

# UNIVERSAL SDK Version 7.7.2

---



## UNIVERSAL SDK

pcProx API DLL Windows,  
Linux & Mac Library

For pcProx® Plus, pcProx,  
pcSwipe™ and pcProx Sonar.

Version 7.7.2

## Support

---

Congratulations on the purchase of the Software Developer's Kit for the pcProx, pcProx Plus reader or Wiegand Converters. rf IDEAS knows you will enjoy using the new device as much as we enjoy creating and developing it! Configuration is easy so you will be able to quickly take advantage of a more secure environment in your business, school, or organization.

We are always discovering new applications for our product line. There are several software developers licensing our technology so the solution you are looking for may already be developed.

Please call our sales department if you have any questions or are interested in our OEM and Independent Developer's programs.

We look forward to your comments and suggestions for future upgrades. Please go to [www.rfIDEAS.com](http://www.rfIDEAS.com) and follow the Support / Learning Center link for more details about our product line.

The rf IDEAS staff

rf IDEAS, Inc.  
425 North Martingale Road,  
Suite 1680.  
Schaumburg, IL 60173

Toll-free: +1(866)439-4884  
Voice: +1(847)-870-1723  
Fax: +1(847)-483-1129

E-mail: [TechSupport@rfIDEAS.com](mailto:TechSupport@rfIDEAS.com)  
<http://www.rfIDEAS.com>

## What's new in Version 7 of this library

---

Beginning with version 7.0.0 this library now supports three products: pcProx, pcSwipe and pcProx Sonar. The basic functionality was derived from the pcProx library version 6.

Version 7.1.2 supports multiple cards types for pcProx Plus.

Version 7.1.3 supports pcProx Plus for Serial and Ethernet TCP/IP, POE and EIP for Rockwell Automation®.

Version 7.1.4 supports Ethernet 241™ Discovery protocol.

Version 7.1.5 supports GetCardList from reader / Ping / IsCardTypeInList. [Config]C++ Code Generation Enable String Pooling YES /GF.

Version 7.1.7 supports libusb for NTWCC Bulk end point 3 CDC like.

Version 7.2.0 supports the raw feature report for NT-ware.

Version 7.2.1 & 7.2.2 fix for rf IDEAS Service Call 103055-regression issue-Caradigm USA LLC.

Version 7.2.3 supports backward compatibility of USB devices.

Version 7.2.4 HealthCast Issue:Added ERROR FLAG in GetActiveID call for serial devices.

Version 7.2.5 Extend,a pcProx Plus feature, was accepting more than 128 bytes.

Version 7.2.6 supports pcProx Plus readers with extended functionality.

Version 7.2.8 supports the changes for backward compatibility of pcProx Plus with extended functionality.

Version 7.2.9 supports the HID API implementation instead of libusb on the Linux.

Version 7.2.11 supports bug fixes on Linux with HID API implementation.

Version 7.2.12 fixed slow connectivity issue of serial devices on Linux Platform.

Version 7.2.13 supports bug fixes of connection.

Version 7.2.14 supports volume control feature of pcProx Plus(version 2 with 4 card configuration).

Version 7.2.15 supports for EV1 Key programming & Bluetooth® Low Energy technology.

Version 7.2.16 supports sending current configuration to the reader before writing EV1 data.

Version 7.2.17 supports removal of name mangling for java escalation.

Version 7.2.18 includes default vid pid list.

## What's new in Version 7 of this library

---

Version 7.2.19 includes change of hidapi libs in linux.

Version 7.2.20 includes issue fix : performing clean installation by installer.

Version 7.2.21 includes GetFWFilename API, new sample applications, issue fix: incorrect definition of the setBTLEConfiguration API in document.

Version 7.2.22 includes reducing sleep time in WriteCfg api for reader based on loon platforms.

Version 7.2.23 includes fixing bug in SDK Examples.

Version 7.2.24 includes new EULA document and fixed issues related to serial loon reader.

Version 7.2.25 includes improved API reference and ReadMe documents.

Version 7.2.26 includes some more improvements in API reference document.

Version 7.2.27 includes adding BLE example in SDK and providing support for Mac OS X in pcProxAPI SDK.

Version 7.2.28Beta includes changes for proxAdv reader.

Version 7.2.28 includes SDK release on RedHat by fixing limitations of SDK on Linux, adding some FAQs in ReadMe related to compatibility issues in Linux and fixing bugs in SDK examples.

Version 7.2.29 includes fixing crash for proxAdv readers and incorporating feedback for BLE Examples.

Version 7.3.0 includes updating documents for new release.

Version 7.3.2 includes GetFullFWVersion API and making SDK thread-safe.

Version 7.3.3 includes Increase the baud rate of serial readers from 9600 to 115200.

Version 7.3.4 includes bug fix for connecting low baud rate serial loon reader.

Version 7.3.5Beta includes issue fix: buffer overflow on calling GetDevName.

Version 7.3.5 includes addition of multiple baud rates functionality for serial loon readers with firmware v16.8.

Version 7.3.6 includes static library for Windows.

Version 7.3.7Beta includes logging feature in SDK.

Version 7.3.7Beta2 changes the location of Log file to C:/pcProxLogs/ProxApiSdk.

Version 7.3.7Beta3 adds Firmware: CPU0 and CPU1 Application and pcProx Library version to logs. Also fixes bug for connecting pcSwipe readers

## What's new in Version 7 of this library

---

Version 7.3.7 adds support for proxAdv readers on Linux platform.

Version 7.3.8 includes bug fixes for: i) Serial readers not getting connected on Linux platform, and ii) USB readers getting connected by passing connection type as SERIAL.

Version 7.3.9 includes bug fixes for: i) Memory Leak in C# example. ii) Fixing hash keys in configuration other than 1st, and iii) NTWare Canon reader listing all cards.

It also includes the updated card list in pcProxAPI.h

Version 7.4.0 includes isASCIIPresent Flag implementation to check support for ASCII-Conversion-Type in the reader and improvement in document for getActiveId32 API.

Version 7.4.1 includes bug fixes for enabling Extended/Hashing functionality for RDR-80x81AKE, AKB, E-P, or B-P with F/W LNC160800SPX700.

Version 7.4.2 includes: i) change in getFWFilename API documentation. ii) added steps to set device rules for rf IDEAS readers in Linux in ReadMe.pdf file. iii) improved API document for LED structure. iv) added FAQ for ethernet reader. v) added signature validation process for libusb0.dll. vi) implemented JM3 reader requirement. vii) Bug fixes for AKE readers and find next IP functionality. viii) updated card list in pcProxAPI.h file.

Version 7.4.3Beta includes: i) changed product id of JM3 reader. ii) added vendor and product id for KPIT. iii) added get Electronic Serial Number functionality for loon USB reader.

Version 7.4.3 includes: i) Added /GUARD:CF compiler option in SDK. ii) Updated SDK and C++, C#, VB-net, and BLE example project files to support Visual Studio 2019 toolset.

Version 7.4.4Beta includes: i) Adding new readers' information in vid-pid list.

Version 7.4.4 includes: i) Exposing some of the APIs of the SDK for Linux and Mac Platforms.

Version 7.4.5 includes: i) Implementing CRC Process for HWG File ii) Adding two new APIs WriteDevCfgToCRCFile and ReadDevCfgFromCRCFile iii) Adding the getESN option in the readme of the Examples.

Version 7.4.6 includes: i) Bug Fix of HWG File CRC Implementation.

Version 7.4.7 includes: i) Changing the name CRC to iEndOfHwgFile in the SDK for Secure hwg files. ii) Updating new window's server IP. iii) Bug Fix related to secure hwg+ files when Value is appended at the end of secure key. iv) Bug Fix related to loading the secure hwg+ files with security key starting with '0'.

Version 7.4.8Beta includes: i) Adding two APIs for some special reader configuration.

Version 7.4.8 includes: i) Adding Latest cardtypes ii) APIs exposure restricted for some APIs.

## What's new in Version 7 of this library

---

Version 7.4.9 includes: i) Bug fix for API documentation.

Version 7.5.0Beta includes: i) Adding exposure for two APIs ReadSecureData and WriteSecureData.

Version 7.5.0Beta2 includes: i) Bug fix for SetComSrchRange for AK0 Readers  
ii) Adding latest cardtypes.

Version 7.5.0Beta3 includes: i) Bug fix for Com port for serial readers on Linux.

Version 7.5.0 includes: i) Removing Beta3 from version.

Version 7.5.1 includes: i) Bug fix to remove Beta3 from API documentation.

Version 7.5.2Beta includes: i) Bug fix for Card data buffer. ii) Adding API for vSDK beta information.

Version 7.6.0 includes i) Exposing WriteDevCfgToSecureFile and ReadDevCfgFromSecureFile for Linux and Mac. ii) Adding the error code for Tampered Files. iii) Exposing WriteSecureData and ReadSecureData for Linux and Mac. iv) Making the Secure hwg file compatible for all three platforms. v) Adding the API for parsing the .ini File. vi) Bug Fixes.

vii) Implementing the Progress Bar at the SDK. viii) Adding API for generating and checking Secure Firmware Files.

Version 7.6.1 includes: i) Bug fix for loadini and documentation. ii) Enabling debugging feature for debug mode

Version 7.6.2 includes: i) Bug fix for documentation and loon reader taking long time to load hwg file. ii) Adding support of Linux-Arm64 and Mac-M series. iii) Adding GetActiveID API in loop for readercomm example. iv) Updating Copyright information. v) Adding support of BLE version information. vi) Adding Updated EULA document.

Version 7.6.3 includes: i) Updating API doc.

Version 7.6.4 includes: i) Bug fix for getting segmentation error for SecureHwg API on Ubuntu22. ii) Bug fix for IOCTL error on console. iii) Changing the folder location for EULA pdf.

## What's new in Version 7 of this library

---

Version 7.7.0 includes: i) Adding new API's to get additional information of reader: GetCopyrightInfo, GetBuildDate, GetCPUInfo, GetPlatformType, GetSerialNumber. ii) Adding new API for BLE readers: WriteBLEDataToFile, WriteBLEKeyToFile, WriteBLEDataToReader WriteBLEKeyToReader. iii) Adding support of GetFWFilename and GetFullFWVersion for Wren Serial Reader. iv) Adding new API SendGetFWVersion to populate firmware version for serial reader.

Version 7.7.1 includes: i) Bug fix for loading BLE HWG files on Wren Mobile reader. ii) Adding Updated EULA document. iii) Updating Readme.

Version 7.7.2 includes: i) Bug fix for writeDevCfgToFile\_char() and readDevCfgFmFile\_char() APIs not working. ii) Bug fix for ReadDevCfgFromSecureFile API not working on Mac Monterey. iii) Adding readDevCfgFmFile\_char and writeDevCfgToFile\_char in C# example. iv) Bug fix for BLE HWG API's WriteBLEKeyToReader not working as expected. v) Updated api documentation for GetLEDCtrl and SetLEDCtrl.

---

All pcProx functions are downward compatible with previous libraries.

New feature include:

Support for up to 127 devices USB and serial in the active list.

Supports three products, pcProx, pcSwipe, pcProx Sonar.

Available as a Linux shared library.

Ability to map com1 through com8 under Linux `"/dev/<name>"`

GetActiveID can be used to read RF ID, Sonar Status and Magnetic Card fields.

pcProx Plus support functions for multiple configurations.

pcProx Plus supports multiple cards types.

Direct Ethernet TCP/IP support for RDR-xxxx-AKE and

Ethernet Industrial Protocol(EIP) readers.

Throttle API calls to GetActiveID when time between calls is under 250 msec.

Reset to Factory Defaults optimized for pcProx Plus.

Reset to Defaults added for pcProx Plus.



## Languages

---

Traditional legacy function starts with a capital letter. These have been used in C, C++, and Visual Basic. Due to C# type safe code, many of the legacy functions that set/get structure members have new functions that begin with a lower case letter and do not take a structure pointer. These are safe for C# managed code.

All function names are not C++ name mangled. They are case sensitive and appear to the linker as is. This allows the functions to be easily called from all versions of Visual Basic, Visual C/C++, GNU C/C++, Tcl/Tk and Autolt. By using Simplified Wrapper Interface Generator (SWIG) a library can easily be created for language such as Java, Lua, Perl, Python, Ruby and others.

## Frequently Asked Questions (FAQ1)

---

### Q1. What is the simplest way to read the Card ID?

There are two main functions that talk with the reader to get the card ID; GetActiveID() (receiving 8 bytes, 64 bits) and GetActiveID32() (receiving 32 bytes up to 255 bits, for FIPS201, Chuid and Legic). The card data must be valid at the time of the function call, card on the reader or the card was present 1 second ago. New firmware may have the function GetQueuedID().

Step #1 Connect to the reader with the USBConnect() function.

Step #1B Optionally select any reader if more than one is on the USB bus.

Step #2 Call one of the GetActiveID functions on a 250 msec timer.

Step #3 When your application exits, call USBDisconnect().

### Q2. How often must I configure the reader?

Once! You don't even have to write code to configure the reader. You may use the pcProx Config Application to set the reader and then save the settings to a .hwg+ file. The Application you write could then load the .hwg+ file into the library by calling ReadDevCfgFmFile() and then write the flash memory by calling WriteCfg().

### Q3. How do I get the Facility Access Code (FAC) out of the GetActiveID bytes?

Card formats vary, but let's take the HID 26 bits card example when the facility access code is one byte and the Card ID is two bytes. Many cards also have a parity bit in the beginning and at the end of the data. First let's look at the data with parity stripped off.

GetActiveID returns an array of bytes. Card manufacturers are different. By the common HID 26 bit format is one leading and one trailing parity bit, 8 bit facility access code and 16 bit ID code. If the facility access code is 183 and the card ID is 32877, You would expect a hex B7 for the FAC of 183, and hex 806D for the ID. Our byte array returned from GetActiveID might look like: 00.00.00.00.00.B7.80.6D. So our FAC would be  $\text{FAC} = \text{Byte}[5]$ . Our ID would be  $\text{ID} = (\text{Byte}[6] * 256) + \text{Byte}[7]$ . Now, with no parity stripped you have two extra bits to throw away. Notice the 03.

---

Our byte array would look like this with parity bits: 00.00.00.00.03.6F.00.DA

FAC = ( ((a[0] & 1) \* 256) + a[1])/2 // Logical and with 1 to discard parity bit.

ID = ( ((a[5] & 1) \* 65536) + (a[6] \* 256) + a[7])/2. Divide by two or shift right to get rid of the least significant parity bit.

**Q4. Can I configure an ethernet reader with the help of SDK ?**

Yes, it can be configured using the SDK but it is advisable to use an ethernet reader with the ACP commands only as the SDK does not support connection to multiple ethernet readers simultaneously.

## Frequently Asked Questions (FAQ2)

---

### Q1. Can I speed up the USBConnect() ?

USBConnect tries to connect to several products and also tries several interfaces; USB, Serial and Ethernet. You can set the product types with the SetConnectProduct function. Some examples below:

```
SetConnectProduct(PRODUCT_PCPROX | PRODUCT_PCSWIPE);  
SetConnectProduct(PRODUCT_PCPROX);  
long ldid=0;  
USBConnect(&ldid);
```

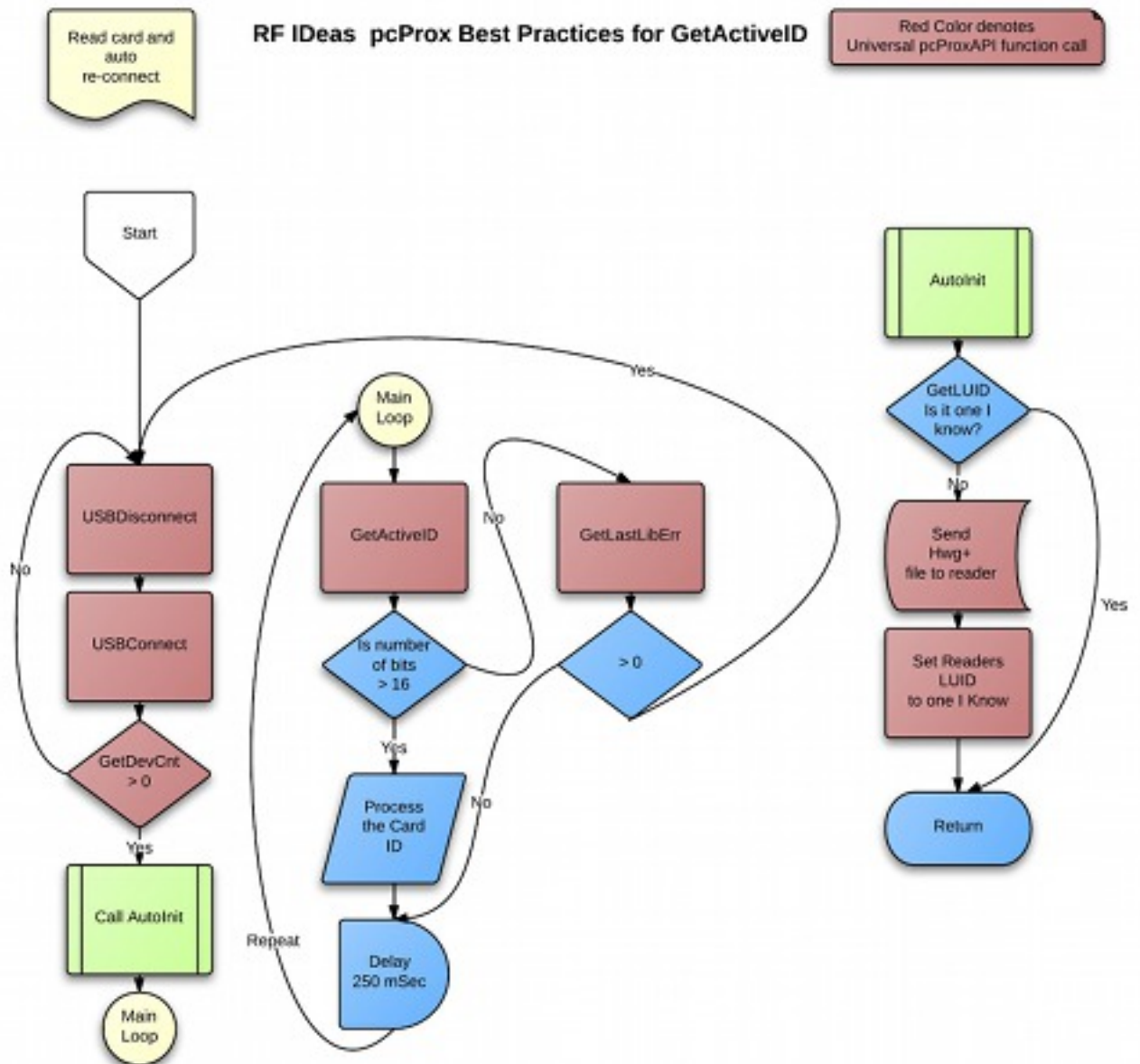
Likewise, you can set the device types you want to search for during USBConnect(). Serial and Ethernet will take longer than USB. See SetDevTypeSrch example below:

```
SetDevTypeSrch(PRXDEVTYP_USB);  
SetDevTypeSrch(PRXDEVTYP_SER);  
SetDevTypeSrch(PRXDEVTYP_TCP);  
SetDevTypeSrch(PRXDEVTYP_ALL);  
long ldid=0;  
USBConnect(&ldid);
```

### Q2. Do all functions in the library generate traffic to the devices?

No, many of the functions return cached values and are documented as such with yellow [Cached] text on the page title. Functions that generate traffic to the device have a green [Traffic] text on the title.

# Best Practices



See the online tools section on our website.

<http://www.rfideas.com/support/software-conversion-tools/read-card-id-tool>

# Data Types

---

The following data types are assumed to be these sizes.

BYTE	1 byte	unsigned char	0 .. 255
WORD	2 bytes	unsigned short	0 .. 65,535
BSHORT	2 bytes	signed short or short	-32,768 .. 0 .. 32,767
SHORT	2 bytes	unsigned short	0 .. 65,535
DWORD	4 bytes	double word	0 .. 4,294,967,295

Treat FALSE as zero and TRUE as NON FALSE.

TRUE may vary from language to language, and /or compiler to compiler.

rf IDEAS defined TRUE to be 0x0001, but other systems may use 0xFFFF or 0xFFFFFFFF or -1. It's safe not to compare with TRUE but to compare with not FALSE.

```
if( FunctionCall() != FALSE )  
{  
    // Success  
}
```

See typedefs in pcProxAPI.h for details.

# sBprRlyCtrl

---

pcProx and OEM W2-USB

---

```
typedef struct sBprRlyCtrl // Used for pcProx and OEM W2-USB
{
    short iPad0;
    short iBeeperState; // 0 == Off, 1 == On
    short iRelayState; // 0 == Off, 1 == On
    short iPad3;
    short iPad4;
    short iPad5;
    short iPad6;
    short bVolatile; // 0 == commit to EE, 1 == Don't store to EE
} tsBprRlyCtrl;
```

# sCfgFlags

pcProx

```
typedef struct sCfgFlags // Used for pcProx
{
    short bFixLenDsp;        // Send as fixed length with leading zeros as needed
    short bFrcBitCntEx;     // Force Rx'd bit count to be exact to be valid
    short bStripFac;        // Strip the FAC from the ID (not discarded)
    short bSndFac;          // Send the FAC (if stripped from data)
    short bUseDelFac2Id;    // Put a delimiter between FAC and ID on send
    short bNoUseELChar;     // Don't use a EndLine char on send (default to ENTER)
    short bSndOnRx;         // Send valid ID as soon as it is received
                           // (iIDLockOutTm timer not used)
    short bHaltKBSnd;       // Don't Send keys to USB (Get ID mechanism)
} tsCfgFlags;
```



## sCfgFlags2

pcProx

```
typedef struct sCfgFlags2 // Used for pcProx
{
    short bUseLeadChrs; // Use leading chars in ID KB send
    short bDspHex;      // Display ID as ASCII Hex [not ASCII decimal]
    short bWiegInvData; // Wiegand data on pins is inverted
    short bUseInvDataF; // Use the bWiegInvData flag over hardware setting
    short bRevWiegBits; // Reverse the Wiegand Rx bits
    short bBeepID;      // Beep when ID received
    short bRevBytes;     // Reverse byte order (CSN reader)
    short isASCIIPresent; //check if ASCII conversion type is present
    //within the reader
} tsCfgFlags2;
```

## sCfgFlags3

pcProx

```
typedef struct sCfgFlags3 // Used for pcProx
{
    short bUseNumKP;        // Euro KB flag
    short bSndSFON;         // Split format ON = 1, old combined scheme = 0
    short bSndSFFC;         // 0 = FAC Decimal, 1 = FAC Hex
    short bSndSFID;         // 0 = ID Decimal, 1 = ID Hex
    short bPrxProEm;        // Use ProxPro emulation
    short bUse64Bit;        // 0 = 32-bit, 1 = 64-bit Display Math
    short bNotBootDev;      // USB Enum: 0=BootDevice, 1=NOTBootDevice
    short bLowerCaseHex;    // Alpha hex output displayed as lower case
} tsCfgFlags3;
```

## sIDBitCnts

pcProx

```
typedef struct sIDBitCnts // Used for pcProx
{
    short iLeadParityBitCnt; // Wiegand Leading Parity bit count to be stripped
    short iTrailParityBitCnt; // Wiegand Trailing Parity bit count to be stripped
    short iIDBitCnt; // If bStripFac, determines bit count of ID and FAC
    short iTotalBitCnt; // If bFrcBitCntEx, card read (including parity)
                        // must match this

    short iPad4;
    short iPad5;
    short iPad6;
    short iPad7;
} tsIDBitCnts;
```

# sIDDispParms

pcProx

```
typedef struct sIDDispParms // Used for pcProx
{
    short iFACIDDelim; // If bStripFac & bSndFac & bUseDelFac2Id, this char
                        // sent between FAC & ID
    short iELDelim;    // If NOT bNoUseELChar, this char sent at end of ID
    short iIDDispLen;  // If bFixLenDsp, ID padded with zeros to this length
    short iFACDispLen; // If bFixLenDsp, FAC padded with zeros to this length
    short iExOutputFormat ; //If iExOutputFormat, Reader will output in Extended
                        //mode for Plus Extended readers

    short iPad5;
    short iPad6;
    short iPad7;
} tsIDDispParms;
```

# sIDDispParms2

pcProx

```
typedef struct sIDDispParms2 // Used for pcProx
{
    short iLeadChrCnt; // Tf bUseLeadChrs, contains the lead char count (<=3)
    short iLeadChr0;   // These lead characters are filled in (up to 3)
    short iLeadChr1;   // Leading character
    short iLeadChr2;   // Leading character
    short iCrdGnChr0;  // If non-zero, sent when ID goes Invalid
    short iCrdGnChr1;  // If this and Chr0 non-zero, sent when ID goes Invalid
    short iPad6;
    short iPad7;
} tsIDDispParms2;
```

## sIDDispParms3

pcProx

```
typedef struct sIDDispParms3 // Used for pcProx
{
    short iTrailChrCnt; // This contains the trail char count (<=3)
    short iTrailChr0;   // These trail characters are filled in (up to 3)
    short iTrailChr1;   // NOTE: LeadChrCnt + TrailChrCnt <= 3
    short iTrailChr2;   //          LeadChrs have priority
    short iPad4;
    short iPad5;
    short iPad6;
    short iPad7;
} tsIDDispParms3;
```

# sIdleParms

pcProx Sonar

```
typedef struct sIdleParms // Used for pcProx Sonar
{
    short Reserved1; //
    short PressRate; // Periodic Press Rate (1 Sec units) - Default Zero Off
    short Key1Mods; // Key 1 modifier bit flags
    short Key1Code; // Key 1 code
    // If bit bDetDeadMan of Flags is SET, DMTODelta determines
    // the spatial window within which the target must remain
    // for DMTOTm seconds before being declared "dead".
    // If DMTODelta is set to 1, the window is 1/3.472 ~= 0.3 inch.
    // If DMTODelta is set to 3, the window is 3/3.472 ~= 0.9 inch.
    //
    // sIdleParms.Flags bit defines...
    #define bDetDeadMan 0 // DeadMan T/O if set, use DMTODelta & DMTOTm above
    #define bDetMinOOR 1 // Target too close to sensor (< MinDist out of range)
    //
    // sIdleParms.Flags.bDetMinOOR can be used in conjunction with
    // sWalkAwayParms.MinFltRepRate to repeatedly send the sWalkAwayParms keys when
    // the target is too close as specified by sSonarParms.MinDist for the time
    // specified by sSonarParms.ORDBTm.
    // This can be used for very close range detection of tape or "dixie cups" over
    // the sensor or other objects placed in front of the sensor in an attempt to
    // disable the send of the WalkAway key set. The repetitive send feature will
    // disallow a manual logon after initial send of the WalkAway keys which results
    // in an unprotected computer.
    //
    short Flags; // see above...
    short DMTODelta; // DeadMan T/O Delta, 3.472 * Distance (in) = DMTODelta
    short DMTOTm; // DeadMan Time Sec. untill Declared Dead while Deltas
    // never exceeded
    short SNBlankTm; // Sonar drive blanking time (43us units)
} tsIdleParms;
```

# sLEDCtrl

pcProx

```
typedef struct sLEDCtrl // Used for pcProx
{
    // Set bAppCtrlsLED to 1 - The user of the SDK can control the LED according to
    // the other settings in the sLEDCtrl struct.
    // Set bAppCtrlsLED to 0 - The reader will control the LED (ignoring the other
    // settings in the sLEDCtrl struct).
    short bAppCtrlsLED;
    // Note:- For AMBER both iRedLEDState and iGrnLEDState will be 1.
    short iRedLEDState; // 0 == Off, 1 == On
    short iGrnLEDState; // 0 == Off, 1 == On
    short iPad3;
    short iPad4;
    short iPad5;
    short iPad6;
    // Set bVolatile to 1 - this will write configuration to RAM and the LED color
    // will change. Setting will not be persistent (it will be volatile). Use this
    // mode if you do not want the LED color to persist. This mode also prevents
    // excessive writes to the reader's flash memory.
    // Set bVolatile to 0 - in order to change the LED color, you must follow this
    // command with a WriteCfg(), which will change the LED's color and also
    // persist the change to flash. Use this mode if you want to keep the setting
    // saved in the reader's flash memory, and avoid calling it frequently since
    // it can cause flash memory wear.
    short bVolatile;
} tsLEDCtrl;
```



# sRangeInfo

pcProx Sonar

```
typedef struct sRangeInfo // Used for pcProx Sonar
{
    short wCurrRange;      // Current Object Range in Inches, Zero for > max or < min
    short bInRange;        // Object In-Range is non-zero
    short bInRangePend;    // Object In-Range Pending (still out-of-range)
    short bOutOfRangePend; // Object Out-of-Range Pending (still in-range)
    short pad4;
    short pad5;
    short pad6;
    short pad7;
} tsRangeInfo;
```

# sSonarParms

pcProx Sonar

```
typedef struct sSonarParms // Used for pcProx Sonar
{
    short LEDFlags; // LED usage flags [0]
    short PingRate; // Ping rate (msec) [332] (200 .. 1020)
    short Reserved1; // 40KHz cycle cnt (READ ONLY)
    short MinDist; // Minimum Distance acceptable (inches) [14] (14 .. 59)
    short MaxDist; // Maximum Distance acceptable (inches) [36] (15 .. 60)
    short ORDBTm; // Debounce out of range time (seconds) [0]
    short IRDBTm; // Debounce in range time (seconds) [0]
    short StartDly; // Cold start delay in seconds before joining USB [0]
} tsSonarParms;
```

# sTimeParms

pcProx

```
typedef struct sTimeParms // Used for pcProx
{
    short iBitStrmTO; // Wiegand read T/O after this msec time (4ms)
    short iIDHoldTO; // Card ID valid for this msec time (48ms)
    short iIDLockOutTm; // Squelch repetitive reader reports (usually > 1000)
                        // in msec (48msec gran.)
    short iUSBKeyPrsTm; // Sets USB inter-key 'Press' time in 4ms units
    short iUSBKeyRlsTm; // Sets USB inter-key 'Release' time in 4ms units
    short ExFeatures01; // Extended Features (big parity Azery ext precision)
    short iPad6; // Spare Unused
    short iTPCfgFlg3; // Bit mapped tp Flags3 -- Use Flags3 structure instead
} tsTimeParms;
```

# sWalkAwayParms

pcProx Sonar

```
typedef struct sWalkAwayParms // Used for pcProx Sonar
{
    short KeyCount;           // Defined keys to follow (max 6 modifiers/keys) [0]
    short InterKeyDelay;      // Time (ms) between keys [512] (range 64..16320)
    short Key1Mods;           // Key Modifiers...[0]
    short Key2Mods;
    short Key3Mods;
    short Key4Mods;
    short Key5Mods;
    short Key6Mods;
    short Flags0;
    short MinFltRepRate;      // Near fault send repeat rate in seconds
    short Key1Code;           // Key Codes...[0]
    short Key2Code;
    short Key3Code;
    short Key4Code;
    short Key5Code;
    short Key6Code;
} tsWalkAwayParms;
```

# sWalkUpParms

pcProx Sonar

```
typedef struct sWalkUpParms // Used for pcProx Sonar
{
    short KeyCount;           // Defined keys to follow (max 6 modifiers/keys) [0]
    short InterKeyDelay;      // Time (ms) between keys [512] (range 64..16320)
    short Key1Mods;           // Key Modifiers...[0]
    short Key2Mods;
    short Key3Mods;
    short Key4Mods;
    short Key5Mods;
    short Key6Mods;
    short Flags0;
    short Reserved2;
    short Key1Code;           // Key Codes...[0]
    short Key2Code;
    short Key3Code;
    short Key4Code;
    short Key5Code;
    short Key6Code;
} tsWalkUpParms;
```

## API Function List

---

The API functions are listed in alphabetical order. The function names are case sensitive, and lower case names are listed after the upper case names. Under each function header line is the supported operating systems shown on the left in red. The right side shows the supported products in blue.

Note: Do not compare version numbers from the pcProx firmware to enable or disable functions in your application as firmware version can change. The firmware version is for reference only.

You can speed up connection to USB readers by not searching for other devices, such as pcProx Sonar and pcSwipe reader, and also by skipping Serial and Ethernet devices.

```
SetDevTypeSrch(PRXDEVTYP_USB);  
SetConnectProduct(PRODUCT_PCPROX);  
rc = usbConnect();
```

# BeepNow

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)

BSHRT BeepNow(BYTE count, BSHRT longBeep)

## DESCRIPTION

Call this function with the number of desired short or long beeps.  
This API is thread-safe.

## PARAMETERS

count 1..N beeps, longBeep = TRUE, short beep = FALSE.  
Range 1..5 short beeps and 1..2 long beeps.

## RETURNS

TRUE = Success / FALSE = Failure / user defined Unsupported.

## SEE ALSO

[pcSwipeGetBeeper\(\)](#)  
[pcSwipeSetBeeper\(\)](#)  
[SetUnsupportedProductErrorCode\(\)](#)

# CheckTamperedFile

[Cached]

Windows Linux Mac

pcProx

```
bool CheckTamperedFile(char *szFileName, int moduleType)
```

## DESCRIPTION

This API verifies CRC of files.

By checking the CRC, it verifies whether the file is tampered or not. If a file is verified, it means that the data in the file has not been altered and could be loaded into a reader.

## PARAMETERS

szfileName, which may include the complete path to the file name.

Module type tells about file type. It can be a .hwg+ or .ini file.

moduleType for hwg+ = 0, ini = 2.

## RETURNS

TRUE - File verified.

FALSE - File tampered.



# ChkAddArrival

[Cached]

Windows

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

```
short ChkAddArrival(char *pBuf)
```

## DESCRIPTION

Check device add arrival

## PARAMETERS

szname atleast of 511 (MAXDEVNAMESZ) characters

## RETURNS

Check to see if device is in the list held by the library  
TRUE = Success / FALSE = Failure

## SEE ALSO

[ChkDelRemoval\(\)](#)  
[chkAddArrival\\_char\(\)](#)  
[chkDelRemoval\\_char\(\)](#)

# ChkDelRemoval

[Cached]

Windows

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** ChkDelRemoval(char \*szName)

## DESCRIPTION

Check device delete removal

## PARAMETERS

szname atleast of 511 (MAXDEVNAMESZ) characters

## RETURNS

Check to see if device is NOT in the list held by the library

TRUE = Success / FALSE = Failure

The user should re-scan the USB device list using USBConnect()  
if this returns TRUE.

## SEE ALSO

[chkDelRemoval\\_char\(\)](#)

[chkAddArrival\\_char\(\)](#)

[ChkAddArrival\(\)](#)

# ComConnect

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)**BSHRT** [ComConnect\(long\\* pIID\)](#)

## DESCRIPTION

Connect to reader attached to Serial COM Port or virtual com port. The com port range set by [SetComSrchRange\(lo,hi\)](#) will be searched during the [ComConnect\(\)](#) [USBConnect\(\)](#) call. The com ports that do not have an attached reader will take longer as they must time-out during the search. Defaults are 1..8 at powerup. Returns the firmware version (DeviceID) after searching ports for a device.

Note: Do not compare version numbers from the pcProx firmware to enable or disable functions in your application as firmware version can change. The firmware version is for reference only.

## PARAMETERS

long pointer to receive DeviceID

## RETURNS

TRUE = Success / FALSE = Failure

## EXAMPLE

```
SetComSrchRange(1,8);  
long DeviceID = 0;  
ComConnect(&DeviceID);  
--OR--  
SetComSrchRange(1,8);  
ComConnect(NULL);
```

## SEE ALSO

[SetComSrchRange\(\)](#)  
[SetComLinux\(\)](#)  
[SendGetFWVersion\(\)](#)

# ComConnectPort

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)**BSHRT** ComConnectPort(WORD iPort, long\* pIIDID)

## DESCRIPTION

Connect to one serial device on COM port iPort (1..256)

Note: Do not compare version numbers from the pcProx firmware to enable or disable functions in your application as firmware version can change. The firmware version is for reference only.

## PARAMETERS

Port number representing COM1 thru COM256 as 1 thru 256 inclusive  
Pointer to get DeviceID or firmware version.

## RETURNS

TRUE = Success connected to device / FALSE = Failure no connection.

## SEE ALSO

[USBConnect\(\)](#)  
[ComConnect\(\)](#)  
[ComDisconnect\(\)](#)  
[SetComLinux\(\)](#)  
[SendGetFWVersion\(\)](#)

# ComDisconnect

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)**BSHRT** `ComDisconnect(void)`

## DESCRIPTION

Disconnect the active device from the serial device and close the COM port.

## PARAMETERS

None

## RETURNS

TRUE = Success / FALSE = Failure not connected, failed to close port, or active device is not serial.

## SEE ALSO

[ComConnect\(\)](#)

# DumpRawFeatureReports

[Cached]

Windows

pcProx

**BSHRT** DumpRawFeatureReports(char \*fname)

## DESCRIPTION

This function writes a USB Raw Feature Report Dump to an ASCII text file. The user provides the name of the text file and then call this function to handle the actually file writing of the 8 byte transmitted and received feature reports.

## PARAMETERS

FileName : \*fname

## RETURNS

TRUE if file is written successfully -- FALSE unsuccessful

## SEE ALSO

# FindXport

[\[Traffic\]](#)

Windows Linux Mac

[pcProx](#)

```
short FindXport(short ip0,  
                short ip1,  
                short ip2,  
                short ip3begin,  
                short ip3end)
```

## DESCRIPTION

Ethernet readers use an Xport interface to convert serial data to Ethernet. This function sends a UDP message to a range of IP address and waits 500 msec for a reply packet identifying it as an Xport. Once the Xport is found it can be connected and you can access the serial reader.

## PARAMETERS

IP address range ip0,ip1,ip2 first 3 digits of IP, ip3begin .. ip3end inclusive range of IP addresses to check for an Xport device.

## RETURNS

1..254 last digit of IP address of first found device. If none found return 0.

## EXAMPLE

```
ip3 = FindXport(192,168,0, 1,254);  
if(ip3)  
{  
    SetIpPort(192.168,0,ip3,10001);  
    SetDevTypeSrch(PRXDEVTYP_TCP);  
    USBConnect(&Luid);  
}
```

# GetAZERTYShiftLock

[Cached]

Windows Linux Mac

pcProx Plus

BSHRT GetAZERTYShiftLock(void)

## DESCRIPTION

Get the state of the Shift Lock. Some keyboards such as French keyboards have a shift lock key in place of the US caps lock key. This affects how the top row of numbers and punctuation are used.

## PARAMETERS

None

## RETURNS

TRUE on / FALSE off

## SEE ALSO

[SetAZERTYShiftLock\(\)](#)



# GetActConfig

[Cached]

Windows Linux Mac

pcProx Plus

short `GetActConfig(void)`

## DESCRIPTION

Get the active configuration ( 0..N ) of the pcProx Plus. For devices that only have one configuration this will return 0. All devices have one configuration, so zero is always valid.

## PARAMETERS

None

## RETURNS

0, 1, 2..N

## SEE ALSO

[GetMaxConfig\(\)](#)

[SetActConfig\(\)](#)

# GetActDev

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)short `GetActDev(void)`

## DESCRIPTION

Return the current active device as an index into the list, 0..(GetDevCnt()-1).

Note: The USB order of devices found can be random. Use the LUID to uniquely tag your devices. Then enumerate through the active device list for the LUID you need.

## PARAMETERS

None

## RETURNS

Active device 0..126

# GetActiveID

[\[Traffic\]](#)

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**short** GetActiveID(BYTE \*pBuf, short wBufMaxSz)

## DESCRIPTION

Reads the active data from the card "presently on the reader". The Least significant bit is Buf[0] bit 0. It is recommended not to call this faster than 250msec, or about twice the data hold time of the active card. Call sooner than 250msec will not be sent to the reader but will return cached data. This API is thread-safe.

## PARAMETERS

pBuf pointer (reference) to character buffer to receive the card ID currently within reader range. wBufMaxSz specifies the size of the byte buffer, and is useful to limit the buffer transfer to less than 8 characters. Values above 8 will transfer 8 bytes max.

## RETURNS

Returns the number of bits (0..63) received from the reader representing the card ID or sometime the card CSN.

The return count represents how many bits in the buffer are valid ID bits. It does NOT include the parity bits that may have been stripped from the ID through the use of the leading and/or trailing parity bit counts.

A return value of zero means there is either no card within RF field or an error was encountered. GetLastError() may be used to differentiate between the two possibilities.

This function was originally designed for pcProx. On pcSwipe it returns the track and field as defined pcSwipeSetActiveID(). New designs should use pcSwipeGetTrackData().

For pcProx Sonar devices the buffer has 3 entries (4 bytes).

[0] = 1 in range, 0 out of range,

[1] Distance

[2] luid low,

[3] luid high.

## SEE ALSO

[GetActiveID32\(\)](#)[GetQueuedID\(\)](#)[pcSwipeGetTrackData\(\)](#)

# GetActiveID32

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

```
short GetActiveID32(BYTE *pBuf, short wBufMaxSz)
```

## DESCRIPTION

This is the 32 byte version of GetActiveID() for FIP-201 and CHUID readers to get the card ID with up to 255 bits of data. Cards with larger ID's than 64 bits will be truncated with GetActiveID(), so this function is recommended instead.

This API is thread-safe.

See GetActiveID() documentation as all else applies.

## PARAMETERS

32 byte buffer, number of bytes max.

## RETURNS

Number of bits read 0..255

## SEE ALSO

[GetActiveID\(\)](#)[GetQueuedID\(\)](#)

# GetBTLEConfiguration

[Cached]

Windows Linux Mac

pcProx

short [GetBTLEConfiguration\(\)](#)

## DESCRIPTION

Gets the BTLE configuration of the reader

## RETURNS

0 for BTLE off/125KHz-13.56MHz radios off  
1 for BTLE off/125KHz-13.56MHz radios on  
2 for BTLE on/125KHz-13.56MHz radios off  
3 for BTLE on/125KHz-13.56MHz radios on

## SEE ALSO

[SetBTLEConfiguration\(\)](#)

# GetBeeperVolume

[Cached]

Windows Linux Mac

pcProx

`short GetBeeperVolume(void)`

## DESCRIPTION

Gets the volume level for the pcProx Plus(version 2).

## RETURNS

Volume level for the pcProx Plus(version 2) Reader.  
For non pcProx Plus(version 2) returns -1.

## SEE ALSO

[SetBeeperVolume\(\)](#)

# GetBprRlyCtrl

[Cached]

Windows Linux Mac

[pcProx](#) (OEM W2-USB)**BSHRT** `GetBprRlyCtrl(tsBprRlyCtrl *psBRCtrl)`

## DESCRIPTION

Get beeper (and relay on OEM W2-USB readers) controls.

## PARAMETERS

psBRCtrl pointer to structure to read beeper/relay control information.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[SetLEDCtrl\(\)](#)

# GetBuildDate

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#)

```
const char* GetBuildDate(void)
```

## DESCRIPTION

Read the device Build Date Information.

## PARAMETERS

None

## RETURNS

Returns the Build Date of reader on success  
In case of failure it returns an empty string.



# GetCPUInfo

[Cached]

Windows Linux Mac

pcProx

```
const char* GetCPUInfo(void)
```

## DESCRIPTION

Read the device CPU Information.

## PARAMETERS

None

## RETURNS

Returns the CPU information on success  
In case of failure it returns an empty string.

# GetCardPriority

[Cached]

Windows Linux Mac

pcProx Plus

```
long GetCardPriority(void)
```

## DESCRIPTION

Get the Card Type Priority for the active configuration. A long integer is used to detect errors. The Priority is 0 or 1, and a -1 indicates an error.

## PARAMETERS

None

## RETURNS

0 = Low priority, 1 = high priority, -1 error.

## SEE ALSO

[SetCardTypePriority\(\)](#)  
[GetActConfig\(\)](#)  
[SetActConfig\(\)](#)

# GetCardType

[Cached]

Windows Linux Mac

pcProx Plus

```
long GetCardType(void)
```

## DESCRIPTION

Get the Card Type 0x0000..0xFFFE for the given configuration.  
See #define CARDTYPE\_<name> in pcProxAPI.h

## PARAMETERS

None

## RETURNS

0..0xFFFE for valid card types. -1 for error, disconnected,  
or non pcProx Plus reader.

## SEE ALSO

[SetCardTypePriority\(\)](#)  
[GetActConfig\(\)](#)  
[SetActConfig\(\)](#)

# GetCopyrightInfo

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#)

```
const char* GetCopyrightInfo(void)
```

## DESCRIPTION

Read the device Copyright Information.

## PARAMETERS

None

## RETURNS

Returns the Copyright information on success  
In case of failure it returns an empty string.

# GetDID

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)short `GetDID(void)`

## DESCRIPTION

Return the firmware version Major,Minor,Build as bits 15..8

Bits 7..4 and bits 3..0 respectively. Example: 0x0F12 = 15.1.2

Note: Do not compare version numbers from the pcProx firmware to enable or disable functions in your application as firmware version can change. The firmware version is for reference only and should not be used to make decision on what features are available.

## PARAMETERS

None

## RETURNS

Returns the Firmware version of the current active device.

## SEE ALSO

[GetFirmwareVersion\(\)](#)

# GetDevByLUID

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

short `GetDevByLUID(short LUID, short index)`

## DESCRIPTION

Find the device index used by SetActDev and GetActDev for the given LUID  
Start at the active device index parameter. If index < 0 then -1 is returned.  
If none of the connected devices has this LUID value return -1.  
This function does not change the active device, you must call SetActDev  
to set the active device by the returned index.

## PARAMETERS

Logical Unit ID 0 .. 65535

## RETURNS

0..N else -1 for not found

## EXAMPLE

```
int FirstIndex = GetDevByLUID(1234, 0);  
int SecondIndex = GetDevByLUID(1234, FirstIndex+1);
```

## SEE ALSO

[SetLUID\(\)](#)  
[GetLUID\(\)](#)

# GetDevCnt

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)`short GetDevCnt(void)`

## DESCRIPTION

Returns the number of pcProx devices found on this machine from USBConnect().

## PARAMETERS

None

## RETURNS

Number of device on bus: 0..127

## SEE ALSO

[GetActDev\(\)](#)

# GetDevName

[Cached]

Windows

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** `GetDevName(char *szName)`

## DESCRIPTION

Get Device Name, this can be a long USB device name the users buffer should be at least 511 (MAXDEVNAMESZ) characters. Can be used under Linux but makes less sense.

## PARAMETERS

szName A string containing COMx, or USB PID VID of the communications port the reader is found on.

## RETURNS

The active RS-232 or Virtual COM port number (1, 2, ...). 999 will be returned for USB devices.

## SEE ALSO

[rf\\_GetDevName\(\)](#)  
[getDevName\\_char\(\)](#)



# GetDevType

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**short** [GetDevType\(void\)](#)

## DESCRIPTION

Same as [rf\\_GetDevType](#). Avoids VB name space problem with VB's [GetDevType\(\)](#)  
VB user can call [rf\\_GetDevType\(\)](#);

## RETURNS

Return PRXDEVTYP\_<name> for USB, Serial or TCP/IP on the Active Device.

## SEE ALSO

[rf\\_GetDevType\(\)](#)

# GetExtendedPrecisionMath

[Cached]

Windows Linux Mac

pcProx Plus

BSHRT `GetExtendedPrecisionMath(void)`

## DESCRIPTION

Return the state of the extended precision math flag. This affects the FAC value when it is longer than 32 bits.

## PARAMETERS

None

## RETURNS

TRUE on / FALSE off

## SEE ALSO

[SetExtendedPrecisionMath\(\)](#)

# GetFWFilename

[Cached]

Windows Linux Mac

[pcProx](#)

```
char * GetFWFilename()
```

## DESCRIPTION

Read the device firmware filename.

## PARAMETERS

None

## RETURNS

Returns the Firmware Filename on success  
In case of failure it returns an empty string.

# GetFirmwareVersion

[Cached]

Windows Linux Mac

[pcProx](#) [Plus](#) [pcSwipe](#)**DWORD** [GetFirmwareVersion](#)(short hardware, short module)

## DESCRIPTION

Some devices have multiple firmware versions to report. This function returns the unsigned long (DWORD) firmware version of the hardware and module pair. Not all modules are available for all hardware. These return 0 for unavailable.

The version 0x01023456 can be expressed as "1.2.34.56".

Note: Do not compare version numbers from the pcProx firmware to enable or disable functions in your application as firmware version can change. The firmware version is for reference only. For pcSwipe only Hardware "Main Cpu" App and bootloader are available.

## PARAMETERS

Hardware 0 = Main CPU, 1 = Aux Cpu, 2-FF unused

Module 0 = Application, 1 = Bootloader, 2 = Modem

## RETURNS

0x000001 .. 0xFFFFFFFF for valid versions, 0 or code set by [SetUnsupportedProductErrorCode\(\)](#) indicates unavailable.

## SEE ALSO

[GetDID\(\)](#)[SetUnsupportedProductErrorCode\(\)](#)

# GetFlags

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** `GetFlags(tsCfgFlags *psCfgFlgs)`

## DESCRIPTION

Get parameters from device. Note pcSwipe and pcProx Sonar only use the bHaltKBSnd flag within this structure.

## PARAMETERS

psCfgFlgs pointer to structure to receive Configuration Flags information.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[SetFlags\(\)](#)

# GetFlags2

[Cached]

Windows Linux Mac

[pcProx](#)**BSHRT** `GetFlags2(tsCfgFlags2 *psCfgFlgs)`

## DESCRIPTION

Get the values from device into the flags2 data structure.

## PARAMETERS

psCfgFlgs2 pointer to structure to receive Configuration Flags2 information.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[SetFlags2\(\)](#)

# GetFlags3

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)**BSHRT** `GetFlags3(tsCfgFlags3 *psCfgFlgs)`

## DESCRIPTION

Get the values from device into the flags3 data structure.  
pcSwipe only uses the bUseNumKP flag. pcSwipe only uses the UseNumKP parameter of this structure.

## PARAMETERS

psCfgFlgs3 pointer to structure to receive Configuration Flags3 information.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[SetFlags3\(\)](#)

# GetFullFWVersion

[Cached]

Windows Linux Mac

pcProx Plus

```
char * GetFullFWVersion()
```

## DESCRIPTION

Reads the device's full Firmware version including the String version number.

## PARAMETERS

None

## RETURNS

Returns the Firmware version with the string version number for loon readers having version number 16.7 or greater.  
In case of failure it returns Null.



# GetHIDGuid

[Cached]

Windows

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** GetHIDGuid(GUID \*pGuid)

## DESCRIPTION

Return the type of interface which is always a (Human Interface Device) HID.  
The return value is a constant.

## PARAMETERS

GUID pointer (from Windows USB SDK)

## RETURNS

Always TRUE = Success

# GetIDBitCnts

[Cached]

Windows Linux Mac

pcProx

**BSHRT** GetIDBitCnts(tslIDBitCnts \*pslIDBitCnts)

## DESCRIPTION

Get information regarding the ID bit and counts.

## PARAMETERS

pslIDBitCnts pointer to structure to receive Bit Count information.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[SetIDBitCnts\(\)](#)

# GetIDDispParms

[Cached]

Windows Linux Mac

[pcProx](#)**BSHRT** `GetIDDispParms(tsIDDispParms *psIDDispParms)`

## DESCRIPTION

Get display parameters.

## PARAMETERS

psIDDispParms pointer to structure to receive ID Display information.

## RETURNS

Returns non-zero on success, zero otherwise.

## SEE ALSO

[SetIDDispParms\(\)](#)

# GetIDDispParms2

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#)

BSHRT GetIDDispParms2(tsIDDispParms2 \*psIDDispParms)

## DESCRIPTION

Get display parameters 2.

## PARAMETERS

psIDDispParms2 pointer to structure to receive ID Display2 information.

## RETURNS

Returns non-zero on success, zero otherwise.

## SEE ALSO

[SetIDDispParms2\(\)](#)

# GetIDDispParms3

[Cached]

Windows Linux Mac

[pcProx](#)**BSHRT** `GetIDDispParms3(tsIDDispParms3 *psIDDispParms)`

## DESCRIPTION

Get display parameters3.

## PARAMETERS

psIDDispParms3 pointer to structure to receive ID Display3 information.

## RETURNS

Returns non-zero on success, zero otherwise.

## SEE ALSO

[SetIDDispParms3\(\)](#)

# GetIdleParms

[Cached]

Windows Linux Mac

pcProx Sonar

**BOOL** GetIdleParms( tssIdleParms\* pssIdleParms )

## DESCRIPTION

Get the pcProx Sonar Idle Parameters as defined in the structure below.  
These key strokes are typed when the object enters the sensor range.

```
typedef struct sIdleParms
{
    short Reserved1; // Leave as-is
    short PressRate; // Periodic Press Rate (1 second units) - [0 disabled]
    short Key1Mods; // Key 1 modifier bit flags
    short Key1Code; // Key 1 code
    short Flags; // see below.
    short DMTODelta; // DeadMan T0Delta, 3.472 * Distance (in) = DMTODelta
    short DMTOTm; // DeadMan Time (seconds) - Time till Declared Dead
                // while Deltas never exceeded
    short SNBlankTm; // pcProx Sonar drive blanking time (43usec units)
} tssIdleParms;
```

## PARAMETERS

tssIdleParms structure

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[SetIdleParms\(\)](#)

# GetLEDCtrl

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** [GetLEDCtrl](#)(tsLEDCtrl \*psLEDCtrl)

## DESCRIPTION

Get LED configuration of the reader.

## PARAMETERS

A pointer to the psLEDCtrl struct which contains the LED configuration.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[SetLEDCtrl\(\)](#)

# GetLUID

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)`short GetLUID(void)`

## DESCRIPTION

Get Logical Unit ID from last ReadCfg() of device.

## RETURNS

LUID a user defined 16-bit ID (0-65536) to be associated with the current selected device.

## SEE ALSO

[SetLUID\(\)](#)  
[SetActDev\(\)](#)



# GetLastLibErr

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

```
long GetLastLibErr(void)
```

## DESCRIPTION

Returns the last library error code (see Library Error Codes) for the active device. The last error code is valid until another library call is made. This does not reset the last library error code. When a function returns FALSE and has failed, it is good practice to call this function to check the error code. See pcProxAPI.h for "Error Bits".

## PARAMETERS

None

## RETURNS

long error code value bits

## SEE ALSO

[WriteCfg\(\)](#)  
[SetActDev\(\)](#)

# GetLibVersion

[Cached]

Windows Linux Mac

[Library](#)

```
BSHRT GetLibVersion(short* piVerMaj,  
                    short* piVerMin,  
                    short* piVerDev)
```

## DESCRIPTION

Get the version of the library code. This does not communicate with any device. It returns constants from the DLL or Linux shared library. The intended interpretation of the version is VerMaj.VerMin.VerDev.

## PARAMETERS

piVerMaj pointer or NULL (reference) to integer to receive the major version  
piVerMin pointer or NULL (reference) to integer to receive the minor version  
piVerDev pointer or NULL (reference) to integer to receive the build version

## RETURNS

Null pointers will not be used to return data.  
Returns Always TRUE / Success

## EXAMPLE

```
short major,minor,build;  
if(GetLibraryVersion(&major,&minor,&build) != FALSE)  
{  
    printf("%d.%d.%d",major,minor,build);  
}  
if(GetLibraryVersion(&major,&minor,NULL) != FALSE)  
{  
    printf("%d.%d.X",major,minor);  
}  
if(GetLibraryVersion(NULL,NULL,&build) != FALSE)  
{  
    printf("Build # ",build);  
}
```

# GetMaxConfig

[Cached]

Windows Linux Mac

pcProx Plus

WORD `GetMaxConfig(void)`

## DESCRIPTION

Get number of pcProx Plus Reader Configurations. For pcProx Plus dual frequency readers this will be 1 or more and 0 for Legacy non pcProx Plus readers.

## PARAMETERS

none

## RETURNS

0, 1..N -- 0 for pcProx, 1 to N where N is max N-1 Config for pcProx Plus.

## SEE ALSO

[GetActConfig\(\)](#)  
[SetActConfig\(\)](#)

# GetMyIpAddress

[Cached]

Windows Linux Mac

pcProx

**ULONG** GetMyIpAddress(void)

## DESCRIPTION

Return the IP address of the local machine as an unsigned long. If dotted quad is 192.168.1.2 then the long in hex will be 0x0201A8C0. To convert to bytes as ip1,ip2,ip3,ip4 shift the byte in multiples of eight as follows:

```
iplong = GetMyIpAddress();
```

```
BYTE ip0 = iplong & 255; // 192
```

```
BYTE ip1 = (iplong >> 8) & 255; // 168
```

```
BYTE ip2 = (iplong >> 16) & 255; // 1
```

```
BYTE ip3 = (iplong >> 24) & 255; // 2
```

## PARAMETERS

None

## RETURNS

Unsigned long. Least significant byte is the first byte of the dotted quad notation.

## SEE ALSO

[SetDevTypeSrch\(PRXDEVTYP\\_TCP\)](#)

[SetIpPort\(\)](#)

[FindXport\(\)](#)

# GetObjRangeInfo

[Cached]

Windows Linux Mac

pcProx Sonar

```
BOOL GetObjRangeInfo( tsRangeInfo* psRangeInfo )
```

## DESCRIPTION

Talks with pcProx Sonar and retrieves info data block.

Get the pcProx Sonar info structure as defined below. Determine if an object is within range and if the time in or out of range has been satisfied.

```
typedef struct sRangeInfo
```

```
{
```

```
    short wCurrRange;    // Range in Inches - 0 if >max or <min
```

```
    short bInRange;      // Object In-Range is non-zero
```

```
    short bInRangePend;  // Object In-Range Pending (still out-of-range)
```

```
    short bOutRangePend; // Object Out-of-Range Pending (still in-range)
```

```
    short pad4;
```

```
    short pad5;
```

```
    short pad6;
```

```
    short pad7;
```

```
} tsRangeInfo;
```

# GetPlatformType

[Cached]

Windows Linux Mac

pcProx

```
const char* GetPlatformType(void)
```

## DESCRIPTION

Read the device Platform Type Example: Wren or Loon.

## PARAMETERS

None

## RETURNS

Returns the Platform Type on success  
In case of failure it returns an empty string.

# GetProduct

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**ULONG** [GetProduct](#)(void)

## DESCRIPTION

Get the active device's product type. The product type is one of the PRODUCT\_<name>'s. See [pcProxAPI.h](#)

## EXAMPLE

```
if(GetProduct() == PRODUCT_PCPROX)
{
    printf("The active device is a pcProx\n");
}
```

## SEE ALSO

[SetConnectProduct\(\)](#)  
[USBConnect\(\)](#)  
[usbConnect\(\)](#)

# GetQueuedID

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**short** [GetQueuedID\(short clearUID, short clearHold\)](#)

## DESCRIPTION

Read the Queued ID data from the reader.

This returns the last card read by the reader, age and overrun counter.

The age is from 0 - 0x0FFFF in 48msec units. The max time is 52 minutes.

The overrun is the number of cards read before the UID was transferred to the PC.

It is recommended not to call this faster than 250msec, or about

twice the data hold time of the active card.

After this function returns TRUE (Success) you may call [GetQueuedID\\_index\(\)](#)

## PARAMETERS

If clearUID is set then the card, and overrun counters will be cleared for the next read. Older firmware sets the age to zero, newer firmware sets the age to 0xFFFF.

If clearHold is set then the reader is ready to read another card immediately.

## RETURNS

Returns TRUE success, or FALSE failed (perhaps function is not available in the firmware). IF TRUE use [GetQueuedID\\_index\(\)](#) to get the rest of the data from the library.

## SEE ALSO

[GetQueuedID\\_index\(short index\)](#)[GetQueuedID\\_index\(\)](#)[GetActiveID\(\)](#)[GetActiveID32\(\)](#)



# GetQueuedID\_index

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)

```
long GetQueuedID_index(short index)
```

## DESCRIPTION

Return specific part of the data read by GetQueuedID()  
GetQueuedID() actually gets data from the reader,  
this functions just marshalls the data back to the caller.  
GetQueuedID\_index()

## PARAMETERS

index 0..35

## RETURNS

index 0..31 = Bytes 0..31 of the UserID  
index 32 = short Number of bits read 0-256.  
index 33 = short Age (16 bits) 0 - 65,535 48msec ticks or 0 to 52 minutes.  
index 34 = short Overrun counter number of cards before UID transferred to PC.  
index 35 = short lockout timer 0-256 (0=ready to read).  
Overrun clips at 255, and age at 0x0FFFF.

## SEE ALSO

[GetQueuedID\(\)](#)

# GetSN

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

DWORD GetSN(void)

## DESCRIPTION

Get serial number from last ReadCfg() on device. Future expansion. Most devices will have an unprogrammed serial number of 0xFFFF.

## PARAMETERS

None

## RETURNS

TRUE = Success / FALSE = Failure

# GetSepFldData

[Cached]

Windows Linux Mac

pcProx

**BSHRT** GetSepFldData(BYTE \*pBuf, short wBufMaxSz)

## DESCRIPTION

Get the FIPS 201 credentials and delimiters for all user defined fields. This is for pcProx or OEM Wiegand converter board that can read FIPS 201 CHUID's. Usually from 75 to 245 bits make up some or all of the FIPS 201 fields. OEM Readers such as the HID-G3 75 bit reader is support by the OEM converter board.

## PARAMETERS

pBuf pointer to memory array to return the FIPS 201 Chuid data.  
wBufMaxSz is always bytes.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[SetSepFldData\(\)](#)

# GetSerialNumber

[Cached]

Windows Linux Mac

[pcProx](#)

```
const char* GetSerialNumber(void)
```

## DESCRIPTION

Read the Electronic Serial Number of Wren Serial Reader.

## PARAMETERS

None

## RETURNS

Returns the Electronic Serial Number on success  
In case of failure it returns an empty string.

## SEE ALSO

[getESN\(\)](#)

# GetSonarParms

[Cached]

Windows Linux Mac

[pcProx](#) [Sonar](#)**BOOL** GetSonarParms( tsSonarParms\* psSonarParms )

## DESCRIPTION

Get pcProx Sonar Parameters. This function returns a structure of values as defined below.

```
typedef struct tsSonarParms
{
    short LEDFlags; // LED usage flags [0]
    short PingRate; // Ping rate (msec) [332] (min 200, max 1020)
    short Reserved1; // 40KHz cycle cnt (READ ONLY)
    short MinDist; // Minimum Distance acceptable (inches) [14] (min 14, max 59)
    short MaxDist; // Maximum Distance acceptable (inches) [36] (min 15, max 60)
    short ORDBTm; // debounce out of range time (seconds) [0]
    short IRDBTm; // debounce in range time (seconds) [0]
    short StartDly; // cold start delay in seconds before joining USB [0]
} tsSonarParms;
```

## PARAMETERS

tsSonarParms structure

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[SetSonarParms\(\)](#)

# GetTimeParms

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** [GetTimeParms](#)(tsTimeParms \*psTimeParms)

## DESCRIPTION

Get device timing information. Times are in milliseconds, keyboard times have a granularity of 4ms, card times have 48ms.

## PARAMETERS

psTimeParms pointer to structure containing timing information from device.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[SetTimeParms\(\)](#)

# GetUnsupportedProductErrorCode

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** [GetUnsupportedProductErrorCode\(void\)](#)

## DESCRIPTION

Read back the unsupported error code set by [SetUnsupportedProductErrorCode\(\)](#).

## PARAMETERS

None

## RETURNS

BSHRT value set by [SetUnsupportedProductErrorCode\(\)](#).

## SEE ALSO

[SetUnsupportedProductErrorCode\(\)](#)

# GetVidPidVendorName

[Cached]

Windows Linux Mac

pcProx

```
char * GetVidPidVendorName(void)
```

## DESCRIPTION

Return the VendorName string for the active device. This comes from the optional pcProxVidPid.txt file.

## PARAMETERS

None

## RETURNS

Returns the pointer to a static buffer. For non pcProx devices a pointer to an empty string is returned.

## SEE ALSO

[getVidPidVendorName\\_char\(\)](#)



# GetWalkAwayParms

[Cached]

Windows Linux Mac

pcProx Sonar

**BOOL** GetWalkAwayParms( tsWalkAwayParms\* psWalkAwayParms )

## DESCRIPTION

Get pcProx Sonar Walk Away Parameters. This function returns a structure of values as defined below. These key strokes are typed when the object leaves the sensor range.

```
typedef struct tsWalkAwayParms
```

```
{  
    short KeyCount;      // Defined keys to follow (max 6 modifiers/keys) [0]  
    short InterKeyDelay; // Time between key presses (msec) [512] (min 64, max 16320)  
    short Key1Mods;      // Key Modifiers..[0]  
    short Key2Mods;      // Bit 0..7 Left Ctrl,Shift,Alt,GUI, Right Ctrl,Shift,Alt,GUI  
    short Key3Mods;  
    short Key4Mods;  
    short Key5Mods;  
    short Key6Mods;  
    short Flags0;  
    short MinFltRepRate; // Near fault send repeat rate in seconds  
    short Key1Code;      // USB Key Scan Codes 'a' = 4, b=5, 'c'=6 etc,  
    short Key2Code;      //  
    short Key3Code;  
    short Key4Code;  
    short Key5Code;  
    short Key6Code;  
} tsSonarParms;
```

## PARAMETERS

tsWalkAwayParms structure

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[SetSonarParms\(\)](#)

# GetWalkUpParms

[Cached]

Windows Linux Mac

pcProx Sonar

**BOOL** GetWalkUpParms( tsWalkUpParms\* psWalkUpParms )

## DESCRIPTION

Get the pcProx Sonar Walk Away Parameters as defined in the structure below. These key strokes are typed when the object enters the sensor range.

```
typedef struct tsWalkAwayParms
```

```
{
    short KeyCount;      // Defined keys to follow (max 6 modifiers/keys) [0]
    short InterKeyDelay; // Time (ms) between key presses [512] (range 64..16320)
    short Key1Mods;      // Key Modifiers..[0]
    short Key2Mods;      // Bit 0..7 Left Ctrl,Shift,Alt,GUI, Right Ctrl,Shift,Alt,GUI
    short Key3Mods;
    short Key4Mods;
    short Key5Mods;
    short Key6Mods;
    short Flags0;
    short MinFltRepRate; // Near fault send repeat rate in seconds
    short Key1Code;      // USB Key Scan Codes 'a' = 4, b=5, 'c'=6 etc,
    short Key2Code;      //
    short Key3Code;
    short Key4Code;
    short Key5Code;
    short Key6Code;
} tsSonarParms;
```

## PARAMETERS

tsWalkAwayParms structure

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[SetWalkUpParms\(\)](#)

# HaltKBSends

[Cached]

Windows Linux Mac

pcProx Sonar

**BOOL** HaltKBSends( **BOOL** bHalt )

## DESCRIPTION

Halt Keyboard Sending of data. Also known as API mode, or silent mode. When set data will not be keystroked out, all pcProx Sonar information must be read using the API.

## PARAMETERS

boolean TRUE = quite, FALSE = normal keystroking mode.

## RETURNS

Return: TRUE / else FALSE = Failed.

# IsBTLEPresent

[Cached]

Windows Linux Mac

pcProx

short IsBTLEPresent()

## DESCRIPTION

To check whether the BTLE is present

## RETURNS

True or False

## SEE ALSO

[GetBTLEConfiguration\(\)](#)

# IsCardTypeInList

[Cached]

Windows Linux Mac

pcProx Plus

BSHRT IsCardTypeInList(WORD findCT)

## DESCRIPTION

Is CardType in list? If the given card type is in the pcProx Plus Firmware List Return TRUE, else return FALSE. A 0 card type (CARDTYPE\_OFF) is always in the list and returns TRUE. If the pcProx Plus does not have a list then all cards types are assumed to be in the list.

See #define CARDTYPE\_<name> in pcProxAPI.h

## PARAMETERS

Card type 0..0xFFFF

## RETURNS

TRUE card type in list -- FALSE not in list

## SEE ALSO

[GetCardType\(\)](#)

# LoadIniFile

[Cached]

Windows Linux Mac

pcProx

short LoadIniFile(char\* fileName)

## DESCRIPTION

Load the Configuration in the reader which is mentioned in the file.

Note: This API only accepts the Secure Blob format of the ini file (generated from the Smartcard Manager utility).

## PARAMETERS

fileName that may include complete path to the file name.

## RETURNS

TRUE / Success - secure key is valid and ASCII file was read.

FALSE / Failed - secure key is invalid, file could not be read.

# Ping

[\[Traffic\]](#)

Windows Linux Mac

[pcProx](#) [pcSwipe](#)

long Ping(void)

## DESCRIPTION

Send a packet to the active device and return the ping time in milliseconds if the device is online. If offline return 0. Serial devices will be slower than USB devices. The ping function is resource hungry therefore pinging the device at the same rate as reading a card ID is not recommended.

## PARAMETERS

None

## RETURNS

1..65,535 milliseconds, 0 = no reply or unsupported.

## SEE ALSO

[USBConnect\(\)](#)

# QuickReadSerialPort

[\[Traffic\]](#)

Windows Linux Mac

[pcProx](#) [pcSwipe](#)**DWORD** QuickReadSerialPort(char \*buf, DWORD count)

## DESCRIPTION

Check for serial data on connected COM port, and receive available bytes. This function sets the timeout values to 75msec so it is quicker than the ReadSerialPort() function.

## PARAMETERS

pointer to char buffer, and count of max desired characters to receive.

## RETURNS

Actual number of characters received.

## SEE ALSO

[readSerialPort\\_char\(\)](#)  
[ReadSerialPort\(\)](#)  
[WriteSerialPort\(\)](#)



# ReadBaudRate

[\[Traffic\]](#)

Windows Linux

[pcProx Plus](#)**BSHRT** `ReadBaudRate()`

## DESCRIPTION

Reads the active baud rate written in the device.

## PARAMETERS

None

## RETURNS

baud rate on success else -1 on failure or if reader is unsupported.

Baud codes meaning :

00 = 9600;

01 = 19200;

02 = 38400;

03 = 57600;

04 = 115200;

## SEE ALSO

[WriteBaudRate\(BSHRT baudrateCode\)](#)

[SetBaudRate\(int baudrate\)](#)

# ReadCfg

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

BSHRT ReadCfg(void)

## DESCRIPTION

A call to this function pulls the device configuration information into the library memory space to be manipulated by the Get\*() and Set\*() functions. After altering the data the user must call WriteCfg() to write the changes back to device so they can take effect. This API is thread-safe.

## PARAMETERS

None

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[WriteCfg\(\)](#)

# ReadDevCfgFmFile

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** `ReadDevCfgFmFile(char *szFileName)`

## DESCRIPTION

Open and read an ASCII file and load the setting into the library memory. The device can be flashed by calling `WriteCfg()`. This function reads files created with the `WriteProxCfgToFile()` function.

Note: This is the ASCII file format. It is not compatible with the simpler Visual Basic `pcProxConfig` format and will return an error if those are read.

## PARAMETERS

`szFileName` file name that may include complete path to the file name.

## RETURNS

TRUE / Success ASCII file was read.

FALSE file may be invalid locked or the path may be invalid.

## SEE ALSO

[WriteDevCfgToFile\(\)](#)[readDevCfgFmFile\\_char\(\)](#)[writeDevCfgToFile\\_char\(\)](#)

# ReadDevCfgFromSecureFile

[Cached]

Windows Linux Mac

pcProx

**BSHRT** ReadDevCfgFromSecureFile(char\* szFileName)

## DESCRIPTION

It will check whether the secure Key which is present in the file is valid or not. Then it will Open and read an ASCII file and load the setting into the library memory. The device can be flashed by calling WriteCfg(). This function reads files created with the WriteDevCfgToSecureFile() function.

Note: This is the ASCII file format. It is not compatible with the simpler Visual Basic pcProxConfig format and will return an error if those are read.

## PARAMETERS

szFileName file name that may include complete path to the file name.

## RETURNS

TRUE / Success secure key is valid and ASCII file was read.

FALSE - secure key is invalid, file could not be read.

# ReadDevTypeFromFile

[Cached]

Windows Linux Mac

[pcSwipe](#) [pcProx](#) [Sonar](#)**WORD** `ReadDevTypeFromFile(char *szFileName)`

## DESCRIPTION

Read Device Type from .mag file. Reading a file written by a USB into a serial device and vice versa will cause incorrect delimiters due to ASCII and scan code differences.

## PARAMETERS

szFileName file name that may include complete path to the file name.

## RETURNS

-1 Error can not read file or determine type.

## SEE ALSO

[ReadDevCfgFmFile\(\)](#)  
[WriteDevCfgToFile\(\)](#)

# ReadSecureData

[Cached]

Windows Linux Mac

pcProx

```
bool ReadSecureData(BYTE *headerBuffer, BYTE *dataBuffer)
```

## DESCRIPTION

Read the configuration as requested in header(blob details: Type, config, length, BSV).

## PARAMETERS

headerBuffer pointer (reference) to character buffer to send the blob details to reader\*

dataBuffer pointer (reference) to character buffer to get the data present on reader\*

## RETURNS

Returns true on success and false on failure.

# ReadSerialPort

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)

DWORD `ReadSerialPort(char *buf, DWORD count)`

## DESCRIPTION

Check for serial data on connected COM port and receive available bytes. This uses a two second timeout.

## PARAMETERS

pointer to char buffer, and count of max desired characters to receive.

## RETURNS

Actual number of characters received.

## SEE ALSO

[QuickReadSerialPort\(\)](#)  
[quickReadSerialPort\\_char\(\)](#)  
[readSerialPort\\_char\(\)](#)

# ResetFactoryDflts

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** `ResetFactoryDflts(void)`

## DESCRIPTION

This sets the devices configuration to the Factory Default values. It is like a WriteCfg() call. Before returning to the caller, this function calls ReadCfg() to reload the configuration information (which may have changed) into the library memory. The GetLastError() return will either the ReadCfg() error code or success. This takes about 1200msec on USB and longer for serial devices.

## PARAMETERS

None

## RETURNS

TRUE = Success / FALSE = Failure



# ResetUserDflts

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [Sonar](#) [pcProx Plus](#)**BOOL** `ResetUserDflts( void )`

## DESCRIPTION

Set the device flash memory of the pcProx Sonar configuration to the last saved user defaults. Execution time ~ 2 seconds.  
For pcProx Plus this recalls the stored settings.

## PARAMETERS

None

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[SaveUserDflts\(\)](#)

# SaveUserDflts

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [Sonar](#) [pcProx Plus](#)**BOOL** `SaveUserDflts( void )`

## DESCRIPTION

Save the current device's working configuration as the user defaults.  
user defaults. Execution time ~ 2 seconds.  
For pcProx Plus this Writes the stored settings.

## PARAMETERS

None

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[ResetUserDflts\(\)](#)

# SendGetFWVersion

[Cached]

Windows Linux

pcProx

```
void SendGetFWVersion(void)
```

## DESCRIPTION

Connect a Serial COM Port.

This API is used to populate the FWVersion[][] for Serial readers.

Sample code

```
SetComSrchRange(1,12)
```

```
long DeviceID = 0;
```

```
ComConnect(&DeviceID);
```

```
SendGetFWVersion();
```

```
GetFirmwareVersion(0,1);
```

--OR--

```
SetComSrchRange(1,12)
```

```
long DeviceID = 0;
```

```
short iPort = 8;
```

```
ComConnectPort(iPort,&DeviceID);
```

```
SendGetFWVersion();
```

```
GetFirmwareVersion(0,1);
```

## PARAMETERS

None

## RETURNS

None

## SEE ALSO

[ComConnect\(\)](#)

[ComConnectPort\(\)](#)

# SetAZERTYShiftLock

[Cached]

Windows Linux Mac

pcProx Plus

BSHRT SetAZERTYShiftLock(short on)

## DESCRIPTION

Set the state of the Shift Lock. Some keyboards such as French keyboards have a shift lock key in place of the US caps lock key. This affects how the top row of numbers and punctuation are used.

## PARAMETERS

boolean value true = on

## RETURNS

TRUE success / FALSE failed  
GetAZERTYShiftLock()

# SetActConfig

[Cached]

Windows Linux Mac

pcProx Plus

BSHRT SetActConfig(BYTE n)

## DESCRIPTION

Set the active configuration ( 0..N ) of the pcProx Plus device.

## PARAMETERS

Configuration number 0..N (GetMaxConfig() - 1)

## RETURNS

TRUE success / FALSE Fail  
Non pcProx Plus readers will return 0.

## SEE ALSO

[GetMaxConfig\(\)](#)  
[GetActConfig\(\)](#)

# SetActDev

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

BSHRT SetActDev(short iNdx)

## DESCRIPTION

Set the active device. Up to MAXDEVSOPEN devices may be present. The selected device is the active device that ReadCfg() and WriteCfg() operate on.

## PARAMETERS

Set the active device where iNdx selects a new device for processing. This does not require a WriteCfg() to be performed as it is only selecting the device to which we are communicating with via the library.

## RETURNS

TRUE = Success / FALSE = Failure index out of range

## SEE ALSO

[GetActDev\(\)](#)

# SetBTLEConfiguration

[Cached]

Windows Linux Mac

pcProx

BSHRT SetBTLEConfiguration(BSHRT level)

## DESCRIPTION

Sets the BTLE configuration of the reader.

## PARAMETERS Level 0,1,2,3,4

0:BTLE off/125KHz-13.56MHz radios off.  
1:BTLE off/125KHz-13.56MHz radios on.  
2:BTLE on/125KHz-13.56MHz radios off.  
3:BTLE on/125KHz-13.56MHz radios on.  
4:BTLE off/125KHz-13.56MHz radios toggle.  
others, BTLE off/125KHz-13.56MHz radios off.

## RETURNS

True or False

## SEE ALSO

[GetBTLEConfiguration\(\)](#)

```
#define BTLEOFF_RADIOOFF 0  
#define BTLEOFF_RADIOON 1  
#define BTLEON_RADIOOFF 2  
#define BTLEON_RADIOON 3  
#define BTLE_RADIOTOGGLE 4
```

# SetBaudRate

[Cached]

Windows Linux

pcProx Plus

```
void SetBaudRate(int baudrate)
```

## DESCRIPTION

Sets the baud rate inside the SDK upon which the reader will communicate.  
By default baud rate will be set to 9600.  
Not to be confused with WriteBaudRate.

## PARAMETERS

baudrate. Such as: 9600, 19200, 38400, 57600, 115200.

## RETURNS

None

## SEE ALSO

[WriteBaudRate\(BSHRT baudrateCode\)](#)  
[ReadBaudRate\(\)](#)



# SetBeeperVolume

[Cached]

Windows Linux Mac

pcProx

**BSHRT SetBeeperVolume(BSHRT volumeLevel)**

## DESCRIPTION

Sets the volume level for the pcProx Plus (version 2) reader.

## PARAMETERS

volumeLevel 0..3,  
0-off, 1-low, 2-med, 3-High

## RETURNS

TRUE = Success / FALSE = Failure / user defined Unsupported.

## SEE ALSO

[GetBeeperVolume\(\)](#)

# SetBprRlyCtrl

[Cached]

Windows Linux Mac

pcProx (OEM W2-USB)

**BSHRT** SetBprRlyCtrl(tsBprRlyCtrl \*psBRCtrl)

## DESCRIPTION

Set beeper (and relay on OEM W2-USB readers) controls.

## PARAMETERS

psBRCtrl pointer to structure containing new beeper/relay control bit.  
Use WriteCfg() to update the hardware outputs.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[GetBprRlyCtrl\(\)](#)  
[GetLEDCtrl\(\)](#)

# SetCardTypePriority

[Cached]

Windows Linux Mac

pcProx Plus

**BSHRT SetCardTypePriority(WORD cardType, BSHRT priority)**

## DESCRIPTION

Set the Card Type 0x0000..0xFFFF for the active configuration.

See #define CARDTYPE\_<name> in pcProxAPI.h

Card types not understood by the device firmware are ignored and will return as (0x0000) Off. The priority bit if non zero sets this configuration to have priority over other configurations that are set to zero. The priority allows dual frequency cards or multiple cards to read in a predictable manner. Only one configuration should have the priority bit set, otherwise unpredictable results may occur on multiple card reads.

## PARAMETERS

Sixteen bit card type, and priority bit.

## RETURNS

TRUE success, FALSE failed, disconnected, or non pcProx Plus reader.

## SEE ALSO

[GetCardType\(\)](#)[SetActConfig\(\)](#)[GetActConfig\(\)](#)

# SetComLinux

[Cached]

Linux Mac

[pcProx](#) [pcSwipe](#)**BSHRT** SetComLinux(WORD index, const char \*devName)

## DESCRIPTION

Map Linux /dev/tty name to serial port number 1..MAXLINUXCOMPORT  
This allows traditional functions calls to work on Linux devices  
using ports as COMx, where x is 1 to MAXLINUXCOMPORT .

## PARAMETERS

COM port 1..MAXLINUXCOMPORT, hold MAXLINUXDEVPATH characters.

## RETURNS

TRUE / Success else FALSE index out of range, or non Linux OS.

## SEE ALSO

[SetComSrchRange\(\)](#)  
[ComConnect\(\)](#)

# SetComSrchRange

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)**BSHRT** SetComSrchRange(WORD iMin, WORD iMax)

## DESCRIPTION

Set serial port COM search range used use by ComConnect.

## PARAMETERS

COM port low, com port high.

Valid range is inclusive from 1 to 256 for Windows and 1 to MAXLINUXCOMPORT for Linux.

## RETURNS

TRUE Success / FALSE values out of range.

## SEE ALSO

[ComConnect\(\)](#)[VirtualComSearchRange\(\)](#)[SetComLinux\(\)](#)

# SetConnectProduct

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

BSHRT SetConnectProduct(ULONG bits)

## DESCRIPTION

Set the Connect functions to only allow these type of products to be connected. by default all products are scanned for. The product type is one of the PRODUCT\_<name>'s see pcProxAPI.h for bits.

## SEE ALSO

[GetProduct\(\)](#)  
[USBConnect\(\)](#)  
[usbConnect\(\)](#)

## EXAMPLE

```
SetConnectProduct(PRODUCT_PCPROX | PRODUCT_PCSWIPE);  
usbConnect();  
SetConnectProduct(PRODUCT_ALL);  
usbConnect();
```

# SetDevTypeSrch

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)**BSHRT** SetDevTypeSrch(short iSrchType)

## DESCRIPTION

Set the type of device interface USB and/or Serial to search for when connecting using USBConnect().

Searching for serial devices is much slower than USB only devices.

This only sets the desired type. It does not search.

## PARAMETERS

iSrchType 0=USB only, 1=Serial (RS-232) Only, -1=Both USB and Serial

## RETURNS

TRUE = Success / FALSE = Failure (invalid parameter)

Default is both devices USB and Serial.

## EXAMPLE

```
SetDevTypeSrch(PRXDEVTYP_ALL);  
SetDevTypeSrch(PRXDEVTYP_USB);  
SetDevTypeSrch(PRXDEVTYP_SER);  
SetDevTypeSrch(PRXDEVTYP_TCP);
```

## SEE ALSO

[ComConnect\(\)](#)

[USBConnect\(\)](#)

Constants in pcProxAPI.h header file

```
#define PRXDEVTYP_ALL -1
```

```
#define PRXDEVTYP_USB 0
```

```
#define PRXDEVTYP_SER 1
```

```
#define PRXDEVTYP_TCP 2
```

# SetExtendedPrecisionMath

[Cached]

Windows Linux Mac

pcProx Plus

BSHRT SetExtendedPrecisionMath(short on)

## DESCRIPTION

Set the state of the extended precision math flag. This affects the FAC value when it is longer than 32 bits. When off FAC values over 32 bits will not be displayed properly.

## PARAMETERS

None

## RETURNS

TRUE on / FALSE off

## SEE ALSO

[GetExtendedPrecisionMath\(\)](#)



# SetFlags

[\[Traffic\]](#)

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** SetFlags(tsCfgFlags \*psCfgFlgs)

## DESCRIPTION

Set parameters to device.

Note pcSwipe and pcProx Sonar only use the bHaltKBSnd flag within this structure.

## PARAMETERS

psCfgFlgs pointer to structure containing new configuration flags information.

This does NOT write the configuration items to the device, just to library memory.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[GetFlags\(\)](#)

# SetFlags2

[Cached]

Windows Linux Mac

[pcProx](#)**BSHRT** SetFlags2(tsCfgFlags2 \*psCfgFlgs)

## DESCRIPTION

Set the values from flags2 data structure into the device. This does NOT write the configuration items to the device, just to library memory.

## PARAMETERS

psCfgFlgs2 pointer to structure to write Configuration Flags2 information.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[GetFlags2\(\)](#)

# SetFlags3

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)**BSHRT** `SetFlags3(tsCfgFlags3 *psCfgFlgs)`

## DESCRIPTION

Set the values from the data structure into the device. This does NOT write the configuration items to the device, just to library memory. pcSwipe only uses the bUseNumKP flag of this structure.

## PARAMETERS

psCfgFlgs3 pointer to structure to write Configuration Flags3 information.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[GetFlags3\(\)](#)

# SetHashKeyData

[Cached]

Windows Linux Mac

pcProx

**BSHRT** SetHashKeyData(tsHashKeyData \*HashKeyData)

## DESCRIPTION

Set the Hash Keys Data in to the Hash field. This does NOT write the configuration items to the device, just to library memory.

## PARAMETERS

HashKeyData pointer to structure containing new key data.

## RETURNS

TRUE = Success / FALSE = Failure

# SetIDBitCnts

[Cached]

Windows Linux Mac

pcProx

**BSHRT** SetIDBitCnts(tsIDBitCnts \*psIDBitCnts)

## DESCRIPTION

Set the number of bits in the ID field. This does NOT write the configuration items to the device, just to library memory.

## PARAMETERS

psIDBitCnts pointer to structure containing new bit count information.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[GetIDBitCnts\(\)](#)

# SetIDDispParms

[Cached]

Windows Linux Mac

pcProx

**BSHRT** SetIDDispParms(tslIDDispParms \*psIDDispParms)

## DESCRIPTION

Set display parameters to control how the card ID is displayed when keystroked out or sent serially on serial and RS-232 readers.

## PARAMETERS

psIDDispParms pointer to structure containing new ID Display information.

## RETURNS

Returns non-zero on success, zero otherwise.

## SEE ALSO

[GetIDDispParms\(\)](#)

# SetIDDispParms2

[Cached]

Windows Linux Mac

pcProx

**BSHRT** SetIDDispParms2(tsIDDispParms2 \*psIDDispParms)

## DESCRIPTION

Set display parameters2. Mainly controls the leading and trailing delimiters. Three delimiters total are allowed and can be split between leading and trailing characters.

## PARAMETERS

psIDDispParms2 pointer to structure to write ID Display2 information.

## RETURNS

Returns non-zero on success, zero otherwise.

## SEE ALSO

[GetIDDispParms2\(\)](#)

# SetIDDispParms3

[\[Cached\]](#)

Windows Linux Mac

[pcProx](#)**BSHRT** SetIDDispParms3(tsIDDispParms3 \*psIDDispParms)

## DESCRIPTION

Set display parameters3.

## PARAMETERS

psIDDispParms3 pointer to structure to write ID Display3 information.

## RETURNS

Returns non-zero on success, zero otherwise.

## SEE ALSO

[GetIDDispParms3\(\)](#)



# SetIdleParms

[Cached]

Windows Linux Mac

pcProx Sonar

**BOOL** SetIdleParms( tsIdleParms\* psIdleParms )

## DESCRIPTION

Set the pcProx Sonar Idle Parameters as as defined in the structure below.  
These key strokes are typed when the object enters the sensor range.

```
typedef struct sIdleParms
{
    short Reserved1; // Leaves as is.
    short PressRate; // Periodic Press Rate (1 second units) - [0 disabled]
    short Key1Mods; // Key 1 modifier bit flags
    short Key1Code; // Key 1 code
    short Flags; // see below.
    short DMTODelta; // DeadMan T0Delta, 3.472 * Distance (in) = DMTODelta
    short DMTOTm; // DeadMan Time (seconds) - Time till Declared Dead
                // while Deltas never exceeded
    short SNBlankTm; // pcProx Sonar drive blanking time (43usec units)
} tsIdleParms;
```

## PARAMETERS

tsIdleParms structure

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[GetIdleParms\(\)](#)

# SetIpPort

[Cached]

Windows

pcProx

```
BSHRT SetIpPort(BYTE i0,  
                BYTE i1,  
                BYTE i2,  
                BYTE i3,  
                unsigned short port)
```

## DESCRIPTION

Set the TCP/IP/UDP source address to be used to connect to Ethernet readers. A call to USBConnect() will check USB, TCP/IP and Serial ports in that order. When TCP/IP devices are checked this is the IP address and port used. The default Xport port value is 10,001.

## PARAMETERS

4 byte dotted ip notation such as 192.168.0.1 port 10,001  
SetIpPort(192,168,0,1, 10001);

## RETURNS

Always returns TRUE / Success

## SEE ALSO

[FindXport\(\)](#)  
[SetDevTypeSrch\(PRXDEVTYP\\_TCP\)](#)

# SetLEDCtrl

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** SetLEDCtrl(tsLEDCtrl \*psLEDCtrl)

## DESCRIPTION

Sets the LED configuration of the reader and controls the color of the LED (red, green, amber, or off).

## PARAMETERS

A pointer to the psLEDCtrl struct which contains the LED configuration.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[GetLEDCtrl\(\)](#)

# SetLUID

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** [SetLUID\(short LUID\)](#)

## DESCRIPTION

Set Logical unit ID to device.

## PARAMETERS

LUID is a user defined 16-bit ID (0-65536) to be associated with the current device. Returns non-zero on Success, zero otherwise.

To help you identify one device from another you can set their LUID values. Devices may enumerate in random order and the LUID is the safest way to tell them apart.

Note: You may have up to MAXDEVSOPEN readers on a given computer. This does NOT write the configuration items to the device, just to library memory. Use WriteCfg() to write to device memory. Setting the reader to defaults does not erase the LUID.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[GetLUID\(\)](#)

# SetMarkerFunction

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#)

```
void SetMarkerFunction(RFPRXLOG pfLog)
```

## DESCRIPTION

Set function to be used as a callback to log errors.

## PARAMETERS

The pfLog should have the prototype of pfCallBackLog(char szBuf, int ilen);

## RETURNS

void

# SetSepFldData

[Cached]

Windows Linux Mac

pcProx

**BSHRT** SetSepFldData(BYTE \*pBuf, short wBufMaxSz)

## DESCRIPTION

Set the FIPS 201 credentials and delimiters for all user defined fields. This is for pcProx or OEM Wiegand converter board that can read FIPS 201 CHUID's. Usually from 75 to 245 bits make up some or all of the FIPS 201 fields. OEM Readers such as the HID-G3 75 bit reader is supported by the OEM converter board.

## PARAMETERS

pBuf pointer to memory array to set the FIPS 201 Chuid data.  
wBufMaxSz is always bytes.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[GetSepFldData\(\)](#)

# SetSonarParms

[Cached]

Windows Linux Mac

[pcProx](#) [Sonar](#)**BOOL** SetSonarParms( tsSonarParms\* psSonarParms )

## DESCRIPTION

Set pcProx Sonar Parameters. This function set the parameters as defined by the structure of values below. Typical default values show in brackets.

```
typedef struct tsSonarParms
```

```
{  
    short LEDFlags; // LED usage flags [0]  
    short PingRate; // Ping rate (msec) [332] (min 200, max 1020)  
    short Reserved1; // 40KHz cycle cnt (READ ONLY)  
    short MinDist; // Minimum Distance acceptable (inches) [14] (min 14, max 59)  
    short MaxDist; // Maximum Distance acceptable (inches) [36] (min 15, max 60)  
    short ORDBTm; // debounce out of range time (seconds) [0]  
    short IRDBTm; // debounce in range time (seconds) [0]  
    short StartDly; // cold start delay in seconds before joining USB [0]  
} tsSonarParms;
```

## PARAMETERS

tsSonarParms structure

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[GetSonarParms\(\)](#)

# SetTimeParms

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** [SetTimeParms](#)([tsTimeParms](#) \*[psTimeParms](#))

## DESCRIPTION

Set device timing information. Set the Key-stroke down and release times in 4 millisecond units. Card times have 48 millisecond granularity.

## PARAMETERS

[psTimeParms](#) pointer to structure containing new Timing information.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[GetTimeParms\(\)](#)



# SetUnsupportedProductErrorCode

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

```
void SetUnsupportedProductErrorCode(BSHRT v)
```

## DESCRIPTION

This function allows you to set the return code for functions not supported by a product to aid you in porting pcProx applications to pcSwipe and/or pcProx Sonar. Set return value for the unsupported functions to return. Some functions that return a BYTE may also return this value. By Default this is 0x00 (FALSE) but can be set to any BSHRT value you desire to help know unsupported errors from function errors. The default value is set each time the library loads.

## PARAMETERS

This value is returned when a function is not supported on a product. For example calling SetFlags2() on a pcSwipe product would return this error code.

## RETURNS

void

## SEE ALSO

[GetUnsupportedProductErrorCode\(\)](#)

# SetWalkAwayParms

[Cached]

Windows Linux Mac

pcProx Sonar

**BOOL** SetWalkAwayParms( tsWalkAwayParms\* psWalkAwayParms )

## DESCRIPTION

Set pcProx Sonar Walk Away Parameters. This function sets the values as defined in the structure below. These key strokes are typed when the object leaves the sensor range.

```
typedef struct tsWalkAwayParms
```

```
{
    short KeyCount;      // Defined keys to follow (max 6 modifiers/keys) [0]
    short InterKeyDelay; // Time between key presses (msec) [512] (min 64, max 16320)
    short Key1Mods;      // Key Modifiers..[0]
    short Key2Mods;      // Bit 0..7 Left Ctrl,Shift,Alt,GUI, Right Ctrl,Shift,Alt,GUI
    short Key3Mods;
    short Key4Mods;
    short Key5Mods;
    short Key6Mods;
    short Flags0;
    short MinFltRepRate; // Near fault send repeat rate in seconds
    short Key1Code;      // USB Key Scan Codes 'a' = 4, b=5, 'c'=6 etc,
    short Key2Code;      //
    short Key3Code;
    short Key4Code;
    short Key5Code;
    short Key6Code;
} tsSonarParms;
```

## PARAMETERS

tsWalkAwayParms structure

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[GetWalkAwayParms\(\)](#)

# SetWalkUpParms

[Cached]

Windows Linux Mac

pcProx Sonar

**BOOL** SetWalkUpParms( tsWalkUpParms\* psWalkUpParms )

## DESCRIPTION

Get the pcProx Sonar Walk Up Parameters as defined in the structure below.  
These key strokes are typed when the object enters the sensor range.

```
typedef struct tsWalkAwayParms
```

```
{
    short KeyCount;      // Defined keys to follow (max 6 modifiers/keys) [0]
    short InterKeyDelay; // Time (ms) between key presses [512] (range 64..16320)
    short Key1Mods;      // Key Modifiers..[0]
    short Key2Mods;      // Bit 0..7 Left Ctrl,Shift,Alt,GUI, Right Ctrl,Shift,Alt,GUI
    short Key3Mods;
    short Key4Mods;
    short Key5Mods;
    short Key6Mods;
    short Flags0;
    short Reserved2;
    short Key1Code;      // Key Codes..[0]
    short Key2Code;      // USB Key Scan Codes 'a' = 4, b=5, 'c'=6 etc,
    short Key3Code;
    short Key4Code;
    short Key5Code;
    short Key6Code;
} tsWalkUpParms; } tsSonarParms;
```

## PARAMETERS

tsWalkAwayParms structure

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[GetWalkUpParms\(\)](#)

# USBConnect

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** [USBConnect\(long\\* pIDID\)](#)

## DESCRIPTION

This is the main function used to connect to USB and/or Serial devices. If no USB devices are found then the serial port range will be searched. This function connects to a USB and/or SERIAL device and returns the Device ID or firmware version. This API is thread-safe.

## PARAMETERS

pIDID is a pointer (reference) to long integer to receive the USB Device ID (firmware version) This function (if successful) opens and maintains a Handle to the USB Device. The Device ID is the firmware version. A long integer is used to avoid sign-extension problems in the return value and its use.

## RETURNS

Returns non-zero on success, zero otherwise. Firmware values of 0x00000111 should be interpreted as v 01.11. When several USB device are on the bus they may not enumerate in the same order on each connect call. Use the LUID to distinguish one device from another. On library version 2.00 and above, all pcProx devices are found in this routine and the DID of the first (index = 0) is returned and the first is made the Active Device.

Note: Do not compare version numbers from the pcProx firmware to enable or disable functions in your application as firmware version can change. The firmware version is for reference only.

## EXAMPLE

```
USBDisconnect(); long DeviceID = 0;
if(USBConnect(&DeviceID) == TRUE) {
    for(;;) {
        BYTE buffer[8];
        int Bits = GetActiveID(buffer,sizeof(buffer));
        Sleep(250); // Sleep a quarter of a second. i.e 250 milliseconds
    }
}
```

## SEE ALSO

[usbConnect\(\)](#)  
[USBDisconnect\(\)](#)  
[SetDevTypeSrch\(\)](#)

# USBDisconnect

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** [USBDisconnect\(void\)](#)

## DESCRIPTION

This function closes the open handle to ALL USB device(s) and/or open serial port(s). It may be called at any time even if [USBConnect\(\)](#) has not been called. It is good practice is to call [USBDisconnect](#) before calling any [Connect](#) function to clear any pending errors.

Errors can happen if the device is physically removed from the USB bus while the user application is open and connected to the device.

This API is thread-safe.

## PARAMETERS

None

## RETURNS

Always TRUE = Success

## SEE ALSO

[USBConnect\(\)](#)

# VirtualComSearchRange

[Cached]

Windows

[pcProx](#) [pcSwipe](#)**BSHRT** VirtualComSearchRange(WORD iMin, WORD iMax)

## DESCRIPTION

Set the virtual com port search range. When this is set the given range of COM ports will be searched during the next Serial port COM connect. Any virtual com ports from USB or PCMCIA adapters with the rf IDEAS USB VID and matching USB PID will be opened and tested against the protocol.

The Windows registry is read to find the VCOM ports. Powerup defaults are 1..256. This function has no effect under Linux.

## PARAMETERS

min serial port 1..256 range  
max serial port 1..256 range  
call with (0,0) to disable all Virtual ports

## RETURNS

Return: TRUE / else FALSE for value out of range.

# WriteBLEDataToFile

[Cached]

Windows Linux Mac

pcProx

**BSHRT** WriteBLEDataToFile(char\* szFileName)

## DESCRIPTION

Write an ASCII file with all the BLE data of most recent device BLE setting. This writes the buffered data stored in the reader to a file.

## PARAMETERS

szFileName file name that may include complete path to the file name.

## RETURNS

TRUE / Success file was written with BLE data.

FALSE file may be locked or the path is invalid or BLE Data could not be calculated.

# WriteBLEDataToReader

[Cached]

Windows Linux Mac

pcProx

**BSHRT** WriteBLEDataToReader(char\* szFileName)

## DESCRIPTION

Configure the reader with new BLE data by sending BLE Data payloads from a file to the reader.

## PARAMETERS

szFileName file name that may include complete path to the file name.

## RETURNS

TRUE / Success reader have written with new BLE data.

FALSE / Failure reader have not written with new BLE data.



# WriteBLEKeyToFile

[Cached]

Windows Linux Mac

pcProx

**BSHRT** WriteBLEKeyToFile(char\* szFileName)

## DESCRIPTION

Write an ASCII file with all the BLE Key of most recent device BLE setting. This writes the buffered key stored in the reader to a file.

## PARAMETERS

szFileName file name that may include complete path to the file name.

## RETURNS

TRUE / Success file was written with BLE key.

FALSE file may be locked or the path is invalid or BLE key could not be calculated.

# WriteBLEKeyToReader

[Cached]

Windows Linux Mac

pcProx

**BSHRT** WriteBLEKeyToReader(char\* szFileName)

## DESCRIPTION

Configure the reader with new BLE key  
by sending BLE key payloads from a file to the reader.

## PARAMETERS

szFileName file name that may include complete path to the file name.

## RETURNS

TRUE / Success reader have written with new BLE key.  
FALSE / Failure reader have not written with new BLE key.

# WriteBaudRate

[\[Traffic\]](#)

Windows Linux

[pcProx Plus](#)**BSHRT** WriteBaudRate(**BSHRT** baudrateCode)

## DESCRIPTION

Writes new baud rate in the reader. However, the baud rate must be supported by the reader.

## PARAMETERS

Baud rate code, where  
00 = 9600;  
01 = 19200;  
02 = 38400;  
03 = 57600;  
04 = 115200;

## RETURNS

1 on success and 0 on failure.

## SEE ALSO

[ReadBaudRate\(\)](#)  
[SetBaudRate\(int baudrate\)](#)

# WriteCfg

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** [WriteCfg\(void\)](#)

## DESCRIPTION

A call to this function writes all configuration information in the library memory space to the device for non-volatile storage. Any changed parameters take effect immediately after WriteCfg(). The actual write internally within the device is not done until all critical pending actions are complete. This may take up to two seconds, typically 1200 msec to complete.

pcProx Plus(version 2) readers takes only 20ms to complete the write.

This API is thread-safe.

## PARAMETERS

None

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[ReadCfg\(\)](#)

# WriteDevCfgToFile

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** WriteDevCfgToFile(char \*szFileName)

## DESCRIPTION

Write an ASCII file with all the setting of most recent the device setting read by ReadCfg(). This writes the buffered data held by the library to a file. The user should call ReadCfg() or WriteCfg() to make sure the correct values are sync'ed with the device memory. The file can be loaded back into the device with function ReadDevCfgFmFile()

Note: The file is in ASCII format. It is not compatible with the simpler Visual Basic pcProxConfig format. You may edit the ASCII text file, taking care to preserve the text formatting.

## PARAMETERS

szFileName file name that may include complete path to the file name.

## RETURNS

TRUE / Success file was written.

FALSE file may be locked or the path is invalid.

## SEE ALSO

[ReadDevCfgFmFile\(\)](#)  
[readDevCfgFmFile\\_char\(\)](#)  
[writeDevCfgToFile\\_char\(\)](#)

# WriteDevCfgToSecureFile

[Cached]

Windows Linux Mac

pcProx

**BSHRT** WriteDevCfgToSecureFile(char\* szFileName)

## DESCRIPTION

Write an ASCII file with all the setting of most recent the device setting read by ReadCfg(). This writes the buffered data held by the library to a file. The user should call ReadCfg() or WriteCfg() to make sure the correct values are sync'ed with the device memory. It will generate the secure key for the file and append at the end/top of the file.

Note: The file is in ASCII format. It is not compatible with the simpler Visual Basic pcProxConfig format. You may edit the ASCII text file, taking care to preserve the text formatting.

## PARAMETERS

szFileName file name that may include complete path to the file name.

## RETURNS

TRUE / Success file was written with secure key.

FALSE file may be locked or the path is invalid or secure key could not be calculated.

# WriteSecureData

[Cached]

Windows Linux Mac

pcProx

```
bool WriteSecureData(BYTE *blobBuffer, int blobLen)
```

## DESCRIPTION

Write the configuration as requested in header(blob details: Type, config, length, BSV).

## PARAMETERS

blobBuffer pointer (reference) to character buffer to send the blob details and data to reader\*  
bloblen is the interger value for length of buffer whose pointer (reference) is sent.

## RETURNS

Returns true on success and false on failure.

# WriteSerialPort

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)

**DWORD** WriteSerialPort(char \*buf, DWORD count)

## DESCRIPTION

Write data to serial port. This allows user to also do their own MFP24 or ACP commands while using the binary protocol on the reader.

## PARAMETERS

pointer to char buffer, and count of max desired characters to transmit.

## RETURNS

Actual number of characters sent.

## SEE ALSO

[QuickReadSerialPort\(\)](#)  
[quickReadSerialPort\\_char\(\)](#)  
[readSerialPort\\_char\(\)](#)



# chkAddArrival\_char

[Cached]

Windows

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

short `chkAddArrival_char`(short index, char c)

## DESCRIPTION

C# interface to Check device add arrival function.

## PARAMETERS

index 0..255. 255 trigger I/O call. When index is 0 internal

DeviceName buffer is zeroed out.

DeviceName atleast of 511 (MAXDEVNAMESZ) characters

## RETURNS

Check to see if device is in the list held by the library

TRUE = Success / FALSE = Failure

## SEE ALSO

[ChkAddArrival\(\)](#)

[ChkDelRemoval\(\)](#)

[chkDelRemoval\\_char\(\)](#)

# chkDelRemoval\_char

[Cached]

Windows

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

short `chkDelRemoval_char`(short index, char c)

## DESCRIPTION

C# interface to Check device removal function.

## PARAMETERS

index 0..255. 255 trigger I/O call. When index is 0 internal

DeviceName buffer is zeroed out.

DeviceName atleast of 511 (MAXDEVNAMESZ) characters

## RETURNS

Check to see if device is in the list held by the library

TRUE = Success / FALSE = Failure

## SEE ALSO

[chkAddArrival\\_char\(\)](#)

[ChkAddArrival\(\)](#)

[ChkDelRemoval\(\)](#)

# comConnect

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)**BSHRT** [comConnect\(void\)](#)

## DESCRIPTION

Connect to Serial COM Port.

The com port range set by [SetComSrchRange\(lo,hi\)](#) will search for a pcProx reader. The com ports search that do not have a reader will take longer as they time-out.

This is a wrapper around [comConnect](#). It returns the same value. It does not require a pointer to return the DeviceID. Use [GetDID\(\)](#) to retrieve the device ID firmware version.

Defaults are com ports 1..8

## PARAMETERS

None

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[ComConnect\(\)](#)[SetComLinux\(\)](#)

# comConnectPort

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)**BSHRT** `comConnectPort(WORD iPort)`

## DESCRIPTION

Connect to one serial device on COM port iPort (1..256)

## PARAMETERS

Port number 1..256 representing COM1 thru COM256 on Windows.  
Linux serial ports 1..MAXLINUXCOMPORT can be mapped to the /dev/ path.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[usbConnect\(\)](#)  
[comConnect\(\)](#)  
[ComConnectPort\(\)](#)  
[SetComLinux\(\)](#)

# getActiveCardData

[\[Traffic\]](#)

Windows Linux Mac

[pcProx](#)

BSHRT getActiveCardData(void)

## DESCRIPTION

Read the Active Card Data. This returns three main pieces of informations: The ScanCount, Card Serial Number (CSN), and Card ID (ID). The Scan count is a one byte value that starts at zero during powerup and increments and wraps from 255 back to zero. The CSN is preceded by the byte count of the CSN size. The ID is preceded by the Bit Count of the card ID. Here is an example of the third card read, scan count = 3, the CSN is 0x12,0x34,0x56,0x78 and a 26 bit ID is 0xAB,0xCD,0xEF,0x80.

The buffer[0x00..0x0A] = 0x03,0x04,0x12,0x34,0x56,0x78,0x1A,0xAB,0xCD,0xEF,0x80  
Retrieving bytes:

getActiveCardData\_byte(0) => 0x03 Scan Count

getActiveCardData\_byte(1) => 0x04 4 Bytes of CSN (2)..(5)

getActiveCardData\_byte(6) => 0x1A Bit Count for following ID

## PARAMETERS

None

## RETURNS

Returns TRUE = Success, FALSE Error or no card read or function not supported in Firmware.

## EXAMPLE

```
BYTE Count = getActiveCardData_byte(0);
BYTE BytesInCSN = getActiveCardData_byte(1);
BYTE BitsInID = getActiveCardData_byte(2+BytesInCSN);
BYTE BytesInID = (BitsInID+7)/8;
BYTE IDBuffer[11];
int idstart = BytesInCSN+3;
for(int i=0; i<BytesInID; i++)
{
    IDBuffer[i] = getActiveCardData_byte(idstart+i);
}
```

## SEE ALSO

[getActiveCardData\\_byte\(\)](#)

# getActiveCardData\_byte

[Cached]

Windows Linux Mac

pcProx

BSHRT getActiveCardData\_byte(short index)

## DESCRIPTION

After the call to getActiveCardData() which reads the card scan count, CSN, and card ID into an internal buffer, the bytes from the buffer are retrieved using this function.

## PARAMETERS

Index into byte buffer.

## RETURNS

Returns 0..FF = Error or no card read. Or return 0 if index is out of range. and return unsupported code for non pcProx devices.

Index Value

-----  
0    ScanCount  
1    CSN Byte that follow  
2..N   CSN Data is [1] is non zero  
N+1   ID Bit Count  
N+2..   Bit of ID  
Zero filled

## SEE ALSO

[getActiveCardData\(\)](#)

# getActiveID

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**short** `getActiveID(short wBufMaxSz)`

## DESCRIPTION

Read the active data from the card presently on the reader. This function will return the number of bits read. Then you call `getActiveID_byte(index)` to return each byte. Index of 0 is the Least Significant Byte holding Bits 0..7. It is recommended you do not call this faster than 250msec, or about twice the data hold time of the active card. This API is thread-safe.

## PARAMETERS

`wBufMaxSz` specifies the size of the character buffer. A max of 8 will be used so this is useful only to limit the buffer transfer to < 8 characters. Values > 8 will still only transfer 8 characters with no error.

## RETURNS

Returns the number of bits received from the reader representing the ID. The return count represents how many bits in the buffer are valid ID bits,

## SEE ALSO

[getActiveID\\_byte\(\)](#)

# getActiveID32

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

short `getActiveID32(short wBufMaxSz)`

## DESCRIPTION

Read the active ID from the device and buffer the data for the function `getActiveID_byte()`.

It is recommended you do not call this faster than 250msec, or about twice the data hold time of the active card.

This API is thread-safe.

## PARAMETERS

`wBufMaxSz` specifies the size of the character buffer. A max of 32 will be used so this is useful only to limit the buffer transfer.

## RETURNS

Number of bits read from card. Zero if no card present or error.

It does NOT include the parity bits that may have been stripped from the ID through the use of the Leading and/or Trailing parity bit counts.

A return of zero means that there is either no card within range or that there was another error encountered. `GetLastLibErr()` may be used to differentiate between the two possibilities.

## SEE ALSO

[GetActiveID32\(\)](#)



# getActiveID\_byte

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)BYTE `getActiveID_byte(short index)`

## DESCRIPTION

This function should be called after `getActiveID32()`  
This function will return the byte at the give index 0..31  
Index 0 is the Least Significant Byte holding Bits 0..7

## PARAMETERS

index value 0 through 7 inclusive.  
if the index is out of range the value returned will be zero.

## RETURNS

Returns the byte at `buffer[index]` from the card read by `GetActiveID()` or `GetActiveID32()` or `getActiveID32()`

# getBprRlyCtrl\_bVolatile

[Cached]

Windows Linux Mac

pcProx

```
short getBprRlyCtrl_bVolatile()
```

## DESCRIPTION

C# interface for safe code. Get BprRlyCtrl member's bVolatile value or error code. You must call ReadCfg() first. This function calls GetBprRlyCtrl() internally. Note: 0 == commit to EE, 1 == Don't store to EE.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsBprRlyCtrl or error if GetBprRlyCtrl() fails.

# getBprRlyCtrl\_iBeeperState

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)

```
short getBprRlyCtrl_iBeeperState()
```

## DESCRIPTION

C# interface for safe code. Get BprRlyCtrl member's iBeeperState value or error code. You must call ReadCfg() first. This function calls GetBprRlyCtrl() internally.  
Note: 0 == Off, 1 == On.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsBprRlyCtrl or error if GetBprRlyCtrl() fails.

# getBprRlyCtrl\_iPad0

[Cached]

Windows Linux Mac

pcProx

```
short getBprRlyCtrl_iPad0()
```

## DESCRIPTION

C# interface for safe code. Get BprRlyCtrl member's iPad0 value or error code. You must call ReadCfg() first. This function calls GetBprRlyCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsBprRlyCtrl or error if GetBprRlyCtrl() fails.

# getBprRlyCtrl\_iPad3

[Cached]

Windows Linux Mac

pcProx

`short getBprRlyCtrl_iPad3()`

## DESCRIPTION

C# interface for safe code. Get BprRlyCtrl member's iPad3 value or error code. You must call ReadCfg() first. This function calls GetBprRlyCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsBprRlyCtrl or error if GetBprRlyCtrl() fails.

# getBprRlyCtrl\_iPad4

[Cached]

Windows Linux Mac

pcProx

`short getBprRlyCtrl_iPad4()`

## DESCRIPTION

C# interface for safe code. Get BprRlyCtrl member's iPad4 value or error code. You must call ReadCfg() first. This function calls GetBprRlyCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsBprRlyCtrl or error if GetBprRlyCtrl() fails.

# getBprRlyCtrl\_iPad5

[Cached]

Windows Linux Mac

pcProx

`short getBprRlyCtrl_iPad5()`

## DESCRIPTION

C# interface for safe code. Get BprRlyCtrl member's iPad5 value or error code. You must call ReadCfg() first. This function calls GetBprRlyCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsBprRlyCtrl or error if GetBprRlyCtrl() fails.

# getBprRlyCtrl\_iPad6

[Cached]

Windows Linux Mac

pcProx

```
short getBprRlyCtrl_iPad6()
```

## DESCRIPTION

C# interface for safe code. Get BprRlyCtrl member's iPad6 value or error code. You must call ReadCfg() first. This function calls GetBprRlyCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsBprRlyCtrl or error if GetBprRlyCtrl() fails.



# getBprRlyCtrl\_iRelayState

[Cached]

Windows Linux Mac

pcProx

```
short getBprRlyCtrl_iRelayState()
```

## DESCRIPTION

C# interface for safe code. Get BprRlyCtrl member's iRelayState value or error code. You must call ReadCfg() first. This function calls GetBprRlyCtrl() internally.  
Note: 0 == Off, 1 == On.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsBprRlyCtrl or error if GetBprRlyCtrl() fails.

# getCfgFlags2\_bBeepID

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)

```
short getCfgFlags2_bBeepID()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags2 member's bBeepID value or error code. You must call ReadCfg() first. This function calls GetFlags2() internally. Note: Beep when ID received.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags2 or error if GetFlags2() fails.

# getCfgFlags2\_bDspHex

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags2_bDspHex()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags2 member's bDspHex value or error code. You must call ReadCfg() first. This function calls GetFlags2() internally. Note: Display ID as ASCII Hex [not ASCII decimal].

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags2 or error if GetFlags2() fails.

# getCfgFlags2\_bRevBytes

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags2_bRevBytes()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags2 member's bRevBytes value or error code. You must call ReadCfg() first. This function calls GetFlags2() internally. Note: Reverse byte order (CSN reader).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags2 or error if GetFlags2() fails.

# getCfgFlags2\_bRevWiegBits

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags2_bRevWiegBits()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags2 member's bRevWiegBits value or error code. You must call ReadCfg() first. This function calls GetFlags2() internally. Note: Reverse the Wiegand Rx bits.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags2 or error if GetFlags2() fails.

# getCfgFlags2\_bUseInvDataF

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags2_bUseInvDataF()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags2 member's bUseInvDataF value or error code. You must call ReadCfg() first. This function calls GetFlags2() internally. Note: Use the bWiegInvData flag over hardware setting.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags2 or error if GetFlags2() fails.

# getCfgFlags2\_bUseLeadChrs

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags2_bUseLeadChrs()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags2 member's bUseLeadChrs value or error code. You must call ReadCfg() first. This function calls GetFlags2() internally. Note: Use leading chars in ID KB send.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags2 or error if GetFlags2() fails.

# getCfgFlags2\_bWiegInvData

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags2_bWiegInvData()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags2 member's bWiegInvData value or error code. You must call ReadCfg() first. This function calls GetFlags2() internally.  
Note: Wiegand data signals are typically active low.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags2 or error if GetFlags2() fails.



# getCfgFlags2\_isASCIIPresent

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags2_isASCIIPresent()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags2 member's isASCIIPresent flag's value or error code.

You must call ReadCfg() first. This function calls GetFlags2() internally.

Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags2 or error if GetFlags2() fails.

# getCfgFlags3\_bLowerCaseHex

[Cached]

Windows Linux Mac

[pcProx](#)

```
short getCfgFlags3_bLowerCaseHex()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags3 member's bLowerCaseHex value or error. code. You must call ReadCfg() first. This function calls GetFlags3() internally. If the flag is set (non zero) then hex output will use lowercase a-z else uppercase A-Z.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags3 or error if GetFlags3() fails.

# getCfgFlags3\_bNotBootDev

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags3_bNotBootDev()
```

## DESCRIPTION

Deprecated function always returns Unsupported API return code.

## PARAMETERS

None

## RETURNS

Unsupported API return code.

# getCfgFlags3\_bPrxProEm

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags3_bPrxProEm()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags3 member's bPrxProEm value or error code. You must call ReadCfg() first. This function calls GetFlags3() internally. Note: Use HID ProxPro card emulation. See HIDCORP.COM for details.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags3 or error if GetFlags3() fails.

# getCfgFlags3\_bSndSFFC

[Cached]

Windows Linux Mac

pcProx

`short getCfgFlags3_bSndSFFC()`

## DESCRIPTION

C# interface for safe code. Get CfgFlags3 member's bSndSFFC value or error code. You must call ReadCfg() first. This function calls GetFlags3() internally.  
Note: 0 = FAC Decimal, 1 = FAC Hex.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags3 or error if GetFlags3() fails.

# getCfgFlags3\_bSndSFID

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags3_bSndSFID()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags3 member's bSndSFID value or error code. You must call ReadCfg() first. This function calls GetFlags3() internally.  
Note: 0 = ID Decimal, 1 = ID Hex.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags3 or error if GetFlags3() fails.

# getCfgFlags3\_bSndSFON

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags3_bSndSFON()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags3 member's bSndSFON value or error code. You must call ReadCfg() first. This function calls GetFlags3() internally. Note: Split format ON = 1, old combined scheme = 0.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags3 or error if GetFlags3() fails.

# getCfgFlags3\_bUse64Bit

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags3_bUse64Bit()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags3 member's bUse64Bit value or error code. You must call ReadCfg() first. This function calls GetFlags3() internally.

It is recommended to leave this on.

Note: 0 = 32-bit, 1 = 64-bit Display Math.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags3 or error if GetFlags3() fails.



# getCfgFlags3\_bUseNumKP

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)

```
short getCfgFlags3_bUseNumKP()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags3 member's bUseNumKP value or error code. You must call ReadCfg() first. This function calls GetFlags3() internally.  
Note: Euro KB flag.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags3 or error if GetFlags3() fails.

# getCfgFlags\_bFixLenDsp

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags_bFixLenDsp()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags member's bFixLenDsp value or error code. You must call ReadCfg() first. This function calls GetFlags() internally. Note: Send as fixed length with leading zeros as needed.

## PARAMETERS

None

## RETURNS

Short value from typedef struct tsCfgFlags or error if GetFlags() fails.

# getCfgFlags\_bFrcBitCntEx

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags_bFrcBitCntEx()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags member's bFrcBitCntEx value or error code. You must call ReadCfg() first. This function calls GetFlags() internally.  
Note: Force Rx'd bit count to be exact to be valid.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags or error if GetFlags() fails.

# getCfgFlags\_bHaltKBSnd

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)`short getCfgFlags_bHaltKBSnd()`

## DESCRIPTION

C# interface for safe code. Get CfgFlags member's bHaltKBSnd value or error code. You must call ReadCfg() first. This function calls GetFlags() internally. Note: Don't Send keys to USB (Get ID mechanism).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags or error if GetFlags() fails.

# getCfgFlags\_bNoUseELChar

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags_bNoUseELChar()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags member's bNoUseELChar value or error code. You must call ReadCfg() first. This function calls GetFlags() internally. Note: Don't use an EndLine char on send (default to ENTER).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags or error if GetFlags() fails.

# getCfgFlags\_bSndFac

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags_bSndFac()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags member's bSndFac value or error code. You must call ReadCfg() first. This function calls GetFlags() internally. Note: Send the FAC (if stripped from data).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags or error if GetFlags() fails.

# getCfgFlags\_bSndOnRx

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags_bSndOnRx()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags member's bSndOnRx value or error code. You must call ReadCfg() first. This function calls GetFlags() internally. Note: Send valid ID as soon as it is received (iDLockOutTm timer not used).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags or error if GetFlags() fails.

# getCfgFlags\_bStripFac

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags_bStripFac()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags member's bStripFac value or error code. You must call ReadCfg() first. This function calls GetFlags() internally. Note: Strip the FAC from the ID (not discarded).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags or error if GetFlags() fails.



# getCfgFlags\_bUseDelFac2Id

[Cached]

Windows Linux Mac

pcProx

```
short getCfgFlags_bUseDelFac2Id()
```

## DESCRIPTION

C# interface for safe code. Get CfgFlags member's bUseDelFac2Id value or error code. You must call ReadCfg() first. This function calls GetFlags() internally.  
Note: Put a delimiter between FAC and ID on send.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsCfgFlags or error if GetFlags() fails.

# getDevName\_char

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)`char getDevName_char(short index)`

## DESCRIPTION

getDeviceName character by character. When the index is zero the device name is gathered from the device. The user may then read the rest of the buffered string character by character.

## PARAMETERS

index value 0 through 126 inclusive.  
if the index is out of range the value returned will be zero.

## RETURNS

Returns the byte at `buffer[index]` from the `GetDevName()` function call.

# getESN

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#)

```
const char* getESN()
```

## DESCRIPTION

Returns the 32 byte electronic serial number embedded in the processor chip.  
Note: For Wren serial reader use GetSerialNumber() API to get ESN.

## PARAMETERS

None

## RETURNS

Returns the 32 byte ESN for loon USB readers and else return NULL.

## SEE ALSO

[GetSerialNumber\(\)](#)

# getIDBitCnts\_iIDBitCnt

[Cached]

Windows Linux Mac

pcProx

```
short getIDBitCnts_iIDBitCnt()
```

## DESCRIPTION

C# interface for safe code. Get IDBitCnts member's iIDBitCnt value or error code. You must call ReadCfg() first. This function calls GetIDBitCnts() internally. Note: If bStripFac, this determines bit count of ID and FAC.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDBitCnts or error if GetIDBitCnts() fails.

# getIDBitCnts\_iLeadParityBitCnt

[Cached]

Windows Linux Mac

pcProx

```
short getIDBitCnts_iLeadParityBitCnt()
```

## DESCRIPTION

C# interface for safe code. Get IDBitCnts member's iLeadParityBitCnt value or error code. You must call ReadCfg() first. This function calls GetIDBitCnts() internally.

Note: Wiegand Leading Parity bit count to be stripped.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDBitCnts or error if GetIDBitCnts() fails.

# getIDBitCnts\_iPad4

[Cached]

Windows Linux Mac

pcProx

`short getIDBitCnts_iPad4()`

## DESCRIPTION

C# interface for safe code. Get IDBitCnts member's iPad4 value or error code. You must call ReadCfg() first. This function calls GetIDBitCnts() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDBitCnts or error if GetIDBitCnts() fails.

# getIDBitCnts\_iPad5

[Cached]

Windows Linux Mac

pcProx

```
short getIDBitCnts_iPad5()
```

## DESCRIPTION

C# interface for safe code. Get IDBitCnts member's iPad5 value or error code. You must call ReadCfg() first. This function calls GetIDBitCnts() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDBitCnts or error if GetIDBitCnts() fails.

# getIDBitCnts\_iPad6

[Cached]

Windows Linux Mac

pcProx

```
short getIDBitCnts_iPad6()
```

## DESCRIPTION

C# interface for safe code. Get IDBitCnts member's iPad6 value or error code. You must call ReadCfg() first. This function calls GetIDBitCnts() internally.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDBitCnts or error if GetIDBitCnts() fails.



# getIDBitCnts\_iPad7

[Cached]

Windows Linux Mac

pcProx

`short getIDBitCnts_iPad7()`

## DESCRIPTION

C# interface for safe code. Get IDBitCnts member's iPad7 value or error code. You must call ReadCfg() first. This function calls GetIDBitCnts() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDBitCnts or error if GetIDBitCnts() fails.

# getIDBitCnts\_iTotalBitCnt

[Cached]

Windows Linux Mac

pcProx

```
short getIDBitCnts_iTotalBitCnt()
```

## DESCRIPTION

C# interface for safe code. Get IDBitCnts member's iTotalBitCnt value or error code. You must call ReadCfg() first. This function calls GetIDBitCnts() internally. Note: If bFrcBitCntEx, card read (including parity) must match this.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDBitCnts or error if GetIDBitCnts() fails.

# getIDBitCnts\_iTrailParityBitCnt

[Cached]

Windows Linux Mac

pcProx

```
short getIDBitCnts_iTrailParityBitCnt()
```

## DESCRIPTION

C# interface for safe code. Get IDBitCnts member's iTrailParityBitCnt value or error code. You must call ReadCfg() first. This calls GetIDBitCnts() internally. Note: Wiegand Trailing Parity bit count to be stripped.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDBitCnts or error if GetIDBitCnts() fails.

# getIDDispParms2\_iCrdGnChr0

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms2_iCrdGnChr0()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms2 member's iCrdGnChr0 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms2() internally. Note: If non-zero, sent when ID goes Invalid on RS-232 readers..

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms2 or error if GetIDDispParms2() fails.

# getIDDispParms2\_iCrdGnChr1

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms2_iCrdGnChr1()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms2 member's iCrdGnChr1 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms2() internally. Note: If this and Chr0 non-zero, sent when ID goes Invalid on RS-232 readers.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms2 or error if GetIDDispParms2() fails.

# getIDDispParms2\_iLeadChr0

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms2_iLeadChr0()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms2 member's iLeadChr0 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms2() internally. Note: These lead characters are filled in (up to 3).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms2 or error if GetIDDispParms2() fails.

# getIDDispParms2\_iLeadChr1

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms2_iLeadChr1()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms2 member's iLeadChr1 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms2() internally.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms2 or error if GetIDDispParms2() fails.

# getIDDispParms2\_iLeadChr2

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms2_iLeadChr2()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms2 member's iLeadChr2 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms2() internally.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms2 or error if GetIDDispParms2() fails.



# getIDDispParms2\_iLeadChrCnt

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms2_iLeadChrCnt()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms2 member's iLeadChrCnt value or error code. You must call ReadCfg() first. This function calls GetIDDispParms2() internally. Note: If bUseLeadChrs, this contains the lead char count (<=3).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms2 or error if GetIDDispParms2() fails.

# getIDDispParms2\_iPad6

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms2_iPad6()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms2 member's iPad6 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms2() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms2 or error if GetIDDispParms2() fails.

# getIDDispParms2\_iPad7

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms2_iPad7()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms2 member's iPad7 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms2() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms2 or error if GetIDDispParms2() fails.

# getIDDispParms3\_iPad4

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms3_iPad4()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms3 member's iPad4 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms3() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms3 or error if GetIDDispParms3() fails.

# getIDDispParms3\_iPad5

[Cached]

Windows Linux Mac

pcProx

`short getIDDispParms3_iPad5()`

## DESCRIPTION

C# interface for safe code. Get IDDispParms3 member's iPad5 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms3() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms3 or error if GetIDDispParms3() fails.

# getIDDispParms3\_iPad6

[Cached]

Windows Linux Mac

pcProx

`short getIDDispParms3_iPad6()`

## DESCRIPTION

C# interface for safe code. Get IDDispParms3 member's iPad6 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms3() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms3 or error if GetIDDispParms3() fails.

# getIDDispParms3\_iPad7

[Cached]

Windows Linux Mac

pcProx

`short getIDDispParms3_iPad7()`

## DESCRIPTION

C# interface for safe code. Get IDDispParms3 member's iPad7 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms3() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms3 or error if GetIDDispParms3() fails.

# getIDDispParms3\_iTrailChr0

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms3_iTrailChr0()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms3 member's iTrailChr0 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms3() internally. Note: These trailing characters are filled in (up to 3).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms3 or error if GetIDDispParms3() fails.



# getIDDispParms3\_iTrailChr1

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms3_iTrailChr1()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms3 member's iTrailChr1 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms3() internally. Note: LeadChrCnt + TrailCheCnt <= 3.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms3 or error if GetIDDispParms3() fails.

# getIDDispParms3\_iTrailChr2

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms3_iTrailChr2()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms3 member's iTrailChr2 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms3() internally. Note: LeadChrs have priority.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms3 or error if GetIDDispParms3() fails.

# getIDDispParms3\_iTrailChrCnt

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms3_iTrailChrCnt()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms3 member's iTrailChrCnt value or error code. You must call ReadCfg() first. This function calls GetIDDispParms3() internally. Note: This contains the trail char count (<=3).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms3 or error if GetIDDispParms3() fails.

# getIDDispParms\_iELDelim

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms_iELDelim()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms member's iELDelim value or error code. You must call ReadCfg() first. This function calls GetIDDispParms() internally. Note: If NOT bNoUseELChar, this character sent at end of ID.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms or error if GetIDDispParms() fails.

# getIDDispParms\_iExOutputFormat

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms_iExOutputFormat()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms member's iExOutputFormat value. You must call ReadCfg() first. This function calls GetIDDispParms() internally. Note : If iExOutputFormat, Reader will output in Extended mode. It is applicable only for Plus Extended readers because these readers have capability to give output in both Conventional and Extended mode.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms or error if GetIDDispParms() fails.

# getIDDispParms\_iFACDispLen

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms_iFACDispLen()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms member's iFACDispLen value or error code. You must call ReadCfg() first. This function calls GetIDDispParms() internally. Note: If bFixLenDsp, FAC padded with zeros to this length.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms or error if GetIDDispParms() fails.

# getIDDispParms\_iFACIDDelim

[Cached]

Windows Linux Mac

pcProx

short `getIDDispParms_iFACIDDelim()`

## DESCRIPTION

C# interface for safe code. Get IDDispParms member's iFACIDDelim value or error code. You must call ReadCfg() first. This function calls GetIDDispParms() internally. Note: If bStripFac and bSndFac and bUseDelFac2Id, this character is sent between FAC and ID.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms or error if GetIDDispParms() fails.

# getIDDispParms\_iIDDispLen

[Cached]

Windows Linux Mac

pcProx

```
short getIDDispParms_iIDDispLen()
```

## DESCRIPTION

C# interface for safe code. Get IDDispParms member's iIDDispLen value or error code. You must call ReadCfg() first. This function calls GetIDDispParms() internally. Note: If bFixLenDsp, ID padded with zeros to this length.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms or error if GetIDDispParms() fails.



# getIDDispParms\_iPad5

[Cached]

Windows Linux Mac

pcProx

`short getIDDispParms_iPad5()`

## DESCRIPTION

C# interface for safe code. Get IDDispParms member's iPad5 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms or error if GetIDDispParms() fails.

# getIDDispParms\_iPad6

[Cached]

Windows Linux Mac

pcProx

`short getIDDispParms_iPad6()`

## DESCRIPTION

C# interface for safe code. Get IDDispParms member's iPad6 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms or error if GetIDDispParms() fails.

# getIDDispParms\_iPad7

[Cached]

Windows Linux Mac

pcProx

`short getIDDispParms_iPad7()`

## DESCRIPTION

C# interface for safe code. Get IDDispParms member's iPad7 value or error code. You must call ReadCfg() first. This function calls GetIDDispParms() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsIDDispParms or error if GetIDDispParms() fails.

# getLEDCtrl\_bAppCtrlsLED

[Cached]

Windows Linux Mac

pcProx

```
short getLEDCtrl_bAppCtrlsLED()
```

## DESCRIPTION

C# interface for safe code. Get LEDCtrl member's bAppCtrlsLED value or error code  
You must call ReadCfg() first. This function calls GetLEDCtrl() internally.  
Note: Display LEDs are controlled by user thru this library.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsLEDCtrl or error if GetLEDCtrl() fails.

# getLEDCtrl\_bVolatile

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

```
short getLEDCtrl_bVolatile()
```

## DESCRIPTION

C# interface for safe code. Get LEDCtrl member's bVolatile value or error code. You must call ReadCfg() first. This function calls GetLEDCtrl() internally. Note: 0 == commit to EE, 1 == Don't store to EE.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsLEDCtrl or error if GetLEDCtrl() fails.

# getLEDCtrl\_iGrnLEDState

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

short `getLEDCtrl_iGrnLEDState()`

## DESCRIPTION

C# interface for safe code. Get LEDCtrl member's iGrnLEDState value or error code. You must call ReadCfg() first. This function calls GetLEDCtrl() internally.  
Note: 0 == Off, 1 == On.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsLEDCtrl or error if GetLEDCtrl() fails.

# getLEDCtrl\_iPad3

[Cached]

Windows Linux Mac

pcProx

`short getLEDCtrl_iPad3()`

## DESCRIPTION

C# interface for safe code. Get LEDCtrl member's iPad3 value or error code. You must call ReadCfg() first. This function calls GetLEDCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsLEDCtrl or error if GetLEDCtrl() fails.

# getLEDCtrl\_iPad4

[Cached]

Windows Linux Mac

pcProx

`short getLEDCtrl_iPad4()`

## DESCRIPTION

C# interface for safe code. Get LEDCtrl member's iPad4 value or error code. You must call ReadCfg() first. This function calls GetLEDCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsLEDCtrl or error if GetLEDCtrl() fails.



# getLEDCtrl\_iPad5

[Cached]

Windows Linux Mac

pcProx

`short getLEDCtrl_iPad5()`

## DESCRIPTION

C# interface for safe code. Get LEDCtrl member's iPad5 value or error code. You must call ReadCfg() first. This function calls GetLEDCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsLEDCtrl or error if GetLEDCtrl() fails.

# getLEDCtrl\_iPad6

[Cached]

Windows Linux Mac

pcProx

`short getLEDCtrl_iPad6()`

## DESCRIPTION

C# interface for safe code. Get LEDCtrl member's iPad6 value or error code. You must call ReadCfg() first. This function calls GetLEDCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsLEDCtrl or error if GetLEDCtrl() fails.

# getLEDCtrl\_iRedLEDState

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

```
short getLEDCtrl_iRedLEDState()
```

## DESCRIPTION

C# interface for safe code. Get LEDCtrl member's iRedLEDState value or error code. You must call ReadCfg() first. This function calls GetLEDCtrl() internally.  
Note: 0 == Off, 1 == On.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsLEDCtrl or error if GetLEDCtrl() fails.

# getLibraryVersion\_Beta

[Cached]

Windows Linux Mac

pcProx

```
const char * getLibraryVersion_Beta(void)
```

## DESCRIPTION

Read the SDK beta version. This is a 8 (MAXBETASZ) character string such as "Beta".

## PARAMETERS

None

## RETURNS

Returns the pointer to a static buffer within the Library or null.

# getLibraryVersion\_Build

[Cached]

Windows Linux Mac

[Library](#)

```
short getLibraryVersion_Build(void)
```

## DESCRIPTION

Get the version of the library code.  
This function returns the build version with the use of pointers.

## PARAMETERS

None

## RETURNS

Returns build version.

## EXAMPLE

```
short Build = getLibraryVersion_Build()
```

## SEE ALSO

[getLibraryVersion\\_Major\(\)](#)  
[getLibraryVersion\\_Minor\(\)](#)  
[GetLibVersion\(\)](#)

# getLibraryVersion\_Major

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[Library](#)

```
short getLibraryVersion_Major(void)
```

## DESCRIPTION

Get the version of the library code.

This function returns the major version without the use of pointers.

## PARAMETERS

None

## RETURNS

Returns major version

## EXAMPLE

```
short Major = getLibraryVersion_Major()
```

## SEE ALSO

[getLibraryVersion\\_Minor\(\)](#)

[getLibraryVersion\\_Build\(\)](#)

[GetLibVersion\(\)](#)

# getLibraryVersion\_Minor

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[Library](#)

```
short getLibraryVersion_Minor(void)
```

## DESCRIPTION

Get the version of the library code.  
This function returns the minor version with the use of pointers.

## PARAMETERS

None

## RETURNS

Returns minor version.

## EXAMPLE

```
short Minor = getLibraryVersion_Minor()
```

## SEE ALSO

[getLibraryVersion\\_Major\(\)](#)  
[getLibraryVersion\\_Build\(\)](#)  
[GetLibVersion\(\)](#)

# getPartNumberString

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

```
const char * getPartNumberString(void)
```

## DESCRIPTION

Read the device part number string. This is a 24 (MAXPRODUCTNAMESZ) character string such as "MS3-00M1AKU" and null terminated.

## PARAMETERS

None

## RETURNS

Returns the pointer to a static buffer within the Library or null.

## SEE ALSO

[getPartNumberString\\_char\(\)](#)



# getPartNumberString\_char

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)

```
char getPartNumberString_char(short index)
```

## DESCRIPTION

Read the device part number string character by character. This is a 24 (MAXPRODUCTNAMESZ) character string such as "MS0-00M1AKU" and null terminated if less than 24 characters are used. When the index is zero, the serial number is received from the device and buffered by the library. The caller can then get the remainder of the string. Therefore the first call must be with index = 0.

## PARAMETERS

index 0 through 23 inclusive.

## RETURNS

Returns the character of index position of the part number string.

## SEE ALSO

[getPartNumberString\(\)](#)

# getTimeParms\_ExFeatures01

[Cached]

Windows Linux Mac

pcProx

short getTimeParms\_ExFeatures01(void)

## DESCRIPTION

C# interface for safe code. Get TimeParms member's iPad5 value or error code. You must call ReadCfg() first. This function calls GetTimeParms() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsTimeParms or error if GetTimeParms() fails.

# getTimeParms\_iBitStrmTO

[Cached]

Windows Linux Mac

pcProx

```
short getTimeParms_iBitStrmTO()
```

## DESCRIPTION

C# interface for safe code. Get TimeParms member's iBitStrmTO value or error code. You must call ReadCfg() first. This function calls GetTimeParms() internally. Note: Wiegand read times out after this msec time (48msec granularity).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsTimeParms or error if GetTimeParms() fails.

# getTimeParms\_iIDHoldTO

[Cached]

Windows Linux Mac

pcProx

```
short getTimeParms_iIDHoldTO()
```

## DESCRIPTION

C# interface for safe code. Get TimeParms member's iIDHoldTO value or error code. You must call ReadCfg() first. This function calls GetTimeParms() internally. Note: Card ID remains valid for this msec time (48msec granularity).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsTimeParms or error if GetTimeParms() fails.

# getTimeParms\_iDLockOutTm

[Cached]

Windows Linux Mac

pcProx

```
short getTimeParms_iDLockOutTm()
```

## DESCRIPTION

C# interface for safe code. Get TimeParms member's iDLockOutTm value or error code. You must call ReadCfg() first. This function calls GetTimeParms() internally. Note: Squelch repetitive reader reports (usually > 1000) in msec (48msec granularity).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsTimeParms or error if GetTimeParms() fails.

# getTimeParms\_iPad6

[Cached]

Windows Linux Mac

pcProx

`short getTimeParms_iPad6()`

## DESCRIPTION

C# interface for safe code. Get TimeParms member's iPad6 value or error code. You must call ReadCfg() first. This function calls GetTimeParms() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

None

## RETURNS

short value from typedef struct tsTimeParms or error if GetTimeParms() fails.

# getTimeParms\_iTPCfgFlg3

[Cached]

Windows Linux Mac

pcProx

```
short getTimeParms_iTPCfgFlg3()
```

## DESCRIPTION

C# interface for safe code. Get TimeParms member's iTPCfgFlg3 value or error code. You must call ReadCfg() first. This function calls GetTimeParms() internally. Note: It is recommended to use the Flags3 structure and not the bits in this byte. Further Flags not related to Time at all (see tsCfgFlags3).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsTimeParms or error if GetTimeParms() fails.

# getTimeParms\_iUSBKeyPrsTm

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

```
short getTimeParms_iUSBKeyPrsTm()
```

## DESCRIPTION

C# interface for safe code. Get TimeParms member's iUSBKeyPrsTm value or error code. You must call ReadCfg() first. This function calls GetTimeParms() internally. Note: Set USB inter-key 'Press' time in msec (4msec granularity).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsTimeParms or error if GetTimeParms() fails.



# getTimeParms\_iUSBKeyRIsTm

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

```
short getTimeParms_iUSBKeyRIsTm()
```

## DESCRIPTION

C# interface for safe code. Get TimeParms member's iUSBKeyRIsTm value or error code. You must call ReadCfg() first. This function calls GetTimeParms() internally. Note: Set USB inter-key 'Release' time in msec (4msec granularity).

## PARAMETERS

None

## RETURNS

short value from typedef struct tsTimeParms or error if GetTimeParms() fails.

# getVidPidVendorName\_char

[Cached]

Windows Linux Mac

pcProx

```
char getVidPidVendorName_char(short index)
```

## DESCRIPTION

Return the VendorName string for the active device. This comes from the optional pcProxVidPid.txt file. is gathered from the device. The user may then read the rest of the buffered string character by character.

## PARAMETERS

index value 0 through 31 inclusive.  
if the index is out of range the value returned will be zero.

## RETURNS

Returns the character at buffer[index] from the GetVidPidVendorName() function call.

## SEE ALSO

[GetVidPidVendorName\(\)](#)

# pcProxPlusDefaults

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx Plus](#)**BOOL** pcProxPlusDefaults( void )

## DESCRIPTION

Reset pcProx Plus to defaults. This copies the Default settings configurations to the Active settings configurations and Stored Settings. You will need to call ReadCfg() to get the changed settings into your application.

## PARAMETERS

None

## RETURNS

TRUE success / FALSE Fail

## SEE ALSO

[ResetUserDflts\(\)](#)

# pcSwipeClearDataAvailable

[Cached]

Windows Linux Mac

[pcSwipe](#)BSHRT [pcSwipeClearDataAvailable\(void\)](#)

## DESCRIPTION

Clear Data available flag. This clears the data available flag and allows the reader to read cards without waiting for the ten second timeout.

Note:

The user must call ClearDataAvailable after getting the track data with GetTrackData(). Once GetTrackData is called you have ten seconds to safely retrieve the next tracks of data without risking another card swipe overwriting this data.

The card reader is locked out from reading cards after the first call to pcSwipeGetTrackData() until you call ClearDataAvailable() or ten seconds which ever occurs first.

After pcSwipeClearDataAvailable() is called the unit will return to the ready state and the LED will return to the ready state color.

## PARAMETERS

None

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[pcSwipeIsDataAvailable\(\)](#)

# pcSwipeClearOverrun

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcSwipe](#)**BSHRT** `pcSwipeClearOverrun(void)`

## DESCRIPTION

Clear overrun counter to zero.

## PARAMETERS

None

## RETURNS

Returns TRUE on success.

## SEE ALSO

[pcSwipeIsDataAvailable\(\)](#)

# pcSwipeGetBeeper

[Cached]

Windows Linux Mac

[pcSwipe](#)

WORD [pcSwipeGetBeeper](#)(WORD i)

## DESCRIPTION

Get the beeper count for one of the 3 states.  
SEE PCSWIPE\_STATE\_\*

## PARAMETERS

state see #define PCSWIPE\_STATE\_

## RETURNS

0..N for short beeps bit 7 set for long beeps  
0,1,2,3,4,80,0x81,0x82

## SEE ALSO

[pcSwipeSetBeeper\(\)](#)  
[BeepNow\(\)](#)

# pcSwipeGetFieldEnable

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcSwipe](#)

BSHRT pcSwipeGetFieldEnable(WORD field)

## DESCRIPTION

Return the state of the the users field enabled TRUE or disabled FALSE.

## PARAMETERS

user field number 1..11.

## RETURNS

TRUE = Success / FALSE = Failure.

## SEE ALSO

[WriteCfg\(\)](#)

[pcSwipeSetFieldEnable\(\)](#)

# pcSwipeGetFieldIgnoreLRC

[Cached]

Windows Linux Mac

[pcSwipe](#)BSHRT [pcSwipeGetFieldIgnoreLRC\(WORD field\)](#)

## DESCRIPTION

Get the state of Ignore LRC flag for a given user field.

## PARAMETERS

user field number 1..11

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[pcSwipeSetFieldIgnoreLRC\(\)](#)



# pcSwipeGetFieldKeyCount

[\[Cached\]](#)

Windows Linux Mac

[pcSwipe](#)BSHRT [pcSwipeGetFieldKeyCount\(WORD field\)](#)

## DESCRIPTION

Get the number of keystrokes for a given user field.

## PARAMETERS

field number 1..N.

## RETURNS

Key count 0..8 (16)

## SEE ALSO

[pcSwipeGetFieldKeydata\(\)](#)  
[pcSwipeSetFieldKeyCount\(\)](#)

# pcSwipeGetFieldKeydata

[Cached]

Windows Linux Mac

[pcSwipe](#)**WORD** [pcSwipeGetFieldKeydata](#)(WORD field, BSHRT kindex)

## DESCRIPTION

Get the keystroke data for a given field. In USB HID keyboard emulation each key is 2 bytes the code and shift modifiers. Even key index bytes are the scan code and odd key index bytes are the shift modifiers. For ASCII devices, each index represents one key and there are twice as many characters available since each key is one byte.

## PARAMETERS

field number 1..N, key index, value.

## RETURNS

return value of key.

## SEE ALSO

[pcSwipeSetFieldKeydata\(\)](#)

# pcSwipeGetFieldLength

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcSwipe](#)

WORD `pcSwipeGetFieldLength`(WORD field)

## DESCRIPTION

Get the user field length in keystrokes or (serial) characters.

## PARAMETERS

field number 1..N

## RETURNS

length 0..N

## SEE ALSO

[pcSwipeSetFieldLength\(\)](#)

# pcSwipeGetFieldMagField

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcSwipe](#)

BSHRT pcSwipeGetFieldMagField(WORD field)

## DESCRIPTION

Get the mag field of the card data for a given user track.

## PARAMETERS

user field number 1..11

## RETURNS

field within card data

## SEE ALSO

[pcSwipeSetFieldMagField\(\)](#)

# pcSwipeGetFieldOffset

[\[Cached\]](#)

Windows Linux Mac

[pcSwipe](#)WORD `pcSwipeGetFieldOffset(WORD field)`

## DESCRIPTION

Get the field offset in characters on where to start keystroking out data.

## PARAMETERS

user field number 1..11

## RETURNS

offset in characters 0..N

## SEE ALSO

[pcSwipeSetFieldOffset\(\)](#)

# pcSwipeGetFieldShowLRC

[Cached]

Windows Linux Mac

pcSwipe

BSHRT pcSwipeGetFieldShowLRC(WORD field)

## DESCRIPTION

Get the state of the field Show LRC flag.

## PARAMETERS

user field number 1..11

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[WriteCfg\(\)](#)

[pcSwipeSetFieldShowLRC\(\)](#)

# pcSwipeGetFieldShowSepSen

[Cached]

Windows Linux Mac

pcSwipe

BSHRT pcSwipeGetFieldShowSepSen(WORD field)

## DESCRIPTION

Get the field Show Separators and Sentinels state.  
When set the mag data separators and sentinels are keystroked out.  
On a typical ISO standard mag card will have one start one end sentinel and zero or more field separators. When a track has one field no FS will be present.

## PARAMETERS

user field number 1..11, TRUE | FALSE

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[WriteCfg\(\)](#)  
[pcSwipeSetFieldShowSepSen\(\)](#)

# pcSwipeGetFieldSkip

[Cached]

Windows Linux Mac

pcSwipe

BSHRT pcSwipeGetFieldSkip(WORD field)

## DESCRIPTION

Get the field skip state.

Tracks 1,2,3 use user fields 9,10, and 11 respectively.

## PARAMETERS

user field number 1..11

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[WriteCfg\(\)](#)

[pcSwipeSetFieldSkip\(\)](#)



# pcSwipeGetFieldTerm

[Cached]

Windows Linux Mac

[pcSwipe](#)

WORD [pcSwipeGetFieldTerm](#)(WORD field)

## DESCRIPTION

Get the field termination character. The keystroked output will terminate when the length is exceeded or term char is found.

## PARAMETERS

field number 1..N.

## RETURNS

byte 0..255

## SEE ALSO

[pcSwipeSetFieldTerm\(\)](#)

# pcSwipeGetFieldTrack

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcSwipe](#)

WORD `pcSwipeGetFieldTrack`(WORD field)

## DESCRIPTION

Get track number for a given user field.

## PARAMETERS

user field number 1..11

## RETURNS

track number 1..7

## SEE ALSO

[WriteCfg\(\)](#)

[pcSwipeGetFieldIgnoreLRC\(\)](#)

# pcSwipeGetLED

[Cached]

Windows Linux Mac

pcSwipe

WORD pcSwipeGetLED(WORD i)

## DESCRIPTION

Get the LED color for one of the 3 states.  
SEE LED\_STATE\_\* #defines and LED\_COLORS

## PARAMETERS

state see #define LED\_STATE\_

## RETURNS

#define one of LED\_COLOR\_

# pcSwipeGetSystemCardsRead

[\[Traffic\]](#)

Windows Linux Mac

[pcSwipe](#)

DWORD pcSwipeGetSystemCardsRead(void)

## DESCRIPTION

This returns the number of cards read minus one for diagnostic purposes. This is the 32 bit number of cards read less one since the units manufacture date. When the unit is manufactured the count is set to 0xFFFFFFFF (4,294,967,295). There is no way to clear the data. The data has no checksum and is never reset to zero. This counter will roll over!

## PARAMETERS

None

## RETURNS

DWORD 0..4,294,967,295

# pcSwipeGetSystemInternalCount

[\[Traffic\]](#)

Windows Linux Mac

pcSwipe

```
long pcSwipeGetSystemInternalCount(int index)
```

## DESCRIPTION

This returns internal pcSwipe diagnostic counters. An index of 0 returns the number of watchdog resets. An index of 1 returns the number of stack underflows. An index of 2 returns the number of stack overflows. An index of 3 returns the number of times the unit self-corrected its flash memory. If power is removed during flash writes the flash memory checksum is not updated, the next power-up will detect an invalid checksum and self-correct to factory defaults.

## PARAMETERS

None

## RETURNS

DWORD 0..4,294,967,295

# pcSwipeGetSystemUptime

[\[Traffic\]](#)

Windows Linux Mac

[pcSwipe](#)

DWORD pcSwipeGetSystemUptime(void)

## DESCRIPTION

Return system "Up Time" in 4msec ticks. This counts from 1 thru 4,294,967,295 (or about 196-200 days) until the counter rolls over to zero.

Note:

The accuracy of the system uptime is about 1% so this should not be used for long term accuracy. It can be used to determine how long the reader has been up since reset.

## PARAMETERS

None

## RETURNS

DWORD 0x00000000 thru 0xFFFFFFFF

# pcSwipeGetTrackData

[Cached]

Windows Linux Mac

pcSwipe

```
const char * pcSwipeGetTrackData(WORD track, BSHRT toAscii)
```

## DESCRIPTION

Get the data from the track. All data is returned including field separators and sentinels and the LRC.

The data format is:

ERR:COUNT START SENTINEL [[ optional data [FIELDSEP]] ENDSENTINEL LRC

Byte[0] = Bit 7 = Set For LRC Error, Bits 0 thru 6 are the character count.

For a track to be valid there must be a start sentinel and end sentinel; the data is optional and so is the field separator. The LRC is an XOR of all data excluding parity inclusively from SS to ES. The LRC does not have parity.

The data can optionally be converted to ASCII so that it is easier to display. The LRC will also be converted to ASCII.

The first call will put the unit in the 'sending data' state and the LED color will be set as the user defined color for that state.

After pcSwipeClearDataAvailable() is called the unit will return to the 'ready state' and the LED will return to the 'ready state' color.

## PARAMETERS

Track 1,2,3

## RETURNS

BYTE \* to 256 bytes. Buffer of nulls returned on any error.

## SEE ALSO

[pcSwipeClearDataAvailable\(\)](#)

[pcSwipeIsDataAvailable\(\)](#)

[GetActiveID\(\)](#)

[pcSwipeSetActiveID\(\)](#)

# pcSwipeGetTrackData\_BYTE

[Cached]

Windows Linux Mac

[pcSwipe](#)

BYTE [pcSwipeGetTrackData\\_BYTE](#)(WORD track, BSHRT toAscii, WORD index)

## DESCRIPTION

Get the data from the track on byte at a time. All data is returned including field separators and sentinels and the LRC.

The data format is:

ERR:COUNT START SENTINEL [[ optional data [FIELDSEP]] ENDSENTINEL LRC

Byte[0] = Bit 7 = Set For LRC Error, Bits 0 thru 6 are the character count.

For a track to be valid there must be a start sentinel and end sentinel; the data is optional and so is the field separator. The LRC is an XOR of all data excluding parity inclusively from SS to ES. The LRC has its own parity bit.

The data can optionally be converted to ASCII so that it is easier to display.

## PARAMETERS

Track 1,2,3, convert to ascii else leave as raw, index for byte 0..255

## RETURNS

BYTE \* to 256 bytes. Buffer of nulls returned on any error.

## SEE ALSO

[pcSwipeGetTrackData\(\)](#)



# pcSwipeGetTrackEnables

[Cached]

Windows Linux Mac

[pcSwipe](#)WORD [pcSwipeGetTrackEnables\(void\)](#)

## DESCRIPTION

Get which tracks are enabled and will be key stroked out. It is possible to enable none of the tracks, you will not see any keystroke data if no track is enabled. The good LED and good beep are based on the enabled tracks.

## PARAMETERS

None

## RETURNS

Flags with bits 0,1,3 set for enabled tracks.

## SEE ALSO

[pcSwipeSetTrackEnables\(\)](#)

# pcSwipeGetTrackFieldOffset

[Cached]

Windows Linux Mac

[pcSwipe](#)

```
const char * pcSwipeGetTrackFieldOffset(WORD track,  
                                         WORD magField,  
                                         WORD offset,  
                                         WORD len,  
                                         WORD term,  
                                         BSHRT toAscii)
```

## DESCRIPTION

This calls GetTrackData() and parses the desired field and offset up to the the length or termination byte. If the field can not be found then the beginning of the track is returned.

## PARAMETERS

Track 1,2,3, field number 1..N, offset in bytes to skip after finding the field, the length in bytes to return unless the termination byte is found first. ToAscii flag can be set to return data in ASCII format.

## RETURNS

BYTE \* up to 256 bytes. Buffer of nulls returned on any error.

## SEE ALSO

[pcSwipeGetTrackData\(\)](#)

# pcSwipelsDataAvailable

[\[Traffic\]](#)

Windows Linux Mac

[pcSwipe](#)WORD `pcSwipelsDataAvailable(void)`

## DESCRIPTION

Check if data is available on any track. If non zero then GetActiveID or GetTrackData() may be called. The value returned has bit 0 through 3 set for data available on tracks 1 through 3 respectively.

If a card is read while data is available but before GetTrack is called then the device's overrun counter is incremented.

The call must call ClearDataAvailable after getting the track data with GetTrackData(). Once GetTrackData is called you have ten seconds to retrieve the next tracks of data and call ClearDataAvailable().

Note:

The card reader is locked out reading after the first call to GetTrackData() when you call ClearDataAvailable() or ten seconds which ever occurs first.

## PARAMETERS

None

## RETURNS

0 = No Card Data Available.

1 = Track 1 has data

2 = Track 2 has data

3 = Tracks 1 & 2 have data

4 = Track 3 has data

5 = Tracks 1 & 3 have data

6 = Tracks 2 & 3 have data

7 = Tracks 1,2 and 3 have data

## SEE ALSO

[pcSwipeClearDataAvailable\(\)](#)

# pcSwipeSetActiveID

[Cached]

Windows Linux Mac

pcSwipe

BSHRT pcSwipeSetActiveID(WORD track,  
WORD magField,  
WORD offset,  
WORD term)

## DESCRIPTION

Set the area of the magstripe card that GetActiveID should return. This makes the pcSwipe backwards compatible with the pcProx API function Call GetActiveID() / GetActiveID32(). It is recommended to use pcSwipeGetTrackData\_BYTE() for newer projects, since GetActiveID and GetActiveID32 are limited to 8 and 32 bytes.

## PARAMETERS

Track 1,2 or 3, field number, offset within field.

## RETURNS

TRUE = Success / FALSE = Failure.

## SEE ALSO

[GetActiveID\(\)](#)  
[GetActiveID32\(\)](#)  
[pcSwipeIsDataAvailable\(\)](#)  
[pcSwipeGetTrackData\\_BYTE\(\)](#)

# pcSwipeSetBeeper

[Cached]

Windows Linux Mac

pcSwipe

**BSHRT** pcSwipeSetBeeper(WORD i, WORD count, BSHRT longBeep)

## DESCRIPTION

Set the beeper length/count color for one of the 2 states, PCSWIPE\_STATE\_GOOD, PCSWIPE\_STATE\_BAD.

SEE BEEPER\_STATE\_\*

WriteCfg must be called to change the LED color on the device.

## PARAMETERS

Set value = BEEPER\_STATE, count, long beep flag

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[pcSwipeGetBeeper\(\)](#)

[BeepNow\(\)](#)

# pcSwipeSetFieldEnable

[Cached]

Windows Linux Mac

pcSwipe

**BSHRT** pcSwipeSetFieldEnable(WORD field, WORD enable)

## DESCRIPTION

Enable the user field of the active device. Enable fields will keystroke out delimiter data, disabled fields will not.

## PARAMETERS

user field number 1..11, TRUE | FALSE

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[WriteCfg\(\)](#)  
[pcSwipeGetFieldEnable\(\)](#)

# pcSwipeSetFieldIgnoreLRC

[Cached]

Windows Linux Mac

pcSwipe

BSHRT pcSwipeSetFieldIgnoreLRC(WORD field, BSHRT enable)

## DESCRIPTION

Set the state of the fields ignore LRC flag. When set the LRC is not checked for accuracy and the track data will be used even if the LRC is incorrect. If not set the LRC will be checked and fields 8 will be used for track 1 errors and field 9 for track 2, and field 10 for track 3 errors.

## PARAMETERS

user field number 1..11, TRUE | FALSE

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[WriteCfg\(\)](#)  
[pcSwipeGetFieldIgnoreLRC\(\)](#)

# pcSwipeSetFieldKeyCount

[Cached]

Windows Linux Mac

[pcSwipe](#)**BSHRT** [pcSwipeSetFieldKeyCount](#)(WORD field, BYTE nKeys)

## DESCRIPTION

Set the number of keystrokes for a given user field.

## PARAMETERS

field number 1..N, key count 0..8 USB (0..16 Serial)

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[pcSwipeGetFieldKeydata\(\)](#)



# pcSwipeSetFieldKeydata

[Cached]

Windows Linux Mac

[pcSwipe](#)**BSHRT** [pcSwipeSetFieldKeydata](#)(WORD field, BSHRT kindex, BSHRT value)

## DESCRIPTION

Set the keystroke data for a given field. In USB HID keyboard emulation each key is 2 bytes the code and shift modifiers. Even key index bytes are the scan code and odd key index bytes are the shift modifiers. For ASCII devices, each index represents one key and there are twice as many characters available since each key is one byte.

## PARAMETERS

field number 1..N, key index, value.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[pcSwipeGetFieldKeydata\(\)](#)

# pcSwipeSetFieldLength

[Cached]

Windows Linux Mac

[pcSwipe](#)**BSHRT** [pcSwipeSetFieldLength](#)(WORD field, WORD length)

## DESCRIPTION

Set the field length of how many characters will be keystroked out or until termination byte/character is found.

## PARAMETERS

user field number 1..11, length 0..N

## RETURNS

offset in characters 0..N

## SEE ALSO

[pcSwipeGetFieldLength\(\)](#)

# pcSwipeSetFieldMagField

[Cached]

Windows Linux Mac

[pcSwipe](#)

BSHRT pcSwipeSetFieldMagField(WORD field, BSHRT fieldNo)

## DESCRIPTION

Set (select) the field with the track number for a given (user) field.

## PARAMETERS

user field number 1..11, field number as define by FS's on mag track.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[pcSwipeGetFieldMagField\(\)](#)

# pcSwipeSetFieldOffset

[\[Cached\]](#)[Windows](#) [Linux](#) [Mac](#)[pcSwipe](#)

BSHRT pcSwipeSetFieldOffset(WORD field, WORD byteOfs)

## DESCRIPTION

Set the field offset in characters on where to start keystroking out data.

## PARAMETERS

user field number 1..11

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[pcSwipeGetFieldOffset\(\)](#)

# pcSwipeSetFieldShowLRC

[Cached]

Windows Linux Mac

pcSwipe

BSHRT pcSwipeSetFieldShowLRC(WORD field, BSHRT enable)

## DESCRIPTION

Set the field Show LRC, when set the LRC (Logical Redundancy Check) byte will be show in hex. Typically this is off. The LRC byte is an exclusive OR of all the data bytes without parity. The LRC parity is set for the LRC byte itself.

## PARAMETERS

user field number 1..11, TRUE | FALSE

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[WriteCfg\(\)](#)

[pcSwipeGetFieldShowLRC\(\)](#)

# pcSwipeSetFieldShowSepSen

[Cached]

Windows Linux Mac

pcSwipe

**BSHRT** pcSwipeSetFieldShowSepSen(WORD field, BSHRT enable)

## DESCRIPTION

Set the field Show Separators and Sentinels state.

When set the mag data separators and sentinels are keystroked out.

On a typical ISO-7811 standard mag card will have one start one end sentinel and zero or more field separators. When a track has one field no FS will be present.

## PARAMETERS

user field number 1..11, TRUE | FALSE

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[WriteCfg\(\)](#)

[pcSwipeGetFieldShowSepSen\(\)](#)

# pcSwipeSetFieldSkip

[Cached]

Windows Linux Mac

pcSwipe

BSHRT pcSwipeSetFieldSkip(WORD field, BSHRT enable)

## DESCRIPTION

Set the field skip state. This allows the field to be enabled for error processing, but skipped during normal field processing. If track 1 has an error and the ignore LRC is off then field 10 can be used to print the delimiters for the error. Tracks 1,2,3 use user fields 9,10, and 11 respectively.

## PARAMETERS

user field number 1..11, TRUE, FALSE.

## RETURNS

TRUE or FALSE.

## SEE ALSO

[WriteCfg\(\)](#)  
[pcSwipeGetFieldSkip\(\)](#)

# pcSwipeSetFieldTerm

[Cached]

Windows Linux Mac

[pcSwipe](#)**BSHRT** [pcSwipeSetFieldTerm](#)(WORD field, WORD term)

## DESCRIPTION

Set the field termination character. The keystroked output will terminate when the length is exceeded or term char is found.

## PARAMETERS

field number 1..N, term byte.

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[pcSwipeGetFieldTerm\(\)](#)



# pcSwipeSetFieldTrack

[Cached]

Windows Linux Mac

pcSwipe

**BSHRT** pcSwipeSetFieldTrack(WORD field, BSHRT trackNo)

## DESCRIPTION

Set track number for a given user field.

## PARAMETERS

user field number 1..11, track number 1..3

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[WriteCfg\(\)](#)

[pcSwipeGetFieldIgnoreLRC\(\)](#)

# pcSwipeSetLED

[\[Traffic\]](#)

Windows Linux Mac

[pcSwipe](#)**BSHRT** pcSwipeSetLED(WORD i, BSHRT v)

## DESCRIPTION

Set the LED color for one of the 5 states.  
SEE LED\_STATE\_\* #defines and LED\_COLORS  
The LED colors are changed in RAM immediately.

## PARAMETERS

Set value = LED\_VALUE

## RETURNS

TRUE = Success / FALSE = Failure

# pcSwipeSetTrackEnables

[Cached]

Windows Linux Mac

pcSwipe

BSHRT pcSwipeSetTrackEnables(WORD trackEnables)

## DESCRIPTION

Set track enables. To Enable mag-track 1 set bit 0, for track 2 set bit 1, for track 3 bit 2. The Good/Bad LED and Beep are based on what tracks are enabled. The card is considered Good is all the Enabled tracks have a valid LRC. If no tracks are enabled no keystroke data will be output and no Good/Bad LED or Beep will occur. It is advised to keep at least one track enabled.

## PARAMETERS

Flags with bits 0,1,2 set to enable and clear to disable tracks.

## RETURNS

TRUE

## SEE ALSO

[pcSwipeGetTrackEnables\(\)](#)

# quickReadSerialPort\_char

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)**DWORD** quickReadSerialPort\_char(short index)

## DESCRIPTION

C# interface to check for serial data on connected COM port and receive available bytes. This function sets the timeout values to 75msec so it is quicker than the ReadSerialPort() function.

## PARAMETERS

index == -1 zero internal buffer calls readSerialPort() and fills up to 1024 byte internal buffer.  
index == 0..1023 returns character from internal buffer.

## RETURNS

index == -1 returns actual number of characters received.  
else returns character at index from internal buffer.

## SEE ALSO

[readSerialPort\\_char\(\)](#)  
[ReadSerialPort\(\)](#)  
[QuickReadSerialPort\(\)](#)  
[WriteSerialPort\(\)](#)

# readDevCfgFmFile\_char

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** readDevCfgFmFile\_char(short index, char c)

## DESCRIPTION

The C# interface to ReadDevCfgToFile(). The file name is sent one character at a time at the given index, at index 255 the function ReadDevCfgToFile is called with it's return value.

## PARAMETERS

index 0..255. 255 trigger I/O call. When index is 0 internal filename buffer is zeroed out.

## RETURNS

**TRUE / Success file was written @ 255, else 0.254 always sucess**

**FALSE** file may be locked or the path is invalid.

## SEE ALSO

[writeDevCfgToFile\\_char\(\)](#)  
[ReadDevCfgFmFile\(\)](#)  
[WriteDevCfgToFile\(\)](#)

# readSerialPort\_char

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)**DWORD** readSerialPort\_char(short index)

## DESCRIPTION

C# interface to check for serial data on connected COM port and receive available bytes. This uses a two second timeout.

## PARAMETERS

index == -1 zero internal buffer calls readSerialPort() and fills up to 1024 byte internal buffer.  
index == 0..1023 returns character from internal buffer.

## RETURNS

index == -1 returns actual number of characters received.  
else returns character at index from internal buffer.

## SEE ALSO

[ReadSerialPort\(\)](#)  
[quickReadSerialPort\\_char\(\)](#)  
[QuickReadSerialPort\(\)](#)

# rf\_GetDevName

[\[Cached\]](#)

Windows

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

BSHRT rf\_GetDevName(char \*szName)

## DESCRIPTION

Same as GetDevName(). Avoids VB name space problem with GetDevName()

## SEE ALSO

[GetDevName\(\)](#)[getDevName\\_char\(\)](#)

# rf\_GetDevType

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)`short rf_GetDevType(void)`

## DESCRIPTION

Same as GetDevType. Avoids VB name space problem with GetDevType()

## RETURNS

Return PRXDEVTYP\_<name> for USB, Serial or TCP/IP on the Active Device.

## SEE ALSO

[GetDevType\(\)](#)



# setBprRlyCtrl\_bVolatile

[Cached]

Windows Linux Mac

pcProx

BSHRT setBprRlyCtrl\_bVolatile(short value)

## DESCRIPTION

C# interface for safe code. Set BprRlyCtrl member's bVolatile value. This is only available on the OEM board reader. You must call ReadCfg() first. This function calls SetBprRlyCtrl() internally. Note: 0 == commit to EE, 1 == Don't store to EE.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetBprRlyCtrl() if failed or SetBprRlyCtrl().

# setBprRlyCtrl\_iBeeperState

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)**BSHRT** setBprRlyCtrl\_iBeeperState(short value)

## DESCRIPTION

C# interface for safe code. Set BprRlyCtrl member's iBeeperState value. You must call ReadCfg() first. This function calls SetBprRlyCtrl() internally. Note: 0 == Off, 1 == On.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetBprRlyCtrl() if failed or SetBprRlyCtrl().

# setBprRlyCtrl\_iPad0

[Cached]

Windows Linux Mac

pcProx

BSHRT setBprRlyCtrl\_iPad0(short value)

## DESCRIPTION

C# interface for safe code. Set BprRlyCtrl member's iPad0 value. You must call ReadCfg() first. This function calls SetBprRlyCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetBprRlyCtrl() if failed or SetBprRlyCtrl().

# setBprRlyCtrl\_iPad3

[Cached]

Windows Linux Mac

pcProx

BSHRT setBprRlyCtrl\_iPad3(short value)

## DESCRIPTION

C# interface for safe code. Set BprRlyCtrl member's iPad3 value. You must call ReadCfg() first. This function calls SetBprRlyCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetBprRlyCtrl() if failed or SetBprRlyCtrl().

# setBprRlyCtrl\_iPad4

[Cached]

Windows Linux Mac

pcProx

BSHRT setBprRlyCtrl\_iPad4(short value)

## DESCRIPTION

C# interface for safe code. Set BprRlyCtrl member's iPad4 value. You must call ReadCfg() first. This function calls SetBprRlyCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetBprRlyCtrl() if failed or SetBprRlyCtrl().

# setBprRlyCtrl\_iPad5

[Cached]

Windows Linux Mac

pcProx

BSHRT setBprRlyCtrl\_iPad5(short value)

## DESCRIPTION

C# interface for safe code. Set BprRlyCtrl member's iPad5 value. You must call ReadCfg() first. This function calls SetBprRlyCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetBprRlyCtrl() if failed or SetBprRlyCtrl().

# setBprRlyCtrl\_iPad6

[Cached]

Windows Linux Mac

pcProx

BSHRT setBprRlyCtrl\_iPad6(short value)

## DESCRIPTION

C# interface for safe code. Set BprRlyCtrl member's iPad6 value. You must call ReadCfg() first. This function calls SetBprRlyCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetBprRlyCtrl() if failed or SetBprRlyCtrl().

# setBprRlyCtrl\_iRelayState

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setBprRlyCtrl\_iRelayState(short value)

## DESCRIPTION

C# interface for safe code. Set BprRlyCtrl member's iRelayState value. You must call ReadCfg() first. This function calls SetBprRlyCtrl() internally. Note: 0 == Off, 1 == On.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetBprRlyCtrl() if failed or SetBprRlyCtrl().



# setCfgFlags2\_bBeepID

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)

BSHRT setCfgFlags2\_bBeepID(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags2 member's bBeepID value. You must call ReadCfg() first. This function calls SetFlags2() internally. Note: True causes the device to beep when a valid ID is received.

## PARAMETERS

short value to set member TRUE or FALSE.

## RETURNS

BSHRT value from GetFlags2() if failed or SetFlags2().

# setCfgFlags2\_bDspHex

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags2\_bDspHex(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags2 member's bDspHex value.  
You must call ReadCfg() first. This function calls SetFlags2() internally.  
Note: Display ID as ASCII Hex not ASCII decimal.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags2() if failed or SetFlags2().

# setCfgFlags2\_bRevBytes

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags2\_bRevBytes(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags2 member's bRevBytes value. You must call ReadCfg() first. This function calls SetFlags2() internally. Note: Reverse byte order (CSN reader).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags2() if failed or SetFlags2().

# setCfgFlags2\_bRevWiegBits

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags2\_bRevWiegBits(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags2 member's bRevWiegBits value. You must call ReadCfg() first. This function calls SetFlags2() internally. Note: Reverse the Wiegand Rx bits.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags2() if failed or SetFlags2().

# setCfgFlags2\_bUseInvDataF

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags2\_bUseInvDataF(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags2 member's bUseInvDataF value. You must call ReadCfg() first. This function calls SetFlags2() internally. Note: Use the bWiegInvData flag over hardware setting.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags2() if failed or SetFlags2().

# setCfgFlags2\_bUseLeadChrs

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setCfgFlags2\_bUseLeadChrs(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags2 member's bUseLeadChrs value. You must call ReadCfg() first. This function calls SetFlags2() internally. Note: Use leading chars in ID KB send.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags2() if failed or SetFlags2().

# setCfgFlags2\_bWiegInvData

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags2\_bWiegInvData(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags2 member's bWiegInvData value. You must call ReadCfg() first. This function calls SetFlags2() internally. Note: Wiegand data on pins is inverted.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags2() if failed or SetFlags2().

# setCfgFlags2\_isASCIIPresent

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setCfgFlags2\_isASCIIPresent(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags2 member's isASCIIPresent value. You must call ReadCfg() first. This function calls SetFlags2() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags2() if failed or SetFlags2()



# setCfgFlags3\_bLowerCaseHex

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags3\_bLowerCaseHex(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags3 member's bLowerCaseHex value. You must call ReadCfg() first. This function calls SetFlags3() internally. if the flag is set (non zero) then hex output will use lowercase a-z else uppercase A-Z.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags3() if failed or SetFlags3().

# setCfgFlags3\_bNotBootDev

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags3\_bNotBootDev(short value)

## DESCRIPTION

Deprecated function always returns Unsupported API return code.

## PARAMETERS

dummy value

## RETURNS

Unsupported API return code.

# setCfgFlags3\_bPrxProEm

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setCfgFlags3\_bPrxProEm(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags3 member's bPrxProEm value. You must call ReadCfg() first. This function calls SetFlags3() internally. Note: Use HID ProxPro card emulation. See [www.HIDGlobal.com](http://www.HIDGlobal.com) for details.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags3() if failed or SetFlags3()

# setCfgFlags3\_bSndSFFC

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags3\_bSndSFFC(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags3 member's bSndSFFC value. You must call ReadCfg() first. This function calls SetFlags3() internally. Note: 0 = FAC Decimal, 1 = FAC Hex.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags3() if failed or SetFlags3().

# setCfgFlags3\_bSndSFID

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setCfgFlags3\_bSndSFID(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags3 member's bSndSFID value. You must call ReadCfg() first. This function calls SetFlags3() internally. Note: 0 = ID Decimal, 1 = ID Hex.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags3() if failed or SetFlags3().

# setCfgFlags3\_bSndSFON

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setCfgFlags3\_bSndSFON(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags3 member's bSndSFON value. You must call ReadCfg() first. This function calls SetFlags3() internally. Note: Split format ON = 1, old combined scheme = 0.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags3() if failed or SetFlags3().

# setCfgFlags3\_bUse64Bit

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags3\_bUse64Bit(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags3 member's bUse64Bit value. You must call ReadCfg() first. This function calls SetFlags3() internally. Note: 0 = 32-bit, 1 = 64-bit Display Math.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags3() if failed or SetFlags3()

# setCfgFlags3\_bUseNumKP

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)

BSHRT setCfgFlags3\_bUseNumKP(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags3 member's bUseNumKP value. You must call ReadCfg() first. This function calls SetFlags3() internally. When true digit are keystroked from the numeric keypad and not the top horizontal row of digits. The digits are the same but the USB scan codes are different for each keyboard key.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags3() if failed or SetFlags3().



# setCfgFlags\_bFixLenDsp

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags\_bFixLenDsp(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags member's bFixLenDsp value. You must call ReadCfg() first. This function calls SetFlags() internally. Note: Send as fixed length with leading zeros as needed.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags() if failed or SetFlags()

# setCfgFlags\_bFrcBitCntEx

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setCfgFlags\_bFrcBitCntEx(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags member's bFrcBitCntEx value. You must call ReadCfg() first. This function calls SetFlags() internally.

Note: Force Rx'd bit count to be exact to be valid.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags() if failed or SetFlags()

# setCfgFlags\_bHaltKBSnd

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#)

BSHRT setCfgFlags\_bHaltKBSnd(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags member's bHaltKBSnd value. You must call ReadCfg() first. This function calls SetFlags() internally.

Note: Don't Send keys to USB (Get ID mechanism).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags() if failed or SetFlags().

# setCfgFlags\_bNoUseELChar

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setCfgFlags\_bNoUseELChar(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags member's bNoUseELChar value. You must call ReadCfg() first. This function calls SetFlags() internally. Note: Don't use an EndLine char on send (default to ENTER).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags() if failed or SetFlags().

# setCfgFlags\_bSndFac

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags\_bSndFac(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags member's bSndFac value. You must call ReadCfg() first. This function calls SetFlags() internally.

Note: Send the FAC (if stripped from data).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags() if failed or SetFlags().

# setCfgFlags\_bSndOnRx

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags\_bSndOnRx(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags member's bSndOnRx value. You must call ReadCfg() first. This function calls SetFlags() internally.

Note: Send valid ID as soon as it is received (iDLockOutTm timer not used).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags() if failed or SetFlags().

# setCfgFlags\_bStripFac

[Cached]

Windows Linux Mac

pcProx

BSHRT setCfgFlags\_bStripFac(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags member's bStripFac value. You must call ReadCfg() first. This function calls SetFlags() internally. Note: Strip the FAC from the ID (not discarded).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags() if failed or SetFlags().

# setCfgFlags\_bUseDelFac2Id

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setCfgFlags\_bUseDelFac2Id(short value)

## DESCRIPTION

C# interface for safe code. Set CfgFlags member's bUseDelFac2Id value. You must call ReadCfg() first. This function calls SetFlags() internally. Note: Put a delimiter between FAC and ID on send.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetFlags() if failed or SetFlags().



# setIDBitCnts\_iIDBitCnt

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDBitCnts\_iIDBitCnt(short value)

## DESCRIPTION

C# interface for safe code. Set IDBitCnts member's iIDBitCnt value. You must call ReadCfg() first. This function calls SetIDBitCnts() internally.

Note: If bStripFac, this determines bit count of ID and FAC.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDBitCnts() if failed or SetIDBitCnts().

# setIDBitCnts\_iLeadParityBitCnt

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setIDBitCnts\_iLeadParityBitCnt(short value)

## DESCRIPTION

C# interface for safe code. Set IDBitCnts member's iLeadParityBitCnt value. You must call ReadCfg() first. This function calls SetIDBitCnts() internally. Note: Wiegand Leading Parity bit count to be stripped.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDBitCnts() if failed or SetIDBitCnts().

# setIDBitCnts\_iPad4

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDBitCnts\_iPad4(short value)

## DESCRIPTION

C# interface for safe code. Set IDBitCnts member's iPad4 value. You must call ReadCfg() first. This function calls SetIDBitCnts() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDBitCnts() if failed or SetIDBitCnts().

# setIDBitCnts\_iPad5

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDBitCnts\_iPad5(short value)

## DESCRIPTION

C# interface for safe code. Set IDBitCnts member's iPad5 value. You must call ReadCfg() first. This function calls SetIDBitCnts() internally. Names with iPad are padding values for the item byte structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDBitCnts() if failed or SetIDBitCnts().

# setIDBitCnts\_iPad6

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDBitCnts\_iPad6(short value)

## DESCRIPTION

C# interface for safe code. Set IDBitCnts member's iPad6 value. You must call ReadCfg() first. This function calls SetIDBitCnts() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDBitCnts() if failed or SetIDBitCnts().

# setIDBitCnts\_iPad7

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDBitCnts\_iPad7(short value)

## DESCRIPTION

C# interface for safe code. Set IDBitCnts member's iPad7 value. You must call ReadCfg() first. This function calls SetIDBitCnts() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDBitCnts() if failed or SetIDBitCnts().

# setIDBitCnts\_iTotalBitCnt

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDBitCnts\_iTotalBitCnt(short value)

## DESCRIPTION

C# interface for safe code. Set IDBitCnts member's iTotalBitCnt value. You must call ReadCfg() first. This function calls SetIDBitCnts() internally.  
Note: If bFrcBitCntEx, card read (including parity) must match this.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDBitCnts() if failed or SetIDBitCnts().

# setIDBitCnts\_iTrailParityBitCnt

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setIDBitCnts\_iTrailParityBitCnt(short value)

## DESCRIPTION

C# interface for safe code. Set IDBitCnts member's iTrailParityBitCnt value. You must call ReadCfg() first. This function calls SetIDBitCnts() internally.  
Note: Wiegand Trailing Parity bit count to be stripped

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDBitCnts() if failed or SetIDBitCnts()



# setIDDispParms2\_iCrdGnChr0

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms2\_iCrdGnChr0(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms2 member's iCrdGnChr0 value. You must call ReadCfg() first. This function calls SetIDDispParms2() internally. Note: If non-zero, sent when ID goes Invalid on RS-232 readers.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms2() if failed or SetIDDispParms2().

# setIDDispParms2\_iCrdGnChr1

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms2\_iCrdGnChr1(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms2 member's iCrdGnChr1 value. You must call ReadCfg() first. This function calls SetIDDispParms2() internally.

Note: If this and Chr0 non-zero, sent when ID goes Invalid on RS-232 readers.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms2() if failed or SetIDDispParms2()

# setIDDispParms2\_iLeadChr0

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms2\_iLeadChr0(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms2 member's iLeadChr0 value. You must call ReadCfg() first. This function calls SetIDDispParms2() internally. Note: These lead characters are filled in (up to 3).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms2() if failed or SetIDDispParms2().

# setIDDispParms2\_iLeadChr1

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms2\_iLeadChr1(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms2 member's iLeadChr1 value. You must call ReadCfg() first. This function calls SetIDDispParms2() internally.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms2() if failed or SetIDDispParms2().

# setIDDispParms2\_iLeadChr2

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setIDDispParms2\_iLeadChr2(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms2 member's iLeadChr2 value. You must call ReadCfg() first. This function calls SetIDDispParms2() internally.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms2() if failed or SetIDDispParms2().

# setIDDispParms2\_iLeadChrCnt

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setIDDispParms2\_iLeadChrCnt(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms2 member's iLeadChrCnt value. You must call ReadCfg() first. This function calls SetIDDispParms2() internally. Note: If bUseLeadChrs, this contains the lead char count (<=3).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms2() if failed or SetIDDispParms2().

# setIDDispParms2\_iPad6

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms2\_iPad6(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms2 member's iPad6 value. You must call ReadCfg() first. This function calls SetIDDispParms2() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms2() if failed or SetIDDispParms2()

# setIDDispParms2\_iPad7

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms2\_iPad7(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms2 member's iPad7 value. You must call ReadCfg() first. This function calls SetIDDispParms2() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms2() if failed or SetIDDispParms2()



# setIDDispParms3\_iPad4

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms3\_iPad4(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms3 member's iPad4 value. You must call ReadCfg() first. This function calls SetIDDispParms3() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms3() if failed or SetIDDispParms3().

# setIDDispParms3\_iPad5

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms3\_iPad5(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms3 member's iPad5 value. You must call ReadCfg() first. This function calls SetIDDispParms3() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms3() if failed or SetIDDispParms3().

# setIDDispParms3\_iPad6

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms3\_iPad6(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms3 member's iPad6 value. You must call ReadCfg() first. This function calls SetIDDispParms3() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms3() if failed or SetIDDispParms3().

# setIDDispParms3\_iPad7

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms3\_iPad7(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms3 member's iPad7 value. You must call ReadCfg() first. This function calls SetIDDispParms3() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms3() if failed or SetIDDispParms3().

# setIDDispParms3\_iTrailChr0

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms3\_iTrailChr0(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms3 member's iTrailChr0 value  
You must call ReadCfg() first. This function calls SetIDDispParms3() internally.  
Note: These trail characters are filled in (up to 3).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms3() if failed or SetIDDispParms3().

# setIDDispParms3\_iTrailChr1

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms3\_iTrailChr1(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms3 member's iTrailChr1 value. You must call ReadCfg() first. This function calls SetIDDispParms3() internally.  
Note: LeadChrCnt + TrailCheCnt <= 3.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms3() if failed or SetIDDispParms3().

# setIDDispParms3\_iTrailChr2

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setIDDispParms3\_iTrailChr2(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms3 member's iTrailChr2 value. You must call ReadCfg() first. This function calls SetIDDispParms3() internally. Note: LeadChrs have priority.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms3() if failed or SetIDDispParms3().

# setIDDispParms3\_iTrailChrCnt

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setIDDispParms3\_iTrailChrCnt(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms3 member's iTrailChrCnt value. You must call ReadCfg() first. This function calls SetIDDispParms3() internally. Note: This contains the trail char count (<=3)

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms3() if failed or SetIDDispParms3().



# setIDDispParms\_iELDelim

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setIDDispParms\_iELDelim(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms member's iELDelim value. You must call ReadCfg() first. This function calls SetIDDispParms() internally. Note: If NOT bNoUseELChar, this character is sent at end of ID.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms() if failed or SetIDDispParms().

# setIDDispParms\_iExOutputFormat

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms\_iExOutputFormat(short value)

## DESCRIPTION

You must call ReadCfg() first. This function calls GetIDDispParms() internally.

Note : If iExOutputFormat, Reader will output in Extended mode. It is applicable only for Plus Extended readers because these readers have capability to give output in both Conventional and Extended mode.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms() if failed or SetIDDispParms().

# setIDDispParms\_iFACDispLen

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms\_iFACDispLen(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms member's iFACDispLen value. You must call ReadCfg() first. This function calls SetIDDispParms() internally.  
Note: If bFixLenDsp, FAC padded with zeros to this length.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms() if failed or SetIDDispParms().

# setIDDispParms\_iFACIDDelim

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms\_iFACIDDelim(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms member's iFACIDDelim value. You must call ReadCfg() first. This function calls SetIDDispParms() internally. Note: If bStripFac and bSndFac and bUseDelFac2Id, this character is sent between FAC and ID.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms() if failed or SetIDDispParms().

# setIDDispParms\_iIDDispLen

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms\_iIDDispLen(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms member's iIDDispLen value. You must call ReadCfg() first. This function calls SetIDDispParms() internally. Note: If bFixLenDsp, ID padded with zeros to this length.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms() if failed or SetIDDispParms().

# setIDDispParms\_iPad5

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms\_iPad5(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms member's iPad5 value. You must call ReadCfg() first. This function calls SetIDDispParms() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms() if failed or SetIDDispParms().

# setIDDispParms\_iPad6

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms\_iPad6(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms member's iPad6 value. You must call ReadCfg() first. This function calls SetIDDispParms() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms() if failed or SetIDDispParms().

# setIDDispParms\_iPad7

[Cached]

Windows Linux Mac

pcProx

BSHRT setIDDispParms\_iPad7(short value)

## DESCRIPTION

C# interface for safe code. Set IDDispParms member's iPad7 value. You must call ReadCfg() first. This function calls SetIDDispParms() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetIDDispParms() if failed or SetIDDispParms()



# setLEDCtrl\_bAppCtrlsLED

[Cached]

Windows Linux Mac

[pcProx](#) [pcProx](#) [Sonar](#)

BSHRT setLEDCtrl\_bAppCtrlsLED(short value)

## DESCRIPTION

C# interface for safe code. Set LEDCtrl member's bAppCtrlsLED value. You must call ReadCfg() first. This function calls SetLEDCtrl() internally.

For pcProx Sonar this is the Yellow Transition state.

Note: Display LEDs are controlled by user thru this library.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetLEDCtrl() if failed or SetLEDCtrl().

# setLEDCtrl\_bVolatile

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** `setLEDCtrl_bVolatile(short value)`

## DESCRIPTION

C# interface for safe code. Set LEDCtrl member's bVolatile value. You must call ReadCfg() first. This function calls SetLEDCtrl() internally.

Note: 0 == commit to EE, 1 == Don't store to EE

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetLEDCtrl() if failed or SetLEDCtrl().

# setLEDCtrl\_iGrnLEDState

[Cached]

Windows Linux Mac

[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)

BSHRT setLEDCtrl\_iGrnLEDState(short value)

## DESCRIPTION

C# interface for safe code. Set LEDCtrl member's iGrnLEDState value. You must call ReadCfg() first. This function calls SetLEDCtrl() internally.

Note: 0 == Off, 1 == On

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetLEDCtrl() if failed or SetLEDCtrl().

# setLEDCtrl\_iPad3

[Cached]

Windows Linux Mac

pcProx

**BSHRT** setLEDCtrl\_iPad3(short value)

## DESCRIPTION

C# interface for safe code. Set LEDCtrl member's iPad3 value. You must call ReadCfg() first. This function calls SetLEDCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetLEDCtrl() if failed or SetLEDCtrl().

# setLEDCtrl\_iPad4

[Cached]

Windows Linux Mac

pcProx

BSHRT setLEDCtrl\_iPad4(short value)

## DESCRIPTION

C# interface for safe code. Set LEDCtrl member's iPad4 value. You must call ReadCfg() first. This function calls SetLEDCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetLEDCtrl() if failed or SetLEDCtrl().

# setLEDCtrl\_iPad5

[Cached]

Windows Linux Mac

pcProx

BSHRT setLEDCtrl\_iPad5(short value)

## DESCRIPTION

C# interface for safe code. Set LEDCtrl member's iPad5 value. You must call ReadCfg() first. This function calls SetLEDCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetLEDCtrl() if failed or SetLEDCtrl().

# setLEDCtrl\_iPad6

[Cached]

Windows Linux Mac

pcProx

BSHRT setLEDCtrl\_iPad6(short value)

## DESCRIPTION

C# interface for safe code. Set LEDCtrl member's iPad6 value. You must call ReadCfg() first. This function calls SetLEDCtrl() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetLEDCtrl() if failed or SetLEDCtrl().

# setLEDCtrl\_iRedLEDState

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** `setLEDCtrl_iRedLEDState(short value)`

## DESCRIPTION

C# interface for safe code. Set LEDCtrl member's iRedLEDState value. You must call ReadCfg() first. This function calls SetLEDCtrl() internally.

Note: 0 == Off, 1 == On.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetLEDCtrl() if failed or SetLEDCtrl().



# setTimeParms\_ExFeatures01

[Cached]

Windows Linux Mac

pcProx

BSHRT setTimeParms\_ExFeatures01(short value)

## DESCRIPTION

C# interface for safe code. Set TimeParms member's iPad5 value. You must call ReadCfg() first. This function calls SetTimeParms() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetTimeParms() if failed or SetTimeParms().

# setTimeParms\_iBitStrmTO

[Cached]

Windows Linux Mac

pcProx

**BSHRT setTimeParms\_iBitStrmTO(short value)**

## DESCRIPTION

C# interface for safe code. Set TimeParms member's iBitStrmTO value. You must call ReadCfg() first. This function calls SetTimeParms() internally.

Note: Wiegand read times out after this msec time (48msec granularity).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetTimeParms() if failed or SetTimeParms().

# setTimeParms\_ildHoldTO

[Cached]

Windows Linux Mac

pcProx

**BSHRT setTimeParms\_ildHoldTO(short value)**

## DESCRIPTION

C# interface for safe code. Set TimeParms member's ildHoldTO value. You must call ReadCfg() first. This function calls SetTimeParms() internally.

Note: Card ID remains valid for this msec time (48msec granularity).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetTimeParms() if failed or SetTimeParms().

# setTimeParms\_ildLockOutTm

[Cached]

Windows Linux Mac

pcProx

BSHRT setTimeParms\_ildLockOutTm(short value)

## DESCRIPTION

C# interface for safe code. Set TimeParms member's ildLockOutTm value. You must call ReadCfg() first. This function calls SetTimeParms() internally. Note: Squelch repetitive reader reports (usually < 1000) in msec (48msec granularity).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetTimeParms() if failed or SetTimeParms().

# setTimeParms\_iPad6

[Cached]

Windows Linux Mac

pcProx

BSHRT setTimeParms\_iPad6(short value)

## DESCRIPTION

C# interface for safe code. Set TimeParms member's iPad6 value. You must call ReadCfg() first. This function calls SetTimeParms() internally. Names with iPad are padding values for the eight item structure, they are available for completeness and for future firmware expansion.

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetTimeParms() if failed or SetTimeParms().

# setTimeParms\_iTPCfgFlg3

[Cached]

Windows Linux Mac

pcProx

BSHRT setTimeParms\_iTPCfgFlg3(short value)

## DESCRIPTION

C# interface for safe code. Set TimeParms member's iTPCfgFlg3 value. You must call ReadCfg() first. This function calls SetTimeParms() internally.

Note: It is recommended to use the Flags3 structure and not the bits in this byte. Further Flags not related to Time at all (see tsCfgFlags3).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetTimeParms() if failed or SetTimeParms().

# setTimeParms\_iUSBKeyPrsTm

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** setTimeParms\_iUSBKeyPrsTm(short value)

## DESCRIPTION

C# interface for safe code. Set TimeParms member's iUSBKeyPrsTm value. You must call ReadCfg() first. This function calls SetTimeParms() internally.

Note: Set USB inter-key 'Press' time in msec (4msec granularity).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetTimeParms() if failed or SetTimeParms().

# setTimeParms\_iUSBKeyRIsTm

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** setTimeParms\_iUSBKeyRIsTm(short value)

## DESCRIPTION

C# interface for safe code. Set TimeParms member's iUSBKeyRIsTm value. You must call ReadCfg() first. This function calls SetTimeParms() internally.  
Note: Set USB inter-key 'Release' time in msec (4msec granularity).

## PARAMETERS

Set member name to value of short.

## RETURNS

BSHRT value from GetTimeParms() if failed or SetTimeParms().



# usbConnect

[\[Traffic\]](#)[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#)BSHRT `usbConnect(void)`

## DESCRIPTION

A C# interface to `USBConnect()` without taking a pointer to a long. Set the types of device to search for when connecting using `USBConnect()`. Searching for serial devices is much slower than search for USB only devices. A wrapper on `USBConnect`. It returns the same value. It does not require a pointer to return the DeviceID. Use `GetDID()` to retrieve the firmware version. Serial port range defaults are 1..8 at powerup. This API is thread-safe.

## PARAMETERS

None

## RETURNS

TRUE = Success / FALSE = Failure

## SEE ALSO

[USBConnect\(\)](#)  
[ComConnect\(\)](#)

# writeDevCfgToFile\_char

[Cached]

[Windows](#) [Linux](#) [Mac](#)[pcProx](#) [pcSwipe](#) [pcProx](#) [Sonar](#)**BSHRT** writeDevCfgToFile\_char(short index, char c)

## DESCRIPTION

The C# interface to WriteDevCfgToFile(). The file name is sent one character at a time at the given index, at index 255 the function WriteDevCfgToFile is called with it's return value.

## PARAMETERS

index 0..255. 255 trigger I/O call. When index is 0 internal filename buffer is zeroed out.

## RETURNS

**TRUE / Success file was written @ 255, else 0.254 always sucess**

**FALSE** file may be locked or the path is invalid.

## SEE ALSO

[readDevCfgFmFile\\_char\(\)](#)  
[ReadDevCfgFmFile\(\)](#)  
[WriteDevCfgToFile\(\)](#)

## Updates

---

From time to time the library will have new features or functions added to it. Check the website for updates.