x24U, **NFC_Iso15693_ResetToReady** = 0x26U, **NFC_Iso15693_Write_AFI** = 0x27U, **NFC_Iso15693_LockAFI** = 0x28U, **NFC_Iso15693_WriteDSFID** = 0x29U, **NFC_Iso15693_LockDSFID** = 0x2AU, **NFC_Iso15693_Get_System_Information** = 0x2BU, **NFC_Iso15693_Get_Multiple_Block_Security_status** = 0x2CU }
- enum **NFC_FelicaCmd** { **NFC_Felica_Raw** = 0xF0U, **NFC_Felica_Check** = 0x00, **NFC_Felica_Update** = 0x01, **NFC_Felica_Invalid** = 0xFFU }
- enum **NFC_JewelCmd** { **NFC_Jewel_ReadAll** = 0x00U, **NFC_Jewel_Read1** = 0x01U, **NFC_Jewel_Read4** = 0x02U, **NFC_Jewel_Read8** = 0x03U, **NFC_Jewel_Write1E** = 0x53U, **NFC_Jewel_Write4E** = 0x54U, **NFC_Jewel_Write8E** = 0x55U, **NFC_Jewel_Write1NE** = 0x1AU, **NFC_Jewel_Write4NE** = 0x1BU, **NFC_Jewel_Write8NE** = 0x1CU, **NFC_Jewel_RID** = 0x78U, **NFC_Jewel_ReadSeg** = 0x10U, **NFC_Jewel_Read** = 0x21U, **NFC_Jewel_Write** = 0x22U, **NFC_Jewel_Invalid** = 0xFFU }

Functions

- BOOL **apiSetNfcPower** (DWORD dwPowerType, DWORD *lpdwError)

  This function turns NFC power on or off.
- BOOL **apiNfcPolling** (DWORD *lpdwPollingType, DWORD *lpdwResponseSize, void *lpResponseData, DWORD *lpdwError)

  This function performs a polling to detect the selected type of card.

  It is called to discover the remote devices.
- BOOL **apiNfcDisconnect** (DWORD *lpdwError)

  This function allows disconnecting the connected tag.

  The client is required to re-enable the apiNfcPolling, it must call the apiNfcDisconnect first.
- DWORD **apiNfcTransceive** (DWORD dwFormatType, DWORD *lpdwOutSize, void *lpOutData, DWORD dwInSize, const void *lpInData, DWORD dwCommand, DWORD dwBlock, DWORD *lpdwError)

  This function is called to transmit and receive data to and from the tag.

Detailed Description

The following definition and functions are for 'RFID'

Enumeration Type Documentation

enum NFC_DATA_FORMAT_TYPE

Used in function **apiNfcPolling()** & **apiNfcTransceive()**

Enumerator:

NFC_DATA_FORMAT_TYPE_A	TYPE A
NFC_DATA_FORMAT_TYPE_B	TYPE B
NFC_DATA_FORMAT_FELICA	FELICA
NFC_DATA_FORMAT_JEWEL	JEWEL
NFC_DATA_FORMAT_MIFARE_1K	MIFARE 1K
NFC_DATA_FORMAT_MIFARE_UL	MIFARE UL
NFC_DATA_FORMAT_MIFARE_DESFIRE	MIFARE DESFIRE
NFC_DATA_FORMAT_ISO15693	ISO15693
NFC_DATA_FORMAT_MIFARE_MINI	MIFARE MINI

enum NFC_FelicaCmd

Used in function **apiNfcTransceive()**

Enumerator:

NFC_Felica_Raw Felica Raw command:

- This command sends the data buffer directly to the remote device

NFC_Felica_Check Felica Check command:

- This command checks the data from the Felica remote device

NFC_Felica_Update Felica Update command:

- This command updates the data onto the Felica remote device

NFC_Felica_Invalid Invalid Command

enum NFC_Iso14443_4_Cmd

Used in function **apiNfcTransceive()**

Enumerator:

NFC_Iso14443_4_Raw ISO 14443-4 Exchange command:

- This command sends the data buffer directly to the remote device

enum NFC_Iso15693_Cmd

Used in function **apiNfcTransceive()**

Enumerator:

NFC_Iso15693_Cmd ISO 15693 Exchange command:

- This command is used to access the card to the remote device

NFC_Iso15693_Stay_Quite Inventory

NFC_Iso15693_Read_Single_Block Stay Quite

NFC_Iso15693_Write_Single_Block Read Single Block

NFC_Iso15693_Lock_Block Write Single Block

NFC_Iso15693_Read_Multiple_Blocks Lock Block

NFC_Iso15693_Write_Multiple_Blocks Read Multiple Blocks

NFC_Iso15693_ResetToReady Write Multiple Blocks

NFC_Iso15693_Write_AFI ResetToReady

NFC_Iso15693_LockAFI Write AFI

NFC_Iso15693_WriteDSFID LockAFI

NFC_Iso15693_LockDSFID WriteDSFID

NFC_Iso15693_Get_System_Information LockDSFID

NFC_Iso15693_Get_Multiple_Block_Security_status Get System Information

enum NFC_JewelCmd

Used in function **apiNfcTransceive()**

Enumerator:

NFC_Jewel_ReadAll Jewel command:

- This command sends the data buffer directly to the remote device

NFC_Jewel_Read1 Read one byte.

NFC_Jewel_Read4 Read four bytes.

NFC_Jewel_Read8 Read eight bytes.

NFC_Jewel_Write1E Erase and Write one byte

NFC_Jewel_Write4E Erase and Write four bytes

NFC_Jewel_Write8E Erase and Write eight bytes

NFC_Jewel_Write1NE Write one byte Without Erase

NFC_Jewel_Write4NE Write four bytes Without Erase

NFC_Jewel_Write8NE Write eight bytes Without Erase

NFC_Jewel_RID Read the identification information

NFC_Jewel_ReadSeg Read the data segment

NFC_Jewel_Read Read the data from the specified block

NFC_Jewel_Write Write the data to specified block

NFC_Jewel_Invalid Invalid jewel command

enum NFC_MifareCmd

Used in function **apiNfcTransceive()**

Enumerator:

NFC_MifareRaw Raw Command Mode.

NFC_MifareAuthentA Mifare Standard:

This command performs an authentication with KEY A for a sector.

Format of the buffer to send with this command :

- first byte : memory (block) adress to authenticate.
- next six bytes : key A.

Example: 0x **08** 01 02 03 04 05 06 authenticates block 08 with the key 0x01[..]06. If this command fails, the user needs to reactivate the remote Mifare card

NFC_MifareAuthentB Mifare Standard:

This command performs an authentication with KEY B for a sector.

Format of the buffer to send with this command :

- first byte : memory (block) adress to authenticate.
- next six bytes : key B.

Example: See **NFC_MifareAuthentA** . If this command fails, the user needs to reactivate the the remote Mifare card

NFC_MifareRead16 Mifare Standard and Ultra Light:

Read 16 Bytes from a Mifare Standard block or 4 Mifare Ultra Light pages.

Format of the buffer to send with this command :

- 1st (and only) byte : memory adress to read.

Example: 0x **08** reads the data (16 bytes) from the specified address. If this command fails, the user needs to reactivate the remote Mifare card

NFC_MifareWrite16 Mifare Standard:

Write 16 bytes.

Format of the buffer to send with this command :

- first byte : start address of memory to write.
- next 16 bytes : Data to write.

Example: 0x **08** 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F If this command fails, the user needs to reactivate the remote Mifare card

NFC_MifareWrite4 Mifare Ultra Light:

Write 4 bytes.

Format of the buffer to send with this command :

- first byte : start address of memory (Mifare Ultra Light page) to write.
- next 4 bytes : Data to write.

Example: 0x **08** 00 01 02 03 If this command fails, the user needs to reactivate the remote Mifare card

NFC_MifareInc Increment.

NFC_MifareDec Decrement.

NFC_MifareTransfer Tranfer.

NFC_MifareRestore Restore.

NFC_MifareReadSector Read Sector.

NFC_MifareWriteSector Write Sector.

NFC_MifareInvalidCmd Invalid Command

enum NFC_POLLING_TYPE

Used in function **apiNfcPolling()**

Enumerator:

NFC_POLLING_TYPE_ISO14443A ISO14443A

NFC_POLLING_TYPE_ISO14443B ISO14443B

NFC_POLLING_TYPE_FELICA212 FELICA21

NFC_POLLING_TYPE_FELICA424 FELICA424

NFC_POLLING_TYPE_ISO15693 ISO15693

enum NFC_POWER

Used in function **apiSetNfcPower()**

Enumerator:

NFC_POWER_OFF NFC power is Off

NFC_POWER_ON NFC power is On

Function Documentation

BOOL apiNfcDisconnect (DWORD **lpdwError*)

This function allows disconnecting the connected tag.

The client is required to re-enable the apiNfcPolling, it must call the apiNfcDisconnect first.

Parameters:

out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to disconnect the opened session.
FALSE - Fail to disconnect the open session. To get more error information, see parameter lpdwError.

BOOL apiNfcPolling (DWORD **lpdwPollingType*, DWORD **lpdwResponseSize*, void **lpResponseData*, DWORD **lpdwError*)

This function performs a polling to detect the selected type of card.

It is called to discover the remote devices.

Parameters:

in,out	*lpdwPollingType*	A DWORD specifying the type of access to the object on using as an input (defined in NFC_POLLING_TYPE). As return value, it will be set in order to retrieve the returning data type (defined in NFC_DATA_FORMAT_TYPE).
out	*lpdwResponseSize*	A pointer to a DWORD, which indicates the size of lpResponseData and also is used to retrieve the number of bytes in response data.
out	*lpResponseData*	A pointer to a buffer that receives the data from remote object.
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Remarks:

Reference the tables below for the formats of returned data:

TypeA	
1 byte	Tagtype=6
1 byte	UidLength
10 bytes[MAX]	Uid
1 byte	AppDateLength
48 bytes[MAX]	AppDate(Application data information of the tag discovered)
1 byte	Sak
2 bytes	AtqA

TypeB	
1 byte	Tagtype=7
12 bytes	AtqRes
1 byte	Afi (Application Family Identifier)

Mifare	

1 byte	Tagtype:
	MIFARE1K=10
	MIFAREUL=11
	MIFAREDESFIRE=12

1 byte	UidLength
10 bytes[MAX]	Uid
1 byte	Sak
2 bytes	AtqA

Felica	
1 byte	Tagtype=8
1 byte	IDmLength
10 bytes[MAX]	IDm
8 bytes	PMm
2 bytes	SystemCode

ISO15693	
1 byte	Tagtype=14
1 byte	UidLength

8 bytes[MAX]	Uid
1 byte	Afi (Application Family Identifier)

Returns:

TRUE - Success to poll the NFC card.
FALSE - Fail to poll the NFC card. To get more error information, see parameter lpdwError.

DWORD apiNfcTransceive (DWORD*dwFormatType*, DWORD **lpdwOutSize*, void **lpOutData*, DWORD*dwInSize*, const void **lpInData*, DWORD*dwCommand*, DWORD*dwBlock*, DWORD **lpdwError*)

This function is called to transmit and receive data to and from the tag.

Parameters:

in	*dwFormatType*	This parameter defines the type of data format (defined in enum NFC_DATA_FORMAT_TYPE).
out	*lpdwOutSize*	Pointer to the buffer which contains data to transmit.
out	*lpOutData*	Pointer to the buffer which contains data to transmit.
in	*dwInSize*	Pointer to the number of bytes which received from remote object.
in	*lpInData*	Pointer to the buffer which receives data from remote object.
in	*dwCommand*	This parameter defines the command of type cards.
in	*dwBlock*	This parameter defines block number. (reserved, set as 0 for now)
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

A negative value indicates error. Otherwise, the value is the number of bytes copied into the location pointed out by InData. To get more error information, see parameter lpdwError.

BOOL apiSetNfcPower (DWORD*dwPowerType*, DWORD **lpdwError*)

This function turns NFC power on or off.

Parameters:

in	*dwPowerType*	A DWORD specifying whether to turn on or off the NFC power (defined in enum NFC_POWER).
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to power NFC on or off .
FALSE - Fail to power NFC on or off. To get more error information, see parameter lpdwError.

Smart Card / SAM Module

Use Microsoft standard API to control Smart Card / SAM Module related functions.

Camera

Use Microsoft standard DirectX interface to control Camera related functions (including Auto-Focus).

Vibrator

The part will be implemented by windows mobile API: NledSetDevice()

Custom ROM File Transfer Sample

Enumerations

- enum **DEVICE_RESET_TYPE** { **DEVICE_RESET_TYPE_FACTORY_RESET**, **DEVICE_RESET_TYPE_CLEAN_BOOT** }

Functions

- BOOL **apiFormatCustomROM** (LPCTSTR lpszPartitionName, DWORD *lpdwError)

  This function will format the Custom ROM partition.
- BOOL **apiBatchCopyToCustomROM** (LPCTSTR lpszPartitionName, DWORD *lpdwError)

  This function will copy all the files in "CopyToCustomROM" folder of micro SD-card to Custom ROM area.

  But excluding "CustomROMTool.exe".
- BOOL **apiFactoryReset** (DWORD dwResetType, DWORD *lpdwError)

  This function will perform Factory Reset or Clean Boot.

  Clean Boot will all remove applications, restore registry and databases to default.

  Factory Reset will do a clean boot AND also format the Custom ROM area and the Flash Disk.

Detailed Description

The following definition and functions are for 'Custom ROM File Transfer Sample'

Enumeration Type Documentation

enum DEVICE_RESET_TYPE

Used in function **apiFactoryReset()**

Enumerator:

DEVICE_RESET_TYPE_FACTORY_RESET	Factory Reset
DEVICE_RESET_TYPE_CLEAN_BOOT	Clean Boot

Function Documentation

BOOL apiBatchCopyToCustomROM (LPCTSTR*lpszPartitionName*, DWORD *_lpdwError_)

This function will copy all the files in "CopyToCustomROM" folder of micro SD-card to Custom ROM area.

But excluding "CustomROMTool.exe".

Parameters:

in	*lpszPartitionName*	A null-terminated string specifying the partition name containing "CopyToCustomROM" folder.
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - All the files in "CopyToCustomROM" folder of micro SD-card are successfully copy to the Custom ROM area.
FALSE - An error occurred. If there is not enough storage to complete this operation, it will return FALSE.

Sample Code:

```
DWORD getErr=0;
BOOL suc = apiBatchCopyToCustomROM(_T("\\CustomROM"), &getErr);
```

BOOL apiFactoryReset (DWORD*dwResetType*, DWORD *_lpdwError_)

This function will perform Factory Reset or Clean Boot.

Clean Boot will all remove applications, restore registry and databases to default.

Factory Reset will do a clean boot AND also format the Custom ROM area and the Flash Disk.

Parameters:

in	*dwResetType*	A DWORD specifying which type of factory reset to perform. Could be any member defined in enum DEVICE_RESET_TYPE.
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to perform factory reset or clean boot.
FALSE - Fail to perform factory reset or clean boot. To get more error information, see parameter lpdwError.

BOOL apiFormatCustomROM (LPCTSTR*lpszPartitionName*, DWORD *_lpdwError_)

This function will format the Custom ROM partition.

Parameters:

in	*lpszPartitionName*	A null-terminated string specifying the name of partition to format.

out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to format the Custom ROM partition.
FALSE - Fail to format the Custom ROM partition. To get more error information, see parameter lpdwError.

Flash LED

The part will be implemented by windows mobile API: NledSetDevice()

Status LED

The part will be implemented by windows mobile API: NledSetDevice()

Keyboard Backlight

The part will be implemented by windows mobile API: NledSetDevice()

Power

Enumerations

- enum **POWER_MODE** { **POWER_MODE_BATTERY**, **POWER_MODE_EXTERNAL** }
- enum **BACKLIGHT_BRIGHTNESS** { **BACKLIGHT_BRIGTHNESS_OFF**, **BACKLIGHT_BRIGTHNESS_1**, **BACKLIGHT_BRIGTHNESS_2**, **BACKLIGHT_BRIGTHNESS_3**, **BACKLIGHT_BRIGTHNESS_4** }
- enum **BACKUP_BATTERY_STATUS** { **BACKUP_BATTERY_STATUS_GOOD**, **BACKUP_BATTERY_STATUS_BAD** }

Functions

- BOOL **apiSetBacklightBrightness** (DWORD dwStatus, DWORD dwValue, DWORD *lpdwError)

  This function will set the brightness level value of screen backlight for either on battery power or on external power.
- BOOL **apiGetBacklightBrightness** (DWORD dwStatus, DWORD *lpdwValue, DWORD *lpdwError)

  This function will get the brightness level value of screen backlight for both on battery power or on external power.
- BOOL **apiGetBackupBatteryStatus** (DWORD *lpdwStatus, DWORD *lpdwError)

  This function will get the status of bakeup battery.
- BOOL **apiSuspendDevice** (DWORD *lpdwError)

  This function can be used to suspend the device.

Detailed Description

The following definition and functions are for 'Power'

Enumeration Type Documentation

enum BACKLIGHT_BRIGHTNESS

Used in function **apiSetBacklightBrightness()** and **apiGetBacklightBrightness()**

Enumerator:

BACKLIGHT_BRIGTHNESS_OFF	Backlight is Off
BACKLIGHT_BRIGTHNESS_1	Backlight brightness level 1 (darkest)
BACKLIGHT_BRIGTHNESS_2	Backlight brightness level 2
BACKLIGHT_BRIGTHNESS_3	Backlight brightness level 3
BACKLIGHT_BRIGTHNESS_4	Backlight brightness level 4 (brightest)

enum BACKUP_BATTERY_STATUS

Used in function **apiGetBackupBatteryStatus()**

Enumerator:

BACKUP_BATTERY_STATUS_GOOD	Backup battery is good
BACKUP_BATTERY_STATUS_BAD	Backup battery is bad

enum POWER_MODE

Used in function **apiSetBacklightBrightness()** and **apiGetBacklightBrightness()**

Enumerator:

POWER_MODE_BATTERY	on battery power
POWER_MODE_EXTERNAL	on external power

Function Documentation

BOOL apiGetBacklightBrightness (DWORD*dwStatus*, DWORD **lpdwValue*, DWORD **lpdwError*)

This function will get the brightness level value of screen backlight for both on battery power or on external power.

Parameters:

in	*dwStatus*	A DWORD specifying whether the brightness level to set is for the status of either battery only or external power. (defined in enum POWER_MODE)
out	*lpdwValue*	A pointer to a DWORD to retrieve brightness level. (defined in enum BACKLIGHT_BRIGHTNESS)
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in

		enum DI_ERRORNO).

Returns:

TRUE - Success to get the brightness level value.
FALSE - Fail to get the brightness level value. To get more error information, see parameter lpdwError.

BOOL apiGetBackupBatteryStatus (DWORD **lpdwStatus*, DWORD **lpdwError*)

This function will get the status of bakeup battery.

Parameters:

out	*lpdwStatus*	A pointer to a DWORD to retrieve brightness level. (defined in enum BACKUP_BATTERY_STATUS)
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to get the backup battery status.
FALSE - Fail to get the backup battery status. To get more error information, see parameter lpdwError.

BOOL apiSetBacklightBrightness (DWORD*dwStatus*, DWORD*dwValue*, DWORD **lpdwError*)

This function will set the brightness level value of screen backlight for either on battery power or on external power.

Parameters:

in	*dwStatus*	A DWORD specifying whether the brightness level to set is for the status of either battery only or external power. (defined in enum POWER_MODE)
in	*dwValue*	A DWORD specifying the brightness level to set. (defined in enum BACKLIGHT_BRIGHTNESS)
out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE - Success to set the brightness level value.
FALSE - Fail to set the brightness level value. To get more error information, see parameter lpdwError.

BOOL apiSuspendDevice (DWORD **lpdwError*)

This function can be used to suspend the device.

Parameters:

out	*lpdwError*	A pointer to a DWORD to retrieve error information (defined in enum DI_ERRORNO).

Returns:

TRUE **-** Success to suspend the device.

FALSE **-** Fail to suspend the device. To get more error information, see parameter lpdwError.

Data Structure Documentation

GSENSOR_ACCELERATION Struct Reference

`#include <IACUTIL.h>`

Data Fields

- DWORD **dwAcclX**
- DWORD **dwAcclY**
- DWORD **dwAcclZ**

Detailed Description

Used in function **apiGetGSensorAcceleration()**

Field Documentation

DWORD dwAcclX

acceleration in the X dimension

DWORD dwAcclY

acceleration in the Y dimension

DWORD dwAcclZ

acceleration in the Z dimension

Index

INDEX