NFC\_Statemachine documentation

Document name: NFC\_Statemachine\_0v2\_21-07-2021.docx



|  |  |
| --- | --- |
| Created by: T. Maier | Released: name, position |
| Date of release: dd.mm.yyyy |
| Printed: 12.03.2023 |
|  | |

# Introduction

This document is valid from the date of release.

## Scope / Purpose

This documentation should give an overview of the statemachine functions and how to use them.

## Terms and Abbreviations

PCB Printed Circuit Board

PCD Proximity Coupling Device (reader

PICC Proximity IC Card

HF High Frequency (13.56MHz)

IC Integrated Circuit

ISO International Standards Organization

LMA Load Modulation Amplitude

PCB Printed Circuit Board

PCD Proximity Coupling Device

PICC Proximity IC Card

Ref PICC Reference PICC defined by ISO

RF Radio Frequency

RFID Radio Frequency Identification

SRF Self Resonance Frequency

# Table of Contents

[1 Introduction 2](#_Toc129543677)

[1.1 Scope / Purpose 2](#_Toc129543678)

[1.2 Terms and Abbreviations 2](#_Toc129543679)

[2 Table of Contents 3](#_Toc129543680)

[3 List of figures 3](#_Toc129543681)

[4 General information 4](#_Toc129543682)

[4.1 Structure 4](#_Toc129543683)

[4.2 Step data / implementation 4](#_Toc129543684)

[5 States / supported features 5](#_Toc129543685)

[5.1 R(NAK) handling 5](#_Toc129543686)

[5.2 S(WTX) handling 6](#_Toc129543687)

[5.3 RUID 7](#_Toc129543688)

[5.4 READ\_RF 8](#_Toc129543689)

[5.5 READ\_SEND\_RF 9](#_Toc129543690)

[5.6 SEND\_RF 9](#_Toc129543691)

[6 Document history and distribution 10](#_Toc129543692)

# List of figures

[Figure 1: R(NAK) handling for PCD RX steps 4](#_Toc80024013)

[Figure 2: S(WTX) handling for PICC RX steps 4](#_Toc80024014)

# General information

The code can be enabled for every TC step. Each of the supported features is mapped to a specific opcode value (1, 2, 3…).

## Structure

The code consists mainly of one large switch/case structure (state-machine). The states are selected via the transmitted opcode. To reduce size and complexity most variables are re-used throughout the states.

## Step data / implementation

To perform features the TC has to send specific OpCode data together with the TC-step data. The format is specified as follows:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Feature | Usecase | OpCode | Pattern trigger flag | Payload | | | | |
| 1st Byte | | 2nd Byte | | Rest |
| R(NAK) handling | Feature | 1 | 1 | Length | Loop limit | | PICC TX data | |
| S(WTX) handling | Feature | 2 | 1 | Length | Loop limit | | Not used | |
| RUID/PUPI | Feature | 3 | 1 | Length | Selection | | Not used | |
| READ\_RF | DEBUG | 4 | 0 | Not used | Not used | | Not used | |
| READ\_SEND\_RF | DEBUG | 5 | 0 | Not used | Not used | | Not used | |
| SEND\_RF | DEBUG | 6 | 0 | Not used | Not used | | Not used | |

Following features are already implemented as ‘Frame options’ which can be set for every step:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | Frame option | Variable explanation | Used step | Description |
| R(NAK) handling | R(NAK)=x | x = consecutive retries | PCD RX | Retransmission handling Example: R(NAK)=3 → handling 3 times |
| S(WTX) handling | S(WTX)=x | x = consecutive retries | PICC RX | Wate time extension handling Example: S(WTX)=3 → handling 3 times |
| RUID/PUPI | RUID | - | PICC RX | RUID/PUPI handling |

As a bonus these features are also included in the User scripting PCD/PICC plugins. For example:



Figure : R(NAK) handling for PCD RX steps



Figure : S(WTX) handling for PICC RX steps

# States / supported features

This section describes all implemented features.

## R(NAK) handling

References:

1. ISO\_IEC\_14443\_4\_2018\_07\_01\_en.pdf
   1. 7.2.2.1 R-block

Transfer values:

1. **Opcode = 1**
2. Opcode payload = Loop limit + required PICC response data
3. Frame option → **R(NAK)=x** where x = allowed consecutive retries (Loop limit)

This case handles the PCD retransmission request R(NAK). The case reacts to a received PCD 0xB2 or 0xB3 byte. The transferred Loop limit byte determines how often the code checks for a R(NAK) request.

1. If the **received byte** is either **0xB2 or 0xB3**:
   1. Send opcode payload data as PICC response
2. **Else / Loop limit reached**
   1. Do nothing (proceed with normal TC execution) and break out of loop.

Code snippet (without UART messages):

**case** RETRANSMISSION: //When opcode == 1 -> PCD retransmission handling [PCD TC FEATURE]

tx\_config = 0x01; //Set TX config to: HEX:1 | CRC:0 | BCC:0  
//while loop for consecutive Retransmission requests

while ((loop\_count < loop\_limit) && (loop\_done == 0))

{

//Read RF

len = read\_nfc\_rx\_stream(rx\_buf, &io\_module);

//Check if first byte is 0xB2 OR 0xB3 (R(NAK) identifier)

**if**(rx\_buf[0] == 0xB2 || rx\_buf[0] == 0xB3)

{

//Send opcode payload data

write\_nfc\_tx\_stream(tx\_buf\_opcode,len\_op,tx\_config,&io\_module);

//Optional UART messages

**#ifdef** UART

}

**else**

{

//Do nothing, proceed with TC

//Optional UART messages

**#ifdef** UART

//If no Retransmission is requested, finish Retransmission handling

loop\_done = 1;

}

loop\_count++;

}

**break**;

## S(WTX) handling

References:

1. ISO\_IEC\_14443\_4\_2018\_07\_01\_en.pdf
   1. 7.2.2.1 S-block & 7.4

Transfer values:

1. **Opcode = 2**
2. Opcode payload = Loop limit
3. Frame option → **S(WTX)=x** where x = allowed consecutive retries (Loop limit)

This case handles the PICC S(WTX) request. The case reacts to a received byte with a value of 0xF2.

The transferred Loop limit byte determines how often the code checks for a S(WTX) request.

1. If the **received first byte** is **0xF2**:
   1. Check if **second byte** (WTXM) is correct (**0x3C – 0x3F & 0xFC – 0xFF not allowed**)
      1. If correct, send S(WTX) response (mirror RX data) - (last loop rotation: only RX!)  
         Otherwise send S(DESELECT) response (protocol error) and break out of loop.
2. **Else / Loop limit reached**
   1. Do nothing (proceed with normal TC execution) and break out of loop.

Code snippet (without UART messages):

**case** WTX: //When opcode == 2 -> PICC WTX handling [PICC TC FEATURE]

tx\_config = 0x01; //Set TX config to: HEX:1 | CRC:0 | BCC:0  
//while loop for consecutive WTX requests

while ((loop\_count <= loop\_limit) && (loop\_done == 0))

{

//Read RF

len = read\_nfc\_rx\_stream(rx\_buf, &io\_module);

//Check if first byte is 0xF2 (S(WTX) identifier)

**if**(rx\_buf[0] == 0xF2)

{

//Check if second byte is correct (WTXM)

**if**(rx\_buf[1] == 0x00 || rx\_buf[1] == 0xC0 || (rx\_buf[1] >= 0x3C && rx\_buf[1] <= 0x3F)

|| rx\_buf[1] >= 0xFC)

{

//Send S(DESELECT) message

len = **sizeof**(deselect);

write\_nfc\_tx\_stream(deselect,len,tx\_config,&io\_module);

//Optional UART messages

**#ifdef** UART

//If no WTX is requested, finish WTX handling

loop\_done = 1;

}

**else**

{

//No TX on last run

**if**(loop\_count < loop\_limit)

{

//Send S(WTX) response

write\_nfc\_tx\_stream(rx\_buf,len,tx\_config,&io\_module);

}

//Optional UART messages

**#ifdef** UART

}

}

**else**

{

//Do nothing, proceed with TC

**#ifdef** UART

//If no WTX is requested, finish WTX handling

loop\_done = 1;

}

loop\_count++;

}

**break**;

## RUID

References:

1. ISO\_IEC\_14443\_3\_2018\_07\_01\_en.pdf
   1. SELECT/ATTRIB

Transfer values:

1. **Opcode = 3**
2. Opcode payload = Selection (RUID/RPUPI)
3. Frame option → **RUID**

This case handles PICCs with random UID/PUPI. The case records the ID and replies with correct framing.

1. If **payload byte == 1**:
   1. RUID handling. TX:
      1. Byte 0 and 1 are fixed (0x93+0x70)
      2. Followed by the previously recorded UID
      3. CRC is calculated and added inside the testcase engine
2. If **payload byte == 2**:
   1. RPUPI handling. TX:
      1. Byte 0 and Byte 5 – 8 are fixed (0x00, 0x00, 0x08, 0x01, 0x00)
      2. Byte 1 – Byte 4 is the previously recorded PUPI
      3. CRC is calculated and added inside the testcase engine
3. **Else**:
   1. Do nothing (proceed with normal TC execution) and break out of loop.

Code snippet (without UART messages):

**case** RUID: //When opcode == 3 -> PICC RUID/PUPI handling

tx\_config = 0x03; //Set TX config to: HEX:1 | CRC:1 | BCC:0

**if**(loop\_limit == 1) // 'loop\_limit' is reused for TypeA/B separation, loop\_limit = 1: Type A

{

//Read RF -> save UID to rx\_buf

len = read\_nfc\_rx\_stream(rx\_buf, &io\_module);

//Add fixed data 0x9370 to beginning of new array, followed by the previously recorded UID

ruid[0] = 0x93;

ruid[1] = 0x70;

for (int i = 0; i < len; i++)

{

ruid[i+2] = rx\_buf[i];

}

//Send generated TX data + CRC

write\_nfc\_tx\_stream(ruid,len+2,tx\_config,&io\_module);

//Optional UART messages

#ifdef UART

}

**else**

{

**if**(loop\_limit == 2) //'loop\_limit' is reused for TypeA/B separation, loop\_limit = 2: Type B

{

//Read RF -> save UID to rx\_buf

len = read\_nfc\_rx\_stream(rx\_buf, &io\_module);

//Add fixed data 0x1D to beginning of new array, followed by the previously

recorded PUPI and fixed bytes 0x00080100

ruid[0] = 0x1D;

for (int i = 1; i < 5; i++)

{

ruid[i] = rx\_buf[i];

}

ruid[5] = 0x00;

ruid[6] = 0x08;

ruid[7] = 0x01;

ruid[8] = 0x00;

len = 9;

//Send generated TX data + CRC

write\_nfc\_tx\_stream(ruid,len,tx\_config,&io\_module);

//Optional UART messages

#ifdef UART

}

**else**

{

#ifdef UART

}

}

break;

## READ\_RF

References:

-

Transfer values:

1. **Opcode = 4**

This case reads one NFC Frame (PICC/PCD) and reports it over UART (if enabled).

Code snippet:

**case** READ\_RF: //When opcode == 3 -> read RF and send over UART [DEBUG]

//Read RF

len = read\_nfc\_rx\_stream(rx\_buf, &io\_module);

**#ifdef** UART

xil\_printf("\r\nRF\_RX length:%i",len);

print("\r\nRF data:\r\n");

write\_uart\_buf(rx\_buf, len);

**#endif**

**break**;

## READ\_SEND\_RF

References:

-

Transfer values:

1. **Opcode = 5**

This case reads one NFC Frame (PICC/PCD) and sends one NFC Frame (PICC/PCD) with a value of 0xAA (defined by tx\_buf) without CRC. If enabled the case reports both received and sent data.

Code snippet:

**case** READ\_SEND\_RF: //When opcode == 4 -> read RF and send RF [DEBUG]

tx\_config = 0x01; //Set TX config to: HEX:1 | CRC:0 | BCC:0

//Read RF

len = read\_nfc\_rx\_stream(rx\_buf,&io\_module);

**#ifdef** UART

xil\_printf("\r\nRF RX length:%i",len);

print("\r\nRF data:\r\n");

write\_uart\_buf(rx\_buf, len);

**#endif**

//Send RF

len = **sizeof**(tx\_buf);

write\_nfc\_tx\_stream(tx\_buf,len,tx\_config,&io\_module);

**#ifdef** UART

xil\_printf("\r\nRF TX length:%i",len);

print("\r\nSent tx data:%d\r\n");

write\_uart\_buf(tx\_buf, len);

**#endif**

**break**;

## SEND\_RF

References:

-

Transfer values:

1. **Opcode = 6**

This case sends one NFC Frame (PICC/PCD) with a value of 0xAA (defined by tx\_buf) without CRC.

If enabled the case reports the sent data over UART.

Code snippet:

**case** READ\_SEND\_RF: //When opcode == 4 -> read RF and send RF [DEBUG]

tx\_config = 0x01; //Set TX config to: HEX:1 | CRC:0 | BCC:0

//Send RF

len = **sizeof**(tx\_buf);

write\_nfc\_tx\_stream(tx\_buf,len,tx\_config,&io\_module);

**#ifdef** UART

xil\_printf("\r\nRF TX length:%i",len);

print("\r\nSent tx data:%d\r\n");

write\_uart\_buf(tx\_buf, len);

**#endif**

**break**;

# Document history and distribution

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision history** | | | |
| Date | Revision | Comments | Author |
| 14.06.2021 | V0.1 | 1st release of draft | T. Maier |
| 21.07.2021 | V0.2 | Added R(NAK)- and S(WTX)-handling | T. Maier |
| 28.07.2021 | V0.2 | Updated R(NAK)- and S(WTX)-handling with loop\_limit | T. Maier |
| 03.08.2021 | V0.2 | Updated features and step data / implementation | T. Maier |
| 16.08.2021 | V.02 | Updated features and step data / implementation | T. Maier |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Distribution log file** | | |
| Original signed document is located at quality manager’s office, actual version is available on cilab’s fileserver, additional external copies are distributed to | | |
| Company / Organization / Name | Hardcopy / pdf | Date |
|  |  |  |
|  |  |  |