|  |  |  |
| --- | --- | --- |
| **Communication Protocol Document**  **OPTR Digital Beamformer Customer P/N:** **XXXXXXX**  **Per customer spec.: RAF#XXXXXXX** | | |
| **Engineering Approval** | **Date** | C:\Users\amit_m\Desktop\LOGOs\LOGO-KRATOS_GMI_EYAL.jpg |
|  |  |
| **QA Approval** | **Date** |
|  |  |
| **Customer Approval** | **Date** |
|  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Revision History** | | | | |
| Revision | ECN | Date | Prepared by | Description |
| 0.1 |  | 11.1.2021 | Arthur D. | Added API:  opcode 0x19 - Sets the temperature gain compensation parameter  opcode 0x20 - Get the temperature gain compensation parameter |
| 0.2 |  | 15.3.2021 | Arthur D. | Added API:  IO FPGA 0x60 – Read IO FPGA register  0x61 – Write IO FPGA register  CCDCUN12 - I2C (Debug Purpose)  0xB1 - Init Sequence  0xB2 - Configuration OUTx  0xB3 - Configuration Input  0xB4 - Configuration dev  0xB5 - Read register  0xB6 - Write register |
| 0.3 |  | 26.04.2021 | Arthur D. | Added API: 0xA0 – Get temperature |
| 0.4 |  | 03.05.2021 | Arthur D. | Added API: 0x1B - Get Madura’s temperature  0x1C - Set Madura’s init calibration  0x1D - Set Rx Tx enable |
| 0.5 |  | 19.05.2021 | Arthur D. | Added support for Madura 1& 2:  0x10 - SetRxGain: mask Rx[1:8]  0x11 - SetRxGain: channel Rx [1:8]  0x12 - SetTxAtten: mask Tx[1:8]  0x13 - GetTxAtten: channel Tx[1:8] 0x14 –Set Local Oscillator (LO) frequency: mask Tx[1:8]  0x15 - Get Local Oscillator (LO) frequency: channel Tx[1:8]  0x17 - Set the Tx NCO test tone configuration: mask Rx[1:8]  0x18 - Get the Tx NCO test tone configuration: channel Rx[1:8]  0x19 - Set temperature gain compensation parameter: mask Rx[1:8]  0x1A - Get temperature gain compensation parameter: channel Rx[1:8]  0x1B - Get Madura’s temperature: device index 0-1  0x1C - Run Madura’s init calibration: mask Rx/Tx[1:8]  0x1D - Madura Rx Tx enable Set: mask Rx & Tx[1:8] |
| 0.6 |  | 24.5.2021 | Arthur D. | * Flash Data: Edited and updated commands (0x30-0x34) * DAC (I2C): added 0xA1 – write register * EEPROM: added   + 0xA2 – Write byte   + 0xA3 – Read byte * Madura: added 0x16 – Validate JESD status |
| 0.7 |  | 3.6.2021 | Arthur D. | * Added Flash API:   + 0x37 – Write byte   + 0x38 – Read byte   + 0x39 – Erase 4K bytes   + 0x40 – Configuration commands * Added Recording API:   + 0x81 – Get recording buffer size |
| 0.8 |  | 9.6.2021 | Arthur D. | * Added API: 0x86 – Get big payload buffer size |
| 0.9 |  | 12.8.2021 | Nora L. | * Added API: 0x92 – Read data from memory |
| 0.10 |  | 23.03.21 | Nora L. | * Added Synthesizer API:   + 0xA4 – Read register   + 0xA5 – Write register   + 0xA6 – Set LO frequency   + 0xA7 – Set Force VCO values |

**Table of Contents**

[1 Scope 7](#_Toc98925803)

[2 Relevant Documents 7](#_Toc98925804)

[3 Communication Protocol 7](#_Toc98925805)

[Introduction 7](#_Toc98925806)

[Hardware definitions 7](#_Toc98925807)

[Frame structure 7](#_Toc98925808)

[3.1.1 Tx frame: 8](#_Toc98925809)

[3.1.2 Rx frame: 8](#_Toc98925810)

[Error codes 9](#_Toc98925811)

[3.1.3 Rx Frame with error f.e: 9](#_Toc98925812)

[4 Commands 10](#_Toc98925813)

[Commands list 10](#_Toc98925814)

[Details command description 13](#_Toc98925815)

[4.1.1 Standard group 13](#_Toc98925816)

[4.1.1.1 Get software version 13](#_Toc98925817)

[4.1.1.2 Get firmware version 13](#_Toc98925818)

[4.1.1.3 Get serial number 14](#_Toc98925819)

[4.1.1.4 Set serial number 14](#_Toc98925820)

[4.1.1.5 Set log level 14](#_Toc98925821)

[4.1.2 Madura (adrv9026) 15](#_Toc98925822)

[4.1.2.1 Madura init HW 15](#_Toc98925823)

[4.1.2.2 Set RX channel gain 15](#_Toc98925824)

[4.1.2.3 Get RX channel gain 15](#_Toc98925825)

[4.1.2.4 Set TX channel attenuation 16](#_Toc98925826)

[4.1.2.5 Get TX channel attenuation 16](#_Toc98925827)

[4.1.2.6 Set Local Oscillator (LO) frequency 17](#_Toc98925828)

[4.1.2.7 Get Local Oscillator (LO) frequency 17](#_Toc98925829)

[4.1.2.8 Validate JESD status 17](#_Toc98925830)

[4.1.2.9 Set the Tx NCO test tone configuration 18](#_Toc98925831)

[4.1.2.10 Get the Tx NCO test tone configuration 18](#_Toc98925832)

[4.1.2.11 Set temperature gain compensation parameter 19](#_Toc98925833)

[4.1.2.12 Get temperature gain compensation parameter 19](#_Toc98925834)

[4.1.2.13 Get Madura’s temperature 19](#_Toc98925835)

[4.1.2.14 Run Madura’s init calibration 20](#_Toc98925836)

[4.1.2.15 Madura Rx Tx enable Set 20](#_Toc98925837)

[4.1.3 Flash Data 21](#_Toc98925838)

[4.1.3.1 Store data 21](#_Toc98925839)

[4.1.3.2 Load data 21](#_Toc98925840)

[4.1.3.3 Get list 22](#_Toc98925841)

[4.1.3.4 Erase 4k sector 22](#_Toc98925842)

[4.1.4 Flash Device 23](#_Toc98925843)

[4.1.4.1 Write byte 23](#_Toc98925844)

[4.1.4.2 Read byte 23](#_Toc98925845)

[4.1.4.3 Erase 4K sector 24](#_Toc98925846)

[4.1.4.4 Configuration commands 24](#_Toc98925847)

[4.1.5 IO FPGA Registers 25](#_Toc98925848)

[4.1.5.1 Read IO FPGA register 25](#_Toc98925849)

[4.1.5.2 Write IO FPGA register 25](#_Toc98925850)

[4.1.6 FPGA Registers 26](#_Toc98925851)

[4.1.6.1 Read FPGA register 26](#_Toc98925852)

[4.1.6.2 Write FPGA register 26](#_Toc98925853)

[4.1.7 Recording system 27](#_Toc98925854)

[4.1.7.1 Record IQ data 27](#_Toc98925855)

[4.1.7.2 Get recording buffer size 27](#_Toc98925856)

[4.1.7.3 Init play IQ data 27](#_Toc98925857)

[4.1.8 System global big payload memory section 28](#_Toc98925858)

[4.1.8.1 Store into big payload 28](#_Toc98925859)

[4.1.8.2 Load from big payload 28](#_Toc98925860)

[4.1.8.3 Get big payload buffer size 28](#_Toc98925861)

[4.1.9 Memory 29](#_Toc98925862)

[4.1.9.1 Read 32bit register from memory 29](#_Toc98925863)

[4.1.9.2 Write 32bit register to memory 29](#_Toc98925864)

[4.1.9.3 Read data from memory 29](#_Toc98925865)

[4.1.10 Thermometers 30](#_Toc98925866)

[4.1.10.1 Read temperature 30](#_Toc98925867)

[4.1.11 DAC – AD9561 (I2C) 30](#_Toc98925868)

[4.1.11.1 Write register 30](#_Toc98925869)

[4.1.12 EEPROM – AT24C02D (I2C) 30](#_Toc98925870)

[4.1.12.1 Write register 30](#_Toc98925871)

[4.1.12.2 Read register 31](#_Toc98925872)

[4.1.13 Synthesizer – LMX2594 (IOFPGA) 31](#_Toc98925873)

[4.1.13.1 Read register 31](#_Toc98925874)

[4.1.13.2 Write register 31](#_Toc98925875)

[4.1.13.3 Set LO frequency 31](#_Toc98925876)

[4.1.13.4 Set VCO force values 32](#_Toc98925877)

[4.1.14 Clock Buffer - CDCUN1208 (I2C) 33](#_Toc98925878)

[4.1.14.1 Init CDCUN1208 33](#_Toc98925879)

[4.1.14.2 Configuration OUTx 33](#_Toc98925880)

[4.1.14.3 Configuration Input 34](#_Toc98925881)

[4.1.14.4 Configuration dev 34](#_Toc98925882)

[4.1.14.5 Read register 35](#_Toc98925883)

[4.1.14.6 Write register 35](#_Toc98925884)

**Figures**

[Figure 1: Tx Serial frame structure 6](#_Toc70948679)

[Figure 2: Rx Serial frame structure 6](#_Toc70948680)

[Figure 3: Rx error Serial frame structure 7](#_Toc70948681)

**Tables**

[Table 1: Errors 7](#_Toc70948682)

[Table 2: Commands 9](#_Toc70948683)

# Scope

The purpose of this document is to detail the serial communication protocol used for controlling the OPTR which is part of DBF system in debug mode from a PC.

# Relevant Documents

1. TBD.

# Communication Protocol

## Introduction

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

## Hardware definitions

The communication protocol between PC and DBF system is a standard TCP/IP transport layer over Ethernet protocol using over 100BT.   
The system is a slave and the PC is the master thus the master initiate all transactions.

Host IPv4 address: 10.0.1.2  
DBF IPv4 address: 10.0.1.12  
Port no on both sides: 5555   
Subnet Mask: 255.255.255.0

## Frame structure

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

The header structure begins with a preamble byte 0x44 (project proprietary).   
The preamble followed by a single command byte, which described in the following tables. The next two bytes are the length of the data, next comes the data, which can be up to 216 bytes long and then the frame ends in a single byte checksum.   
The checksum is the result of the sum of all bytes in the frame.

### Tx frame:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x0044 | 2bytes | 4 bytes | TD0 | … | TDn | 2bytes |
| Preamble | Command op code | Data length | Data[0] |  | Data[n] | Check Sum |

Figure 1: Tx Serial frame structure

Preamble - The header structure begins with a preamble byte, which is project proprietary. For this project: is 0x0044 (ASCII 'D').   
  
Command - 2 bytes command, which described in the following tables.   
  
Data length - 4 bytes are the length of the data.

Data - Next comes the data, which can be up to 232 bytes long

Check sum - 2 bytes. Is the result of the sum of all bytes in the frame except the CRC field.

### Rx frame:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x0044 | 2bytes | 4 bytes | RD0 | … | RDn | 2bytes |
| Preamble | Command op code | Data length | Data[0] |  | Data[n] | Check Sum |

Figure 2: Rx Serial frame structure

Preamble - The header structure begins with a preamble byte, which is project

proprietary. For this project: is 0x0044 (ASCII 'D').

Command - 2 bytes command, which described in the following tables.   
  
Data length - 4 bytes are the length of the data.

Data - Next comes the data, which can be up to 232 bytes long.

Check sum - 2 bytes. Is the result of the sum of all bytes in the frame except the CRC

Field.

## Error codes

The target will return an error code in the CMD field when an error occurs. The following error codes 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 |  |
| Time-out Error – Type 3 | 0xF7 |  |
| Message length error | 0xF8 |  |
| Data length | 0xF9 |  |
|  |  |  |
|  |  |  |
| Reserved | 0xFB |  |
| Reserved | 0xFC |  |
| Reserved | 0xFD |  |
| Reserved | 0xFE |  |
| Reserved | 0xFF |  |

Table 1: Errors

### Rx Frame with error f.e:

|  |  |  |  |
| --- | --- | --- | --- |
| 0x0044 | 0x00F4 | 0x00000000 | 0x0138 |
| 2bytes | 2bytes | 4 BYTES | 2bytes |
| Preamble | Command op code | Data length | Check Sum |

Figure 3: Rx error Serial frame structure

Preamble - For this project: is 0x0044 (ASCII 'D').   
  
Command - 2 bytes command. Only from error table  
  
Data length – 0, no data

Check sum - 2 bytes. Is the result of the sum of all bytes in the frame except the CRC field. 🡪 0x0044 + 0x00F4 + 0x00000000 = 0x0138

# 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 | Reserved, Not done yet |
| 0x08 | Get system type | Read | Reserved, Not done yet |
| 0x09 | Get eyal PN | PN | Reserved, Not done yet |
|  |  |  |  |  |
| Madura | 0x0A | Init Madura HW | Write |  |
| 0x10 | Set RX channel gain | Write |  |
| 0x11 | Get RX channel gain | Read |  |
| 0x12 | Set TX channel attenuation | Write |  |
| 0x13 | Get TX channel attenuation | Read |  |
| 0x14 | Set Local Oscillator (LO) frequency | Write |  |
| 0x15 | Get Local Oscillator (LO) frequency | Read |  |
| 0x16 | Validate JESD status | Read |  |
| 0x17 | Set the Tx NCO test tone configuration | Write |  |
| 0x18 | Get the Tx NCO test tone configuration | Read |  |
| 0x19 | Set the temperature gain compensation parameter | Write |  |
| 0x1A | Get the temperature gain compensation parameter | Read |  |
| 0x1B | Get Madura’s temperature | Read |  |
| 0x1C | Set Madura’s init calibration | Write |  |
| 0x1D | Set Rx Tx enable | Write |  |
| 0x85 | Get memory dump from Madura's ARM | Read |  |
|  |  |  |  |  |
| Data Flash | 0x30 | Store data | Write |  |
| 0x31 | Load data | Read |  |
| 0x33 | Get memory list | Read | N/A |
| 0x34 | Erase 4k sector | Write | N/A |
|  |  |  |  |  |
| Flash | 0x37 | Write byte | Write |  |
| 0x38 | Read byte | Read |  |
| 0x39 | Erase 4K | Write |  |
| 0x40 | Configuration commands | Read/Write |  |
|  |  |  |  |  |
| IO FPGA | 0x60 | Read IO FPGA register | Read |  |
| 0x61 | Write IO FPGA register | 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 |  |
|  |  |  |  |  |
| Local big payload | 0x83 | Store big payload to flash device | Write |  |
| 0x84 | Load data from big payload | Read |  |
| 0x86 | Get big payload buffer size | Read |  |
|  |  |  |  |  |
| Memory | 0x90 | Read 32bit register | Read |  |
| 0x91 | Write 32bit register | Write |  |
| 0x92 | Read data from memory | Read |  |
|  |  |  |  |  |
| Thermometer | 0xA0 | Read Temperature | Read |  |
|  |  |  |  |  |
| DAC | 0xA1 | Write register | Write |  |
|  |  |  |  |  |
| EEPROM | 0xA2 | Write byte | Write |  |
| 0xA3 | Read byte | Read |  |
|  |  |  |  |  |
| Synthesizer | 0xA4 | Read register | Read |  |
| 0xA5 | Write register | Write |  |
| 0xA6 | Set LO frequency | Write |  |
| 0xA7 | Set VCO force values | Write |  |
|  |  |  |  |  |
| CCDCUN12 - I2C | 0xB1 | Init Sequence | Write |  |
| 0xB2 | Configuration OUTx | Write |  |
| 0xB3 | Configuration Input | Write |  |
| 0xB4 | Configuration dev | Write |  |
| 0xB5 | Read register | Read |  |
| 0xB6 | Write register | Write |  |

Table 2: Commands

## Details command description

### Standard group

#### Get software version

**Description:** Get the SW version: ICD version + UUT version and date.

**Command:** 0x01

**TX** **data:** N.A

**TX** **frame:** 0x0044 0x0001 0x00000000 0x45

**RX data:** 8 bytes version structure

ICD major version – 1 byte

ICD minor version – 1 byte

Unit major version – 1 byte

Unit minor version – 1 byte

Version day – 1 bytes

Version month – 1 bytes

Version year – 2 bytes

**RX** **frame:** 0x0044 0x0001 0x00000008 + RX Data + CS

#### Get firmware version

**Description:** Get the FW version: version and date.

**Command:** 0x02

**TX** **data:** N.A

**TX** **frame:** 0x0044 0x0002 0x00000000 0x46

**RX data:** 6 bytes version structure

Unit major version – 1 byte

Unit minor version – 1 byte

Version day – 1 bytes

Version month – 1 bytes

Version year – 2 bytes

**RX** **frame:** 0x0044 0x0002 0x00000006 + RX Data + CS

#### Get serial number

**Description:** Get the unit-stored serial number.

**Command:** 0x04

**TX** **data:** N.A

**TX** **frame:** 0x0044 0x0004 0x00000000 0x48

**RX data:** 2 bytes Serial number:

2 bytes - Serial number, range: 0 – 65535

**RX** **frame:** 0x0044 0x0004 0x00000002 + RX Data + CS

#### Set serial number

**Description:** Set the unit serial number.

**Command:** 0x05

**TX** **data:** 2 bytes Serial number:

2 bytes - Serial number, range: 0 – 65535

**TX** **frame:** 0x0044 0x0005 0x00000002 + TX Data + CS

**RX data:** N.A

**RX** **frame:** 0x0044 0x0005 0x00000000 0x49

#### Set log level

**Description:** Set log level severity.

**Command:** 0x06

**TX** **data:** 2 bytes:

1 byte - log severity: 0- log off, 1 (critical error) - 7(trace). Note: the selected severity means “up to” this severity

**TX** **frame:** 0x0044 0x0006 0x00000002 + TX Data + CS

**RX data:** N.A

**RX** **frame:** 0x0044 0x0006 0x00000000 0x4A

### Madura (adrv9026)

#### Madura init HW

**Description:** Init Madura HW (Both ADRV9026 devices)

**Command:** 0x000A

**TX** **data:** N.A

**TX** **frame:** 0x0044 0x000A 0x00000000 0x4E

**RX data:** N.A

**RX** **frame:** 0x0044 0x000A 0x00000000 0x4E

#### Set RX channel gain

**Description:** Set the Gain to the RF Cards

**Command:** 0x0010

**TX** **data:** 2 bytes:

1 byte - Madura bitwise RX channel selection: RX1=0X01, RX2=0X02,   
RX3=0X4, RX4=0X8, RX5=0X10, RX6=0X20, RX7=0X40, RX8=0X80

1 byte - Gain index. 183 – 255

**TX** **frame:** 0x0044 0x0010 0x00000002 + TX Data + CS

RX data: N.A

**RX** **frame:** 0x0044 0x0010 0x00000002 0x56

#### Get RX channel gain

**Description:** Get the Gain to the RF Cards

**Command:** 0x0011

**TX** **data:** 1 byte

Madura RX channel: RX1=0X01, RX2=0X02, RX3=0X4, RX4=0X8, RX5=0X10, RX6=0X20, RX7=0X40, RX8=0X80

**TX** **frame:** 0x0044 0x0011 0x00000001 + TX Data + CS

**RX data:** 1 byte

Gain index, range: 0-255

**RX** **frame:** 0x0044 0x0011 0x00000001 + RX Data + CS

#### Set TX channel attenuation

**Description:** Set the Gain to the RF Cards

**Command:** 0x12

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

1 byte – Madura TX channel: TX1 = 1, TX2 = 2, TX3 = 4, TX4 = 8,

TX5=0x10, TX6=0x20, TX7=0x40, TX8=0x80

4 bytes - TX attenuation (mdB). Range: 0 to 41950 mdB

**TX** **frame:** 0x0044 0x0012 0x00000005 + TX Data + CS

**RX data:** N.A

**RX** **frame:** 0x0044 0x0012 0x00000000 0x56

#### Get TX channel attenuation

**Description:** Get the Gain to the RF Cards

**Command:** 0x13

**TX** **data:** 1 byte:

Madura TX channel: TX1 = 1, TX2 = 2, TX3 = 4, TX4 = 8

TX5=0x10, TX6=0x20, TX7=0x40, TX8=0x80

**TX** **frame:** 0x0044 0x0013 0x00000001 + TX Data + CS

**RX data:** 4 bytes attenuation value

TX attenuation (mdB). Range: 0 to 41950 mdB

**RX** **frame:** 0x0044 0x0013 0x000000004 + RX Data + CS

#### Set Local Oscillator (LO) frequency

**Description:** Sets the RF PLL local oscillator (LO) frequency (RF carrier frequency)

**Command:** 0x14

**TX** **data:** 9 bytes control structure

1 byte - PLL name (LO1 =0, LO2 = 1, AUX = 2)

8 byte - PLL Frequency [Hz]

**TX** **frame:** 0x0044 0x0014 0x00000009 + TX Data + CS

**RX data:** N.A

**RX** **frame:** 0x0044 0x0014 0x00000000 0x58

#### Get Local Oscillator (LO) frequency

**Description:** Get the RF PLL local oscillator (LO) frequency (RF carrier frequency)

**Command:** 0x15

**TX** **data:** 1 byte:

PLL name (LO1 =0, LO2 = 1, AUX = 2)

**TX** **frame:** 0x0044 0x0015 0x00000001 + TX Data + CS

**RX data:** 8 bytes PLL frequency

PLL Frequency [Hz]

**RX** **frame:** 0x0044 0x0015 0x00000008 + RX Data + CS

#### Validate JESD status

**Description:** Plot to terminal and validate JESD status in particular Madura

**Command:** 0x16

**TX** **data:** 1 byte -Madura index [0-1]

**TX** **frame:** 0x0044 0x0016 0x00000001 + TX Data + CS

**RX data:** N.A

**RX** **frame:** 0x0044 0x0016 0x00000000 0x5A

#### Set the Tx NCO test tone configuration

**Description:** Generate tone from JESD. Enables/disables a digital numerically

controlled oscillator.

**Command:** 0x17

**TX** **data:** 7 bytes:

1 byte – bitwise Tx channel mask: ch0 = 0x1, ch1 = 0x2, ch2 = 0x4,

ch3 = 0x8, ch4 = 0x1, ch5 = 0x20, ch6 = 0x40, ch7 = 0x80

1 byte – 0 = Disable Tx NCO, 1 = Enable Tx NCO on both transmitters

4 byte – Signed frequency in [Hz] of the desired Tx tone

1 byte – NCO gain: 0 = -18 dB, 1 = -12 dB, 2 = -6 dB, 3 = 0 dB

**TX** **frame:** 0x0044 0x0017 0x00000007 + TX Data + CS

**RX data:** N.A

**RX** **frame:** 0x0044 0x0017 0x00000000 0x69

#### Get the Tx NCO test tone configuration

**Description:** Get configuration of generated tone from JESD.

**Command:** 0x18

**TX data:** 1 byte:

bitwise Tx channel mask: ch0 = 0x1, ch1 = 0x2, ch2 = 0x4,

ch3 = 0x8, ch4 = 0x1, ch5 = 0x20, ch6 = 0x40, ch7 = 0x80

**TX** **frame:** 0x0044 0x0018 0x00000007 + TX Data + CS

**RX data:** 7 bytes:

1 byte – Tx selected channel: ch0 = 0x1, ch1 = 0x2, ch2 = 0x4,

ch3 = 0x8, ch4 = 0x1, ch5 = 0x20, ch6 = 0x40, ch7 = 0x80

1 byte – 0 = Disable Tx NCO, 1 = Enable Tx NCO on both transmitters

4 byte – Signed frequency in [Hz] of the desired Tx tone

1 byte – NCO gain: 0 = -18 dB, 1 = -12 dB, 2 = -6 dB, 3 = 0 dB

**RX** **frame:** 0x0044 0x0018 0x00000007 0x69

#### Set temperature gain compensation parameter

**Description:** Sets the temperature gain compensation parameter.

**Command:** 0x19

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

1 byte – bitwise Rx channel mask: ch0 = 0x1, ch1 = 0x2, ch2 = 0x4,

ch3 = 0x8, ch4 = 0x1, ch5 = 0x20, ch6 = 0x40, ch7 = 0x80

4 byte – [float] Gain can be configured from -6.3 dB to 6.3 dB in 0.1 dB

increments.

**TX** **frame:** 0x0044 0x0019 0x00000007 + TX Data + CS

**RX data:** N.A

**RX** **frame:** 0x0044 0x0019 0x00000000 0x63

#### Get temperature gain compensation parameter

**Description:** Gets the temperature gain compensation parameter.

**Command:** 0x1A

**TX** **data:** 1 byte – Rx channel sel : ch0 = 0x1, ch1 = 0x2, ch2 = 0x4,

ch3 = 0x8, ch4 = 0x1, ch5 = 0x20, ch6 = 0x40, ch7 = 0x80

**TX** **frame:** 0x0044 0x0019 0x00000007 + TX Data + CS

**RX data:** 4 bytes – [float] Gain value retrieved [db]

**RX** **frame:** 0x0044 0x0019 0x00000004 + RX Data + CS

#### Get Madura’s temperature

**Description:** Read temperature of Madura

**Command:** 0x1B

**TX** **data:** 1 byte – device index: 0x0, 0x1

**TX** **frame:** 0x0044 0x001B 0x00000000 + TX Data + CS

**RX data:** 2 bytes -12bit temperature sensor value in degrees C

**RX** **frame:** 0x0044 0x001B 0x00000002 + RX Data + CS

#### Run Madura’s init calibration

**Description:** Runs the ADRV9026 initialization calibrations

**Command:** 0x1C

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

1 byte - 0x01-0xFF: Madura#1- Tx/Rx[0:3], Madura#2- Tx/Rx[4:7]

Mask to hold Channel to run on init, 1 bit per channel. Applies to

the specific cals selected.

4 bytes - Calibration init mask

**TX** **frame:** 0x0044 0x001C 0x00000005 + TX Data + CS

**RX data:** N.A

**RX** **frame:** 0x0044 0x001C 0x00000000 + 0x60

#### Madura Rx Tx enable Set

**Description:** ADRV9026 Rx TX enable channels

**Command:** 0x1D

**TX** **data:** 2 bytes

1 byte - Mask to Rx channels to turn on init. (valid 0x01-0xFF), 1 bit per

channel. Applies to the specific cals selected.  
 Madura#1- Rx[0:3], Madura#2- Rx[4:7]

1 byte - Mask to Tx channels to turn on init. (valid 0x01-0xFF), 1 bit per

channel. Applies to the specific cals selected.

Madura#1- Tx [0:3], Madura#2- Tx[4:7]

**TX** **frame:** 0x0044 0x001D 0x00000002 + TX Data + CS

**RX data:** N.A

**RX** **frame:** 0x0044 0x001D 0x00000000 + 0x61

### Flash Data

Data Flash fields:

enum

{

DB\_SERIAL\_NUMBER\_ADDR = 0x0000,

DB\_ETHERNET\_CFG = 0x1000,

DB\_MAC\_ADDR = 0x2000,

DB\_MADURA\_LO\_FREQ\_ADDR = 0x3000,

DB\_FEM\_BIAS\_ADDR = 0x4000,

DB\_DAC\_TRIM\_28V\_ADDR = 0x5000,

DB\_LOVCO\_FORCED\_VAL\_ADDR = 0x6000,

DB\_LAST\_SECTOR\_ADDR = 0x7000

};

#### Store data

**Description:** Store data in flash device

**Command:** 0x0030

**TX** **data:** 8 bytes + N bytes:

4 byte – Destination address [4k address]

4 bytes – Data length

N bytes – Data content

**TX** **frame:** 0x0044 0x0030 + TX len + TX Data + CRC

**RX data:** 4 bytes - Start address where data has been amounted

**RX** **frame:** 0x0044 0x0030 0x00000004 + RX data + CRC

#### Load data

**Description:** Load data from flash

**Command:** 0x0031

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

4 byte – address [4k address]

4 bytes – size of loaded data (bytes)

**TX** **frame:** 0x0044 0x0031 0x00000005 + TX Data + CRC

**RX data:** N bytes - RX data

**RX** **frame:** 0x0044 0x0031 + size + RX Data + CRC

#### Get list

**Description:** Get list of information about all stored data

**Command:** 0x0033

**TX** **data:** N.A

**TX** **frame:** 0x0044 0x0033 0x000000010 0x87

**RX data:** N bytes - Array of data information.

Each cell in array is structure of: 12 bytes - File name

4 bytes - Address

4 bytes - Data size

**RX** **frame:** 0x0044 0x0033 + size + RX Data + CRC

#### Erase 4k sector

**Description:** Erase 4K sector in memory

**Command:** 0x0034

**TX** **data:** 4 bytes address: start address of sector to be erased

**TX** **frame:** 0x0044 0x0034 0x00000004 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x0034 0x00000000 0x78

### Flash Device

#### Write byte

**Description:** Write byte

**Command:** 0x0037

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

4 byte – Destination address

1 byte - data

**TX** **frame:** 0x0044 0x0037 0x00000005 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x0037 0x00000000 0x7B

#### Read byte

**Description:** Read byte from flash.

**Command:** 0x0038

**TX** **data:** 4 bytes – address

**TX** **frame:** 0x0044 0x0038 0x00000004 + TX Data + CRC

**RX data:** 1 byte - data

**RX** **frame:** 0x0044 0x0038 0x00000001 + RX Data + CRC

#### Erase 4K sector

**Description:** Erase 4k sector

**Command:** 0x0039

**TX** **data:** 4 bytes – Any address within sector

**TX** **frame:** 0x0044 0x0039 0x00000004 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x0039 0x00000000 0x7D

#### Configuration commands

**Description:** Read and write configuration commands

**Command:** 0x0040

**TX** **data:** 1 bytes – Selection of configuration command:

0x0 – Print status register to terminal

0x1 – Print flash ID to terminal

0x2 – Set write enable (WEL)

0x3 – Disable write protection (SRWD)

0x4 – Enable write protection (SRWD)

**TX** **frame:** 0x0044 0x0040 0x00000001 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x0040 0x00000000 0x7E

### IO FPGA Registers

#### Read IO FPGA register

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

**Command:** 0x0060

**TX** **data:** 1 bytes - register number

**TX** **frame:** 0x0044 0x0060 0x00000004 + TX Data + CRC

**RX data:** 3 byte – retrieved data

**RX** **frame:** 0x0044 0x0060 0x00000004 + RX Data + CRC

#### Write IO FPGA register

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

**Command:** 0x0061

**TX** **data:** 4 bytes:

1 byte – register number

3 bytes – register value

**TX** **frame:** 0x0044 0x0061 0x00000008 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x0061 0x00000004 + RX Data + CRC

### FPGA Registers

#### Read FPGA register

**Description:** Read FPGA 32bit register

**Command:** 0x0070

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

**TX** **frame:** 0x0044 0x0070 0x00000004 + TX Data + CRC

**RX data:** 4 byte – retrieved data

**RX** **frame:** 0x0044 0x0070 0x00000004 + RX Data + CRC

#### Write FPGA register

**Description:** Write FPGA 32bit register

**Command:** 0x0071

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

4 bytes – register number

4 bytes – register value

**TX** **frame:** 0x0044 0x0071 0x00000008 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x0071 0x00000004 + RX Data + CRC

### Recording system

#### Record IQ data

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

**Command:** 0x0080

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

1 byte – Channels mask to record

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

**TX** **frame:** 0x0044 0x0080 0x00000005 0xC9

**RX data:** 1-200MBytes – sequence of recorded IQ data

**RX** **frame:** 0x0044 0x0080 + size + RX data + CRC

#### Get recording buffer size

**Description:** Get amounted size for recording buffer

**Command:** 0x0081

**TX** **data:** N.A

**TX** **frame:** 0x0044 0x0082 + size + TX Data + CRC

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

**RX** **frame:** 0x0044 0x0082 0x00000000 0XC6

#### Init play IQ data

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

**Command:** 0x0082

**TX** **data:** 1-200Mbytes – sequence of loaded data

**TX** **frame:** 0x0044 0x0082 + size + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x0082 0x00000000 0XC6

### System global big payload memory section

#### Store into big payload

**Description:** Store sequence of data into big payload memory section

**Command:** 0x0083

**TX** **data:** 1-100Mbytes – sequence of data

**TX** **frame:** 0x0044 0x0083 + data len + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x0083 + data len + CRC

#### Load from big payload

**Description:** Load data from big payload memory section

**Command:** 0x0084

**TX** **data:** 4 bytes – length of data to load

**TX** **frame:** 0x0044 0x0084 0x00000004 + TX Data + CRC

**RX data:** 1-100Mbytes – sequence of loaded data

**RX** **frame:** 0x0044 0x0084 + data len + RX Data + CRC

#### Get big payload buffer size

**Description:** Get amounted bytes for big payload buffer

**Command:** 0x0086

**TX** **data:** N.A

**TX** **frame:** 0x0044 0x0086 0x00000000 0xCA

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

**RX** **frame:** 0x0044 0x0086 0x00000004 + RX Data + CRC

### Memory

#### Read 32bit register from memory

**Description:** Read 32bit register memory from memory

**Command:** 0x0090

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

**TX** **frame:** 0x0044 0x0090 0x00000004 + TX Data + CRC

**RX data:** 4 bytes – value

**RX** **frame:** 0x0044 0x0090 0x00000004 + RX Data + CRC

#### Write 32bit register to memory

**Description:** Write 32bit register to memory

**Command:** 0x0091

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

4 bytes – address

4 bytes – 32bit register value

**TX** **frame:** 0x0044 0x0091 0x00000008 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x0091 0x00000004 + RX Data + CRC

#### Read data from memory

**Description:** Read n bytes from memory, starting from specific address

**Command:** 0x0092

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

**TX** **data:** 4 bytes – length

**TX** **frame:** 0x0044 0x0092 0x00000008 + TX Data + CRC

**RX data:** 1- 0x6400000 – sequence of loaded data

**RX** **frame:** 0x0044 0x0090 0x00000008 + RX Data + CRC

### Thermometers

#### Read temperature

**Description:** Read temperature from selected thermometer

**Command:** 0x00A0

**TX** **data:** 1 byte – channel/thermometer

**TX** **frame:** 0x0044 0x00A0 0x00000001 + TX Data + CRC

**RX data:** 2 bytes – temperature (12 left aligned bits)

**RX** **frame:** 0x0044 0x00A0 0x00000002 + RX Data + CRC

### DAC – AD9561 (I2C)

#### Write register

**Description:** Write DAC register

**Command:** 0x00A1

**TX** **data:** 2 bytes – 12 bit value [0-4095]

**TX** **frame:** 0x0044 0x00A1 0x00000002 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x00A1 0x00000000 0xE5

### EEPROM – AT24C02D (I2C)

#### Write register

**Description:** Write byte

**Command:** 0x00A2

**TX** **data:** 2 bytes:

1 byte – address

1 byte – data

**TX** **frame:** 0x0044 0x00A2 0x00000002 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x00A2 0x00000000 0xE6

#### Read register

**Description:** Read byte

**Command:** 0x00A3

**TX** **data:** 1 byte - address

**TX** **frame:** 0x0044 0x00A3 0x00000001 + TX Data + CRC

**RX data:** 1 byte – data

**RX** **frame:** 0x0044 0x00A3 0x00000001 + RX Data + CRC

### Synthesizer – LMX2594 (IOFPGA)

#### Read register

**Description:** Read register

**Command:** 0x00A4

**TX** **data:** 1 byte – register address [0-112]

**TX** **frame:** 0x0044 0x00A4 0x00000001 + TX Data + CRC

**RX data:** 1 byte – data

**RX** **frame:** 0x0044 0x00A4 0x00000001 + RX Data + CRC

#### Write register

**Description:** Write register

**Command:** 0x00A5

**TX** **data:** 2 bytes:

1 byte – register address

1 byte – data

**TX** **frame:** 0x0044 0x00A5 0x00000002 + TX Data + CRC

**RX data:** 1 byte – result

**RX** **frame:** 0x0044 0x00A5 0x00000001 + RX Data + CRC

#### Set LO frequency

**Description:** Set LO frequency

**Command:** 0x00A6

**TX** **data:** 1 byte:

1 byte – frequency index [0-80]

**TX** **frame:** 0x0044 0x00A6 0x00000001 + TX Data + CRC

**RX data:** 1 byte – result

**RX** **frame:** 0x0044 0x00A6 0x00000001 + RX Data + CRC

#### Set VCO force values

**Description:** Set VCO force values. This function uses vco\_sel,

vco\_dacisel and vco\_capCtrl according to frequency index.

This values are stored in the data flash during calibration process.

**Command:** 0x00A7

**TX** **data:** 1 byte:

1 byte – frequency index [0-80]

**TX** **frame:** 0x0044 0x00A7 0x00000001 + TX Data + CRC

**RX data:** 1 byte – result

**RX** **frame:** 0x0044 0x00A6 0x00000007 + RX Data + CRC

### Clock Buffer - CDCUN1208 (I2C)

#### Init CDCUN1208

**Description:** Run commands to init CDCUN1208 device.

**Command:** 0x00B1

**TX** **data:** N.A

**TX** **frame:** 0x0044 0x00B1 0x00000000 0xF5

**RX data:** N.A

**RX** **frame:** 0x0044 0x00B1 0x00000000 0xF5

#### Configuration OUTx

**Description:** Set configuration of OUTx.

**Command:** 0x00B2

**TX** **data:** 2 bytes or 5 bytes or 6 bytes:

1 byte channel – decimal 1-8

1 byte state - '1'- DISABLED, '0'-'Enabled'

Optional bytes:

1 byte type - '0'-LVDS, '1'-LVCMOS, '3'-HCSL

1 byte enable - '0'-OFF, '1'-OFF\_ON, '2'-ON\_OFF, '3'-ON

1 byte rate - '0'-SLOW, '4'-FAST, '7'-MEDIUM"

for type = '0'-LVDS or '3'-HCSL:

1 byte cmos\_mode - '0'-IN\_PHASE, '1'-COMP

**TX** **frame:** 0x0044 0x00B2 + data len + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x00B2 0x00000000 0xF6

#### Configuration Input

**Description:** Set configuration of input

**Command:** 0x00B3

**TX** **data:** 3 bytes:

1 byte inmux - '0'-IN1, '1'-IN2, '2'-SMART, '3'-PIN\_SEL

1 byte type '0'-LVDS, '1'-LVCMOS, '3'-HCSL

1 byte div '0'-DIV 1, '1'-DIV 2, '2'-DIV 4, '3'-DIV 8

**TX** **frame:** 0x0044 0x00B3 0x00000003 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x00B3 0x00000000 0xF7

#### Configuration dev

**Description:** Set configuration of input

**Command:** 0x00B4

**TX** **data:** 3 bytes:

1 byte dev\_reset '0'-Run device, '1'-Reset device

1 byte dev\_pd '0'-Device is active, '1'-Device is powered down

**TX** **frame:** 0x0044 0x00B4 0x00000002 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x00B4 0x00000000 0xF8

#### Read register

**Description:** Read value from CCDCUN1208 register

**Command:** 0x00B5

**TX** **data:** 1 bytes - register address – 0 … 15

**TX** **frame:** 0x0044 0x00B5 0x00000001 + TX Data + CRC

**RX data:** 2 bytes - register value

**RX** **frame:** 0x0044 0x00B5 0x00000001 + RXData + CRC

#### Write register

**Description:** Write 16bit data to CCDCUN1208 register

**Command:** 0x00B6

**TX** **data:** 3 bytes

1byte - register address – 0 … 15

2bytes – data

**TX** **frame:** 0x0044 0x00B6 0x00000006 + TX Data + CRC

**RX data:** N.A

**RX** **frame:** 0x0044 0x00B6 0x00000000 0xFA