

Semiconductors	AN108511		
	PN51x Design-In Kit Quick Start Guide		
	Rev. 01.10 — 24. May 2006	Application note	
		108511	

Document information	
Info	Content
Keywords	PN51x, NFC
Abstract	This document describes how to set up first communication using the PN51x Design-In Kit

PHILIPS

Revision history

Rev	Date	Description
1.00	2005-04-11	First version along with BFL v3.1
1.10	2006-03-30	Update due to BFL v4.0 which follows Philips MoReUse Standardization

Contact information

For additional information, please visit: <http://www.semiconductors.philips.com>

For sales office addresses, please send an email to: sales.addresses@www.semiconductors.philips.com

1. Introduction

The aim of this document is to show how to establish a first communication by use of the parts sent within the PN51x Design-In Kit (Hardware) and the parts sent via Philips document control (Documentation and Software).

The naming "PN51x" refers to both products available - PN511 and PN512 - as there is no difference with respect to the initial steps.

After enumerating the prerequisites two different tools are presented. First a very simple piece of software which allows basic register operations, the second one will be a quick introduction to the NFC-BFL.

2. Prerequisites

2.1 System Requirements

- Two RS232 interfaces on host side
- Microsoft <Visual Studio (VS) .NET 2003>
- Acrobat Reader

2.2 References

Near Field Communication (NFC)

- General Information
<http://www.semiconductors.philips.com/markets/identification/products/nfc/index.html>
- ECMA Near Field Communication (NFC) White Paper (PDF)
<http://www.ecma-international.org/activities/Communications/2004tg19-001.pdf>
- ECMA-340 Near Field Communication – Interface and Protocol (NFCIP1)
This 2nd edition fully matches ISO/IEC 18092.
<http://www.ecma-international.org/publications/standards/Ecma-340.htm>
- ISO18092 Near Field Communication – Interface and Protocol (NFCIP1)
<http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMB ER=38578&COMMID=&scopelist=>

- PN511 – Transmission module, Short Form Specification (PDF)
http://www.semiconductors.philips.com/acrobat_download/other/identification/sfs_pn511_rev2.0.pdf

MIFARE® 1k Classic

- General Information
<http://www.semiconductors.philips.com/markets/identification/products/mifare/classic/index.html>
- MIFARE® Functional Specification (PDF)
http://www.semiconductors.philips.com/acrobat_download/other/identification/m001051.pdf

MIFARE® UltraLight (UL)

- General Information
<http://www.semiconductors.philips.com/markets/identification/products/mifare/ulight/index.html>
- MIFARE® UL Functional Specification (PDF)
http://www.semiconductors.philips.com/acrobat_download/other/identification/M028630.pdf

2.3 Abbreviations

BFL	“Basic Function Library” written in C and C++ to control our PN51x products
MoReUse	“Module Re-Use” Standard which is standardizing software for better reuse within Philips (naming of interfaces, variables, parameters etc.)
PCD	Proximity Coupled Device, “Reader/ Writer”
PICC	Proximity Integrated Chip Card, “Tag, Card”

3. Design-In-Kit: PN51x with NFC-BFL

Latest Demo-Boards have the type and hardware version printed on the antenna part of the PCB. You might find PN511 with hardware version 0xa.0 or PN512 with hardware version 0x8.0 inside your hardware package. We will refer to them as PN51x Demo-Board. The hardware version can be read at location 0x37 of the registers, too (Some hint for the already advanced users).

3.1 Hardware: Design-In Kit PN51x

- 2 Demo Boards PN51x
- 2 Power Supplies
- 3 MIFARE® UltraLight Cards
- 3 MIFARE® Classic 1k Cards
- Two serial cables

3.2 Software: SDK BFL PN51x

Inside the BFL Package you find this self extracting file: "NFC-Tools-Setup-V4_x.exe". The default installation directory is C:\Program Files\Philips Semiconductors\NFC PN51x Tools v4.x.x\. Please double-click on it and follow the instructions of the install shield.

To check that all components are installed correctly go to **Start/Programs/Philips Semiconductors/NFC PN51x Tools/** or to the given hard disk directories and compare to following list:

1. JoinerPCSerial and Scripts (*.jcf files)
C:\Program Files\Philips Semiconductors\NFC PN51x Tools V4.x.x\JoinerPcSerial
2. Example Project VS .NET solution file (ExampleProject.sln)
C:\Program Files\Philips Semiconductors\NFC PN51x Tools V4.x.x\Bfl\ex
3. makefile (see Annex B)
C:\Program Files\Philips Semiconductors\NFC PN51x Tools V4.x.x\Bfl\ex
4. BFL Reference Manual
C:\Program Files\Philips Semiconductors\NFC PN51x Tools V4.x.x\Bfl\docs\14_user_doc

Further Documents which will be provided via document control (either sent by e-mail or on a CD). It is up to the user to put them into a specific (project-) folder and put its links into the start menu.

5. AN1085: PN51x Design-In Kit Quick Start Guide
6. AN1087: Tips and Tricks when integrating PN51x
7. UM1194: PN51x Demo-Board Description
8. AN1007: NFC Transmission Module Antenna and RF Design Guide
9. Datasheet PN51x Transmission Module
 - a. 1113 Data Sheet PN512 or
 - b. 0797 Data Sheet PN511
10. 1013 Anomaly Sheet for PN51x
11. AN10362: Near Field Communication FAQ

4. Using the JoinerPCSerial

This application is provided as an executable with a graphical user interface. In principal it is build on very low level – just offering read and write register operations basically. It can be compared to the Bus-Abstraction Layer which is part of the NFC-BFL software package.

The software can be used to communicate with PN51x on register level. By use of script files the single register operations provide the necessary settings to configure the IC for a certain operating mode. The script files have the extension *.jcf.

Note that this application is intended to configure the PN51x. In order to perform more complex tasks (including control flow,...) the ExampleProject shall be used.

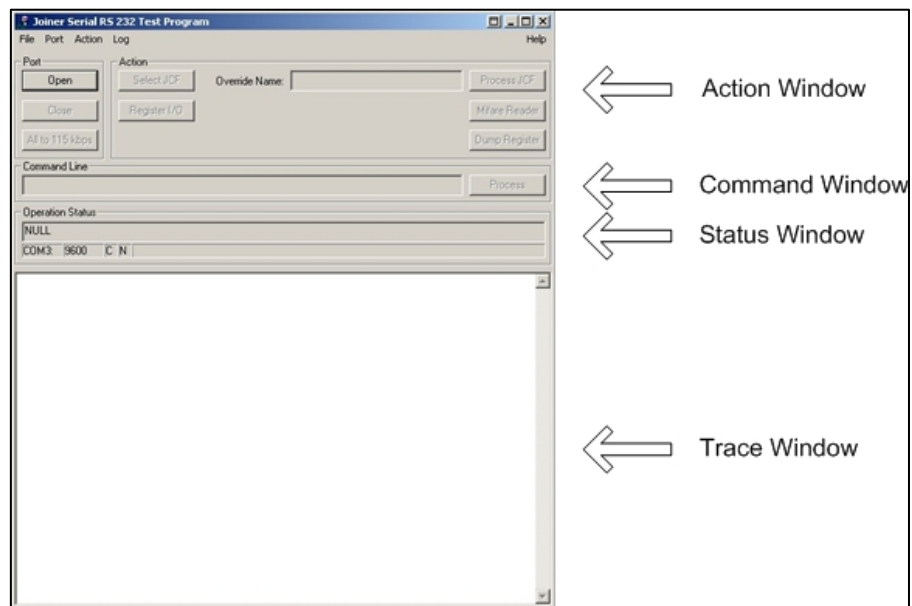
The following script files are provided:

- mifare_reader.jcf - act as Mifare® reader
- nfc_initiator_106kbps.jcf - act as active NFC Initiator
- nfc_target_106kbps.jcf - act as passive NFC Target
- passive_target.jcf - act as passive PICC Target
- passive_initiator_212.jcf - act as FeliCa™ reader
- loop_jmp_example.jcf - example how to implement JCF loops

4.1 Getting Started

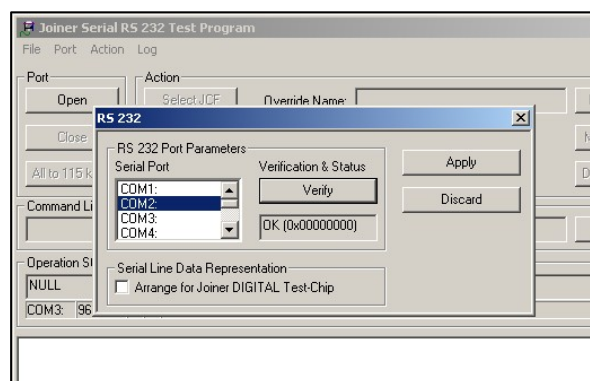
Please connect one PN51x Demoboard to the RS232 interface and power it up using the enclosed switching power supply.

1. Start JoinerPCSerial by clicking on: **Start/Programs/Philips Semiconductors/NFC PN51x Tools/JoinerPCSerial**. An empty window opens:



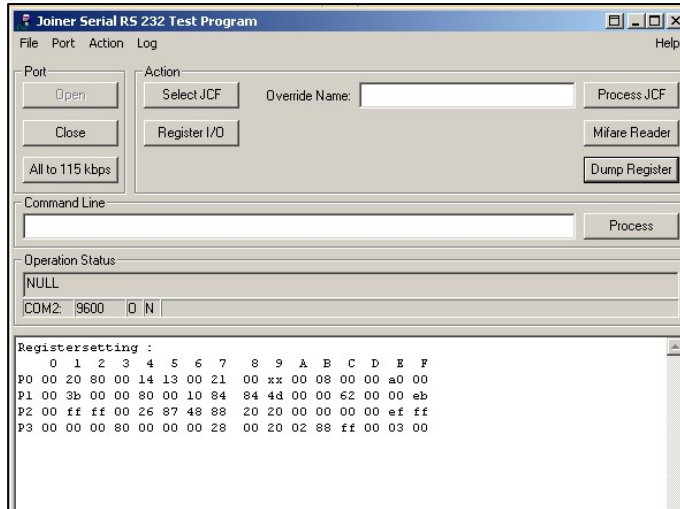
The *Joiner Serial RS232 Test Program* (=JoinerPCSerial) is a simple User Interface (UI) utility for register-based access to the Joiner hardware using the serial interface and the *Basic Function Library (BFL)*.

2. The RS232 COM port has to be configured: **Port/Settings...**

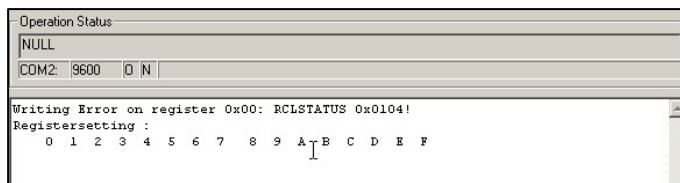


Choose the appropriate RS232 COM port for your serial communication, **Verify** (checks if the port is used or not) your selection and press **Apply**.

- By using the **Open** button the communication channel should be established. An indicator of correct settings is the changing of all buttons but Open from inactive to active state.

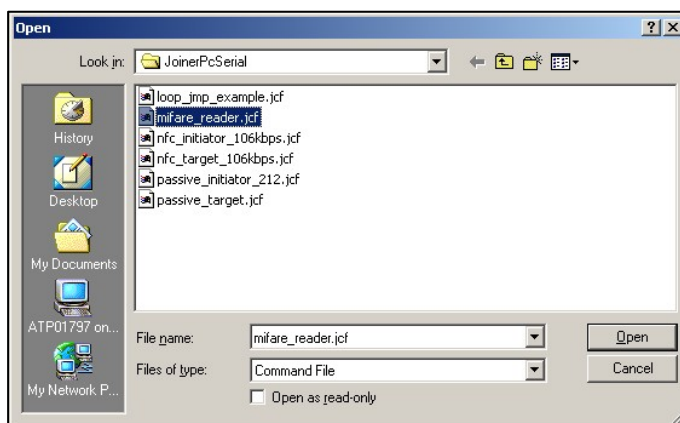


An additional check can be performed using the **Dump Register** button. It returns the actual PN51x settings. The result can be read off in the trace window.



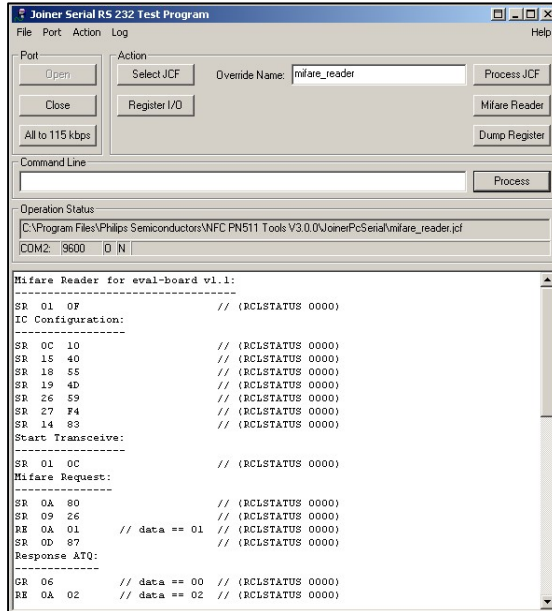
If a timeout occurred something in the previous setup procedure went wrong. Compare with the window on the left.

- By pressing **Select JCF** a window with all delivered scripts is opened. For this first try the file "mifare_reader.jcf" is taken.

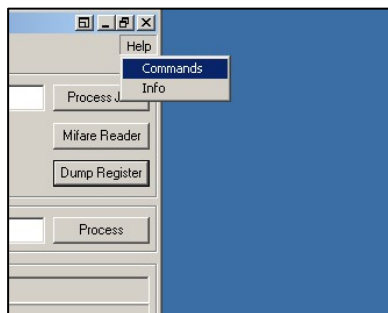


It might be useful while investigating the different modes to open the files in a text editor, too, due to inline documentation which will not be displayed in the trace window.

- Put a MIFARE® Classic 1k or MIFARE® UltraLight (UL) on the Demo-Board and start the communication by using the **Process JCF** button.



The first line sets register 01 with value 0x0F. A soft reset is performed. The status variable RCLSTATUS showing 0 indicates that no error in the communication has occurred.



All the possible commands (e.g. SR, RE...) used in the scripts are explained when selecting **Help/Commands**.

All commands are explained in Appendix A, too.

As mentioned above more details on the scripts can be viewed at by opening the *.jcf file with a text editor.

The first comments of the script chosen: mifare_reader.jcf to compare it to the output of the trace window:

```
CLL    // Clear History

//> Mifare Reader for eval-board v1.1:
//> -----

SR 01 0F // Softreset

//> IC Configuration:
//> -----
SR 0c 10 // ControlReg - Initiator
SR 15 40 // TxAutoReg
SR 18 55 // RxThresholdReg - MinLevel, CollLevel
SR 19 4D // DemodReg
SR 26 59 // RFCfgReg - RxGain
SR 27 F4 // GsNReg - CWGsN, ModGsN
SR 14 83 // TxControlReg - InvTx2On=1, Tx2RFEn, Tx1RFEn

//> Start Transceive:
//> -----
SR 01 0c // CommandReg - transceive

//> Mifare Request:
//> -----
SR 0a 80 // flush FIFO
SR 09 26 // FIFO - Request code
RE 0a 01 // Read FIFOLevel
...
..
.
```

```

Joiner Serial RS 232 Test Program
File Port Action Log Help
Port:
  Open
  Close
  All to 115 kbps
Action:
  Select JCF
  Register I/O
  Process JCF
  Mifare Reader
  Dump Register
Override Name: mifare_reader
Command Line:
  Process
Operation Status:
  C:\Program Files\Philips Semiconductors\WFC PN511 Tools V3.0.0\JoinerPcSerial\mifare_reader.jcf
  COM2: 9600
GR 06 // data == 00 // (RCLSTATUS 0000)
RE 0A 05 // data == 05 // (RCLSTATUS 0000)
GR 09 // data == 88 // (RCLSTATUS 0000)
GR 09 // data == 04 // (RCLSTATUS 0000)
GR 09 // data == B0 // (RCLSTATUS 0000)
GR 09 // data == D5 // (RCLSTATUS 0000)
GR 09 // data == E9 // (RCLSTATUS 0000)
Mifare Select:
SR 0A 80 // (RCLSTATUS 0000)
SR 09 93 // (RCLSTATUS 0000)
SR 09 70 // (RCLSTATUS 0000)
SR 12 80 // (RCLSTATUS 0000)
SR 13 80 // (RCLSTATUS 0000)
SR 09 88 // (RCLSTATUS 0000)
SR 09 04 // (RCLSTATUS 0000)
SR 09 B0 // (RCLSTATUS 0000)
SR 09 D5 // (RCLSTATUS 0000)
SR 09 E9 // (RCLSTATUS 0000)
SR 0D 80 // (RCLSTATUS 0000)
Response SAK:
GR 06 // data == 00 // (RCLSTATUS 0000)
RE 0A 01 // data == 01 // (RCLSTATUS 0000)
GR 09 // data == 04 // (RCLSTATUS 0000)
Response SAK: 0x04 Mifare UL (UltraLight) detected

```

Scrolling down to the end of the trace window the Select Acknowledge (SAK) can be read off. For this example with the MIFARE® UL card it has to be 0x04. If the MIFARE® Classic 1k is placed on the antenna the SAK is supposed to be 0x08.

This code is used to determine the type of card and its capabilities respectively.

4.2 Command Line

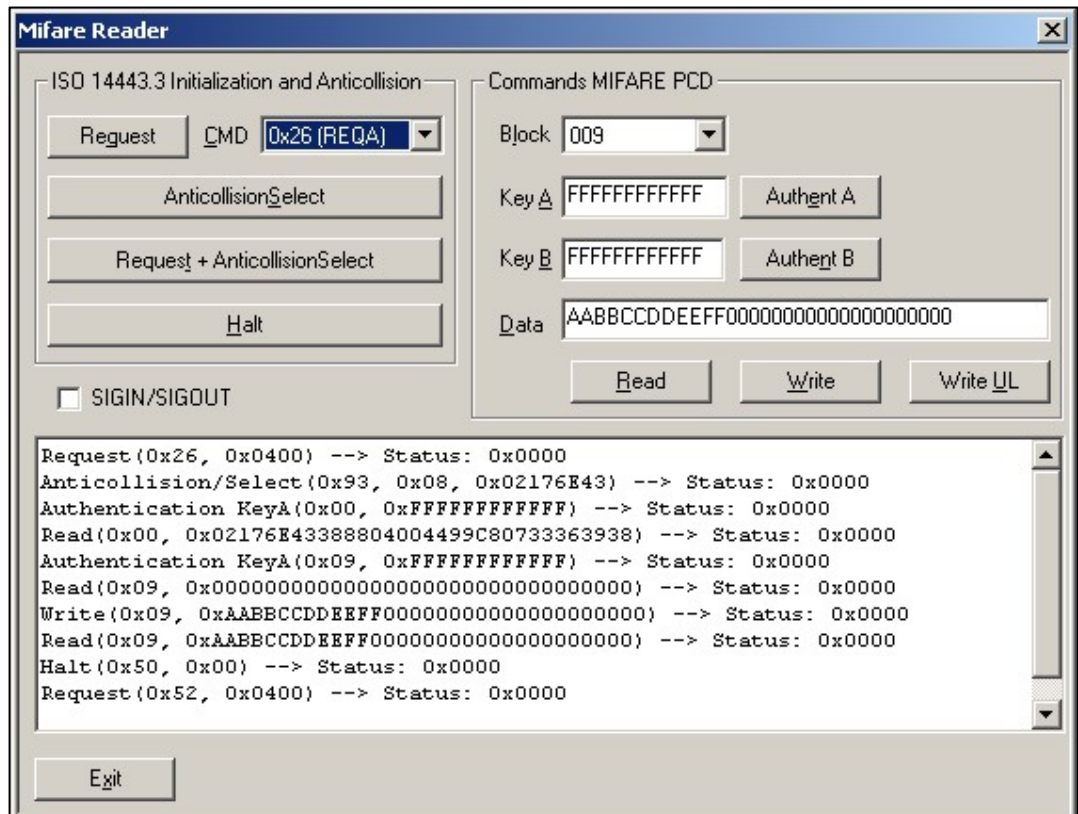
Besides the scripts single commands can be handled, too. This can be done by typing e.g. **SR 01 0F** (Set Register 01 to 0x0F to perform a Soft-Reset) in the Command Line and execute it with **Process**. In principle all lines of the scripts provided could be fed into the command line sequentially, each followed with a **Process**. Its main intention is to try out small sequences in interaction with the Dump Register function to verify modifications visually.

4.3 Register I/O

The button Register I/O can be used to read from and write to certain registers.

4.4 MIFARE® Reader Window

In order to operate a MIFARE® (Classic 1k or Ultra Light) card it is required to step through Initialisation and Anti-collision. After successfully performing this ISO14443.3 procedure the card is ready to receive MIFARE® Proximity Coupled Device (PCD) commands. After selecting a block (refer to the MIFARE® specification for detailed information about available blocks and their types) and Authentication (except MIFARE® Ultra Light) data can be written to or read from the Proximity Integrated Circuit Card (PICC).



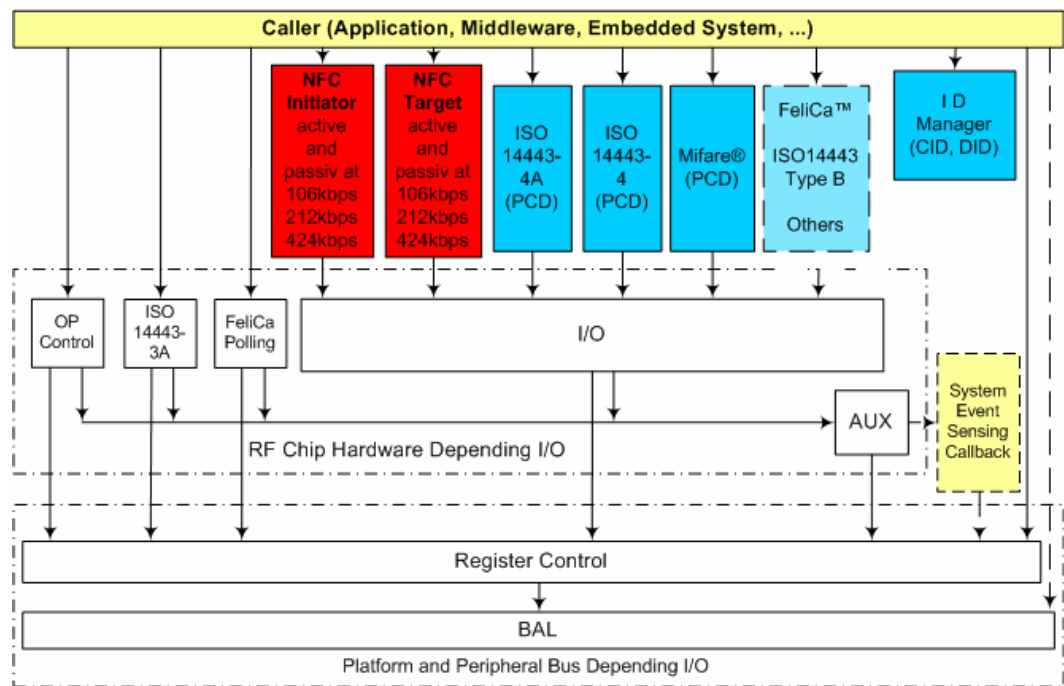
5. Using the ExampleProject (VS .NET)

The BFL is delivered in plain source and shall be used for testing purposes only. For the exact liability information please refer to the “BFL Reference Manual” section 1.3 “Liability Information”.

5.1 Remark on NFCIP1 Compliancy

Due to the BFL component approach it is a very flexible piece of software. It allows a large number of different implementations using more or less BFL components/ functionality.

To be compliant to ISO18092 NFCIP1 specifications it is mandatory to include the Active and Passive NFC communication components with its three data transmission speeds each. All other components provided are ADD-ONS, which allow additional communication modes.



- ISO 18092 / NFCIP1 Compliant Components
- ADD-ONS
- Mandatory Stack or Hardware Related Components
- Application Accessible Components

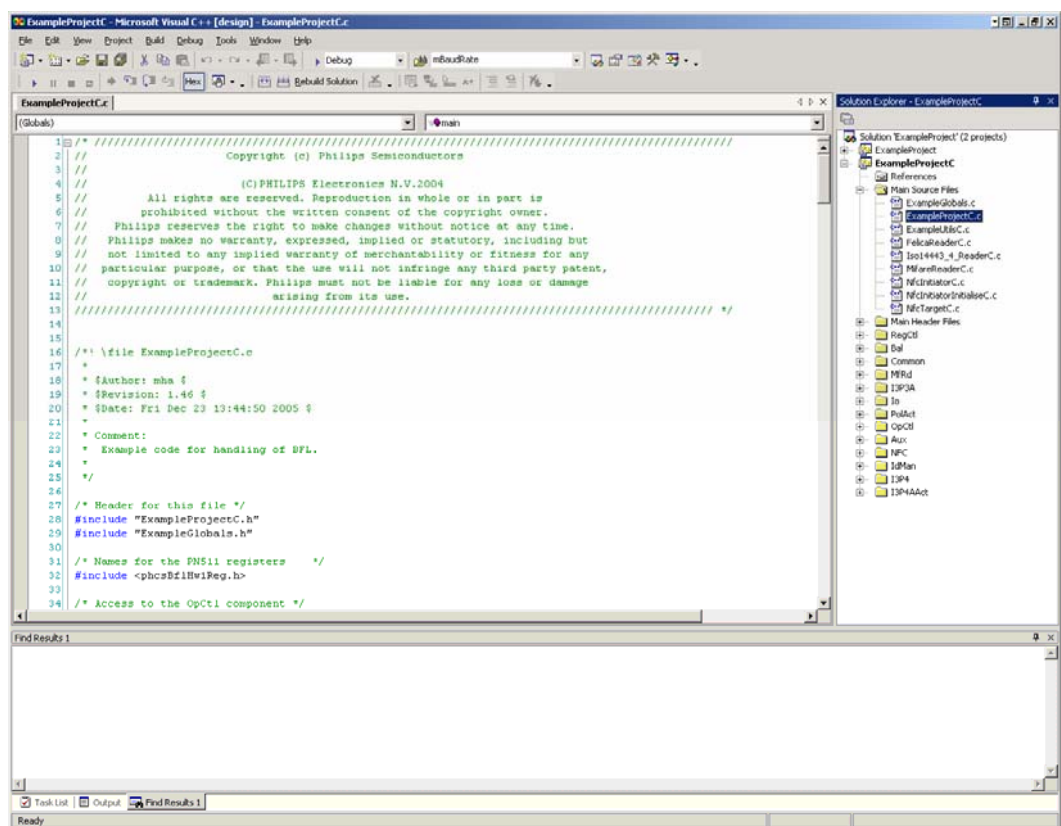
Just to clarify: The components of “RF Chip Hardware Depending I/O” and “Platform and Peripheral Bus Depending I/O” have to be included in any of all possible solutions.

5.2 BFL C Version

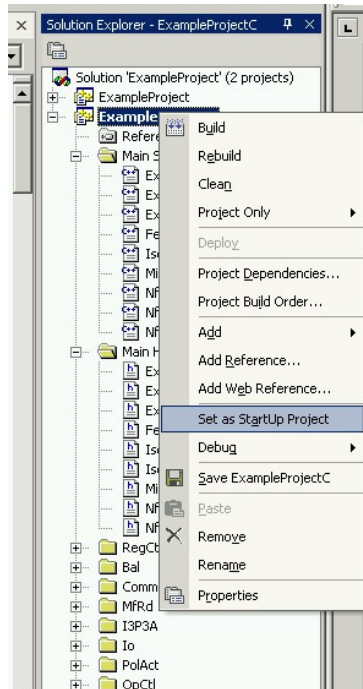
The example project should help you to take the first steps inside the Basic Function Library (BFL). This example was set up by using Visual Studio .NET. We will choose the C version as default and set up the MIFARE® reader functionality. Please start up the project: **Start/Programs/Philips Semiconductors/NFC PN51x Tools/ExampleProject.NET**

The mapped window is opened. To get a better overview you should enable the Solution Explorer. If not opened by default click on: **View/Solution Explorer**

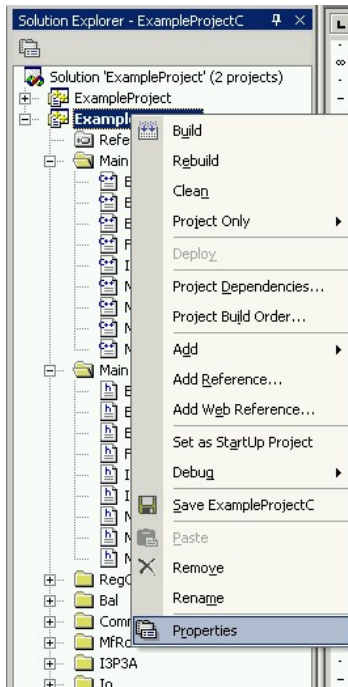
To open the ExampleProjectC: Double click in the Solution Explorer in the Source-Files-Folder the ExampleProjectC.c file.



The BFL consists of two projects. Both the C (ExampleProjectC) and the C++ (ExampleProject) versions are included. In principle all files should appear in the Solution Explorer. Here only the C tree was expanded.



As mentioned we will start to work in the C environment: Right-click on the ExampleProjectC in the Solution Explorer. In the flip down menu select **Set as StartUp Project**.

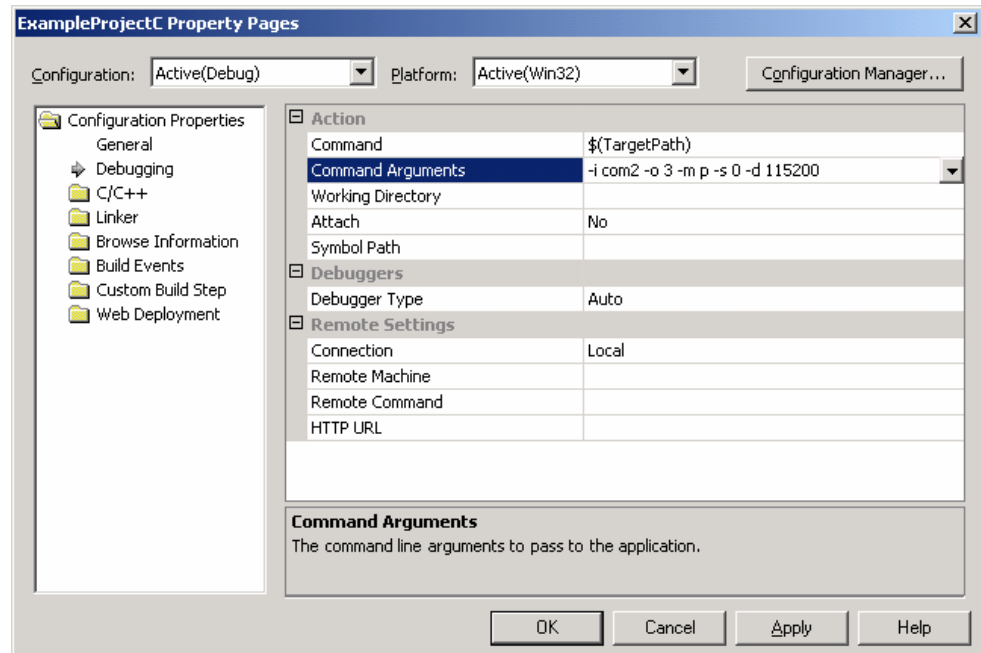


Select with a right mouse click: **ExampleProjectC/ Properties**

Some further parameters have to be customized like the COM port used and the type of communication wanted.

The project property window opens:

In the example given we would choose NFC Initiator, passive @106kbps communication. Select the sub directory **Debugging**.



Insert the command arguments according to following scheme (Run the executable and add `-h` or `-?` as parameter):

```

C:\winnt\system32\cmd.exe
C:\Program Files\Philips Semiconductors\NFC PN51x Tools U4.1.0\Bfl\ex\DebugC>ExampleProjectC.exe -?
ExampleProject/BFL Source Code Examples, (c) Philips Semiconductors 2004, 2005
Windows : ExampleProject[C].exe [-i ComIfc] [-o OpModel] [-m pial] [-s 0-2] [-h]
Linux   : C[pp]Example [-i ComIfc] [-o OpModel] [-m pial] [-s 0-2] [-h]

-h          Show this help
-i ComIfc  RS232 Interface (e.g. -i COM1 or -i /dev/ttyS0)
-d ComBr   RS232 Baudrate (default: 9600; e.g. -d 115200)
           Following baudrates are supported:
           9600, 115200, 128000, 230400, 460800, 921600, 1288000
-o [1-6]  Operating Mode (e.g. -o 3 for NFC Initiator)
           Following Operating Modes are supported:
           1 ... Mifare Reader
           2 ... FeliCa Reader (just Polling)
           3 ... NFC Initiator
           4 ... NFC Target
           5 ... ISO14443-4 Reader
           6 ... Power Down Example
-m [pial] Passive/Active Mode (only relevant for NFC-I)
           p ... Passive Mode
           a ... Active Mode
-s [0-2]  Bit Rate (only relevant for NFC-I)
           0 ... 106kBit/s
           1 ... 212kBit/s
           2 ... 424kBit/s

C:\Program Files\Philips Semiconductors\NFC PN51x Tools U4.1.0\Bfl\ex\DebugC>

```

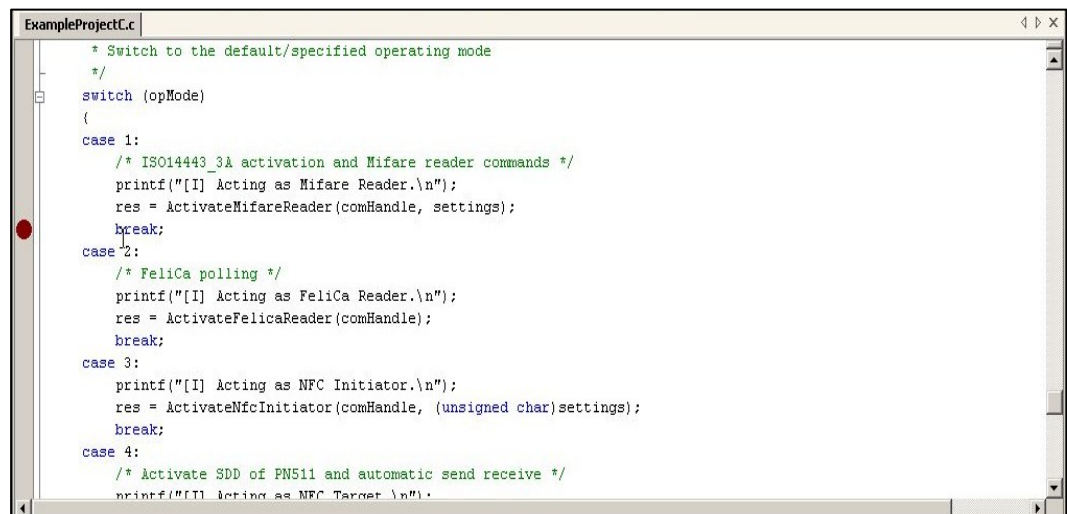
Attention: These properties exist for the C as for the C++ part. If no connection could be established with your PN51x-demoboard first check if the property page has been filled out correctly!

Complete the setup with **Apply** and **OK**.

The project is now ready to be build and linked: Build/Build Solution

To avoid any conflict on the serial port please close the port you are using in the Joiner-PC-Serial if the window is still open.

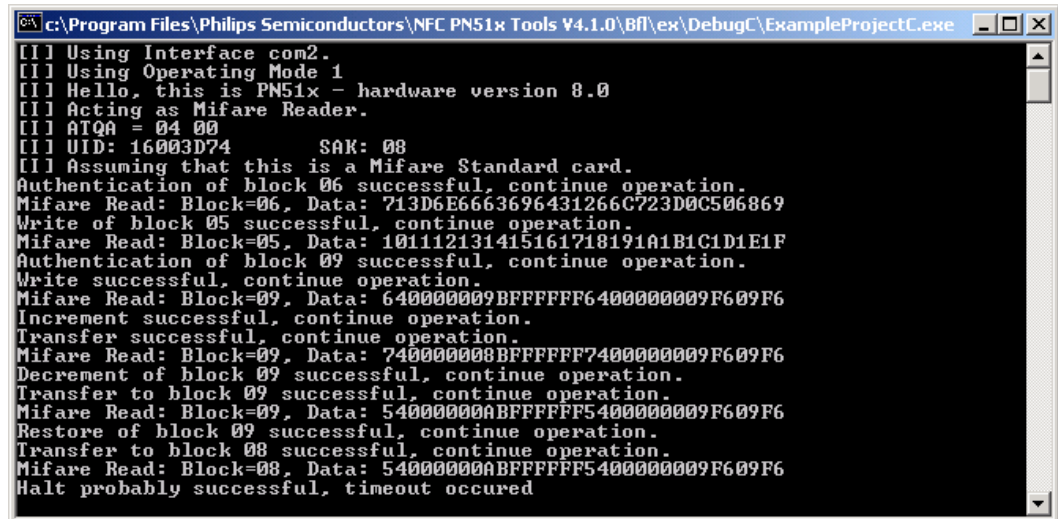
To see any result we have to set a breakpoint after the MIFARE® reader example. Please set the parameters in the properties page accordingly (e.g. -i com2 -o 1).



```
ExampleProjectC.c
/* Switch to the default/specified operating mode
*/
switch (opMode)
{
case 1:
/* ISO14443_3A activation and Mifare reader commands */
printf("[I] Acting as Mifare Reader.\n");
res = ActivateMifareReader(comHandle, settings);
break;
case 2:
/* FeliCa polling */
printf("[I] Acting as FeliCa Reader.\n");
res = ActivateFeliCaReader(comHandle);
break;
case 3:
printf("[I] Acting as NFC Initiator.\n");
res = ActivateNfcInitiator(comHandle, (unsigned char)settings);
break;
case 4:
/* Activate SDD of PN511 and automatic send receive */
printf("[I] Acting as NFC Target.\n");
```

A breakpoint is set (ExampleProjectC.c) either by double-clicking on the gray left strip where the red dot appears or by setting the cursor to the wanted line and selecting Debug/New Breakpoint.

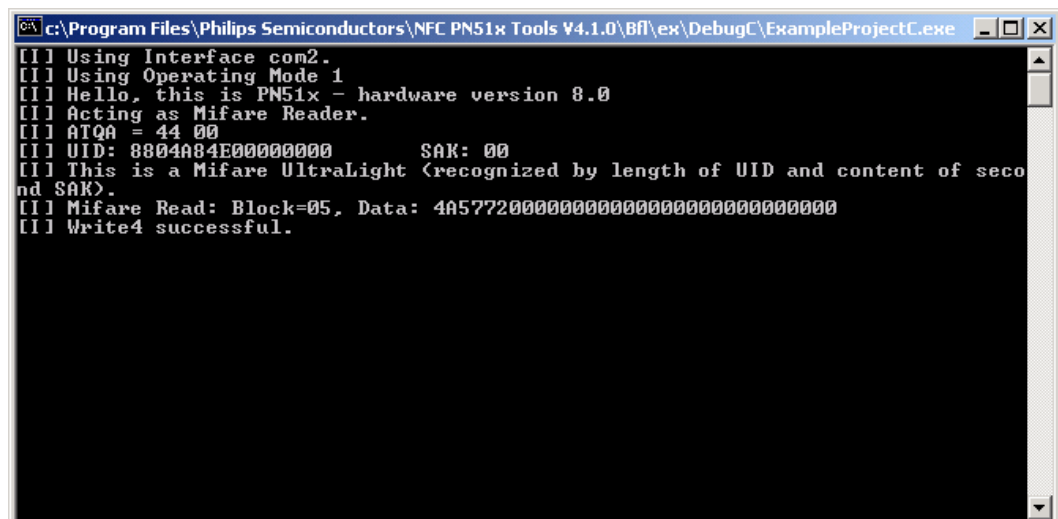
Take a MIFARE® Classic 1k card and apply it onto the Demoboard. Start the communication with Debug/Start. The DOS debug window informs you about the several steps taken:



```
c:\Program Files\Philips Semiconductors\NFC PN51x Tools V4.1.0\Bfl\ex\DebugC\ExampleProjectC.exe
[I] Using Interface com2.
[I] Using Operating Mode 1
[I] Hello, this is PN51x - hardware version 8.0
[I] Acting as Mifare Reader.
[I] ATQA = 04 00
[I] UID: 16003D74 SAK: 08
[I] Assuming that this is a Mifare Standard card.
Authentication of block 06 successful, continue operation.
Mifare Read: Block=06, Data: 713D6E6663696431266C723D0C506869
Write of block 05 successful, continue operation.
Mifare Read: Block=05, Data: 101112131415161718191A1B1C1D1E1F
Authentication of block 09 successful, continue operation.
Write successful, continue operation.
Mifare Read: Block=09, Data: 640000009BFFFFFF640000009F609F6
Increment successful, continue operation.
Transfer successful, continue operation.
Mifare Read: Block=09, Data: 740000008BFFFFFF740000009F609F6
Decrement of block 09 successful, continue operation.
Transfer to block 09 successful, continue operation.
Mifare Read: Block=09, Data: 54000000ABFFFFFF540000009F609F6
Restore of block 09 successful, continue operation.
Transfer to block 08 successful, continue operation.
Mifare Read: Block=08, Data: 54000000ABFFFFFF540000009F609F6
Halt probably successful, timeout occurred
```

To continue either press F5 or use the cursor to select **Debug/Continue**. The communication is cleaned up and closed.

The output for a Mifare UL card (using the same configuration):



```
c:\Program Files\Philips Semiconductors\NFC PN51x Tools V4.1.0\Bfl\ex\DebugC\ExampleProjectC.exe
[I] Using Interface com2.
[I] Using Operating Mode 1
[I] Hello, this is PN51x - hardware version 8.0
[I] Acting as Mifare Reader.
[I] ATQA = 44 00
[I] UID: 8804A84E00000000 SAK: 00
[I] This is a Mifare UltraLight (recognized by length of UID and content of second SAK).
[I] Mifare Read: Block=05, Data: 4A577200000000000000000000000000
[I] Write4 successful.
```

5.3 The Command Window

If you want to try out your project by using the command window – this is possible, too. Parse through your file system until you finally arrive at the ExampleProjectC.exe location. You may choose between different arguments to set and configure the executable file without recompiling the whole project. For an overview just enter **ExampleProjectC.exe -h** or **ExampleProjectC.exe -?**.

```

C:\winnt\system32\cmd.exe
C:\Program Files\Philips Semiconductors\NFC PN51x Tools U4.1.0\Bf1\ex\DebugC>ExampleProjectC.exe -?

ExampleProject/BFL Source Code Examples, (c) Philips Semiconductors 2004, 2005

Windows : ExampleProject[C].exe [-i ComIfc] [-o OpMode] [-m p|a] [-s 0-2] [-h]
Linux   : C[pp]Example [-i ComIfc] [-o OpMode] [-m p|a] [-s 0-2] [-h]

-h          Show this help
-i ComIfc   RS232 Interface (e.g. -i COM1 or -i /dev/ttyS0)
-d ComBr    RS232 Baudrate (default: 9600; e.g. -d 115200)
            Following baudrates are supported:
            9600, 115200, 128000, 230400, 460800, 921600, 1288000
-o [1-6] Operating Mode (e.g. -o 3 for NFC Initiator)
            Following Operating Modes are supported:
            1 ... Mifare Reader
            2 ... FeliCa Reader (just Polling)
            3 ... NFC Initiator
            4 ... NFC Target
            5 ... ISO14443-4 Reader
            6 ... Power Down Example
-m [p|a] Passive/Active Mode (only relevant for NFC-I)
-p ... Passive Mode
-a ... Active Mode
-s [0-2] Bit Rate (only relevant for NFC-I)
            0 ... 106kBit/s
            1 ... 212kBit/s
            2 ... 424kBit/s

C:\Program Files\Philips Semiconductors\NFC PN51x Tools U4.1.0\Bf1\ex\DebugC>

```

Congratulations! You managed to set-up and use the BFL. If you want to gain more knowledge about its structure - there is no better way than stepping through the examples looking at the parameters and functions.

Further information can be obtained by reading the documentation provided within the Design-In-Kit like the “PN51x Programmers Manual”.

Good Speed for your Design-In!

6. APPENDIX A: Parser Command Overview of JoinerPCSerial

The parser supports commands for both PC (host) and Joiner control. Two-letter commands control the chip three-letter commands represent the host-side. Always keep a space between command and parameter.

6.1 Joiner-Related Commands

Table 1: Joiner Commands

Commands	Synopsis	Description
SR	SR <address> <data>	The SR function sets a Joiner register, located at address <address> according to data <data>, both specified as an 8-bit HEX value.
GR	GR <address>	The GR function gets data from a Joiner register, located at address <address> (8-bit HEX). The retrieved value is stored in IOR.
MR	MR <address> <mask> <set>	The MR function modifies a Joiner register, located at address <address>. The mask <mask> specifies which bits to modify by having the corresponding bits set. If the <set> parameter is nonzero, the corresponding bits are set, otherwise cleared. All values are in 8-bit HEX format.
RE	RE <address> <data>	The RE function compares a Joiner register, located at address <address> to data, specified in the <data> parameter. If equal, IOR is 0, otherwise 1. All values are in 8-bit HEX format.
RF	RF <address> <data> <mask>	The RF function compares a Joiner register, located at address <address> to data, specified in the <data> parameter, AND' ed with the content of <mask>. If equal, IOR is 0, otherwise 1. All values are in 8-bit HEX format.

6.2 Host-Related Commands

Table 2: Host Related Commands

Commands	Synopsis	Description
CHB	CHB <bit-rate>	The CHB function sets the <bitrate> (in bps) of the PC serial port. Possible values are: {9600, 19200, 38400, 57600, 115200}.
WIE	WIE < timeout ms>	The WIE function waits for an edge at the serial port's RI pin. Maximum waiting time is specified by <timeout ms>, in [ms]. This function should be used with caution only (not recommended).
WIL	WIL <level> <timeout ms>	The WIL function waits for the serial port's RI pin to reach a certain logical level, specified by <level>, (= {0, 1}) Maximum waiting time is specified by <timeout ms>, in [ms]. This is the preferred intr. function.
SLP	SLP <timeout ms>	The SLP function waits for the time is specified by <timeout ms>, in [ms] to expire.
CLL	CLL	The CLL function removes all content from the application's LOG window.
//	// <Comment Text>	The // function does nothing but allow comments being added to a script. The text <Comment Text> must be separated from the command by at least one blank.
//>	//> <Message Text>	The //> function allows messages to be displayed during script execution. The text <Message Text> must be separated from the command by at least one blank.
//#	//# <data>	The //# function allows data to be displayed during script execution. The <data> parameter can be either plain data (8-bit HEX) or a User Register.
JMP	JMP <destination>	The JMP function skips script commands until a label with the name <destination> is found. The label name <destination> should contain only {a..z, A..Z, _}.
JNE	JNE <value> <compare value> <destination>	The JNE function compares User Register or plain data <value> to <compare value>. If unequal, the function skips script commands until a label with the name <destination> is found. Data are in 8-bit HEX format.
:::	::: <destination>	The ::: function is the <destination> of the JUMP commands. The label name <destination> should contain only {a..z, A..Z, _}.
MOV	MOV <destination> <source>	The MOV function copies User Register or plain data from <source> to <destination>. Data are in 8-bit HEX format.
INC	INC <user register>	The INC function increments a user register.
DEC	DEC <user register>	The DEC function decrements a user register.

6.3 Registers

Table 3: Registers

Registers	Scope	Description
ML0..ML7	General Purpose	These registers can be used to store internal variables, loop counters, comparison references and other types of items useful for script execution control.
IOR	I/O Result	This variable receives the result of an I/O operation (see Joiner-Related Commands). This can either be any numerical value in case of register content retrieval or a Boolean value pointing out the result of a comparison.
IOE	I/O Error	This register served as an I/O error indicator. It merely points out the fact that an error has occurred, not the type of error itself. In case of success the value is 0 , otherwise 1 .

7. Appendix B: BFL, GCC, Linux

If you want to run the BFL under Linux there is a makefile available for GNU Compiler Collection (GCC) v3.3. It was verified to run under Fedora Linux Core 2.

The makefile is located in the same directory as the VS .NET project files.

You will find further information in the "BFL Reference Manual" section 2.24 "Using the ExampleProject".

8. Disclaimers

Life support — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no licence or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

9. Licenses

Purchase of Philips I²C components



Purchase of Philips I²C components conveys a license under the Philips' I²C patent to use the components in the I²C system provided the system conforms to the I²C specification defined by Philips. This specification can be ordered using the code 9398 393 40011.

Purchase of Philips RC5 components

Purchase of Philips RC5 components conveys a license under the Philips RC5 patent to use the components in RC5 system products conforming to the RC5 standard UATM-5000 for allocation of remote control commands defined by Philips.

10. Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — <patent owner>

11. Trademarks

MIFARE® — is a registered trademark of Royal Philips Electronics

FeliCa™ — is a trademark of Sony Cooperation

12. Contents

1.	Introduction	3
2.	Prerequisites.....	3
2.1	System Requirements.....	3
2.2	References.....	3
2.3	Abbreviations	4
3.	Design-In-Kit: PN51x with NFC-BFL	5
3.1	Hardware: PN51x.....	5
3.2	Software: SDK BFL PN51x	5
4.	Using the JoinerPCSerial.....	7
4.1	Getting Started.....	8
4.2	Command Line.....	12
4.3	Register I/O.....	12
4.4	MIFARE® Reader Window	13
5.	Using the ExampleProject (VS .NET).....	14
5.1	Remark on NFCIP1 Compliancy	14
5.2	BFL C Version.....	15
5.3	The Command Window	20
6.	APPENDIX A: Parser Command Overview	21
6.1	Joiner-Related Commands	21
6.2	Host-Related Commands.....	22
6.3	Registers.....	23
7.	Appendix B: BFL, GCC, Linux.....	23
8.	Disclaimers	24
9.	Licenses	24
10.	Patents	24
11.	Trademarks	24
12.	Contents.....	25



© Koninklijke Philips Electronics N.V. 2006

All rights are reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner. The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent- or other industrial or intellectual property rights.

Date of release: 24. May 2006
Document number: 108511

Published in Austria