APPLICATION NOTE

mifare®

MF RC500

Basic Function Library

Revision 2.0

March 2001

PUBLIC

Philips Semiconductors

PHILIPS

CONTENTS

MIFARE is a registered trademark of Philips Electronics N.V.

1 GENERAL INFORMATION

1.1 Scope

This document describes the MF RC500's Basic Function library. Implementing this library into a specific µController environment gives access a MIFARE® Classic card.

A complementary document '*MIFARE® MF RC500 Highly Integrated ISO 1443A Reader IC'* describes the functionality and electrical specifications of this IC.

1.2 General Description

The MF RC500 is a combined digital and analogue reader IC for the communication with ISO 14443A (MIFARE®) compatible contactless smart card ICs. It supports both, the MIFARE® Classic protocol (used with MIFARE® Standard, Light) and the open protocol T=CL as proposed for ISO 14443A-4 for contactless µController cards.

The MF RC500 is designed to work in a µController environment and provides the functionality needed to access contactless cards to such a system. For the connection to a µController system the MF RC500 utilises a parallel interface which supports various kinds of 8-bit busses.

1.3 Available Libraries

As an example how to access the MF RC500 in a proper way in order to work with MIFARE® cards the Basic Function Library is published. It is programmed in the programming language 'C' and allows communication with the MIFARE® cards on a functional level.
This document describes two modules of the library hierarchy. The first module is named MfRc500uC.c and covers the basic MIFARE® functionality. Its interface to the application software is described in MfRc500.h. The Second module contains two files, PcdShared.c and PcdUtils.c and includes the functions to access the MF RC500. Both interfaces to the MfRc500.h module are described in PcdShared.h and PcdUtils.h.

1.4 Programming Environment

In this document the user will find the specification of a Basic Function Library which allows access to the MF RC500 and perform all its functionality correctly. The Low Level Library is programmed in the programming language 'C' and implemented to work on a µC. Not tricky programming technique is used, so the user can make a very easy and straight forward implementation on other controllers.

In this document the user will find a functional description of the available interface. Comments on programming techniques and structures can be found in the inline documentation and header files of the source code.

1.5 File List

The Basic Function Library contains the following files:

File	Content
MfRc500Reg.h	Register flag definitions for the MF RC500
MfRc500.h	Header for Mfrc500uc.c
MfRc500uc.c	Implementation of the basic MIFARE® functionality
PcdShared.c	basic MF RC 500 functions
PcdShared.h	header for PcdShared.c
PcdUtils.c	basic internal MF RC500 functions
PcdUtils.h	header for PcdUtils.c

2 HANDLING THE MIFARE® WORLD

Cards of the MIFARE® Classic family (MIFARE® Standard, MIFARE® Light) support a defined set of instructions. The MF RC500 fully supports communication with these cards. Additionally, MIFARE® Classic instructions have to be sent to the card in correct sequences. To apply these sequences in the appropriate way is the responsibility of the application software.
For further information on the cards command set please refer to the according product description of the MIFARE Standard or the MIFARE Light IC.

2.1 MIFARE® Classic Command Set

The MIFARE® Classic command set can be divided in to 2 parts . The identification and selection procedure of the MIFARE® protocol is implemented in an ISO14443A compliant way. These commands are marked with a grey background in the following table. Having identified and selected the MIFARE® card the MIFARE® specific authentication procedure can be started. Finally, having passed the authentication procedure memory operations are allowed.

Command	Abbr.	Code	Argument	Response	Possible After
Request ALL	ATR	52	None	Tag Type (ATQ)	card's POR, HALT, communication failure
Request IDLE	ATR	26	None	Tag Type (ATQ)	card's POR, communication failure
Anticollision	AC	93,95, 97	(optional parts of the card's serial number)	(rest of) card's serial number	ATR, AC
Select	SEL	93,95, 97	Card serial number	Answer to select (ATS)	ATR, AC
Authentication	AUT	60 61	Block address	Acknowledge	SEL, AUT, RD, WR, TRANS
Read	RD	30	Block address	16 byte data block	SEL*), AUT, RD, WR, TRANS
Write	WR	A0	Block address and 16 byte data block	Acknowledge	SEL*), AUT, RD, WR, TRANS
Decrement	DEC	C0	Block address and 4 byte value	Acknowledge	SEL*), AUT, RD, WR, TRANS
Increment	INC	C1	Block address and 4 byte value	Acknowledge	SEL*), AUT, RD, WR, TRANS
Restore	REST	C2	Block address and 4 byte dummy value	Acknowledge	SEL*), AUT, RD, WR, TRANS
Transfer	TRANS	B0	Block address	Acknowledge	DEC, INC, REST
Halt	HALT	50	Dummy address	None	SEL, AUT, RD, WR, TRANS

A command can be executed successfully only if it is carried out after a function listed in the column 'Possible After'. Otherwise a failure is returned and the card falls back into the initial state.

*) Although the command might be executed after a SEL command, it will fail since the card is not authenticated.

2.1.1 REQUEST ALL / IDLE

If a MIFARE® card enters the operating field the card is set into its initial state. The very first command a MIFARE® card accepts in this initial state is a 'Request' command. This command is just seven bits long. The MIFARE® card accepts two different request codes.
A 'Request All' (52_{hex}) to which all cards in the initial state respond.
A 'Request Idle' (26_{hex}) to which only cards respond that were not set into the halt state before.

The MIFARE® card responds with its card type depending tag-type (ATQ according to ISO 14443).

Relevant functions: Mf500PiccRequest()
Mf500PiccCommonRequest()

2.1.2 ANTICOLLISION

After a successful request, the MIFARE® card accepts an 'Anticollision' command (93_{hex}) and returns its serial number. If there are more than one card within the operating field, the MF RC500 automatically starts an internal procedure resulting in a serial number of one card.
To support the contactless interface Standard ISO 14443, further collision codes (95_{hex}, 97_{hex}) for extended serial numbers are implemented.

Relevant functions: Mf500PiccAnticoll()
Mf500PiccCascAnticoll()

2.1.3 SELECT

After a successful 'Request' or 'Anticollision' command, the MIFARE® card accepts an 'Select' command. Due to the 'Select' command only the card with the specified serial number is selected and will continue the communication with the MF RC500. All other cards are not longer involved during the communication and will fall back to the initial state again.
The card responds to a 'Select' command with a card type specific code called 'answer to select' (ATS).
To support the contactless interfaGce Standard ISO 14443, further select codes (95_{hex}, 97_{hex}) for extended serial numbers are implemented.

Relevant functions: Mf500PiccSelect()
Mf500PiccCascSelect()

2.1.4 AUTHENTICATION

To get access to the MIFARE® Classic card's EEPROM and the stored data, the card has to be authenticated. Therefore, an authentication procedure has to be started which is implemented completely inside the MF RC500. The user has to address the appropriate keys in the key memory. If these keys match to the keys stored in the card's sector trailer further access to the card is allowed.
There are three possibilities to store the master keys for authentication. Keys can be stored by the uC and passed to the RC500 with Mf500PiccAuth(). Keys can be passed directly to the MF RC500 by calling the authentication function Mf500PiccAuthKey(), or keys can be stored in the internal non volatile key memory of

the MF RC500 used for authentication with Mf500PiccAuthE2(). To be able to load the keys in the EEPROM the function Mf500PcdLoadkeyE2 has to be used. Furthermore the keys have to be stored in a specific format. Therefore the function Mf500HostCodeKey has to be taken.

Relevant functions: Mf500PiccAuth()
Mf500PiccAuthE2()
Mf500PiccAuthKey()
Mf500HostCodeKey()
Mf500PcdLoadKeyE2()

Note: These commands are part of the MIFARE® Classic protocol. It is not applicable with part 4 of ISO 14443.

2.1.5 READ

Having passed authentication procedure the 'Read' command reads out the card EEPROM. Depending on the card IC either a complete block of 16 bytes (MIFARE® Standard IC) or 2 pages of 4 bytes each and 8 dummy bytes (MIFARE Light IC) are read by the reader.
If the content of a sector trailers is read out the keys are masked to zero by the card, because keys could not be read back.
In addition to a valid authentication of the MIFARE® Classic card, data can only be read if the card's access condition for that block allows it.

Relevant function: Mf500PiccRead()

Note: This Command is part of the MIFARE® Classic protocol. It is not applicable with part 4 of ISO 14443.

2.1.6 WRITE

Having passed authentication procedure the 'Write' command writes data to the card EEPROM if the access conditions allows it. Depending on the card IC either a complete block of 16 bytes (MIFARE® Standard IC) or 1 page of 4 bytes (MIFARE Light IC) is written to the card ICs EEPROM.
In addition to a valid authentication of the MIFARE® Classic card, data can only be read if the card's access condition for that block allows it.

Relevant function: Mf500PiccWrite()

Note: Special care has to be taken when writing to sector trailers on the card. These blocks hold the card's keys and the access conditions. Cards should be presented in a short fixed distance to the MIFARE® reader antenna during writing the card's sector trailers.

Note: This Command is part of the MIFARE® Classic protocol. It is not applicable with part 4 of ISO 14443.

2.1.7 INCREMENT

The 'Increment' command reads the addressed value block of a MIFARE® Standard card IC, checks the data structure and increases the content of the value block by the transmitted value and stores the result in the card's internal register. No write operation to the EEPROM is done.
This command is not implemented as a single function, the implemented function MfPiccValue includes all value format related functions.

Relevant functions MfPiccValue()

Note: This Command is part of the MIFARE® Classic protocol. It is not applicable with part 4 of ISO 14443.

2.1.8 DECREMENT

The 'Decrement' command reads the addressed value block of a MIFARE® Standard card IC, checks the data structure and decreases the content of the value block by the transmitted value and stores the result in the card's internal register. No write operation to the EEPROM is done.
This command is not implemented as a single function, the implemented function MfPiccValue includes all value format related functions.

Relevant function: MfPiccValue()

Note: This Command is part of the MIFARE® Classic protocol. It is not applicable with part 4 of ISO 14443.

2.1.9 RESTORE

The 'Restore' command reads the addressed value block of a MIFARE® Standard card IC, checks the data structure and stores the content of the value block in the card's internal register. No write operation to the EEPROM is done.
This command is not implemented as a single function, the implemented function MfPiccValue includes all value format related functions.

Relevant function: MfPiccValue()

Note: This Command is part of the MIFARE® Classic protocol. It is not applicable with part 4 of ISO 14443.

2.1.10 TRANSFER

The ‘Transfer’ command transfers the content of the cards internal register to the EEPROM address. This function can only be called after the increment, decrement or restore function.
This command is not implemented as a single function, the implemented function MfPiccValue includes all value format related functions.

Relevant function: MfPiccValue()

Note: This Command is part of the MIFARE® Classic protocol. It is not applicable with part 4 of ISO 14443.

2.1.11 HALT

The ‘Halt’ command sets the MIFARE® card into a special state, the halt state. To activate the MIFARE® card again, a power down reset (leaving and re-entering the field) or the ‘Request All’ command has to be used.

Relevant function: Mf500PiccHalt()

3 ISO 14443A COMMAND SET

The MIFARE® Classic command set includes partly the ISO/IEC14443A command set. Common commands are the *Request*, *Anticollision*, *Select* and the *Halt* command, which are described in the MIFARE® command set.
Additionally commands are described to activate a PICC in the IDLE mode, to activate a PICC in the field and to exchange data blocks as described in the open protocol definition of the ISO 14443A .

3.1.1 COMMON REQUEST

Performs either a Request command (Request idle) or a wake up (Request all) command. For detailed information refer to 2.1.1.

Relevant function: MfPiccCommonRequest()

3.1.2 CASCADED ANTICOLLISION

Performs the anticollision routine for cascaded serial numbers. For detailed information refer to 2.1.2.

Relevant function: MfPiccCascAnticollision()

3.1.3 CASCADED SELECT

Performs the select command for cascaded serial numbers. For detailed information refer to 2.1.3.

Relevant function: MfPiccCascSelect()

3.1.4 ACTIVATE WAKEUP

This function activates one PICC in the field by performing a Request-All, Anticollision, Select sequence and change its state from IDLE to ACTIVE state. Cascaded serial numbers are handled correctly.

Relevant function: Mf500PiccActivateWakeup()

3.1.5 ACTIVATE IDLE

This function activates one PICC in the field by performing a Request-IDLE, Anticollision, Select sequence and change its state from IDLE to ACTIVE state. Cascaded serial numbers are handled correctly.

Relevant function: Mf500ActivateIdle()

3.1.6 EXCHANGE BLOCKS

This function gives the user the possibility to exchange data blocks between the PCD and the PICC.
The data blocks are exchanged from the PCD to the PICC and back to the PCD.

Relevant function: Mf500PiccExchangeBlock()

4 SYSTEM CONFIGURATION AND CONTROL

4.1 Configurations settings of the MF RC500

It is necessary to configure MF RC500' registers before starting to work with it. These are referring to blocks in the MF RC500 like the RF interface and Interface Definition.

Relevant function: Mf500PcdConfig()

4.2 Configuration settings for the active antenna concept

The MF RC500 can be configured to be used in an active antenna concept. Therefore configuration settings for a master and slave MF RC 500 have to be done.

Relevant functions: MfActiveAntennaMasterConfig()
MfActiveAntennaSlaveConfig()

4.3 Return Values Overview

For communication commands with PICC's and special commands for the reader IC, there is a set of messages, which can always occur and therefore they are not described explicitly at function level.

These error messages are noted as **C**ommon **C**ommunication **R**eturn **C**odes **CCRC** :

Value	Name
0	MI_OK
-1	MI_NOTAGERR
-2	MI_CRCERR CRC
-5	MI_PARITYERR
-19	MI_OVFLERR FIFO
-21	MI_FRAMINGERR
-22	MI_ACCESSTIMEOUT
-24	MI_COLLERR
-100	MI_NY_IMPLEMENTED

List of all error codes defined for Philips MIFARE® Libraries.

Value	Name	Description
0	MI_OK	function was executed without failure
- 1	MI_NOTAGERR	no card in the operating area of the antenna
- 2	MI_CRCERR	card's CRC was not correct
- 3	MI_EMPTY	value overflow during increment or underflow during decrement
- 4	MI_AUTHERR	no card authentication possible due to Crypto I algorithm failure
- 5	MI_PARITYERR	card's parity was not correct
- 6	MI_CODEERR	NACK received, card detected CRC or parity error
- 8	MI_SERNRERR	check byte of card's serial number (got from anticollision) is not correct
-9	MI_KEYERR	Authentication key error

Value	Name	Description
- 10	MI_NOTAUTHERR	card's sector is not authenticated
- 11	MI_BITCOUNTERR	Wrong number of bits received from the card
-12	MI_BYTECOUNTERR	Wrong number of bytes received from the card
-13	MI_IDLE	IDLE command active
- 14	MI_TRANSERR	NACK received after TRANS
- 15	MI_WRITEERR	NACK received after WR
- 16	MI_INCRERR	NACK received after INC
-17	MI_DECRERR	NACK received after DEC
- 18	MI_READERR	NACK received after sending READ
-19	MI_OVFLERR	FIFO overflow error
- 20	MI_POLLING	message after initialising the polling mode
-21	MI_FRAMINGERR	Framing error of the RF interface, start bit not valid
-22	MI_ACCESSERR	Timout accessing the reader module
-23	MI_UNKNOWN_COMMAND	Unknown Command
-24	MI_COLLERR	collision during RF data transfer
-25	MI_RESETERR	Error during reset of the MF RC500
-25	MI_INITERR	Error during initialisation of the MF RC500
-26	MI_INTERFACEERR	MF RC500 is not responding correctly
-27	MI_ACCESSTIMEOUT	MF RC500 is not responding in time
-28	MI_NOBITWISEANTICOLL	card does not support bitwise anticollision
- 50	MI_RECBUF_OVERFLOW	T=CL receive buffer overflow
- 51	MI_SENDBYTENR	T=CL wrong send byte number
- 53	MI_SENDBUF_OVERFLOW	T=CL send buffer overflow
- 54	MI_BAUDRATE_NOT_ SUPPORTED	wrong baud rate selected
- 55	MI_SAME_BAUDRATE_ REQUIRED	receiving and sending with the same baudrate
- 60	MI_WRONG_PARAMETER _VALUE	wrong parameter was passed to the function
- 100	MI_NY_IMPLEMENTED	this function is not available or implemented
- 101	MI_NO_MFRC	the hardware version is not 2.x
- 102	MI_MFRC_NOTAUTH	MF RC500 is not authenticated (and therefore denies this function)
- 103	MI_WRONG_DES_MODE	the selected DES mode does not match the MF RC500 configuration

Value	Name	Description
- 104	MI_HOST_AUTH_FAILED	the MF RC500 rejects the host during host authentication
- 106	MI_WRONG_LOAD_MODE	the mode for master key and access condition loading does not match the MF RC500 configuration
- 107	MI_WRONG_DESKEY	the least significant bits of the DES key bytes are not zero
- 108	MI_MKLOAD_FAILED	master keys or access conditions were rejected by the MF RC
-109	MI_FIFOERR	Error during FIFO handling
- 110	MI_WRONG_ADDR	the specified address is out of range
- 111	MI_DESKEYLOAD_FAILED	DES keys were rejected by the MF RC500
- 114	MI_WRONG_SEL_CNT	this select counter is not valid
- 117	MI_WRONG_TEST_MODE	this test mode is not valid
- 118	MI_TEST_FAILED	the self-test routine failed
- 119	MI_TOC_ERROR	the time out counter did not behave as expected
-120	MI_COMM_ABORT	communication aborted
- 121	MI_INVALID_BASE	the base address is not in the range of 200_{hex} to $3F0_{hex}$
- 122	MI_MFRC_RESET	MF RC500 is busy with start up or it is in reset
- 123	MI_WRONG_VALUE	wrong value passed
-124	MI_VALERR	Value format error
- 149	MI_WRONG_MAC_TOKEN	wrong MAC token passed
- 150	MI_WRONG_TOKEN	TokenAB is not correct, host rejects MF RC500
- 151	MI_NO_VALUE	the data is not a valid value format
- 152	MI_MFRC150	MF RC150 detected
- 153	MI_MFRC170	MF RC170 detected
- 180	MI_WRONG_BASEADDR	wrong base address for the parallel I/O port
- 199	MI_NO_ERROR_TEXT_AVAIL	wrong error number, no error text available

5 FUNCTION REFERENCE

5.1 Function Overview

Name	Description
MIFARE® Card Communication Functions	
Mf500PiccRequest	Request Command defined in ISO14443 (MIFARE®)
Mf500PiccAnticoll	Anti-Collision Command for MIFARE®
Mf500PiccSelect	Select Command defined in ISO14443 (MIFARE®)
Mf500PiccAuth	Card Authentication and Crypto1-Initialising
Mf500PiccAuthE2	Key loading into the MF RC 500's EEPROM
Mf500PiccAuthKey	Authentication with direct key loading from the µC.
Mf500HostKeyCodeKey	Authentication key coding
Mf500PcdLoadKeyE2	Key loadinfg into the MF RC500's EEPROM
Mf500PiccRead	Write to MIFARE® card
Mf500PiccWrite	Read from MIFARE® card
Mf500PiccValue	Value format operations for MIFARE® Standard card ICs
Mf500PiccValueDebit	Value Block operation for PICCs with automatic transfer as MIFARE® Light or Plus
Mf500PiccHalt	Set card in HALT-state
ISO14443 Commands	
Mf500PiccCommonRequest	Request Command defined in ISO14443 (MIFARE®)
Mf500PiccCascAnticoll	Cascaded Anti-Collision Command defined in ISO14443 (MIFARE®)
Mf500PiccCascSelect	Cascaded Select Command defined in ISO14443 (MIFARE®)
Mf500PiccActivateWakeup	Activates a PICC in the RF field
Mf500PiccActivate Idle	Activates a PICC in Idle mode
Mf500PiccExchangeBlock	Exchange data blocks
PCD Configuration	
Mf500PcdConfig	Configures the MF RC500
Mf500PcdActiveAntennaMasterConfig()	Active antenna master configuration of the MF RC500
Mf500PcdActiveAntennaSlaveConfig()	Active antenna master configuration of the MF RC500

5.2 Function Reference

The function reference is given in alphabetic order.

5.2.1 MF500ACTIVEANTENNAMASTERCONFIG ()

Active Antenna Master IC configuration

Syntax:

Mf500ActiveAntennaMasterConfig (void)

Description:

This function initialises the master reader IC to use it in an active antenna configuration. This function is additional to the standard configuration "Mf500PcdConfig". The MF RC500 reader IC configured in the master configuration is able to communicate with another MF RC500 configured in the slave configuration via the digital ‘MFin’ and ‘MFout’ pins. The corresponding slave configuration routine for the slave MF RC 500 can be initialised by the function *MF500ActiveAntennaSlaveConfig*. .

Parameters:

none

Return Value:

CCRC

5.2.2 MF500ACTIVEANTENNASLAVECONFIG ()

Active Antenna Slave configuration.

Syntax:

Mf500ActivateAntennaSlaveConfig (void);

Description:

The MF RC500 reader IC configured in the slave configuration is able to communicate with another MF RC500 configured in the master configuration via the digital 'MFIn' and 'MFOut' pins. The master MF RC500 reader IC sends commands and data using the 'MFOut' pin. The slave reader IC receives the data via 'MFIn' pin. Sending data back from the slave IC is done connecting the 'MF Out' for the slave IC and 'MFIn' for the master MF RC500.

In this configuration the slave module can not be initialised by the micro-controller because only the "MF In/Out" interface is connected between both MF RC500's.The slave module has to be initialised before the connection is established. During this initialisation the appropriate parameter settings are written to the E2PROM. After POR (power on reset) the IC reads these settings and initialises itself automatically as slave IC.

Additionally, it is possible to connect the slave reader IC to the µC to have the possibility to change the setting in the application later.

Parameters:

none

Return Value:

CCRC

5.2.3 MF500HOSTCODEKEY()

Authentication Key Coding.

Syntax:

void *Mf500HostCodeKey* (unsigned char* uncoded,

unsigned char* coded);

Description:

To pass the authentication procedure a coded master key is needed and this key has to be stored in the MF RC500's internal key buffer. The coding of the 6 byte long uncoded master key to a 12 byte long coded master key is performed using this function.

Parameters:

uncoded (IN) 6 bytes uncoded authentication key

coded (OUT) *12 bytes coded authentication key*

Return Value:

MI_OK

5.2.4 MF500PCDCONFIG ()

Configures the MF RC500

Syntax:

Mf500PcdConfig (void)

Description:

This function has to be called before the first data is written to the MF RC500 to perform the internal configuration. A reset of the MF RC500 is done and several registers are set. Further more, the backward compatible key storage and authentication method are initialised.

Parameters:

none

Return Value:

MI_OK, MI_RESETERR, MI_INTERFACERR

5.2.5 MF500PCDLOADKEYE2()

Key Loading into the MF RC500's EEPROM.

Syntax:

Mf500PcdLoadKeyE2 (unsigned char *key_type*,
unsigned char *sector*,
unsigned char * *uncoded_keys*)

Description:

This function stores the keys in the reader internal E2PROM. After successful loading, these keys are available for the use by function Mf500PiccAuthE2.

Parameters:

key_type	(IN) selects master key A or master key B PICC_AUTHENT1A PICC_AUTHENT1B
sector	(IN) Range: [0..15] key sector number
uncoded_keys	(IN) 6 bytes key values

Return Value:

CCRC, MI_KEYERR

5.2.6 MF500PICCACTIVATEIDLE ()

Activation of a PICC in IDLE mode.

Syntax:

Mf500PiccActivateIdle (unsigned char *br*,
unsigned char * *atq*,
unsigned char * *sak*,
unsigned char * *uid*,
unsigned char * *uid_len*)

Description:

This function performs a 'Request-Idle', 'Anticollision', 'Select' sequence to activate the PICC and change its state from IDLE to ACTIVE state. Cascaded serialnumbers are handled correctly.

Parameters:

br	(*IN*) Baudrate for MIFARE® communication 0 106 kBaud
atq	(*OUT*) Answer to Request
sak	(*OUT*) Select acknowledge
uid	(*OUT*) up to 10 bytes UID
uid_len	(*OUT*) length of the UID

Return Value:

CCRC, MI_BITCOUNTERR, MI_NOBITWISEANTICOLL, MI_BAUDRATE_NOT_SUPPORTED, MI_SERNRERR

5.2.7 MF500PICCACTIVATEWAKEUP ()

Activates one PICC in the field.

Syntax:

Mf500PiccActivateWakeup (unsigned char *br*,
unsigned char * *atq*,
unsigned char * *sak* ,
unsigned char * *uid*,
unsigned char * *uid_len*)

Description:

This function performs a ‘Request-All’, ‘Anticollision’, ‘Select’ sequence to activate the PICC and change its state from IDLE to ACTIVE state. Cascaded serial numbers are handled correctly.

Parameters:

br (*IN*) Baudrate for MIFARE® communication
0 106 kBaud

atq (*OUT*) Answer to Request

sak (*OUT*) Select acknowledge

uid_len (*OUT*) up to 10 bytes UID

uid_len (*OUT*) length of the UID

Return Value:

CCRC, MI_BITCOUNTERR, MI_NOBITWISEANTICOLL, MI_BAUDRATE_NOT_SUPPORTED, MI_SERNRERR

5.2.8 MF500PICCANTICOLL ()

Syntax:

Mf500PiccAnticoll (unsigned char *bcnt*,

unsigned char **snr*);

Description:

This function reads the four byte serial number of one of the cards within the operating field. This function calls MF500PiccCascAnticoll with select_code 0x93 to perform the anitcollision for MIFARE® Classic card ICs .

Parameters:

bcnt (IN) can take a value from 0 to 32. This defines the number of serial number bits sent out with the anticollision command to the card.
If *bcnt* is 0 all cards will return their serial number.
If *bcnts* is 32 only the one card will respond which serial number is equal *snr*.
For other values of *bcnt* only cards respond which first *bcnt* serial number bits correspond to the sent bits.

snr (IN/OUT) 4 bytes serial number (number of bits, which are known and indicated by 'bcnt'). The value is valid only if MI_OK is returned. In addition this parameter can be used to pass forward the *bcnts* starting bits of the serial number.

Return Value:

CCRC, MI_BITCOUNTERR, MI_SERNRERR

5.2.9 MF500PICCAUTH ()

MIFARE® Authhentication

Syntax:

Mf500PiccAuth (unsigned char *auth_mode*,

unsigned char *key_sector*,

unsigned char *block*)

Description:

This function authenticates one card's sector (according to the block address) using the specified master key A or B, addressed with auth_mode. Having send the command to the card the function waits for the card's answer. This function is calling compatible with authentication functions former reader IC's. The keys are stored by the microcontroller, which should be capable for the key management.

Parameters:

auth _mode (IN) selects master key A or master key B

PICC_AUTHENT1A

PICC_AUTHENT1B

key_sector (IN) Range [0..15] specifies the key RAM address from which the data should be taken

block (IN) Range [0..dep.card type] addresses the card's block address on the card, which shall be authenticated. For MIFARE® Standard cards, block addr can take a value from 0 to 63, for other card types please refer to the according product description.

Return Value:

CCRC

5.2.10 MF500PICCAUTHE2()

MIFARE® Authentication with keys stored in the MF RC 500's EEPROM.

Syntax:

Mf500PiccAuthKey(unsigned char auth_*mode*,
unsigned char* snr;
unsigned char key_sector,
unsigned char *block*);

Description:

This function authenticates one card's sector using the specified mode. After sending the command to the card the function waits for the card's answer. The keys for authentication have to be stored at the corresponding location in the E2PROM.

Parameters:

auth_mode (IN) selects master key A or master key B
PICC_AUTHENT1A
PICC_AUTHENT1B

snr (IN) 4 byte serial number of the card, that should be authenticated

key_sector (IN) Range [0..15] specifies the EEPROM address where the keys are stored in the MF RC 500

block (IN) Range [0..dep.card type] addresses the block address on the card, which shall be authenticated. For MIFARE standard cards, block can take a value from 0 to 63, for other card types please refer to the according product description.

Return Value:

CCRC, MI_BITCOUNTERR, MI_KEYERR, MI_AUTHERR

5.2.11 MF500PICCAUTHKEY()

Authentication with direct key loading form the microcontroller.

Syntax:

Mf500PiccAuthKey (

unsigned char *auth_mode*,

unsigned char * *snr*,

unsigned char * *keys*,

unsigned char *sector*)

Description:

This function authenticates one card's sector using keys stored in the µController. The keys are first loaded to the reader module and used for authentication of the specified sector. In order to get the required keys coded, the function Mf500HostCodeKey has to be used .

Parameters:

auth_mode (IN) selects master key A or master key B

PICC_AUTHENT1A

PICC_AUTHENT1B

snr (IN) 4 byte serial number of the card, which should be authenticated

keys (IN) 12 bytes coded master keys for card authentication

block (IN) Range [0..dep.card type] addresses the card's block address, which shall be authenticated. For MIFARE® Standard cards, block can take a value from 0 to 63, for other card types please refer to the according product description.

Return Value:

CCRC, MI_BITCOUNTERR, MI_KEYERR, MI_AUTHERR

5.2.12 MF500PICCCASCANTICOLL()

PICC Anticollision Command for ISO 14443 A-4 Command set.

Syntax:

Mf500PiccCascAnticoll(unsigned char *select_code*,

unsigned char *bcnt*;

unsigned char* *snr*);

Description:

Corresponding to the specification in ISO 14443 A, this function handles extended serial numbers. Therefore more than one select_code is possible. The function transmits a select code and all ready tags are responding. The return value of this function will be the serial number of one PICC.

Parameters:

select_code	(IN)	0x93 standard select code
		0x95 cascaded level 1
		0x97 cascaded level 2
bcnt	(IN) Range: [0..32] Number of SNR-bits that are known (default value is 0);	
snr	(IN/OUT) 4 bytes serial number (number of bits, which are known and indicated by bcnt	

Return Value:

CCRC, MI_BITCOUNTERR, MI_SERNRERR

5.2.13 MF500PICCCASCSELECT()

PICC Select Command for ISO 14443 A-4 Command set.

Syntax:

Mf500PiccCascSelect(unsigned char *select_code*,
unsigned char* *snr*;
unsigned char* *sak*);

Description:

This functions selects a UID level, depending on select code and returns a Select Acknowledge byte. Corresponding to the specification in ISO 14443 A, this function is able to handle extended serial numbers. So that more than one select_code is possible. Relevant bit positions in SAK are 6 and 1. All other bit positions are RFU.
Valid combinations are:

XX1XX0XX UID complete, ATS available

XX0XX0XX UID complete, ATS not available

XXXXX1XX UID not complete

Parameters:

select_code (IN) 0x93 standard select code
0x95 cascaded level 1
0x97 cascaded level 2

snr (IN) 4 bytes serial number

sak (OUT) 1 byte select acknowledge

Return Value:

CCRC, MI_BITCOUNTERR

5.2.14 MF500PICCCOMMONREQUEST ()

PICC Request command for ISO 14443 A-4 Command set.

Syntax:

Mf500PiccCommonRequest (unsigned char *req_code*,
unsigned char * *atq*) ,

Description:

PICC Request command for ISO 14443 A-4 Command set.
Please note, that this function has an identical functionality to function Mf500PiccRequest.

Parameters:

req_code (IN) PICC_REQALL Request Code 52hex
PICC_REQIDL Request Code 26hex

atq (OUT) 16 bit ATQ (answer to request). atq[0] .. LSByte; atq[1] .. MSByte

Return Value:

CCRC, MI_BITCOUNTERR

5.2.15 MF500PICCEXCHANGE BLOCKS ()

Exchange data blocks.

Syntax:

Mf500PiccExchangeBlock (
unsigned char * *send_data*,
unsigned char *send_bytelen*,
unsigned char * *rec_data*,
unsigned char * *rec_bytelen*,
unsigned char *append_crc*,
unsigned short *timeout*)

Description:

This function exchanges data blocks between the PCD and PICC.

Parameters:

send_data (IN) Data to be send to the PICC

send_bytelen (IN) bytelength of send data

rec_data (OUT) received data from the PICC

rec_bytelen (OUT) bytelength of the received data

append_crc (IN) enables CRC calculation

timeout (IN) sets the timeout length

Note: if append_crc is enabled, two CRC bytes are included in send_bytelen and rec_bytelen. The received CRC bytes in the receive buffer are always set to zero.

Return Value:

CCRC

5.2.16 MF500PICCHALT()

Sends the PICC into the halt state.

Syntax:

Mf500PiccHalt (void);

Description:

This function sets a MIFARE® Classic compatible card into the halt state. Having send the command to the card, the function does not expect a cards response. Only in case of any error the card sends back a NACK. If the command was successful, the card does not return with an ACK. Thus, the function is successful, if a time out in the MF RC500 is indicated.

Parameters:

none

Return Value:

CCRC

5.2.17 MF500PICCREAD()

Reads out EEPROM data.

Syntax:

Mf500PiccRead (unsigned char *addr*,

unsigned char **data*);

Description:

This function directly reads out a 16 byte block from the specified card's block address *addr*.

Parameters:

addr (*IN*) Range [0..depending oncard type]. Addresses the card's block address from which data shall be read. For MIFARE® Standard cards, *addr* can take a value from 0 to 63 (255 for MIFARE® Pro),for other card types please refer to the according product description.

data (*OUT*) is a pointer to the 16 byte data block read from the card's memory.

Return Value:

CCRC, MI_NOTAUTHERR, MI_BYTECOUNTERR, MI_CODEERR

5.2.18 MF500PICCREQUEST ()

PICC Request command

Syntax:

Mf500PiccRequest (unsigned char *req_code*,

unsigned char* atq);

Description:

This function accesses the reader module and activates sending the REQ code to the MIFARE® card. After sending the command to the card the function waits for the card's answer.
This function has an identical functionality to the Mf500PiccCommonRequest function, which is desribed by ISO 14443A command set. Depending on the Request Code and the state of the cards in the field all cards reply with their Tag-Type synchronously. The time between end of the Request command and start of reply of the card is exactly 8 * 9.44 us long. The Tag-Type field is 16 bits long and only one bit out of 16 is set. When cards with different Tag-Types are in field, the MF RC500 is able to identify all types of cards in the RF-field. Further more, the Tag-Type is used to identify a card with cascaded serial number. Double and Triple serial numbers are possible.
Relevant bit positions LSByte:

[8..7] UID size

00 standard 32bit long UID
01 UID size double (56 bit long)
10 UID size triple

[5..1] if any bit is set, frame anticollision is supported; tag type recognition

The complete MSByte is RFU.

Note: Future cards might also work with other request codes.

Parameters:

rq_code *(IN)* can take the following values

Value	**Meaning**
ALL	Request Code 52_{hex} is sent out to get also a response from cards in halt state.
IDLE	Request Code 26_{hex} is sent out to get a response only from cards that are not in halt state.

atq (*OUT*) 16 bit ATQ (answer to request). Gives back the card type specific 'answer to request' received from the card. This value is valid only if MI_OK is returned. For actual MIFARE® cards this value is 4. , for future card types please refer to the according product description.

The 'atq' shall not be evaluated.

Return Value:

CCRC, MI_BITCOUNTERR

5.2.19 MF500PICCSELECT()

PICC Select Command.

Syntax:

Mf500PiccSelect (unsigned char* *snr*,

unsigned char* *sak*);

Description:

This function selects a card by the specified serial number. All other cards in the field fall back into the idle mode and they are not longer involved during the communication. The actual select procedure is done by the function "Mf500PiccCascSelect", which is called with select_code 0x93.

Parameters:

snr (*IN*) 4 bytes serial number

sak (*OUT*) 1 byte select acknowledge

Return Value:

CCRC, MI_BITCOUNTERR

5.2.20 MF500PICCVALUE ()

PICC Value Block Operation.

Syntax:

Mf500PiccValue (unsigned char *dd_mode*,
unsigned char *addr*,
unsigned char * *value*,
unsigned char *trans_addr*)

Description:

This function performs the INCMENT, DECREMENT or RESTORE command. To be able to perform a value format related operation the data block has to be formatted as value block. For INCREMENT and DECREMENT, the command doesn't write back the value to the memory location directly, but loads the transfer buffer with the increased value, which could be transferred to any authenticated block by the TRANFER command. The RESTORE command loads the transfer buffer with the value stored at datablock address, while the given value is only a dummy value, which only have to be in valid range. With a subsquent TRANSFER command a backup management for Value Blocks is established. After sending the command to the card the function waits for the card's answer. In case of an error Mf500PiccValue() generates a return code according to the MF RC's error flags, otherwise the value is sent to the card and then it waits for a NACK. As an exception in this protocol step only a NACK is sent by the card in case of an error. Thus, the function is successful, if a time out occurs.
After the calculation is done, a TRANSFER is automatically performed to the block address trans_addr. After sending the command to the card the function waits for the card's answer and generates a return code according to the MF RC's error flags. A TRANSFER command is only possible directly after a RESTORE, INCREMENT or DECREMENT command.
The value inside a Value Block is four bytes wide and stored tow times in normal and one time in bit-inverted manner for data security issues. Additionally the initial address of the Value Block is stored two times normal and two times bit-inverted. In case of a backup of a Value Block, this addres contains the original address of the Value Block.
Note: Only positive numbers are alowed for the parameter value.

Parameters:

dd_mode (*IN*) selects the value format related operation

PICC_INC Increment , PICC_DEC Decrement , PICC_REST Restore

addr (*IN*) Range [0..dep.card type]. Addresses the card's data block address. The card IC internally reads the stored value and takes it as initial value for the calculation. For MIFARE® standard cards, ***addr*** can take a value from 0 to 63,for MIFARE® Pro cards, addr can take values from 0 to 255), for other card types please refer to the according product description.

value (*IN*) is a pointer to a 4 byte positive value.

trans_addr (*IN*) Range [0..dep.card type] .Addresses the card's block address to which the result of the calculation shall be transferred. For MIFARE® standard cards, ***trans_addr*** can take a value from 0 to 63 (255 for Mifare Pro), for other card types please refer to the according product description.

Return Value: CCRC, MI_BITCOUNTERR, MI_NOTAUTHERR, MI_VALERR, MI_CODEERR

5.2.21 MF500PICCVALUEDEBIT ()

PICC Value Block Operation for Cards with automatic transfer.

Syntax:

```
Mf500PiccValueDebit (   unsigned char  dd_mode,
                        unsigned char addr,
                        unsigned char * value);
```

Description:

This function executes calculations on value debit blocks with cards, that support automatic transfer (MIFARE light, MIFARE PLUS, MIFARE PRO, MIFARE PROX, ...). After sending the command to the card the function waits for the card's answer. In case of an error it generates a return code according to the MF RC's error flags.

Parameters:

dd_mode	(IN) PICC_DECREMENT only decrement operations are allowed
addr	(IN) Range [depends on card type] address of the block on the card with which calculation shall be performed. A valid address range can be obtained from the card description.
value	(IN) is a pointer to a 4 byte positive value.

Return Value:

CCRC, MI_BITCOUNTERR, MI_NOTAUTHERR, MI_VALERR, MI_CODEERR

5.2.22 MF500PICCWRITE()

Writes data to the cards EEPROM:

Syntax:

Mf500PiccWrite (unsigned char *addr*,

unsigned char **data*);

Description:

This function writes a 16 byte block to the specified card's block address addr. Having send the command the card indicates with an ACK, that the direct memory access is possible. Having received the ACK, the MF RC500 sends the 16 bytes data block and waits for an ACK again. In case of an error a return code according to the MF RC500's error flags is generated..

Parameters:

addr (IN) Range [0..dep.card type] Addresses the card's block address to which data shall be written. For MIFARE® Standard cards, addr can take values from 0 to 63 (255 for MIFARE® Pro), for other card types please refer to the according product description.

data (OUT) is a pointer to the 16 byte data block, which should be written to the card

Return Value:

CCRC, MI_NOTAUTHERR, MI_BITCOUNTERR, MI_WRITEERR, MI_CODEERR

6 EXAMPLE PROGRAMS

6.1 General

The following program samples show the usage of functions defined in the basic function library and give an overview about the functionality of the MF RC500.

6.2 Program Samples

6.2.1 HOW TO INITIALIZE THE PCD

```
//*****************************************************************************
signed int CALL_CONV InitReader(void)
//*****************************************************************************
{
  signed int status = MI_OK;
  if ((status = Mf500PcdConfig()) == MI_OK )
  {
     printf("\nConfiguration done\n”);
     status = MI_OK;
  }
  return (status);
}
```

6.2.2 HOW TO TURN OFF THE RF-FIELD

```
//*****************************************************************************
signed int SwitchOff(void)
//*****************************************************************************
{
      return(PcdRfReset(0));
}
```

6.2.3 HOW TO RESET ALL CARDS IN THE RF-FIELD

```
//*****************************************************************************
signed int ResetAllCards(void)
//*****************************************************************************
{
      return(PcdRfReset(50); // turn off for 50 * 50 microseconds
}
```

6.2.4 HOW TO ACCESS A MIFARE® STANDARD CARD

The following function demonstrates MIFARE® Classic Standard card access to every sector. Authentication is performed either with key A or key B.

```
//*****************************************************************************
signed int CardAccessLoop(void)
//*****************************************************************************
{
  unsigned char i = 0;
  unsigned int j = 0;
  signed int   failure = 0;
  unsigned char keytype[6] = { KS0|KEYA, KS1|KEYA, KS2|KEYA,
                               KS0|KEYB, KS1|KEYB, KS2|KEYB};

  resetCard();
  failure = 0;
  for(i=0; i<16; i++) // for every master key addres within one set
  {
     for(j=0; j<6; j++) // for every combination between key set and key A or B
     {
        if ((failure = accessCard(i, keytype[j])) != 0)
            break;
     }
     if (failure)
```

```
        break;
  }
  return(failure);
}

//*****************************************************************************
signed int accessCard(unsigned char sector, unsigned char ks)
//*****************************************************************************
{
    unsigned char tt[2];
    unsigned char snr[4];
    unsigned char size;
    unsigned int i = 0;
    signed   int status = 0;

  for ( i = 0; i < 16; i++ )
  {
     // send a REQUEST command for all cards in the field
     // and use RF unit 0
     if ((status = Mf500PiccRequest(ALL, tt)) != MI_OK)
        break;
     // determine one card out of the field
     if ((status = Mf500PiccAnticoll(0, snr)) != MI_OK)
        break;
     // select this card
     if ((status = Mf500PiccSelect(snr, &size)) != MI_OK)
        break;

     // authenticate the desired sector ( address is passed as an argument
     // to the function. Each sector consists of 4 Blocks
     if ((status = Mf500PiccAuth(ks, snr, sector, 4*sector)) != MI_OK)
        break;
```

```
      // get 16 bytes of random data which should be written to the card
      FillRandom(wdata,16);

      if ((status = Mf500PiccWrite((unsigned char)(4*sector+1), wdata)) != MI_OK)
         break;

      // read the same sector and check, if data is correct
      if ((status = Mf500PiccRead ((unsigned char)(4*sector+1), rdata)) != MI_OK)
         printf ("read error (%02d %1d)",sector,ks);

      // compare read and written data (should be the same)
      if (memcmp (wdata, rdata, 16))
         printf ("(%02d %1d)", sector, ks);MfMsgDisp(MSG_ERR,MfBuffer);

      // put the card into halt mode
      Mf500PiccHalt();

      status = 0;
   }
   return(status);
}
```

7 TERMS AND ABBREVIATIONS

7.1 General

Designation:	Description:
µC	Micro Controller
µP	Micro Processor
AC	Access Conditions: applies for MIFARE® Classic cards. define the rights for the kind of access (read, write, decrement, increment) for each data block on MIFARE® Classic cards.
ACK	Acknowledge: to confirm a communication.
CRC	Cyclic Redundancy Check
Dec	decimal value
E²PROM	Electrically Erasable and Programmable Read Only Memory
FIFO	First In First Out Memory
FWT	Frame Waiting Time: maximum time delay between last bit transmitted by the reader and first bit received from the card's response.
Hex	hexadecimal value
I/O	Input and Output
LSB	Least Significant Bit
Master Key	48 Bit Key for Crypto1 (MIFARE® Classic) to authenticate card sectors
MF	MIFARE®
MIFARE® Standard	Functionality of the contactless smart card IC MF1 ICS50
MSB	Most Significant Bit
NACK	Not acknowledge to confirm a communication or protocol failure.
PC	Personal Computer
PCD	Proximity Coupling Device
PICC	Proximity Integrated Circuit Card
PIO	Programmable Input or Output
POR	Power On Reset
RAM	Random Access Memory
RD (X)	read from register x
RF	Radio Frequency
T=CL	Designation for the transparent protocol for contactless µC ICs as proposed for ISO14443-4.

7.2 Card Commands

Designation:	Description:
AC	Anticollision Procedure
AUT	Authentication of the Card
ATR	Answer To Request
DEC	Decrement Value
HALT	Set Card into Halt State
INC	Increment Value
RD	Read Data Block
REST	Restore Value
SEL	Select
TRANS	Transfer Value
WR	Write Data Block

8 DEFINITIONS

Data sheet status	
Objective specification	This data sheet contains target or goal specifications for product development.
Preliminary specification	This data sheet contains preliminary data; supplementary data may be published later.
Product specification	This data sheet contains final product specifications.
Limiting values	
Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics section of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.	
Application information	
Where application information is given, it is advisory and does not form part of the specification.	

9 LIFE SUPPORT APPLICATIONS

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 customers using or selling these products for use in such applications do so on their own risk and agree to fully indemnify Philips for any damages resulting from such improper use or sale.

10 REVISION HISTORY

Table 1 Revision History

REVISION	DATE	CPCN	PAGE	DESCRIPTION
2.0		-		second published version
1.0		-		Internal Version.

Philips Semiconductors - a worldwide company

Argentina: see South America
Australia: 34 Waterloo Road, NORTHRYDE, NSW 2113,
Tel. +612 9805 4455, Fax. +612 9805 4466
Austria: Computerstraße 6, A-1101 WIEN, P.O.Box 213,
Tel. +431 60 101, Fax. +431 30 101 1210
Belarus: Hotel Minsk Business Centre, Bld. 3, r.1211, Volodarski Str. 6,
220050 MINSK, Tel. +375172 200 733, Fax. +375172 200 773
Belgium: see The Netherlands
Brazil: see South America
Bulgaria: Philips Bulgaria Ltd., Energoproject, 15th floor,
51 James Bourchier Blvd., 1407 SOFIA
Tel. +3592 689 211, Fax. +3592 689 102
Canada: Philips Semiconductors/Components,
Tel. +1800 234 7381
China/Hong Kong: 501 Hong Kong Industrial Technology Centre,
72 Tat Chee Avenue, Kowloon Tong, HONG KONG,
Tel. +85223 19 7888, Fax. +85223 19 7700
Colombia: see South America
Czech Republic: see Austria
Denmark: Prags Boulevard 80, PB 1919, DK-2300 COPENHAGEN S,
Tel. +4532 88 2636, Fax. +4531 57 1949
Finland: Sinikalliontie 3, FIN-02630 ESPOO,
Tel. +3589 61 5800, Fax. +3589 61 580/xxx
France: 4 Rue du Port-aux-Vins, BP 317, 92156 SURESNES Cedex,
Tel. +331 40 99 6161, Fax. +331 40 99 6427
Germany: Hammerbrookstraße 69, D-20097 HAMBURG,
Tel. +4940 23 53 60, Fax. +4940 23 536 300
Greece: No. 15, 25th March Street, GR 17778 TAVROS/ATHENS,
Tel. +301 4894 339/239, Fax. +301 4814 240
Hungary: see Austria
India: Philips INDIA Ltd., Shivsagar Estate, A Block, Dr. Annie Besant Rd.
Worli, MUMBAI 400018, Tel. +9122 4938 541, Fax. +9122 4938 722
Indonesia: see Singapore
Ireland: Newstead, Clonskeagh, DUBLIN 14,
Tel. +3531 7640 000, Fax. +3531 7640 200
Israel: RAPAC Electronics, 7 Kehilat Saloniki St., TEL AVIV 61180,
Tel. +9723 645 0444, Fax. +9723 649 1007
Italy: Philips Semiconductors, Piazza IV Novembre 3,
20124 MILANO, Tel. +392 6752 2531, Fax. +392 6752 2557
Japan: Philips Bldg. 13-37, Kohnan 2-chome, Minato-ku, TOKYO 108,
Tel. +813 3740 5130,Fax. +813 3740 5077
Korea: Philips House, 260-199, Itaewon-dong, Yonsan-ku, SEOUL,
Tel. +822 709 1412, Fax. +822 709 1415
Malaysia: No. 76 Jalan Universiti, 46200 PETALING JAYA, Selangor,
Tel. +60 3750 5214, Fax. +603 757 4880
Mexico: 5900 Gateway East, Suite 200, EL PASO, Texas 79905,
Tel. +9 5800 234 7381
Middle East: see Italy
Netherlands: Postbus 90050, 5600 PB EINDHOVEN, Bldg. VB,
Tel. +3140 27 82785, Fax +3140 27 88399
New Zealand: 2 Wagener Place, C.P.O. Box 1041, AUCKLAND,
Tel. +649 849 4160, Fax. +649 849 7811
Norway: Box 1, Manglerud 0612, OSLO,
Tel. +4722 74 8000, Fax. +4722 74 8341
Philippines: Philips Semiconductors Philippines Inc.,
106 Valero St. Salcedo Village, P.O.Box 2108 MCC, MAKATI,
Metro MANILA, Tel. +632 816 6380, Fax. +632 817 3474
Poland: Ul. Lukiska 10, PL 04-123 WARSZWA,
Tel. +4822 612 2831, Fax. +4822 612 2327
Portugal: see Spain
Romania: see Italy
Russia: Philips Russia, Ul. Usatcheva 35A, 119048 MOSCOW,
Tel. +7095 247 9145, Fax. +7095 247 9144
Singapore: Lorong 1, Toa Payoh, SINGAPORE 1231,
Tel. +65350 2538, Fax. +65251 6500
Slovakia: see Austria
Slovenia: see Italy
South Africa: S.A. Philips Pty Ltd., 195-215 Main Road Martindale,
2092 JOHANNESBURG, P.O.Box 7430 Johannesburg 2000,
Tel. +2711 470 5911, Fax. +2711 470 5494
South America: Rua do Rocio 220, 5th floor, Suite 51,
04552-903 Sao Paulo, SAO PAULO - SP, Brazil,
Tel. +5511 821 2333, Fax. +5511 829 1849
Spain: Balmes 22, 08007 BARCELONA,
Tel. +343 301 6312, Fax. +343 301 4107
Sweden: Kottbygatan 7, Akalla, S-16485 STOCKHOLM,
Tel. +468 632 2000, Fax. +468 632 2745
Switzerland: Allmendstraße 140, CH-8027 ZÜRICH,
Tel. +411 488 2686, Fax. +411 481 7730
Taiwan: Philips Taiwan Ltd., 2330F, 66,
Chung Hsiao West Road, Sec. 1, P.O.Box 22978,
TAIPEI 100, Tel. +8862 382 4443, Fax. +8862 382 4444
Thailand: Philips Electronics (Thailand) Ltd.,
209/2 Sanpavuth-Bangna Road Prakanong, BANGKOK 10260,
Tel. +662 745 4090, Fax. +662 398 0793
Turkey: Talapasa Cad. No. 5, 80640 GÜLTEPE/ISTANBUL,
Tel. +90212 279 2770, Fax. +90212 282 6707
Ukraine: Philips Ukraine, 4 Patrice Lumumba Str., Building B, Floor 7,
252042 KIEV, Tel. +38044 264 2776, Fax. +38044 268 0461
United Kingdom: Philips Semiconductors Ltd., 276 Bath Road, Hayes,
MIDDLESEX UM3 5BX, Tel. +44181 730 5000, Fax. +44181 754 8421
United States: 811 Argues Avenue, SUNNYVALE, CA94088-3409,
Tel. +1800 234 7381
Uruguay: see South America
Vietnam: see Singapore
Yugoslavia: Philips, Trg N. Pasica 5/v, 11000 BEOGRAD,
Tel. +38111 625 344, Fax. +38111 635 777

Published by:
Philips Semiconductors Gratkorn GmbH, Mikron-Weg 1, A-8101 Gratkorn, Austria Fax: +43 3124 299 - 270

For all other countries apply to: Philips Semiconductors, Marketing & Sales Communications, Internet: http://www.semiconductors.philips.com
Building BE-p, P.O.Box 218, 5600 MD EINDHOVEN, The Netherlands, Fax: +3140 27 24825

Philips Semiconductors

PHILIPS