BioControl
Technology for biology
www.biocontrol.no

iMAN

D00G1.1 iMan Manual for System Integrators

January 2013

This documentation applies to the product iMan manufactured by BioControl, Norway. Documentation version 1.1 completely replaces previous versions.

Modifications since last documentation version:

- none – version 1.0
- new functions in "RFD1:" driver, new interface in C# dll

Note:
All information in this document describes product and details 'as is'. BioControl cannot be held liable for (consequences of) incorrect or missing information in this document.
Check our website for latest version of this document and information on this product.

1 Content

2 General description

iMan has several interfaces and drivers that can be used by System Integrators. This document describes how to use these features from programmer point of view.
The iMan interfaces and features described in this manual are:

- RS232 port on D-SUB25 iMan connector
- RS485 port on side connectors
- RGB Led driver
- RFID functionality

3 Using RS232 interface

RS232 interface physically is accessible through D-SUB25 connector using iMan RS232 cable or docking block (optional).

This interface from the operating system side is accessible through «COM2:» logical name. Note that this is 4-wire interface:

- Two data lines Rx and Tx
- Two control lines CTS and RTS

To use this interface in C/C++ languages use CreateFile function to open the port and ReadFile, WriteFile functions to read and write to the port. Please refer to MSDN documentation for more details about serial port usage.

4 Using RS485 interface

RS485 interface is physically available on side connectors on the iMan or through docking block (optional). The side interfaces have 4 pins:

- A and B data lines
- Power (this pin is connected to the iMan's battery)
- GND

This interface is accessible through «COM8:» logical name in the operating system.
To use this interface in C/C++ languages use CreateFile function to open the port and ReadFile, WriteFile functions to read and write to the port. Please refer to MSDN documentation for more details.

IMPORTANT: To set RS485 driver in proper mode write at least one 'junk' byte to the port using WriteFile function
IMPORTANT: RS485 interface is not accessible when RFID driver is operating. Please unload the RFID driver first (Turn OFF button in RFID Manager) before accessing «COM8:»

5 Using LED driver

iMan has RGB LED on the front panel. By the default it is used by the charger to indicate status of battery charging. In addition it is also used by the RFID software to indicate tag reading; also it indicates successful or not successful tag reading. It is possible to the user to use this LED to own purposes. From the operating system side LED is accessible through «LED1:» port.

First, open this port using CreateFile function. Then write one byte to the port to set LED colour:

Value	Colour
0	Off
1	Red
2	Green
3	Yellow
4	Blue
5	Magenta
6	Cyan
7	White

IMPORTANT: LED driver is not accessible when readers driver is operating. Please unload the readers driver first (Turn OFF button in Readers Manager) before accessing «LED1:» port.

6 Using Readers functionality in iMan

6.1 Readers Functionality overview

Readers in the iMan are connected through RS485 port. Readers in the iMan system are autonomic devices with own microcontroller. All readers are accessible through one instance of readers driver.

Reader software structure in the iMan is as follows:

- Readers communication protocol (refer Reader documentation for more details) using «COM8:»
- Readers driver with notification functionality
- Readers Manager
- Readers COM port
- iMan dll for C#

There are several methods to use readers functionality in the iMan. Choose of the right method is application dependent. The most 'low level' method is to implement reader protocol in own application. It gives the most flexibility but requires implementation of all readers functions. The 'highest level' method is to use WEDGE functionality, which sends TAG number to the keyboard buffer, thus end application can use readers functionality without any programming work.

6.2 Using Reader protocol

Please refer to the reader manual to get more information about reader protocol. In this method it is possible to use LED driver to add specific user notifications. It is also possible to use other RS485 devices with the reader (please refer to reader manual for correct COM port settings).

Power management is automatically handled by the reader. Reader goes into sleep mode after 10 minutes of inactivity. Thus it is reasonable to send 'Power On' command before other commands to ensure that the reader is in active mode.

6.3 Using readers driver port «RFD1:»

Readers driver functionality is accessible through «RFD1:» port. Open this port to access readers functionality. Readers driver accepts following commands:

- **PAR** – get reader parameters
- **NTF** – set notifications
- **KEY** – read tag
- **PWR** – set reader active or inactive mode
- **SND** – set notification sounds
- **PRG** – update reader's firmware
- **REL** – reload information about readers from registry
- **VER** – get driver's version
- **TUN** – run autotuning of reader

Syntax of each command (except Key command) is as following:

XXXaa???,

where *XXX* is three-letter command code, *aa* is reader address (in HEX) on RS485 bus, *???* is optional command information (see command description for more details).

NOTE: ISO reader default address is *0x01*. If iMan is supplied with BarCode reader, default BarCore reader address is *0xB0*

After each command the driver will answer with:

XXXaa:???

where *XXX* is the last command sent to the driver, *aa* is reader address and *???* is result of the command. The result *???* can be:

OK – command was successful (instead 'OK' some commands return additional information, see command description for more details)

ERR – command failed – this could mean wrong parameter or communication problem with the reader.

If command is not recognized by the driver it returns:

?????:ERR

After writing the message You can use ReadFile() function to get response:

```
WCHAR inBuffer[255];
DWORD bwritten;

ReadFile(hReaderDriver,
      (LPVOID)inBuffer,255*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

`hReaderDriver` is handle to the driver returned by the CreateFile() function:

```
HANDLE hReaderDriver =
      CreateFile(L"RFD1:",GENERIC_READ|GENERIC_WRITE,FILE_SHARE_READ|FILE_SHAR
      E_WRITE,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
```

6.3.1 PAR Message

"PAR" message returns parameters of selected reader. To send this message define string «PARaa» and pass this to the «RFD1:» device, *aa* is reader address. Here is an example:

```
WriteFile(hReaderDriver,
      L"PAR01",5*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

The returned additional message information from the driver will be in format:
"vX.Y.Z AxTy"
`vX.Y.Z` is the version of the reader firmware, X is version major number, Y is minor version number and Z is revision.
`AxTy` – y and x are reader parameters. The proper ranges for these parameters are from 1 to 14 for y and from 100 to 200 for x.

NOTE: All strings used in this protocol are wide characters!

6.3.2 NTF Message

"NTF" message allows to change notifications for selected reader. To send this message pass the string «NTFaaxxxxx» to the «RFD1:» device. *XXXXX* are notifications options (see below), *aa* is reader address. Here is an example how to turn on all notifications:

```
WriteFile(hReaderDriver,
      L"NTF01WSLPI",10*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

Where:
W – WEDGE on/off
S - sound on/off
L - use LED
P - popup window
I - taskbar icon

To turn off all notifications send following string to the driver:

```
WriteFile(hReaderDriver,
      L"NTF01",5*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

To turn on only WEDGE and sounds send the following string:

```
WriteFile(hReaderDriver,
      L"NTF01WS",7*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

You are allowed to use any combination of notification options to manage reading notifications.
The driver will answer with **OK** if command was succesfull or **ERR** if command failed.

6.3.3 KEY Message

"KEY" message is used to read TAG ID or BarCode number. To send this message pass «KEYX» string to the «RFD1:» device.
X is parameter which selects the reader:

R - reader that is triggered by READ key
1 - reader that is triggered by F1 key
2 - reader that is triggered by F2 key
3 - reader that is triggered by F3 key
4 - reader that is triggered by F4 key
A - address of the reader, the *A* option must be followed by two digits containing HEX address of the reader on RS485 bus.

Following command will cause reader with READ button assigned to read the TAG or BarCode:

```
WriteFile(hReaderDriver,
  L"KEYR",4*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

If You don't want to use triggering key or the triggering key is not set (refer to iMan start-up guide to see how to set triggering key for the reader) You can pass address of the reader:

```
WriteFile(hReaderDriver,
  L"KEYA01",6*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

Driver will answer with: "KEY01:yyyyy" where yyyyy is the RFID or BarCode number if reading wass successful.

6.3.4 PWR Message

"PWR" message is used to set reader in active or inactive (power saving) mode. To send this message define string «PWRaay» and pass this to the «RFD1:» device; *aa* is reader address, Y is the power state:
1 – full power,
0 – power saving mode.

To set active mode send:

```
WriteFile(hReaderDriver,
   L"PWR011",6*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

To set inactive mode (sleep mode) send:

```
WriteFile(hReaderDriver,
   L"PWR010",6*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

The driver will answer with **OK** if command was succesfull or **ERR** if command failed.
NOTE: Readers driver is managind readers power consumption automatically, so this command doesn't have to be used in order to save battery.

6.3.5 SND Message

"SND" message is used to change sound notifications for selected reader. To send this message define string «SNDaaxyyy» and pass this to the «RFD1:» device, *aa* is reader address.

The x parameter represents type of sound notification:
S – Start reading sound
F – Reading failed sound
E – Reading TAG successful sound

The yyy parameter is the path to the .wav file. Example:

```
WriteFile(hReaderDriver,
   L"SNDS\MyDevice\start.wav",23*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

The driver will answer with **OK** if command was succesfull or **ERR** if command failed.

6.3.6 PRG Message

"PRG" message is used to update firmware of selected reader. To send this message define string «PRGaayyy» and pass this to the «RFD1:» device, *aa* is reader address, yyy is a path to programming file. Example:

```
 WriteFile(hReaderDriver,
   L"PRG01\MyDevice\ISO_Reader_0_3_4.hex",35*sizeof(WCHAR),(LPDWORD)&bwriten,N
ULL);
```

The driver will answer with **OK** if command was succesfull or **ERR** if command failed.
NOTE: Reader firmware reprogramming may take up to 5 minutes.

6.3.7 REL Message

"REL" message is used to reload readers settings from the registry. It can be useful if some changes were made in the registry while driver was working – this command will apply this changes to the driver. To send this message define string «REL00» and pass this to the «RFD1:» device:

```
WriteFile(hReaderDriver,
   L"REL00",5*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

The driver will answer with **OK** if command was succesfull or **ERR** if command failed.

6.3.8 VER Message

"VER" message is used to get the version of the driver. To send this message define string «VER00» and pass this to the «RFD1:» device:

```
WriteFile(hReaderDriver,
   L"VER00",5*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

The driver will return version of the driver in the additional message information field in the format:"v.x.y.z" where *x* is major version, *y* is minor version and *z* is revision number.

6.3.9 TUN Message

"TUN" message can be used to force the driver to go into autotuning mode. To send this message define string «TUNaa» and pass this to the «RFD1:» device, *aa* is reader address:

```
WriteFile(hReaderDriver,
   L"TUN01",5*sizeof(WCHAR),(LPDWORD)&bwriten,NULL);
```

The driver will answer with **OK** if command was succesfull or **ERR** if command failed.
NOTE1: Autotuning mode is available in ISO type readers only
NOTE2: Autotuning is performed automatically by the driver, this command is optional and should be used only in special cases.

6.4 Using Reder COM port driver

Reader COM port is available in the operating system as «COM6:» port. To use this port run Reader Manager Application and turn on the driver. Then in the application open port «COM6:» and listen to the port. When read button is pressed on the iMan transponder number will appear on this port. This port should be used as standard serial port. Baud rate, parity, stop bits and other settings are not relevant, you can use any settings to get this port working.

6.5 Using WEDGE functionality

WEDGE functionality can be set in Readers Manager Application. It sends tag number to the keyboard buffer. Because this functionality sends keys to the active window please put focus on the desired window to receive keystrokes. WEDGE functionality can send <CR><LF> keys after each tag, please refer to iMan start-up manual for more details how to setup WEDGE options.

6.6 Readers Registry settings

The following registry settings are example of readers configuration in the iMan:

```
[HKEY_LOCAL_MACHINE\BioControl\RfidDriver1]
      "Dll"="RfidDriver.dll"
      "Order"=dword:4
      "Prefix"="RFD"
      "Index" =dword:1
      "Flags"=dword:10
      "iManAddress"=dword:f8
      "RunAtStartup"=dword:0
      "UseVirtualComPort"=dword:1
```

`"RunAtStartup"` – set to `1` if driver should start autamatically with the system or to `0` to start driver manually from Readers Manager application.
`"UseVirtualComPort"` - set to `1` if You need to use <COM6:> Reader port (see 6.4)
NOTE: DO NOT CHANGE OTHER SETTINGS IN THIS REGISTRY KEY

You can define up to 16 readers in iMan system. Please note: each reader has to have different address. Each reader has its own configuration key on the registry. The key names have to be consecutive integers starting from 1. Example of registry settings for two readers:

```
[HKEY_LOCAL_MACHINE\BioControl\Readers\1]
      "ReaderAddress"=dword:1
      "DeviceType"=dword:0
      "UseNotificationLed"=dword:1
      "UseNotificationPopup"=dword:0
      "UseNotificationWedge"=dword:1
```

```
        "UseNotificationWedgeLC"=dword:1
        "UseNotificationIcon"=dword:1
        "UseNotificationSounds"=dword:1
        "NotificationWedgeLastChars"=dword:16
        "NotificationWedgePutCR"=dword:1
        "ReadKey"=dword:2A
        "StartSoundFile"=""
        "FailSoundFile"=""
        "EidSoundFile"=""

[HKEY_LOCAL_MACHINE\BioControl\Readers\2]
        "ReaderAddress"=dword:b0
        "DeviceType"=dword:0
        "UseNotificationLed"=dword:1
        "UseNotificationPopup"=dword:0
        "UseNotificationWedge"=dword:1
        "UseNotificationWedgeLC"=dword:0
        "UseNotificationIcon"=dword:1
        "UseNotificationSounds"=dword:1
        "NotificationWedgeLastChars"=dword:30
        "NotificationWedgePutCR"=dword:0
        "ReadKey"=dword:73
        "StartSoundFile"=""
        "FailSoundFile"=""
        "EidSoundFile"=""
```

The following values can be set in the reader registry key:

`"ReaderAddress"` – set this value to match physical address of the reader
`"DeviceType"` – set to 0 for ISO reader type or to 1 for BarCode Reader

The following entries are to chose which notifications should be used with particular reader. Set to 1 to use the notification or to 0 if the notification is not needed:

```
"UseNotificationLed"
"UseNotificationPopup
"UseNotificationWedge
"UseNotificationIcon
"UseNotificationSounds
```

If You do not need all the digits in the RFID od BarCode number set following key to 1:

```
"UseNotificationWedgeLC"=dword:1
```

And define how many last characters You want to use:

```
"NotificationWedgeLastChars"=dword:5
```

You can also decide if <CR> character should follow RFID or BarCode number. If Yes, set the following value to 1:

```
"NotificationWedgePutCR"=dword:1
```

There are 5 keys that can trigger reader to start reading: Read Key (0x2a), F1 Key (0x73) , F2 Key (0x74) , F3 Key (0x75) , F4 Key (0x76). Set the following registry value if You to want to use one of this keys to read RFID or BarCode number:

```
"ReadKey"=dword:73
```

NOTE: DO NOT USE THE SAME TRIGERING KEY FOR MORE THAN ONE READER

Sound files that are to be played when reading starts, reading failed or reading succeded are defined in following entries:

```
"StartSoundFile"=""
"FailSoundFile"=""
"EidSoundFile"=""
```

In the iMan default registry settings for the readers are:

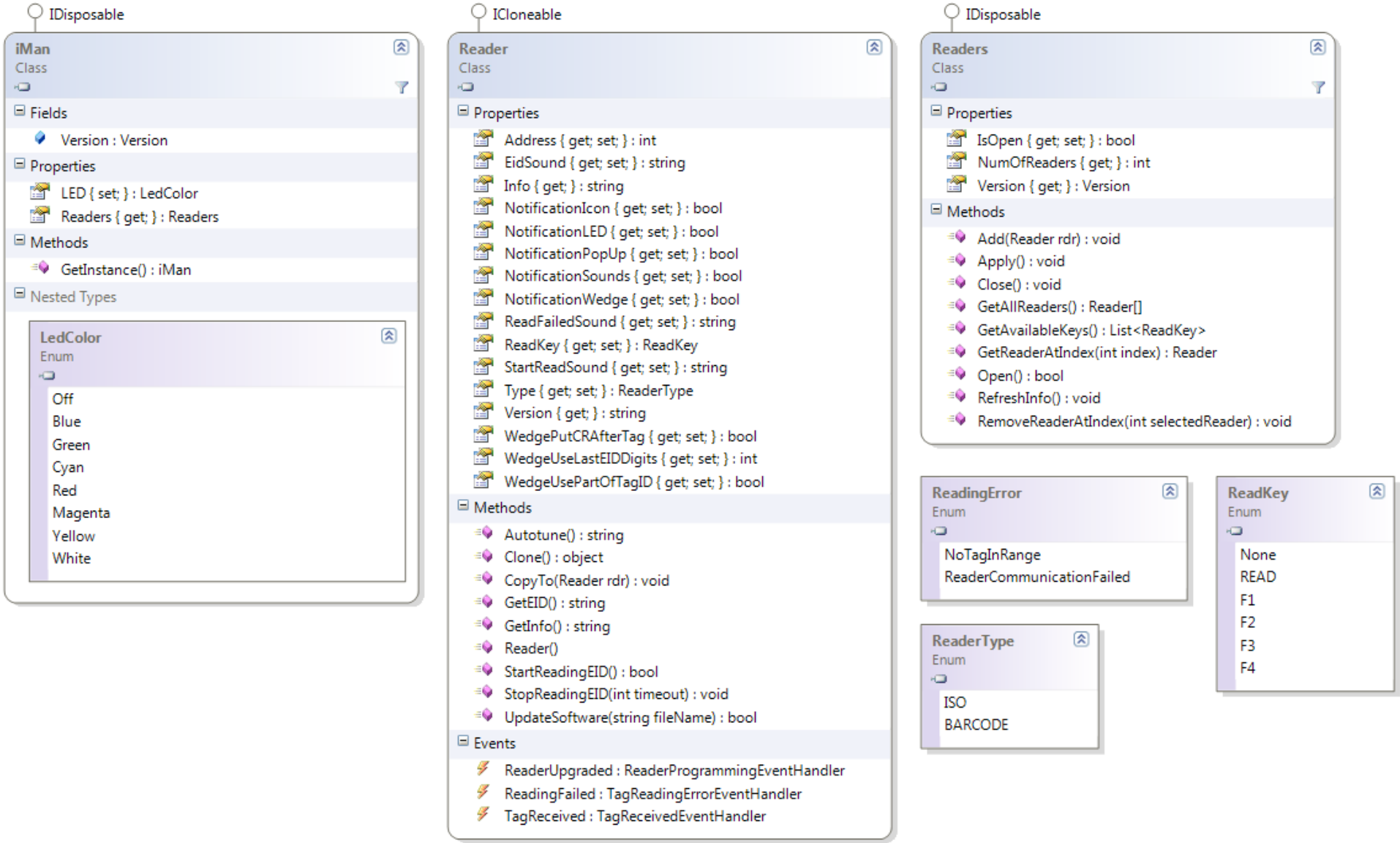
```
[HKEY_LOCAL_MACHINE\BioControl\RfidDriver1]
      "Dll"="RfidDriver.dll"
      "Order"=dword:4
      "Prefix"="RFD"
      "Index" =dword:1
      "Flags"=dword:10
      "iManAddress"=dword:f8
      "RunAtStartup"=dword:0
      "UseVirtualComPort"=dword:1

[HKEY_LOCAL_MACHINE\BioControl\Readers\1]
      "ReaderAddress"=dword:1
      "DeviceType"=dword:0
      "UseNotificationLed"=dword:1
      "UseNotificationPopup"=dword:0
      "UseNotificationWedge"=dword:1
      "UseNotificationWedgeLC"=dword:1
      "UseNotificationIcon"=dword:1
      "UseNotificationSounds"=dword:1
      "NotificationWedgeLastChars"=dword:16
      "NotificationWedgePutCR"=dword:1
      "ReadKey"=dword:2A
      "StartSoundFile"=""
      "FailSoundFile"=""
      "EidSoundFile"=""
```

7 Using iMan library in C#

7.1 iMan library class overview

The structure of classes in the iMan library is showr on the following picrute:


IDisposable
iMan
Class
Fields
Version : Version
Properties
LED { set; } : LedColor
Readers { get; } : Readers
Methods
GetInstance() : iMan
Nested Types
LedColor
Enum
Off
Blue
Green
Cyan
Red
Magenta
Yellow
White
ICloneable
Reader
Class
Properties
Address { get; set; } : int
EidSound { get; set; } : string
Info { get; } : string
NotificationIcon { get; set; } : bool
NotificationLED { get; set; } : bool
NotificationPopUp { get; set; } : bool
NotificationSounds { get; set; } : bool
NotificationWedge { get; set; } : bool
ReadFailedSound { get; set; } : string
ReadKey { get; set; } : ReadKey
StartReadSound { get; set; } : string
Type { get; set; } : ReaderType
Version { get; } : string
WedgePutCRAfterTag { get; set; } : bool
WedgeUseLastEIDDigits { get; set; } : int
WedgeUsePartOfTagID { get; set; } : bool
Methods
Autotune() : string
Clone() : object
CopyTo(Reader rdr) : void
GetEID() : string
GetInfo() : string
Reader()
StartReadingEID() : bool
StopReadingEID(int timeout) : void
UpdateSoftware(string fileName) : bool
Events
ReaderUpgraded : ReaderProgrammingEventHandler
ReadingFailed : TagReadingErrorEventHandler
TagReceived : TagReceivedEventHandler
IDisposable
Readers
Class
Properties
IsOpen { get; set; } : bool
NumOfReaders { get; } : int
Version { get; } : Version
Methods
Add(Reader rdr) : void
Apply() : void
Close() : void
GetAllReaders() : Reader[]
GetAvailableKeys() : List<ReadKey>
GetReaderAtIndex(int index) : Reader
Open() : bool
RefreshInfo() : void
RemoveReaderAtIndex(int selectedReader) : void
ReadingError
Enum
NoTagInRange
ReaderCommunicationFailed
ReadKey
Enum
None
READ
F1
F2
F3
F4
ReaderType
Enum
ISO
BARCODE

7.2 BioControl.iMan Class Reference

7.2.1 Member Functions

static iMan BioControl.iMan.GetInstance ()[static]
Gets instance of the iMan singleton class

7.2.2 Properties

LedColor BioControl.iMan.LED[set]
Sets color of LED or turns it off

Version BioControl.iMan.Version [get]
iMan library version

Readers BioControl.iMan.Readers [get]
Object containing readers registered in the system

7.3 BioControl.Readers Class Reference

7.3.1 Member Functions

void BioControl.Readers.Add (Reader *rdr*)
Adds a reader to the system

void BioControl.Readers.Apply ()
Forces Readers driver to reload readers configuration from the registry

void BioControl.Readers.Close ()
Closes communication with the Readers driver

Reader[] BioControl.Readers.GetAllReaders ()
Gets collection of readers registered in the system

List<ReadKey> GetAvailableKeys()
Returns list of available keys that can be used as reading trigger

Reader BioControl.Readers.GetReaderAtIndex (int *index*)
Returns Reader object at specified index

bool BioControl.Readers.Open ()
Opens communication with the readers driver

void BioControl.Readers.RefreshInfo ()
Forces driver to refresh readers status of all readers

void BioControl.Readers.RemoveReaderAtIndex (int *selectedReader*)
Removes reader at the given index from the system

7.3.2 Properties

bool BioControl.Readers.IsOpen[get]
True if communication with the driver is established, false otherwise

int BioControl.Readers.NumOfReaders[get]
Number of readers registered in the system

Version BioControl.Readers.Version[get]
Returns version of Readers driver

7.4 BioControl.Reader Class Reference

7.4.1 Member Functions

string BioControl.Reader.Autotune ()
Performs autotuning of the antenna. Works only with ISO readers

object BioControl.Reader.Clone ()
Clones the Reader object

void BioControl.Reader.CopyTo (Reader *rdr*)
Copies reader object status to another reader object

string BioControl.Reader.GetEID ()
Returns TAG ID. Function blocks until answer from reader is received

string BioControl.Reader.GetInfo ()
Gets firmware version and reader parameters string

bool BioControl.Reader.StartReadingEID ()
Starts reading a TAG ID. This is not blocking function

void BioControl.Reader.StopReadingEID (int *timeout*)
Stops reading a TAG ID.

bool UpdateSoftware(string FileName)
Updates reader's firmware. Non-blocking function

7.4.2 Property Documentation

int BioControl.Reader.Address[get], [set]
Defines reader address on the RS485 bus

string BioControl.Reader.EidSound[get], [set]
Defines sound to be played when Tag ID is received

string BioControl.Reader.Info[get]
Gets Reader information field

bool BioControl.Reader.NotificationIcon[get], [set]
Defines if tray icon should be used as notification method

bool BioControl.Reader.NotificationLED[get], [set]
Defines if LED should be used as notification method

bool BioControl.Reader.NotificationPopUp[get], [set]
Defines if pop-up window should be used as notification method

bool BioControl.Reader.NotificationSounds[get], [set]
Defines if sounds should be used as notification method

bool BioControl.Reader.NotificationWedge[get], [set]
Defines if WEDGE notification should be used

string BioControl.Reader.ReadFailedSound[get], [set]
Defines sound to be played when reading failed

ReadKey BioControl.Reader.ReadKey[get], [set]
Gets or sets key to trigger reading

string BioControl.Reader.StartReadSound[get], [set]
Defines sound to be played when reading is started

ReaderType BioControl.Reader.Type[get], [set]
Gets or sets reader type

string BioControl.Reader.Version[get]
Gets reader firmvare version

bool BioControl.Reader.WedgePutCRAfterTag[get], [set]
Defines how many last digits of the tag should be passed by the driver. This property is used only if WedgeUsePartOfTagID is set to true

int BioControl.Reader.WedgeUseLastEIDDigits[get], [set]
Returns version of reader firmware

bool BioControl.Reader.WedgeUsePartOfTagID[get], [set]
Defines if only part of the TAG number should be passed by the driver

7.4.3 Events

TagReadingErrorEventHandler BioControl.Reader.ReadingFailed
This event is triggered when reading TAG ID fails

TagReceivedEventHandler BioControl.Reader.TagReceived
This event is triggered when TAG ID is received from the reader

ReaderProgrammingEventHandler BioControl.Reader.ReaderUpgraded
This event is triggered when upgrading reader's firmawe is finished

7.5 Example

7.5.1 Example code - reader

To set-up reader in Your application You can use following code:

```
//declare iMan and Reader objects
iMan iMan;
Reader reader = null;

// get instance of iMan object and open communication with readers driver
iMan = iMan.GetInstance();
iMan.Readers.Open();

// if You have standard iMan setup with one ISO reader select this reader
reader = iMan.Readers.GetReaderAtIndex(0);

// add event handlers
reader.TagReceived += new TagReceivedEventHandler(TagReceived);
reader.ReadingFailed += new TagReadingErrorEventHandler(ReadingFailed);
```

To start reading RFID or barcode use following code:

```
reader.StartReadingEID();
```

Add the code in event handlers

```
void TagReceived(Reader sender, string tag)
{
      // add Your code here
}


void ReadingFailed(Reader sender, ReadingError error)
{
      // add Your code here
}
```

7.5.2 Example code – LED

```
//declare iMan object
iMan iMan;

// get instance of iMan object
iMan = iMan.GetInstance();

// set green LED color on iMan
iMan.LED = iMan.LedColor.Green;
```