UM 119221

PCSerial

Rev. 2.1 — 07 July 2007 User manual

Document information

Info	Content
Keywords	*PCSerial*, MFRD52x development board, MFRC52x, PN51x design-in kit
Abstract	This document describes the demo program *PCSerial*. The demo program *PCSerial* has been developed to test functionality of the design-in board MFRD52x which is based on the NXP reader IC MFRC522 or MFRC523.

Revision history

Rev	Date	Description
2.1	July 2007	Changed to NXP Layout Add Chapter 2.7
2.0	December 2005	Changed status to preliminary
1.0	September 2005	Initial Version

Contact information

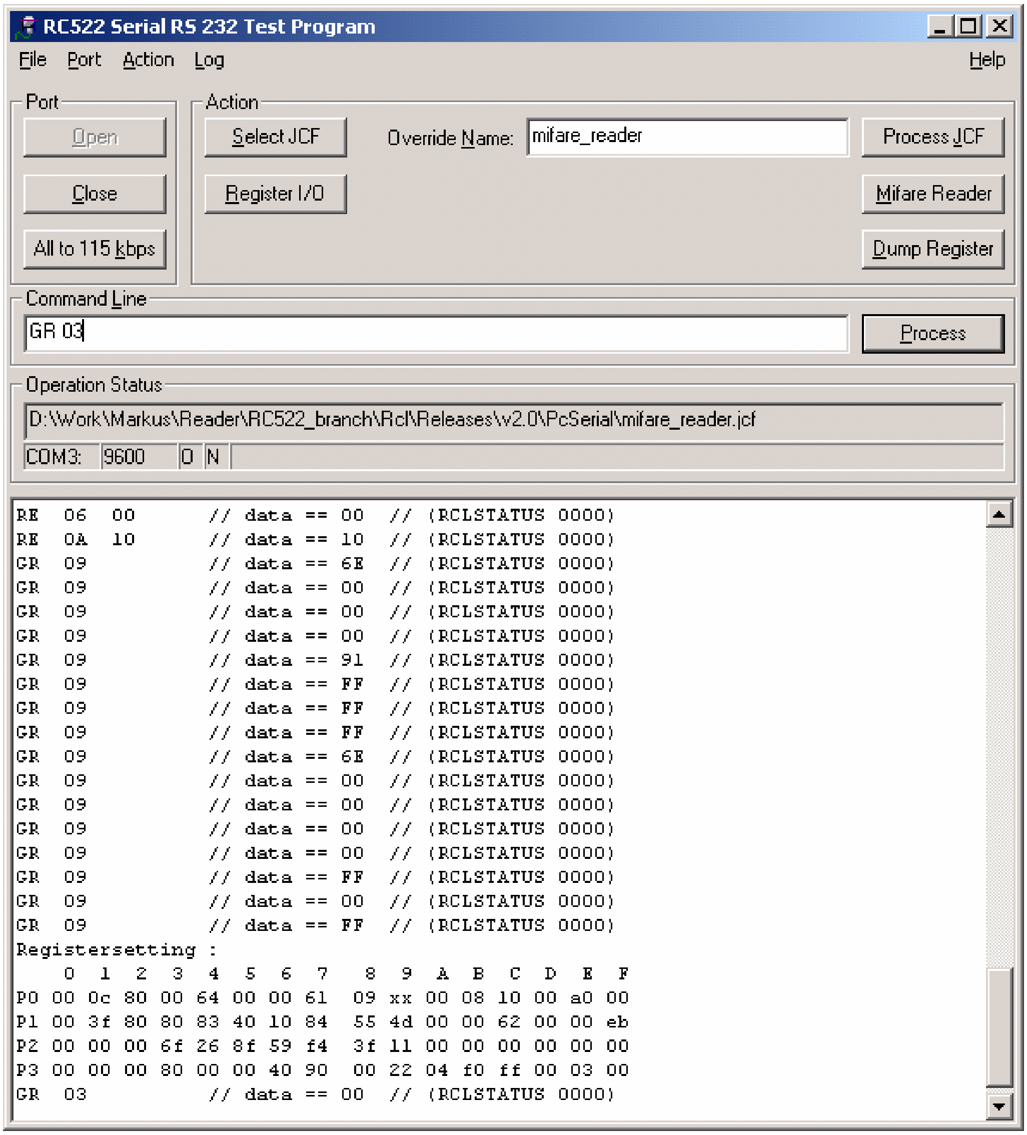
For additional information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

1. Introduction

This document describes how to use the PC-Serial demonstration program in combination with NXP Semiconductors MIFARE® readers based on the MFRC52x.

The MFRC52x Serial RS232 Test Program is a simple graphical user interface (GUI) utility for register based access to the hardware using the serial interface and the Basic Function Library (BFL). In addition, there is the possibility to execute MIFARE commands. Fig 1 shows *PCSerial* graphical user interface.



Override Name: mifare_reader

Command Line: GR 03

D:\Work\Markus\Reader\RC522_branch\Rcl\Releases\v2.0\PcSerial\mifare_reader.jcf

COM3: 9600 0 N

```
RE  06  00       // data == 00  // (RCLSTATUS 0000)
RE  0A  10       // data == 10  // (RCLSTATUS 0000)
GR  09           // data == 6E  // (RCLSTATUS 0000)
GR  09           // data == 00  // (RCLSTATUS 0000)
GR  09           // data == 00  // (RCLSTATUS 0000)
GR  09           // data == 00  // (RCLSTATUS 0000)
GR  09           // data == 91  // (RCLSTATUS 0000)
GR  09           // data == FF  // (RCLSTATUS 0000)
GR  09           // data == FF  // (RCLSTATUS 0000)
GR  09           // data == FF  // (RCLSTATUS 0000)
GR  09           // data == 6E  // (RCLSTATUS 0000)
GR  09           // data == 00  // (RCLSTATUS 0000)
GR  09           // data == 00  // (RCLSTATUS 0000)
GR  09           // data == 00  // (RCLSTATUS 0000)
GR  09           // data == 00  // (RCLSTATUS 0000)
GR  09           // data == FF  // (RCLSTATUS 0000)
GR  09           // data == 00  // (RCLSTATUS 0000)
GR  09           // data == FF  // (RCLSTATUS 0000)
Registersetting :
    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
P0 00 0c 80 00 64 00 00 61  09 xx 00 08 10 00 a0 00
P1 00 3f 80 80 83 40 10 84  55 4d 00 00 62 00 00 eb
P2 00 00 00 6f 26 8f 59 f4  3f 11 00 00 00 00 00 00
P3 00 00 00 80 00 00 40 90  00 22 04 f0 ff 00 03 00
GR  03           // data == 00  // (RCLSTATUS 0000)
```

Fig 1. *PCSerial* Graphical User Interface

2. Operating Instructions

Several operating instructions give the possibility to adapt the PC-Serial demonstration program to a specific environment.

2.1 File Menu

Table 1. File Menu Commands

Operating Mode	Description
Exit	Exits the program

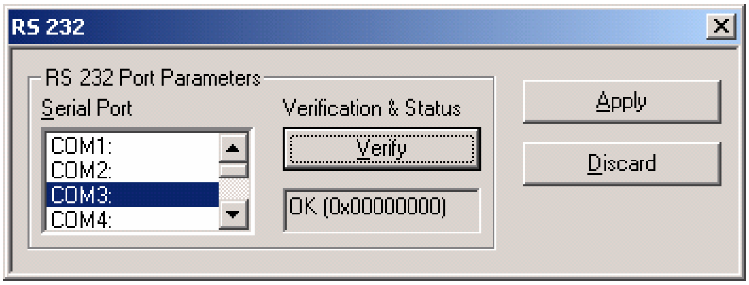
2.2 Port Menu

Table 2. Port Menu Commands

Operating Mode	Description
Settings…	Opens a dialog to configure the RS232 port
Open	Opens the defined port
Close	Closes the defined port

2.2.1 RS 232 Settings Dialog

This dialog establishes contact with the hardware. Fig 2 shows the RS232 configuration dialog.



Serial Port: COM3:

OK (0x00000000)

Fig 2. *RS 232* Settings Dialog

Please select the appropriate serial port and apply following sequence to verify:

- select the correct port in the *Serial Port* list of the dialog depicted above,
- verify the settings with the *Verify* button (status must by 0x00000000) and
- apply the settings using the *Apply* button.

The dialog stores the port setting in a file after they have been applied. It is therefore not required to enter this section upon each program start when the external hardware configuration hasn't been changed.

Note: Even if ports seem to be available, other application can lock requiring access to external devices, such as PDA's, phones or modems. In this case, the program reports an error upon port verification. Select another port or terminate the application locking the resource.

2.2.2 Open and Close Button

After successfully configuring, the serial interface the link can be established by pressing the Open button.

Vice versa the Close Button disconnects from the peripheral. This operation is performed automatically when the program exits.

The status line reflects the state by displaying an "O" instead of a "C".

Errors are indicated by a message in the log window.

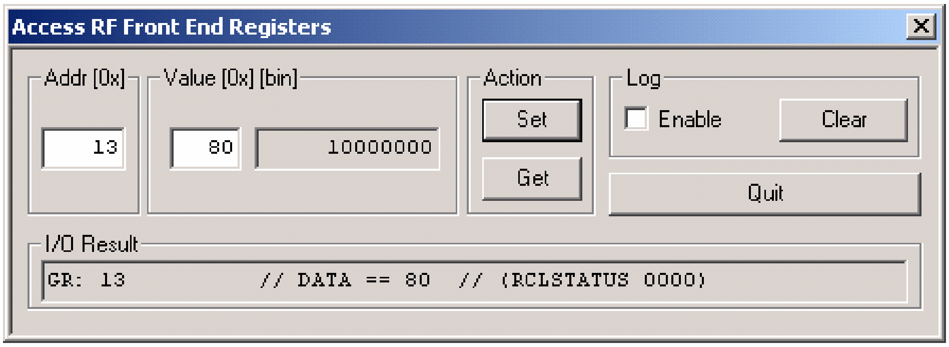
2.3 Action Menu

Table 3. Action Menu Commands

Operating Mode	Description
Access Register	Opens a dialog to configure the RS232 port
Select Command File	Opens the defined port
Process Command File	Closes the defined port
Change Host Baudrate	

2.3.1 Access Register Dialog

This dialog can be accessed when pressing *Access Register* button or the *Register I/O* button. Fig 3 shows the dialog to access individual registers of the reader IC.



Addr [0x]: 13

Value [0x] [bin]: 80 10000000

□ Enable

```
GR: 13          // DATA == 80   // (RCLSTATUS 0000)
```

Fig 3. *Access RF Front End Register* Dialog

When the port has been successfully opened, the program is ready to perform register operations like:

- Specify the register address in the *Addr[0x]* edit box.
- For writing data to the peripheral, specify the value to write to a specific register in the *Value[0x][bin]* section.
- Access registers with the *Set* or *Get* buttons.
- If the *Enable* box in the *Log* section is checked, the I/O result is not only visible in the dialog's *I/O Result* field but also in the main window's message log.

Please note that the MFRC52x supports paged and non-paged register access. Register 0x00 specifies the addressing mode.

2.3.2 Select Command File Button

A unique feature of the program is the built-in line parser which is capable of processing ASCII text files containing command lines. To get the line parser working it is required to select a command file by pressing *Select Command File* button or *Select JCF* button. A file pop-up allows to select a .jcf file (Command File).

2.3.3 Process Command File Button

When pressing *Process Command File* button or the *Process JCF* button, the selected command file will be executed.

If no file has been selected, the program shows the file selection dialog to select a file to execute.

Note: You can change the file name directly by typing into the *Override Name* text box.

The parser supports commands for both PC (host) and hardware control. Two-letter commands are used to control the reader IC and three-letter commands are used to control the host. Following Table 4, Table 5 and Table 6 show an overview about the command set.

Table 4. Hardware Related Commands

Command	Synopsis	Description
SR	SR <address> <data>	The SR function sets a 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 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 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 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 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.

Table 5. Host Related Commands

Command	Synopsis	Description
CHB	CHB <bitrate>	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

Command	Synopsis	Description
		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.
BRK	BRK	The BRK function stops the execution of the current script.
SAV	SAV [<File Name>]	The SAV function stores the log output to the current working directory. The File Name is used if present (max. length of 32 char). If no parameter is present a file dialog is opened to specify the location and the name.

Table 6. Marker Related Commands

Register	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 Hardware-Related Commands). This can either be any numerical value in case of register content retrieval or a boolean value pointing out

Register	Scope	Description
		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**.

2.3.4 Script Example

The script “language” is similar to various types of assembly language. The built-in line parser steps through the lines of the script which implies that each line can hold only one command. If a line has more than one command, only the first one is executed, all subsequent instructions are ignored.

The example in Fig 4 illustrates the usage of a subset of commands available for scripting. Written into a standard-ASCII text file of type .jcf, the script can be loaded and executed by PCSerial.

```
...
//> *** Write a byte into the FIFO, read it back and increment result:
SR 09 AB
// Error in accessing Hardware ?
JNE IOE 0 FWD
GR 09
// Value is now in IOR (I/O Result):
INC IOR
//> Content of IOR (Must be 0xAC):
//# IOR

//> *** Sample loops, nested (8*7*6 turns):
MOV ML0 8
::: OUTER_LOOP
    DEC ML0
    //# ML0

    MOV ML1 7
    ::: MID_LOOP
        DEC ML1

        MOV ML2 6
        ::: INNER_LOOP
            DEC ML2
        JNE ML2 0 INNER_LOOP
    JNE ML1 0 MID_LOOP
JNE ML0 0 OUTER_LOOP

// End of the script (and jump destination in case of error):
:::FWD
//> END.
```

Fig 4. Script Example

2.3.5 Change Host Baudrate Button

This function allows to set the host to either 9600 bps or 115000 bps. This function could be used after a RF-Reset to switch back to 9600 bps after having used the *All to 115kbps* button before.

2.4 Log Menu

The software is capable of logging actions. The log appears in the main window’s text field. The *Log* menu allows to save the log window content to a file or to remove it.

Table 7. Table title here

Operating Mode	Description
Save	Saves the content of the Log window to a file.
Clear	Removes all content from the Log window.

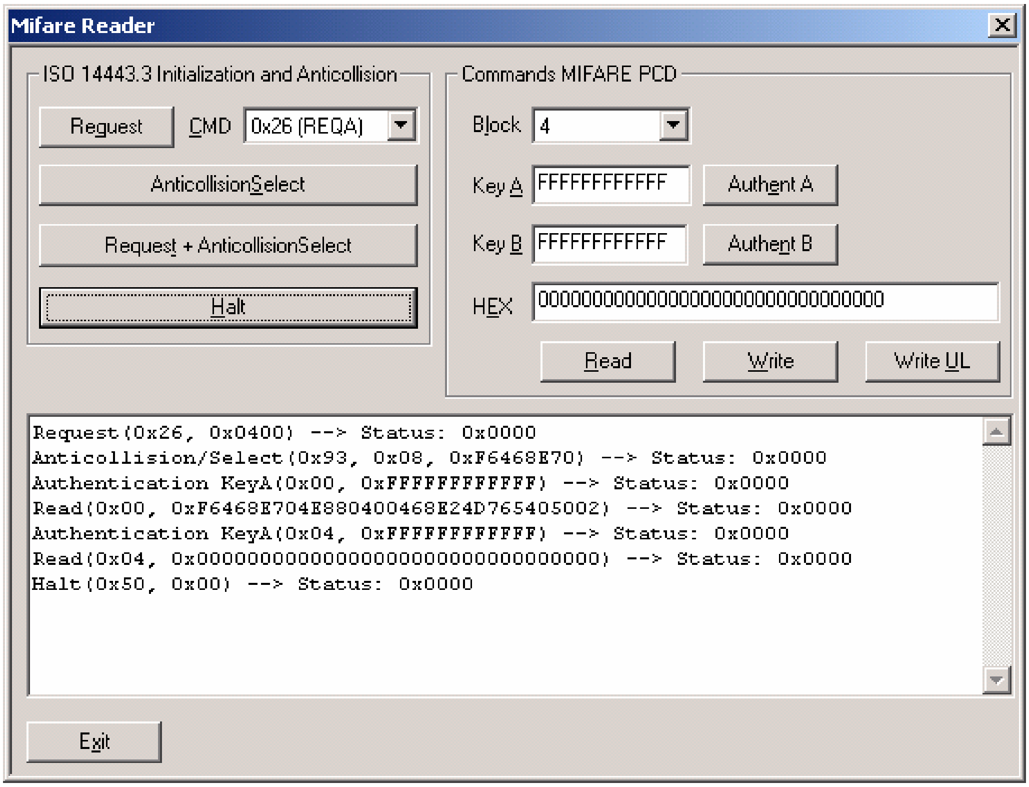
2.5 All to 115kbps Button

The program allows to set both PC (host) and MFRC52x (peripheral) to a serial communication speed of 115200 bps using the *All to 115kbps* button.

Please note that the program performs a register access which means that the peripheral must be already operational at this time. If the command fails to execute properly (error message) the settings are not applied.

2.6 Mifare Reader Button

This feature implements a subset of the MIFARE Reader functionality and is activated via the Mifare Reader button in the main GUI. Fig 5 shows the Mifare Reader dialog.



CMD 0x26 (REQA)

Block 4

Key A FFFFFFFFFFFF

Key B FFFFFFFFFFFF

HEX 00000000000000000000000000000000

```
Request(0x26, 0x0400) --> Status: 0x0000
Anticollision/Select(0x93, 0x08, 0xF6468E70) --> Status: 0x0000
Authentication KeyA(0x00, 0xFFFFFFFFFFFF) --> Status: 0x0000
Read(0x00, 0xF6468E704E880400468E24D765405002) --> Status: 0x0000
Authentication KeyA(0x04, 0xFFFFFFFFFFFF) --> Status: 0x0000
Read(0x04, 0x00000000000000000000000000000000) --> Status: 0x0000
Halt(0x50, 0x00) --> Status: 0x0000
```

Fig 5. Mifare Reader Dialog

In order to operate a MIFARE (Standard or Ultra Light) card it is required to step through *Request* and *AnticollisionSelect*. After successfully performing this ISO 14443-3 procedure the card is ready to receive MIFARE 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 / read from the card (PICC).

2.7 Error and Status Messages

For communication commands with PICC's and special commands for the IC, there is a set of success/error codes.

The messages are grouped into following categories:

- Success Indicator,
- Communication Errors between Reader and Card,
- Interface Errors on each component's interface,
- MIFARE Protocol Errors,
- ISO/IEC 14443 Part 3 Errors,
- other Errors.

2.7.1 Success Messages

RCLSTATUS_SUCCESS (0x0000) - Returned in case of no error when there isn't any more appropriate code.

2.7.2 Communication Error/Status Messages

RCLSTATUS_IO_TIMEOUT (0x0001) - No reply received, e.g. PICC removal.

RCLSTATUS_CRC_ERROR (0x0002) - Wrong CRC detected by RC or library.

RCLSTATUS_PARITY_ERROR (0x0003) - Parity error detected by RC or library.

RCLSTATUS_BITCOUNT_ERROR (0x0004) - Typically, the RC reports such an error.

RCLSTATUS_FRAMING_ERROR (0x0005) - Invalid frame format.

RCLSTATUS_COLLISION_ERROR (0x0006) - Typically, the RC repors such an error.

RCLSTATUS_BUFFER_TOO_SMALL (0x0007) - Communication buffer size insufficient.

RCLSTATUS_ACCESS_DENIED (0x0008) - Access has not been granted (readonly?).

RCLSTATUS_BUFFER_OVERFLOW (0x0009) - Attempt to write beyond the end of a buffer.

RCLSTATUS_PROTOCOL_ERROR (0x000B) - Mifare start bit wrong, buffer length error.

RCLSTATUS_ERROR_NY_IMPLEMENTED (0x000C) - Feature not yet implemented.

RCLSTATUS_FIFO_WRITE_ERROR (0x000D) - Error caused because of interface conflict during write access to FIFO.

RCLSTATUS_USERBUFFER_FULL (0x000E) - The user buffer is full, the calling application/routine gets the chance to save user buffer data and start over.

2.7.3 Interface Error/Status Messages

RCLSTATUS_INVALID_PARAMETER (0x0101) - Parameter is invalid (range, format).

RCLSTATUS_UNSUPPORTED_PARAMETER (0x0102) - Parameter value/format is correct but not supported in the current configuration.

RCLSTATUS_UNSUPPORTED_COMMAND (0x0103) - The device does not support the command.

RCLSTATUS_INTERFACE_ERROR (0x0104) - Host-peripheral interface error.

RCLSTATUS_INVALID_FORMAT (0x0105) - The data format does not match the spec.

RCLSTATUS_INTERFACE_NOT_ENABLED (0x0106) - This interface is currently(!) not supported (e.g. function ptr. to NULL).

RCLSTATUS_UNKNOWN_HARDWARE_TYPE (0x0107) - The chosen hardware for configurations is not known by the initialisation function. The default hardware (HW_1) is used instead.

2.7.4 Mifare Error/Status Messages

RCLSTATUS_AUTHENT_ERROR (0x0201) - Authentication failure (e.g. key mismatch).

RCLSTATUS_ACK_SUPPOSED (0x0202) - Single byte or nibble received, CRC error detected, possibly MF (N)ACK response.

RCLSTATUS_NACK_RECEIVED (0x0203) - NACK detected.

2.7.5 ISO 14443-3 Error/Status Messages

RCLSTATUS_WRONG_UID_CHECKBYTE (0x0501) - UID check byte is wrong.

RCLSTATUS_WRONG_HALT_FORMAT (0x0502) - HALT Format error.

2.7.6 Miscellaneous Error/Status Messages

RCLSTATUS_OTHER_ERROR (0x7E01) - Unspecified, error, non-categorised.

RCLSTATUS_INSUFFICIENT_RESOURCES (0x7E02) - The system runs low of resources!

RCLSTATUS_INVALID_DEVICE_STATE (0x7E03) - The (sub-)system is in a state which does not allow the operation.

RCLSTATUS_DEVICE_TEMP_ERROR (0x7E04) - Temperature error indicated by MFRC52x HW.

3. Legal information

3.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

3.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

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

3.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

3.4 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> — owned by <Company name>

3.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP B.V.

4. Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section ‘Legal information’.

For more information, please visit: http://www.nxp.com.
For sales office addresses, email to: salesaddresses@nxp.com.

Date of release: 07 July 2007

Document identifier: 119221

founded by PHILIPS