BIS C-870 handheld driver for .Net
User's Manual

Table of Content

1 Introduction

BIS_C_WinCE_dotNet.dll is a wrapper *ClassLibrary* for BIS_C_WinCE_DLL.dll driver. It provides a reference to .Net solutions, thus user can simply use his existing driver of BIS C-870 module.
The wrapper is suitable *only* for Zebra Technologies (PSION Teklogix) or equal (Pocket PC) mobile devices, which have *Compact Framework 2.0* (CF 2.0) or 3.5 (CF 3.5) installed. *The driver runs only under Windows CE 6.0, higher versions are not tested.*

As the class library wraps the BIS C-870 driver, it has the same implemented Balluff Dialog function set:

- *ReadData* – read data from data carrier
- *WriteData* – write data to data carrier
- *WritePattern* – write a constant character to the data carrier
- *InitDataCarrier* – init data carrier for CRC16
- *InitReader* – set configuration of the processor
- *ResetReader* – reset read/write processor
- *StopCommand* – stop actual running command
- *MemoryLock* – locks the desired area of the code tag

There are other functions which are not part of the Balluff Dialog protocol, but can be called via the driver:

- *USBPortState* – gets the state of the COM port (open, closed, exist etc.)
- *OpenUSBPort* – opens the port
- *CloseUSBPort* - closes the port
- *OpenReader* – initializes and tests the read/write head
- *CloseReader* – closes the read/write head
- *GetDLLVersion* – gets the version info of the driver (version number, release date, BIS system type)
- *GetLastError* – gets the error number of the last error within the driver
- *PowerOffRWHead* – turns off the power of the USB port, thus the RW head
- *UnLoadUSB* – close the USB port, but keeps the connection handle to opened BIS
- *ReLoadUSB* – re-opens the USB port to use the original connection handle

2 Additional files to the projects

In order to use the driver properly, the following files have to be added to the projects (or installation folder during runtime) as minimum:

- *SIUSBXP_LIB.dll* USB driver interface to Silabs
- *PtxSdkCommon.dll* C++ Common Library File (PsionTeklogixCE600)
- *BIS_C_WinCE_DLL.dll* BIS C-870 native coded driver
- *BIS_C_WinCE_dotNet.dll* Wrapper class library to BIS C-870 driver

3 Namespaces, classes, structures

The Class library contains the following namespaces, classes and structures.

3.1 Namespace Balluff.RFID

This is the "main" namespace of the wrapper. If user uses this namespace in his project, then he can reach all the classes and structures, related to BIS C-870 module.

3.2 Structure BISVERSION

This structure is created to store version information of the BIS_C_WinCE_DLL driver, such as:

- Version Contains the version number of the driver in 4 character format; e.g. "1.00"
- DateOfRelease Contains the date of the driver when it was created in format MM.DD.YYYY
- SystemType Contains the BIS system type (1 char); e.g. in case of BIS C-870 this is "C"

3.3 Structure ConnectionHandle

ConnectionHandle is a special handle, which stores information about the opened device, such as:

- COMPort Contains the port number on which the BIS device is opened
- LOCK indicates whether the RW head works with a command; RFU
- Process Contains the pseudo handle of the process, which opened the BIS device
- CodeTagType Shows, which Code Tag types can be read with the RW head (see types later)
- CRCCheck Indicates, whether the BIS module uses CRC16 check for data safety

After a reader is opened successfully, every function of the driver can be called only with this connection handle*. User should never change this handle manually!*

3.4 Class Errors

This is a public class, which contains constant values for the error numbers. The error numbers can be divided into 2 parts.

The first part of error codes covers the errors of BIS system. (For details see manuals.) These error codes are the return values of the functions, which implement Balluff Dialog protocol.
The second part of errors contains error codes, which have a meaning only within the driver, such as e.g. error during reading from the port, invalid connection handle was provided etc.

3.5 Class DataCarrierTypes

When user wants to open the read/write head, he has to provide a parameter, in which he specifies for the module, which Code Tags he wants to be recognized.

This class contains the following constant values:

- BIS_C_Unknown = 0x00; not known by Balluff
- BIS_C_1xx_04_05 = 0x01; (BIS C-1xx-04 and BIS C-1xx-05)
- BIS_C_1xx_11_32 = 0x02; (BIS C-1xx-11 and BIS C-1xx-32)

3.6 Class BISCReader

This is the main class of the wrapper class library. User can reach the implemented functions of Balluff Dialog via an object of this class.

This class also contains some properties, about the wrapper class, like:

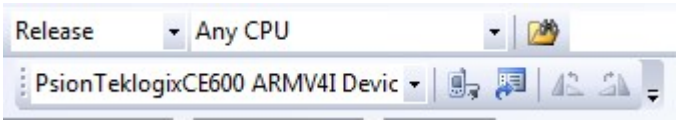
- dotNetVersion the version number of the wrapper class library
- dotNetDate the compilation date of the wrapper class library

4 Usage of the driver

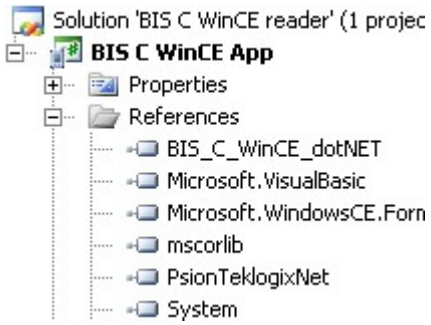
We recommend using Microsoft Visual Studio 2005 or higher to develop RFID applications for mobile devices. *Visual Studio maintains mobile solutions from Professional version!*

Before starting of development, Psion Teklogix Mobile Devices SDK for Windows® CE .NET CE 6.0 has to be installed (Find in this SDK or download from the homepage of Zebra Technologies: *PlatformCE600.msi*).

Then in Visual Studio *PsionTeklogixCE600 ARMV4I Device* has to be selected as target platform. This option allows the developer to develop against the appropriate hardware and environment running on PSION WorkaboutPro4. (See picture below.)



BIS_C_WinCE_dotNet.dll has to be added to the new .Net project as a reference.



Then *Balluff.BIS* namespace has to be added in the code for usage. (Although, it is not necessary makes development easier later on.)

```
using Balluff.RFID;
```

As a minimum, user has to create an object of the BISCReader class via which the read/write head functions and a variable for connection handle can be accessed.

```
BISCReader oReader = BISCReader.Instance;
```

```
//His connection handle will contain information about his opened device
ConnectionHandle MyConnection = new ConnectionHandle();

oReader.OpenUSBPort(1, ref MyConnection); //Use port '1' on PSION WorkaboutPro

oReader.OpenReader(ref MyConnection, false, DataCarrierTypes.BIS_C_1xx_04_05);
```

If any of the opening function fails, user has to check error codes and open the port and the device again. After these steps, user can use the RW head, e.g. read from a Code Tag:

Important!
Every function call returns with 0 on success (ERR_NOERROR) with the exception of OpenUSBPort and OpenReader functions, of which return codes are ERR_PORT_OPENED and ERR_READER_OPENED respectively.

```
String DataBack;          //variable which will store the data from the Code Tag

//read 50 bytes of data from address 0
oReader.ReadData(MyConnection, 0, 50, out DataBack);
```

For the parameters of the driver functions, please read "*BIS C-870 handheld driver - User's Manual*".