Communication protocol

**Table of Contents**

[1. Serial port parameters 2](#_Toc429402652)

[2. Brief command list 2](#_Toc429402653)

[3. Extended command list 3](#_Toc429402654)

[3.1. getFirmwareID 3](#_Toc429402655)

[3.2. takeMeasEis 3](#_Toc429402656)

[3.3. giveMeasChunkEis 5](#_Toc429402657)

[3.4. endMeasEis 5](#_Toc429402658)

[3.5. takeMeasCv 6](#_Toc429402659)

[3.6. giveMeasChunkCV 7](#_Toc429402660)

[3.7. endMeasCV 8](#_Toc429402661)

[4. Crc calculation 9](#_Toc429402662)

[4.1. Algorithm 9](#_Toc429402663)

[4.2. Example 9](#_Toc429402664)

# Serial port parameters

|  |  |
| --- | --- |
| Parameter name | Value |
| Baudrate | 115200 |
| Data bits | 8 |
| Parity | Even |
| Stop bits | 1 |
| Flow control | None |

# Brief command list

|  |  |  |  |
| --- | --- | --- | --- |
| Command | Code | Sender | Description |
| Generic commands | | | |
| getFirmwareID | 0x01 | PC1 | Requests the firmware version of embedded system. Also, provides information for the PC app that the system is connected. |
| Electrochemical Impedance Spectroscopy (EIS) commands | | | |
| takeMeasEis | 0x02 | PC2 | All required parameters are sent from PC to ES in order for ES to start the EIS measurement. |
| giveMeasChunkEis | 0x03 | ES | ES sends a chunk of measured data to the PC |
| endMeasEis | 0x04 | ES | After the last chunk of EIS data is sent, ES sends this command in order to close the measurement. |
| Cyclic Voltammetry (CV) commands | | | |
| takeMeasCv | 0x05 | PC | All required parameters are sent from PC to ES in order for ES to start the CV measurement. |
| giveMeasChunkCV | 0x06 | ES | ES sends a chunk of measured data to the PC |
| endMeasCV | 0x07 | ES | After the last chunk of CV data is sent, ES sends this command in order to close the measurement. |

1PC – Computer user application,

2ES – Embedded system.

# Extended command list

## getFirmwareID

PC sends getFirmwareID to the embedded system in order to retrieve the current firmware version of the embedded system and also obtain the information about connection. If the firmware version is returned, it means that embedded system is connected. This command should be used as communication initialization command.

**Communicate:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | CRC 0 | CRC 1 |
| ‘?’ (0x3F) | 0x01 | 0x02 | 0x00 | 0x00 | 0x00 | 0xXX | 0xXX |

**Answer:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | Firmware 0 | Firmware 1 |
| ‘?’ (0x3F) | 0x01 | 0x06 | 0x00 | 0x00 | 0x00 | 0xXX | 0xXX |

|  |  |  |  |
| --- | --- | --- | --- |
| Byte 8 | Byte 9 | Byte 10 | Byte 11 |
| Firmware 2 | Firmware 3 | CRC 0 | CRC 1 |
| 0xXX | 0xXX | 0xXX | 0xXX |

## takeMeasEis

PC sends all required parameters for the EIS measurement to the ES. After ES obtains the command it then checks the parameters. If they are all ok, he sends an ok ACK, if not, he sends an error code ACK.

**Communicate:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | amplitude | Freq  Range  Start 0 |
| ‘?’ (0x3F) | 0x02 | 0x0E | 0x00 | 0x00 | 0x00 | 0-100 | 0xXX |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 8 | Byte 9 | Byte 10 | Byte 11 | Byte 12 | Byte 13 | Byte 14 | Byte 15 |
| Freq  Range  Start 1 | Freq  Range  Start 2 | Freq  Range  Start 3 | Freq  Range  End 0 | Freq  Range  End 1 | Freq  Range  End 2 | Freq  Range  End 3 | Freq  Range  Step 0 |
| 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX |

|  |  |  |  |
| --- | --- | --- | --- |
| Byte 16 | Byte 17 | Byte 18 | Byte 19 |
| Freq  Range  Step 1 | StepType | CRC 0 | CRC 1 |
| 0xXX | 0-1 | 0xXX | 0xXX |

**amplitude (Byte 6):** Maximum signal amplitude expressed in mili volts, values from 0 to 100 (unsigned char).

**FreqRangeStart (Bytes 7-10):** The starting frequency of the measurement expressed as float value.

**FreqRangeEnd (Bytes 11-14):** The ending frequency of the measurement expressed as float value.

**FreqRangeStep (Bytes 15-16):** Numbers of steps/ measurements to take between the starting and ending frequency.

**StepType (Byte 17):** 0 – Linear, 1 – Logarithmical.

**Answer:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | ACK | CRC 0 |
| ‘?’ (0x3F) | 0x02 | 0x03 | 0x00 | 0x00 | 0x00 | 0-1 | 0xXX |

|  |
| --- |
| Byte 8 |
| CRC 1 |
| 0xXX |

**ACK (Byte 6):** 0 – parameters OK, 1 – parameters invalid, measurement won’t be started.

## giveMeasChunkEis

ES sends measurement results to the PC, for a certain frequency step. This message required no answer.

**Communicate:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | Real 0 | Real 1 |
| ‘?’ (0x3F) | 0x03 | 0x0E | 0x00 | 0x00 | 0x00 | 0xXX | 0xXX |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 8 | Byte 9 | Byte 10 | Byte 11 | Byte 12 | Byte 13 | Byte 14 | Byte 15 |
| Real 2 | Real 3 | Imag 0 | Imag 1 | Imag 2 | Imag 3 | Freq 0 | Freq 1 |
| 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX |

|  |  |  |  |
| --- | --- | --- | --- |
| Byte 16 | Byte 17 | Byte 18 | Byte 19 |
| Freq 2 | Freq 3 | CRC 0 | CRC 1 |
| 0xXX | 0xXX | 0xXX | 0xXX |

**Real (Bytes 6-9):** Real part of measured impedance expressed as float.

**Imag (Bytes 10-13):** Imaginary part of measured impedance expressed as float.

**Freq (Bytes 14-17):** Frequency of the signal witch which measurement was taken.

## endMeasEis

ES sends this command to the PC right after the last giveMeasChunk was sent in order to close the measurement process.

**Communicate:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | CRC 0 | CRC 1 |
| ‘?’ (0x3F) | 0x04 | 0x02 | 0x00 | 0x00 | 0x00 | 0xXX | 0xXX |

**Answer:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | CRC 0 | CRC 1 |
| ‘?’ (0x3F) | 0x04 | 0x02 | 0x00 | 0x00 | 0x00 | 0xXX | 0xXX |

## takeMeasCv

PC sends all required parameters for the CV measurement to the ES. After ES obtains the command it then checks the parameters. If they are all ok, he sends an ok ACK, if not, he sends an error code ACK.

**Communicate:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | Start Potential 0 | Start Potential 1 |
| ‘?’ (0x3F) | 0x05 | 0x0B | 0x00 | 0x00 | 0x00 | 0xXX | 0xXX |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 8 | Byte 9 | Byte 10 | Byte 11 | Byte 12 | Byte 13 | Byte 14 | Byte 15 |
| End Potential 0 | End Potential 1 | Number of cycles | Potential step 0 | Potential step 1 | Scanning speed 0 | Scanning speed 1 | CRC 0 |
| 0xXX | 0xXX | 0xXX | 0x00 | 0x00 | 0x00 | 0xXX | 0xXX |

|  |
| --- |
| Byte 16 |
| CRC 1 |
| 0xXX |

**Start potential (bytes 6-7):** The first working signal potential [mV]. Values from 1 to 0xFFFF (16 bit).

**End potential (bytes 8-9):** The last working signal potential [mV]. Values from 1 to 0xFFFF (16 bit).

**Number of cycles (byte 10):** A number of full measure cycles to make. Values from 1 to 0xFF (8 bit).

**Potential step (bytes 11-12):** Each measurement step is greater than the last by the factor of Potential step [mV]. When it exceeds End Potential value, the measurement is finished. Values from 1 to 0xFFFF (16 bit).

**Scanning speed (bytes 13-14):** Time interval in between which the measurements should be taken [ms]. Values from 1 to 0xFFFF (16 bit).

**Answer:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | ACK | CRC 0 |
| ‘?’ (0x3F) | 0x05 | 0x03 | 0x00 | 0x00 | 0x00 | 0-1 | 0xXX |

|  |
| --- |
| Byte 8 |
| CRC 1 |
| 0xXX |

**ACK (Byte 6):** 0 – parameters OK, 1 – parameters invalid, measurement won’t be started.

## giveMeasChunkCV

ES sends measurement results to the PC, for a certain potential step. This message required no answer.

**Communicate:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | Sample number 0 | Sample number 1 |
| ‘?’ (0x3F) | 0x06 | 0x0C | 0x00 | 0x00 | 0x00 | 0xXX | 0xXX |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 8 | Byte 9 | Byte 10 | Byte 11 | Byte 12 | Byte 13 | Byte 14 | Byte 15 |
| Current value 0 | Current value 1 | Current value 2 | Current value 3 | Voltage value 0 | Voltage value 1 | Voltage value 2 | Voltage value 3 |
| 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX | 0xXX |

|  |  |
| --- | --- |
| Byte 16 | Byte 17 |
| CRC 0 | CRC 1 |
| 0xXX | 0xXX |

**Sample number (bytes 6-7):** number of the currently sent sample. Values from 0 to 0xFFFF (16 bit).

**Current value (bytes 8-11):** Current measurement for a sample. 32 bit float value.

**Voltage value (bytes 8-11):** Voltaget measurement for a sample. 32 bit float value.

## endMeasCV

ES sends this command to the PC right after the last giveMeasChunk was sent in order to close the measurement process.

**Communicate:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | CRC 0 | CRC 1 |
| ‘?’ (0x3F) | 0x07 | 0x02 | 0x00 | 0x00 | 0x00 | 0xXX | 0xXX |

**Answer:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| Sync Byte | Command | Length 0 | Length 1 | Length 2 | Length 3 | CRC 0 | CRC 1 |
| ‘?’ (0x3F) | 0x07 | 0x02 | 0x00 | 0x00 | 0x00 | 0xXX | 0xXX |

# Crc calculation

Crc is stored as an 2 byte unsigned integer value. CRC is calculated for every communicate and answer command and added at the end of that command. After the PC/ ES receives a message, it calculates the CRC for it and compares with the CRC of this sent message.

## Algorithm

|  |
| --- |
| unsigned short GetCrc(unsigned char\* buffer, unsigned short bytes)  {  unsigned short i;  unsigned short temp = 0;  for(i = 0; i < bytes; i++)  {  temp += \*buffer++;  }  temp =~ temp;  return temp;  } |

## Example

getFirmwareID is sent from PC to ES, the whole message is:

0x3F 0x01 0x02 0x00 0x00 0x00 0xBD 0xFF

Embedded system receives this message and calculates the CRC of it (blue bytes):

0x3F + 0x01 + 0x02 + 0x00 + 0x00 + 0x00 = 0x0043

~0x0043 = 0xFFBD

Calculated CRC for this message matches the one that came along with it. That means the frame is corrent and an answer can be sent to the PC:

0x3F 0x01 0x06 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0xB8 0xFF

PC receives the message and checks it by calculating CRC:

0x3F + 0x01 + 0x06 + 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0x00 + 0x01 = 0x0047

~0x0047 = 0xFFB8

Calculated CRC matches the sent one. ES answered that its firmware number is 1.0.0.0 .