AN12657

Using the RC663 without library

Rev. 1.0 — 7 January 2020 Application note
581610 COMPANY PUBLIC

Document information

Information	Content
Keywords	RC663
Abstract	This document describes how to communicate with the RC663 without a library

Revision history

Rev	Date	Description
1.0	20200107	Initial version

1 Introduction

This document describes how a host processor interacts with the RC663 via an SPI interface.

The behavior of the RC663 is determined by a state machine capable to perform a certain set of commands. By writing a command-code to the command register, the command is executed.

The command register is register 0, all other registers are configuration registers except of the FIFO buffer.

The size of a configuration is 8 bit and the size of the FIFO buffer is 512 byte.

2 Host interface

The CLRC663 supports direct interfacing of various hosts as the SPI, I2C, I2CL and serial UART interface type. The CLRC663 resets its interface and checks the current host interface type automatically having performed a power-up or resuming from power down. The CLRC663 identifies the host interface by the means of the logic levels on the control pins after the Cold Reset Phase. This is done by a combination of fixed pin connections

In the example in section 4, an SPI interface is used.

2.1 SPI interface

The CLRC663 acts as a slave during the SPI communication. The SPI clock SCK has to be generated by the master. Data communication from the master to the slave uses the Line MOSI. Line MISO is used to send data back from the CLRC663 to the master.

NSS (Not Slave Select) enables or disables the SPI interface. When NSS is logical high, the interface is disabled and reset. Between every SPI command, the NSS must go to logical high to be able to start the next command read or write.

2.1.1 Communication protocol

Via the SPI communication the host processor can either read a register or write into a register.

An SPI frame always consists of at least 2 Bytes. The first byte is the address of the register and the second byte is the value written into the register.

The first byte (address byte) also defines if the host wants to read the value of the register or write a new value into the register.

2.1.1.1 Address byte

The LSB (least significant bit) of the address byte defines if the master wants to read the register or write into the register.

- If the LSB is 0 the host want to write
- If the LSB is 1 the host want to read

The 7 MSB (most significant bits) define the register address.

Table 1. Address byte

7	6	5	4	3	2	1	0
address 6	address 5	address 4	address 3	address 2	address 1	address 0	write (0) read (1)
MSB							LSB

2.1.1.2 Write register

Most of the time a write command consists of a 2 byte frame (address byte + data), because for altering the value of a register only 1 byte of data is needed.

But it is possible to have longer frames to, for example, fill the FIFO with more than 1 byte at a time. Then the frame looks like this: address byte + data0 + data1 + … + data n

Table 2. Write register frame

	byte 0	byte 1	...	byte n + 1
MOSI	address byte	data 0	...	data n
MISO	X*	X	X	X

* X …. random value send by the slave.

2.1.1.3 Read register

Most of the time a read command consists of a 2 byte frame (address byte + random value), because for reading the value of a register only 1 byte of data is needed.

But it is possible to have longer frames to, for example, read more than 1 byte at the time from the FIFO. Then the frame looks like this: address byte + register address 1 + register address 2 + … + register address n + random value.

Table 3.

	byte 0	byte 1	….	byte n	byte n + 1
MOSI	address byte	register address 1	….	register address n	random value (00h)
MISO	X*	Data of register address 0	….	Data of register address n-1	Data of register address n

X …. random value send by the slave

3 RC663 command set

By writing a command-code to the command register, the command is executed.

Arguments and/or data necessary to process a command, are exchanged via the FIFO buffer.

- Each command that needs a certain number of arguments will start processing only when it has received the correct number of arguments via the FIFO buffer.
- The FIFO buffer is not cleared automatically at command start. It is recommended to write the command arguments and/or the data bytes into the FIFO buffer and start the command afterwards.
- Each command may be stopped by the host by writing a new command code into the command register e.g.: the Idle-Command.

4 out of 17 commands are described in this section. Only these 4 commands are used in the example in section 4.

3.1 Idle command

Command (00h);

This command indicates that the CLRC663 is in idle mode. This command is also used to terminate the actual command.

3.2 Transceive command

Command (07h);

This command transmits data from FIFO buffer and automatically activates the receiver after a transmission is finished. Each transmission process starts by writing the command into CommandReg.

3.3 LoadProtocol command

Command (0Dh), Parameter1 (Protocol number RX), Parameter2 (Protocol number TX);

Reads out the EEPROM Register Set Protocol Area and overwrites the content of the Rx- and Tx- related registers. These registers are important for a Protocol selection.

Abort condition: Insufficient parameter in FIFO

3.4 SoftReset command

Command (1Fh);

This command is performing a soft reset. Triggered by this command all the default values for the register setting will be read from the EEPROM and copied into the register set.

4 Interaction

4.1 ISO 14443

4.1.1 Example Code - REQA - ISO 14443

This code snippet initializes the RC663 to the ISO14443 protocol, sends a REQA command and listens to the response of a tag.

```
1: writeRegister(0x00, 0x00);
2: writeRegister(0x02, 0xB0);
3: writeRegister(0x05, 0x00, 0x00);
4: writeRegister(0x00, 0x0D);
5: writeRegister(0x02, 0xB0);
6: writeRegister(0x28, 0x8E);
7: writeRegister(0x06, 0x7F);
8: writeRegister(0x2C, 0x18);
9: writeRegister(0x2D, 0x18);
10: writeRegister(0x2E, 0x0F);
11: writeRegister(0x05, 0x26);
12: writeRegister(0x00, 0x07);
13: waitForCardResponse();
14: readRegister(0x05, 0x05, 0x00);
```

The writeRegister function executes a logic left bit shift to the first argument, to confirm the communication protocol. (section 2.1.1)

Then the writeRegister function sends the calculated new value and all the other unmodified values as one SPI frame to the slave.

The readRegister function executes a logic left bit shift to the first argument and adds 1, to confirm the communication protocol. (section 2.1.1)

Then the readRegister function sends the calculated new value and all the other unmodified values as one SPI frame to the slave.

4.1.2 Line by line description - REQA - ISO 14443

1: Cancels previous executions and the state machine returns into IDLE mode

2: Flushes the FIFO and defines FIFO characteristics

3: Fills the FIFO with 0x00 and 0x00.

4: Executes LoadProtocol command with parameters 0x00 and 0x00 (FIFO). This translates to load protocol ISO14443A - 106

5: Flushes the FIFO and defines FIFO characteristics

6: Switches the RF filed ON.

7: Clears all bits in IRQ0

8: Switches the CRC extention *OFF* in tx direction

9: Switches the CRC extention *OFF* in rx direction

10: Only the 7 last bits will be sent via NFC

11: Fills the FIFO with 0x26 (REQA)

12: Executes Transceive routine

13: Waits until a card has responded by checking the IRQ0 register

14: Reads 2 byte from the FIFO to get ATQA

4.1.3 Detailed frame description - REQA - ISO 14443

1: writeRegister(0x00, 0x00);

Table 4. Line 1

	Value	Description
Address Byte	0x00	Write into command Register
Data	0x00	Perform the Idle command (0x00)

2: writeRegister(0x02, 0xB0);

Table 5. Line 2

	Value	Description
Address Byte	0x02	Write into Register FIFOControlReg
Data	0xB0	Flush FIFO, Set FIFO size to 255, High alert deactivated

3: writeRegister(0x05, 0x00, 0x00);

Table 6. Line 3

	Value	Description
Address Byte	0x05	Write into Register FIFOBuffer
Data	0x00	0x00 ISO14443A - 106 Protocol in rx direction
Data	0x00	0x00 ISO14443A - 106 Protocol in tx direction

4: writeRegister(0x00, 0x0D);

Table 7. Line 4

	Value	Description
Address Byte	0x00	Write into command Register
Data	0x0D	Load Protocol (the parameters for this command are taken from the FIFO)

5: writeRegister(0x02, 0xB0);

Table 8. Line 5

	Value	Description
Address Byte	0x02	Write into Register FIFOControlReg
Data	0xB0	Flush FIFO and maintain old FIFO characteristics

6: writeRegister(0x28, 0x8E);

Table 9. Line 6

	Value	Description
Address Byte	0x28	Write into Register DrvMode
Data	0x8E	Switches the RF field ON

7: writeRegister(0x06, 0x7F);

Table 10. Line 7

	Value	Description
Address Byte	0x06	Write into Register IRQ0
Data	0x7F	Clear all interrupts

8: writeRegister(0x2C, 0x18);

Table 11. Line 8

	Value	Description
Address Byte	0x2C	Write into Register TxCrcPreset
Data	0x18	do NOT use CRC

9: writeRegister(0x2D, 0x18);

Table 12. Line 9

	Value	Description
Address Byte	0x2D	Write into Register RxCrcPreset
Data	0x18	do NOT use CRC

10: writeRegister(0x2E, 0x0F);

Table 13. Line 10

	Value	Description
Address Byte	0x2E	Write into Register TxDataNum
Data	0x0F	Only send last 7 bits of the next transmit

11: writeRegister(0x05, 0x26);

Table 14. Line 11

	Value	Description
Address Byte	0x05	Write into Register FIFOBuffer
Data	0x26	0x26 = REQA defined by ISO 14443-3 A

12: writeRegister(0x00, 0x07);

Table 15. Line 12

	Value	Description
Address Byte	0x00	Write into command Register
Data	0x07	Execute Transceive command

14: readRegister(0x05, 0x05, 0x00);

Table 16. Line 14

	Value	Description
Address Byte	0x05	Read FIFOBuffer
Data	0x05	Read FIFOBuffer (While sending this byte on MOSI line, the RC663 answers with the first byte of the FIFOBuffer)
Data	0x00	random byte to read second byte of the FIFOBuffer (While sending this byte on MOSI line, the RC663 answers with the second byte of the FIFOBuffer)

4.2 ISO 15693

4.2.1 Example Code - Inventory - ISO 15693

This code snippet initializes the RC663 to the ISO 15693 protocol, sends an inventory request and listens to the response of a tag.

```
1: writeRegister(0x00, 0x00);
2: writeRegister(0x02, 0xB0);
3: writeRegister(0x05, 0x0A, 0x0A);
4: writeRegister(0x00, 0x0D);
5: writeRegister(0x02, 0xB0);
6: writeRegister(0x28, 0x8E);
7: writeRegister(0x05, 0x01, 0x94, 0x28, 0x12);
8: writeRegister(0x00, 0x0C);
9: writeRegister(0x00, 0x00);
10: writeRegister(0x02, 0xB0);
11: writeRegister(0x06, 0x7F);
12: writeRegister(0x05, 0x06, 0x01, 0x00);
13: writeRegister(0x00, 0x07);
14: for(counter = 0; counter < 16; counter++);
{
15: if(CardResponded())
{
16: readRegister(0x05, UID);
}
17: writeRegister(0x33, 0x0C);
18: writeRegister(0x2E, 0x00);
19: writeRegister(0x02, 0xB0);
20: writeRegister(0x00, 0x07);
}
21:writeRegister(0x28, 0x86);
```

The writeRegister function executes a logic left bit shift to the first argument, to confirm the communication protocol. (section 2.1.1)

Then the writeRegister function sends the calculated new value and all the other unmodified values as one SPI frame to the slave.

The readRegister function executes a logic left bit shift to the first argument and adds 1, to confirm the communication protocol. (section 2.1.1)

Then the readRegister function sends the calculated new value and all the other unmodified values as one SPI frame to the slave.

4.2.2 Line by line description - inventory - ISO 15693

1: Cancels previous executions and the state machine returns into IDLE mode

2: Flushes the FIFO and defines FIFO characteristics

3: Fills the FIFO with 0x0A and 0x0A.

4: Executes LoadProtocol command with parameters 0x0A and 0x0A (FIFO). This translates to load protocol ISO 15693

5: Flushes the FIFO and defines FIFO characteristics

6: Switches the RF filed ON.

7: Fills the FIFO with 0x01 (upper part EEPROM address), 0x94 (lower part EEPROM address), 0x28 (Register Address) and 0x12 (number of bytes)

8: Executes LoadReg command with parameters from FIFO. This translates to load from the EEPROM address 0x0194 into the the Registers starting at 0x28 the next 0x12 bytes. (The EEPROM of the RC663 has a lot of different predefined configurations saved which can be loaded via LoadReg)

9: Cancels previous executions and the state machine returns into IDLE mode

10: Flushes the FIFO and maintain FIFO characteristics.

11: Clears all bits in IRQ0

12: Fills the FIFO with 0x06 (Inventory Flag), 0x01 (inventory command), 0x00 (inventory command)

13: Executes Transceive routine

14: A loop that repeats 16 rimes since an inventory command consists of 16 time slots

15: The functions CardResponded reads the IRQ0 register and checks if a Card has responded

16: Reads the FIFO and saves it into the UID buffer.

17: Switches off the Start symbol

18: At the next Transceive routine no data is sent.

19: Flushes the FIFO and maintain FIFO characteristics.

20: Executes Transceive routine, without start symbol and without data => only EOF is sent, which indicates that the next timeslot is reached

21: Switch OFF RF filed.

4.2.3 Detailed frame description - Inventory - ISO 15693

1: writeRegister(0x00, 0x00);

Table 17. Line 1

	Value	Description
Address Byte	0x00	Write into command Register
Data	0x00	Perform the Idle command (0x00)

2: writeRegister(0x02, 0xB0);

Table 18. Line 2

	Value	Description
Address Byte	0x02	Write into Register FIFOControlReg
Data	0xB0	Flush FIFO, Set FIFO size to 255, High alert deactivated

3: writeRegister(0x05, 0x00, 0x00);

Table 19. Line 3

	Value	Description
Address Byte	0x05	Write into Register FIFOBuffer
Data	0x0A	0x0A ISO 15693 in rx direction
Data	0x0A	0x0A ISO 15693 in tx direction

4: writeRegister(0x00, 0x0D);

Table 20. Line 4

	Value	Description
Address Byte	0x00	Write into command Register
Data	0x0D	Load Protocol (the parameters for this command are taken from the FIFO)

5: writeRegister(0x02, 0xB0);

Table 21. Line 5

	Value	Description
Address Byte	0x02	Write into Register FIFOControlReg
Data	0xB0	Flush FIFO and maintain old FIFO characteristics

6: writeRegister(0x28, 0x8E);

Table 22. Line 6

	Value	Description
Address Byte	0x28	Write into Register DrvMode

	Value	Description
Data	0x8E	Switches the RF field ON

7: writeRegister(0x05, 0x01, 0x94, 0x28, 0x12);

Table 23. Line 7

	Value	Description
Address Byte	0x05	Write into Register FIFOBuffer
Data	0x01	Upper part of the EEPROM Address => EEPROM address = 0x0194
Data	0x94	Lower part of the EEPROM Address => EEPROM address = 0x0194
Data	0x28	Starting point of the Register addresses
Data	0x12	number of Registers, which should be copied

8: writeRegister(0x00, 0x0C);

Table 24. Line 8

	Value	Description
Address Byte	0x00	Write into command Register
Data	0x0C	Excute "LoadReg" with parameters from FIFO. Load from address 0x0194 the next 0x12 Byte into the Registers starting at address 0x28

9: writeRegister(0x00, 0x00);

Table 25. Line 9

	Value	Description
Address Byte	0x00	Write into command Register
Data	0x00	Execute IDLE - Cancels previous executions and the state machine returns into IDLE mode

10: writeRegister(0x02, 0xB0);

Table 26. Line 10

	Value	Description
Address Byte	0x02	Write into Register FIFOControlReg
Data	0xB0	Flush FIFO and maintain old FIFO characteristics

11: writeRegister(0x06, 0x7F);

Table 27. Line 11

	Value	Description
Address Byte	0x06	Write into Register IRQ0
Data	0x7F	Clear all Interrupts

12: writeRegister(0x05, 0x06, 0x01, 0x00);

Table 28. Line 12

	Value	Description
Address Byte	0x05	Write into Register FIFOBuffer
Data	0x06	Inventory Flag - Inventory with 16 slots
Data	0x01	Inventory Command defined by ISO 15693
Data	0x00	Inventory Command defined by ISO 15693

13: writeRegister(0x00, 0x07);

Table 29. Line 13

	Value	Description
Address Byte	0x00	Write into command Register
Data	0x07	Execute Transceive command

14: for(counter = 0; counter < 16; counter++)

A loop which counts from 0 to 15, which is used to check all time slots in an Inventory request

15: if(CardResponded)

The function CardResponded reads the IRQ0 register, which indicates if a card has responded or not.

16: readRegister(0x05, UID);

Table 30. Line 16

	Value	Description
Address Byte	0x05	Read from the FIFOBuffer Register
Buffer	UID	Save FIFO into UID Array

17: writeRegister(0x33, 0x0C);

Table 31. Line 17

	Value	Description
Address Byte	0x33	Write into Register FrameControl
Data	0x0C	Switch off the Start symbol and select correct Stop symbol

18: writeRegister(0x2E, 0x00);

Table 32. Line 18

	Value	Description
Address Byte	0x2E	Write into Register TxDataNum

	Value	Description
Data	0x00	By setting all bits to zero no data is transmitted at the next transceive command, but only the start and stop symbols-

19: writeRegister(0x02, 0xB0);

Table 33. Line 19

	Value	Description
Address Byte	0x02	Write into Register FIFOControlReg
Data	0xB0	Flush FIFO and maintain old FIFO characteristics

20: writeRegister(0x00, 0x07);

Table 34. Line 20

	Value	Description
Address Byte	0x00	Write into command Register
Data	0x07	Execute Transceive command (This command only sends the Stop symbol)

21: writeRegister(0x28, 0x86);

Table 35. Line 21

	Value	Description
Address Byte	0x28	Write into Register DrvMode
Data	0x86	Switch RF field OFF

5 Legal information

5.1 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.

5.2 Disclaimers

Limited warranty and liability — 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. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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 life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at 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. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer. In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages. Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — While NXP Semiconductors has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP Semiconductors accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

5.3 Licenses

Purchase of NXP ICs with NFC technology

Purchase of an NXP Semiconductors IC that complies with one of the Near Field Communication (NFC) standards ISO/IEC 18092 and ISO/IEC 21481 does not convey an implied license under any patent right infringed by implementation of any of those standards. Purchase of NXP Semiconductors IC does not include a license to any NXP patent (or other IP right) covering combinations of those products with other products, whether hardware or software.

5.4 Trademarks

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

Tables

Contents

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

Date of release: 7 January 2020
Document identifier: AN12657
Document number: 581610