Communication Protocol Document

NANREC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Revision History** | | | | | |
| Revision | ECN | Date | Prepared by | Description |
| 1.0 |  | 23.07.2023 | O. Saferman | Created |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Table of Contents**

[1 Scope 5](#_Toc141019193)

[2 Relevant Documents 5](#_Toc141019194)

[3 Communication Protocol 5](#_Toc141019195)

[3.1 Introduction 5](#_Toc141019196)

[3.2 Hardware definitions 5](#_Toc141019197)

[3.3 Frame structure 5](#_Toc141019198)

[3.3.1 Tx frame: 6](#_Toc141019199)

[3.3.2 Rx frame: 6](#_Toc141019200)

[3.4 Error codes 7](#_Toc141019201)

[3.4.1 Example of Rx Frame with error 7](#_Toc141019202)

[4 Commands 8](#_Toc141019203)

[4.1 Commands list 8](#_Toc141019204)

[4.2 Detailed command description 10](#_Toc141019205)

[4.2.1 Standard group 10](#_Toc141019206)

[4.2.1.1 Get software version 10](#_Toc141019207)

[4.2.1.2 Get firmware version 10](#_Toc141019208)

[4.2.1.3 Get serial number 11](#_Toc141019209)

[4.2.1.4 Set serial number 11](#_Toc141019210)

[4.2.1.5 Set log level 11](#_Toc141019211)

[4.2.2 RF Cards 12](#_Toc141019212)

[4.2.2.1 RF1 MFCU1 SPI 12](#_Toc141019213)

[4.2.2.2 RF1 MFCU2 SPI 12](#_Toc141019214)

[4.2.2.3 RF2 MFCU1 SPI 12](#_Toc141019215)

[4.2.2.4 RF2 MFCU2 SPI 13](#_Toc141019216)

[4.2.3 Lattice FPGA Registers 13](#_Toc141019217)

[4.2.3.1 Lattice FPGA SPI 13](#_Toc141019218)

[4.2.4 FPGA Registers 13](#_Toc141019219)

[4.2.4.1 Read FPGA register 13](#_Toc141019220)

[4.2.4.2 Write FPGA register 14](#_Toc141019221)

[4.2.5 Recording system 14](#_Toc141019222)

[4.2.5.1 Record IQ data 14](#_Toc141019223)

[4.2.5.2 Get recording buffer size 14](#_Toc141019224)

[4.2.5.3 Init play IQ data 15](#_Toc141019225)

[4.2.6 Memory 15](#_Toc141019226)

[4.2.6.1 Read memory address 15](#_Toc141019227)

[4.2.6.2 Write memory address 15](#_Toc141019228)

**Figures**

[Figure 1: Tx frame structure 6](#_Toc141019188)

[Figure 2: Rx frame structure 6](#_Toc141019189)

[Figure 3: Example of Rx error frame structure 7](#_Toc141019190)

**Tables**

[Table 1: Errors 7](#_Toc141019191)

[Table 2: Commands 9](#_Toc141019192)

# Scope

The purpose of this document is to detail the communication protocol used for controlling the NANREC product.

# Relevant Documents

1. TBD.

# Communication Protocol

## Introduction

The host PC communicates with the NANREC system through a standard Ethernet protocol carrying Eyal proprietary protocol frame.

## Hardware definitions

The communication protocol between PC and NANREC system is a standard TCP/IP transport layer over Ethernet protocol using 1 GbE.   
The system is a slave and the PC is the master thus the master initiates all transactions.

Host IPv4 address: 10.0.1.1  
NANREC IPv4 address: 10.0.1.15  
Port number on both sides: 5555   
Subnet Mask: 255.255.255.0

## Frame structure

The frame consists of four header bytes then the variable length data payload and completed with a checksum byte.

The header structure begins with a preamble word (project proprietary).   
The preamble is followed by a command word, which is described in the following tables. The next four bytes are the length of the data, next comes the data, which can be up to 232 bytes long and then the frame ends in a checksum word.   
The checksum is the result of the sum of all bytes in the frame excluding the checksum word.

### Tx frame:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x004E | 2 bytes | 4 bytes | TD0 | … | TDn | 2 bytes |
| Preamble | Command | Data length | D[0] |  | D[n] | Checksum |

Figure 1: Tx frame structure

Preamble - The header structure begins with a preamble byte, which is project proprietary. For this project: 0x004E (ASCII 'N').   
  
Command - 2 bytes command. All commands are described in the following tables.   
  
Data length - 4 bytes. Length of transmitted data.

Data – Data. Up to 232 bytes long.

Checksum - 2 bytes. The sum of all bytes in the frame except the checksum field.

### Rx frame:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x004E | 2bytes | 4 bytes | RD0 | … | RDn | 2bytes |
| Preamble | Command | Data length | D[0] |  | D[n] | Checksum |

Figure 2: Rx frame structure

Preamble - The header structure begins with a preamble byte, which is project proprietary. For this project: 0x004E (ASCII 'N').

Command - 2 bytes command. All commands are described in the following tables.   
  
Data length - 4 bytes. Length of received data.

Data – Data. Up to 232 bytes long.

Checksum - 2 bytes. The sum of all bytes in the frame except the checksum field.

## Error codes

The target will return an error code in the CMD field when an error occurs. The following error codes are supported:

|  |  |  |
| --- | --- | --- |
| **Command** | **Opcode** | **Remarks** |
| Header error | 0xF0 |  |
| Command error | 0xF1 |  |
| Checksum error | 0xF2 |  |
| Data error | 0xF3 |  |
| Execution Error | 0xF4 |  |
| Time-out Error – Type 1 | 0xF5 |  |
| Time-out Error – Type 2 | 0xF6 | Not used for this project |
| Time-out Error – Type 3 | 0xF7 | Not used for this project |
| Message length error | 0xF8 |  |
| Data length error | 0xF9 |  |
| Reserved | 0xFA |  |
| Reserved | 0xFB |  |
| Reserved | 0xFC |  |
| Reserved | 0xFD |  |
| Reserved | 0xFE |  |
| Reserved | 0xFF |  |

Table 1: Errors

### Example of Rx Frame with error

|  |  |  |  |
| --- | --- | --- | --- |
| 0x004E | 0x00F4 | 0x00000000 | 0x0142 |
| 2bytes | 2bytes | 4 Bytes | 2 bytes |
| Preamble | Command | Data length | Checksum |

Figure 3: Example of Rx error frame structure

Preamble - For this project: is 0x004E (ASCII 'N').   
  
Command - 2 bytes command. One of the error codes from error table.  
  
Data length – 0x00000000, no data.

Checksum - 2 bytes. Is the result of the sum of all bytes in the frame except the checksum field. 🡪 0x004E + 0x00F4 + 0x00000000 = 0x0142

# Commands

## Commands list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Group** | **Command** | **Function** | **Direction** | **Remarks** |
| Standard | 0x01 | Get software version | Read |  |
| 0x02 | Get firmware version | Read |  |
| 0x04 | Get serial number | Read |  |
| 0x05 | Set serial number | Write |  |
| 0x06 | Set log level | Write |  |
| 0x07 | Is recording system busy | Read |  |
| 0x08 | Get system type | Read |  |
| 0x09 | Get Eyal PN | Read |  |
| 0x0A | Get DPC serial number | Read |  |
| 0x10 | Set DPC serial number | Write |  |
|  |  |  |  |  |
| RF Cards | 0x20 | RF1 MFCU1 SPI | Read/Write |  |
| 0x21 | RF1 MFCU2 SPI | Read/Write |  |
| 0x22 | RF2 MFCU1 SPI | Read/Write |  |
| 0x23 | RF2 MFCU2 SPI | Read/Write |  |
|  |  |  |  |  |
| Lattice FPGA | 0x60 | Lattice FPGA SPI | Read/Write |  |
|  |  |  |  |  |
| FPGA Registers | 0x70 | Read FPGA register | Read |  |
| 0x71 | Write FPGA register | Write |  |
|  |  |  |  |  |
| Recording | 0x80 | Record IQ data | Write |  |
| 0x81 | Get recording buffer size | Read |  |
| 0x82 | Init play IQ data | Write |  |
|  |  |  |  |  |
| Memory | 0x90 | Read 32-bit memory address | Read |  |
| 0x91 | Write 32-bit memory address | Write |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Table 2: Commands

## Detailed command description

### Standard group

#### Get software version

**Description:** Get the software version

**Command:** 0x0001

**TX** **data:** NA

**TX** **frame:** 0x004E 0x0001 0x00000000 0x004F

**RX data:** 6 bytes version structure

Major version – 1 byte

Minor version – 1 byte

Version day – 1 bytes

Version month – 1 bytes

Version year – 2 bytes

**RX** **frame:** 0x004E 0x0001 0x00000006 + RX Data + CHK

#### Get firmware version

**Description:** Get firmware version

**Command:** 0x0002

**TX** **data:** NA

**TX** **frame:** 0x004E 0x0002 0x00000000 0x0050

**RX data:** 6 bytes version structure

Major version – 1 byte

Minor version – 1 byte

Version day – 1 bytes

Version month – 1 bytes

Version year – 2 bytes

**RX** **frame:** 0x004E 0x0002 0x00000006 + RX Data + CS

#### Get serial number

**Description:** Get serial number

**Command:** 0x0004

**TX** **data:** NA

**TX** **frame:** 0x004E 0x0004 0x00000000 0x0052

**RX data:** 2-byte serial number

**RX** **frame:** 0x004E 0x0004 0x00000002 + RX Data + CS

#### Set serial number

**Description:** Set serial number.

**Command:** 0x0005

**TX** **data:** 2-byte serial number

**TX** **frame:** 0x004E 0x0005 0x00000002 + TX Data + CS

**RX data:** NA

**RX** **frame:** 0x004E 0x0005 0x00000000 0x0053

#### Set log level

**Description:** Set log level.

**Command:** 0x0006

**TX** **data:** 1 byte. Log level: 0- log off, 1 (critical error) - 7(trace). Note: the selected level means “up to” this level. Default level is TBD (info).

**TX** **frame:** 0x004E 0x0006 0x00000002 + TX Data + CS

**RX data:** NA

**RX** **frame:** 0x004E 0x0006 0x00000000 0x0054

### RF Cards

#### RF1 MFCU1 SPI

**Description:** Send RF1 MFCU1 an SPI command

**Command:** 0x0020

**TX** **data:** 8-byte message

**TX** **frame:** 0x004E 0x0020 0x00000008 + Tx data + CS

**RX data:** 2-byte response if SPI message is read otherwise NA

**RX** **frame:** 0x004E 0x0020 0x00000002 + Rx data + CS

0x004E 0x0020 0x00000000 0x006E

#### RF1 MFCU2 SPI

**Description:** Send RF1 MFCU2 an SPI command

**Command:** 0x0021

**TX** **data:** 8-byte message

**TX** **frame:** 0x004E 0x0021 0x00000008 + Tx data + CS

**RX data:** 2-byte response if SPI message is read otherwise NA

**RX** **frame:** 0x004E 0x0021 0x00000002 + Rx data + CS

0x004E 0x0021 0x00000000 0x006F

#### RF2 MFCU1 SPI

**Description:** Send RF2 MFCU1 an SPI command

**Command:** 0x0022

**TX** **data:** 8-byte message

**TX** **frame:** 0x004E 0x0022 0x00000008 + Tx data + CS

**RX data:** 2-byte response if SPI message is read otherwise NA

**RX** **frame:** 0x004E 0x0022 0x00000002 + Rx data + CS

0x004E 0x0022 0x00000000 0x0070

#### RF2 MFCU2 SPI

**Description:** Send RF2 MFCU2 an SPI command

**Command:** 0x0023

**TX** **data:** 8-byte message

**TX** **frame:** 0x004E 0x0023 0x00000008 + Tx data + CS

**RX data:** 2-byte response if SPI message is read otherwise NA

**RX** **frame:** 0x004E 0x0023 0x00000002 + Rx data + CS

0x004E 0x0023 0x00000000 0x0071

### Lattice FPGA Registers

#### Lattice FPGA SPI

**Description:** Lattice FPGA SPI command

**Command:** 0x0060

**TX** **data:** 8-byte message

**TX** **frame:** 0x004E 0x0060 0x00000008 + TX Data + CRC

**RX data:** 2-byte response if SPI message is read otherwise NA

**RX** **frame:** 0x004E 0x0060 0x00000002 + Rx data + CS

0x004E 0x0060 0x00000000 0x00AE

### FPGA Registers

#### Read FPGA register

**Description:** Read FPGA 32-bit register

**Command:** 0x0070

**TX** **data:** 4 bytes - register number

**TX** **frame:** 0x004E 0x0070 0x00000004 + TX Data + CRC

**RX data:** 4 bytes

**RX** **frame:** 0x004E 0x0070 0x00000004 + RX Data + CRC

#### Write FPGA register

**Description:** Write FPGA 32-bit register

**Command:** 0x0071

**TX** **data:** 8 bytes:

4 bytes – register number

4 bytes – register value

**TX** **frame:** 0x004E 0x0071 0x00000008 + TX Data + CRC

**RX data:** NA

**RX** **frame:** 0x004E 0x0071 0x00000000 0x00BF

### Recording system

#### Record IQ data

**Description:** Start recording ‘I’ & ’Q’ data recording

**Command:** 0x0080

**TX** **data:** 5 bytes

1 byte – Channels mask to record

4 byte – Num of samples per channel. Note: each sample is 32-bit.

**TX** **frame:** 0x004E 0x0080 0x00000005 + Tx data + CS

**RX data:** 0-4Gbyte – Recorded IQ data

**RX** **frame:** 0x004E 0x0080 + size + Rx data + CS

#### Get recording buffer size

**Description:** Get size of recording buffer

**Command:** 0x0081

**TX** **data:** NA

**TX** **frame:** 0x004E 0x0081 0x00000000 0x00CF

**RX data:** 4 bytes – size of buffer in bytes

**RX** **frame:** 0x004E 0x0081 0x00000004 + Rx Data + CS

#### Init play IQ data

**Description:** Configure the playback system without starting

**Command:** 0x0082

**TX** **data:** 0-4Gbyte – Data for playback

**TX** **frame:** 0x004E 0x0082 + size + TX Data + CS

**RX data:** NA

**RX** **frame:** 0x004E 0x0082 0x00000000 0xD0

### Memory

#### Read memory address

**Description:** Read 32-bit memory address

**Command:** 0x0090

**TX** **data:** 4 bytes - address

**TX** **frame:** 0x004E 0x0090 0x00000004 + TX Data + CS

**RX data:** 4 bytes – value

**RX** **frame:** 0x004E 0x0090 0x00000004 + RX Data + CS

#### Write memory address

**Description:** Write 32-bit memory address

**Command:** 0x0091

**TX** **data:** 8 bytes:

4 bytes – address

4 bytes – value

**TX** **frame:** 0x004E 0x0091 0x00000008 + TX Data + CRC

**RX data:** NA

**RX** **frame:** 0x004E 0x0091 0x00000000 0x00DF