* Table of Contents

[1. Introduction 2](#__RefHeading___Toc2915_3497913849)

[2. General message format 2](#__RefHeading___Toc2917_3497913849)

[3. Message definitions 3](#__RefHeading___Toc2919_3497913849)

[Get status command 3](#__RefHeading___Toc2921_3497913849)

[Get status response 3](#__RefHeading___Toc2923_3497913849)

[Set humidity threshold command 4](#__RefHeading___Toc2925_3497913849)

[Set humidity threshold response 4](#__RefHeading___Toc2927_3497913849)

[Get humidity threshold command 5](#__RefHeading___Toc2929_3497913849)

[Get humidity threshold response 5](#__RefHeading___Toc2931_3497913849)

[Get humidity command 6](#__RefHeading___Toc2933_3497913849)

[Get humidity response 6](#__RefHeading___Toc2935_3497913849)

[Set a TEC set point temperature command 7](#__RefHeading___Toc2937_3497913849)

[Set a TEC set point temperature response 7](#__RefHeading___Toc2939_3497913849)

[Get a TEC set point temperature command 8](#__RefHeading___Toc2941_3497913849)

[Get a TEC set point temperature response 8](#__RefHeading___Toc2943_3497913849)

[Set chiller set point temperature command 9](#__RefHeading___Toc2945_3497913849)

[Set chiller set point temperature response 9](#__RefHeading___Toc2947_3497913849)

[Get chiller set point temperature command 10](#__RefHeading___Toc2949_3497913849)

[Get chiller set point temperature response 10](#__RefHeading___Toc2951_3497913849)

[Enable TECs command 11](#__RefHeading___Toc2953_3497913849)

[Enable TECs response 11](#__RefHeading___Toc2955_3497913849)

[Disable TECs command 11](#__RefHeading___Toc2957_3497913849)

[Disable TECs response 12](#__RefHeading___Toc2959_3497913849)

[Start up command 12](#__RefHeading___Toc2961_3497913849)

[Start up response 13](#__RefHeading___Toc2963_3497913849)

[Shutdown command 13](#__RefHeading___Toc2965_3497913849)

[Shutdown response 14](#__RefHeading___Toc2967_3497913849)

### Introduction

This section describes the protocol between the control PC and the unit.

The protocol is command and response and has the following attributes.

* Start of packet character
  + ‘#’ for command from a master
  + ‘!’ for a response from a slave
* An address, master is 0, slaves are 1 - n
* Sequence number
  + The sequence number sent by the master will be returned back by the slave in replies, this will allow the master to verify whether the response is for the current command.
* Packet length and CRC to protect against corrupt packets.
* End of packet character

Master ID 0

Slave ID n

Slave ID 1

### **General message format**

Following message attributes are present in all messages.

|  |  |  |
| --- | --- | --- |
| **Byte(s)** | **Data type** | **Description / Value(s)** |
| 1 | uint8\_t | Start of packet byte  - ‘#’ for packets from master  - ‘!’ for packets from slave |
| 2 | uint8\_t | Packet length  – this is the entire length of the packet |
| 3, 4 | uint16\_t | Address  – 0 for master, 1 – n for slaves |
| 5 | uint8\_t | Sequence number  - chosen by the master, can be any number  - returned by the slave in command response |
| 6 | uint8\_t | Message number  - identifies the command, i.e. getStatus, startUp, etc. |
| **7 – N** | **\*** | **Message body if present required for the command** |
| N + 1 | uint16\_t | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| N + 3 | uint16\_t | End of packet byte  - ‘\r’ (0x000D) |

### Message definitions

### Get status command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x0A | Packet length (hex value for length 10) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x00 | Message number |
| 7, 8 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 9, 10 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### **Get status response**

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x10 | Packet length (hex value for length 16) |
| 3, 4 | uint16\_t | 0x0000 | Master address |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x01 | Message number |
| 7, 8 | uint16\_t | 0x0000  or  0x0001 | Humidity alert  0 – no alert present  1 – alert present – humidity is at or above the threshold |
| 9, 10 | uint16\_t | 0x0000  or  0x0001 | TECs running  0 – TECs are not running  1 – TECs are running |
| 11, 12 | uint16\_t | 0x0000  or  0x0001 | Chiller running  0 – chiller is not running  1 – chiller is running |
| 13, 14 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 15, 16 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Set humidity threshold command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x000C | Packet length (hex value for length 12) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x02 | Message number |
| 7, 8 | uint16\_t | 0x0020 | Humidity threshold in network byte order, hex value, for example:  - 0x0020 is 32 – 32% humidity will be the new threshold  - 0x0030 is 48 – 48% humidity will be the new threshold |
| 9, 10 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 11, 12 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Set humidity threshold response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x0C | Packet length (hex value for length 12) |
| 3, 4 | uint16\_t | 0x0000 | Master address in network byte order |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x03 | Message number |
| 7, 8 | uint16\_t | 0x0000  or  0x0001 | Result  0 – fail  1 – success |
| 9, 10 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 11, 12 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Get humidity threshold command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x000A | Packet length (hex value for length 10) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x04 | Message number |
| 7, 8 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 9, 10 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Get humidity threshold response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x000C | Packet length (hex value for length 12) |
| 3, 4 | uint16\_t | 0x0000 | Master address in network byte order |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x05 | Message number |
| 7, 8 | uint16\_t | 0x0040 | Current threshold in network byte order, hex value, for example  - 0x0040 – current threshold is 64% humidity |
| 9, 10 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 11, 12 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Get humidity command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x000A | Packet length (hex value for length 10) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x06 | Message number |
| 7, 8 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 9, 10 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Get humidity response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x0012 | Packet length (hex value for length 18) |
| 3, 4 | uint16\_t | 0x0000 | Master address in network byte order |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x06 | Message number |
| 7 - 14 | uint8\_t | 8 byte null term string | Current humidity in null terminated ASCII string, for example client will send:  - “47.9”0x00 for 47.9% humidity  - “58.3”0x00 for 58.3% humidity |
| 15, 16 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 17, 18 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Set a TEC set point temperature command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x0014 | Packet length (hex value for length 20) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x08 | Message number |
| 7, 8 | uint16\_t | 0x0002 | TEC address to set, 16bit integer value in network byte order |
| 9 - 16 | uint8\_t | 8 byte null term string | TEC temperature in null terminated ASCII string, for example master will send something like the following, leading +/- sign supported, two decimal points supported.  - “-23.71”0x00 for -23.71C for the TEC  - or “5”0x00 for +5.00C for the TEC  - or “+10.83” for +10.83C for the TEC |
| 17, 18 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 19, 20 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Set a TEC set point temperature response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x000E | Packet length (hex value for length 14) |
| 3, 4 | uint16\_t | 0x0000 | Master address in network byte order |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x09 | Message number |
| 7, 8 | uint16\_t | 0x02 | TEC address, hex, network byte order |
| 9, 10 | uint16\_t | 0x0000  or  0x0001 | Result  0 – fail  1 – success |
| 11, 12 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 13, 14 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Get a TEC set point temperature command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x000C | Packet length (hex value for length 12) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x0A | Message number |
| 7, 8 | uint16\_t | 0x0002 | TEC address, hex, network byte order |
| 9, 10 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 11,12 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Get a TEC set point temperature response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x0014 | Packet length (hex value for length 20) |
| 3, 4 | uint16\_t | 0x0000 | Master address in network byte order |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x0B | Message number |
| 7, 8 | uint16\_t | 0x0002 | TEC address, hex, network byte order |
| 9 - 16 | uint8\_t | 8 byte null term string | TEC temperature in null terminated ASCII string, for example client will send something like the following, leading +/- sign supported, two decimal points supported.  - “-23.71”0x00 for -23.71C for the TEC  - or “5”0x00 for +5.00C for the TEC  - or “+10.83” for +10.83C for the TEC |
| 17, 18 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 19, 20 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Set chiller set point temperature command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x0012 | Packet length (hex value for length 18) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x0C | Message number |
| 7 - 14 | uint8\_t | 8 byte null term string | Chiller set point temperature in null terminated ASCII string, for example master will send something like the following, leading +/- sign supported, two decimal points supported.  - “-23.71”0x00 for -23.71C  - or “5”0x00 for +5.00C  - or “+10.83” for +10.83C |
| 15, 16 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 17, 18 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Set chiller set point temperature response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x000C | Packet length (hex value for length 14) |
| 3, 4 | uint16\_t | 0x0000 | Master address in network byte order |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x0D | Message number |
| 7, 8 | uint16\_t | 0x0000  or  0x0001 | Result  0 – fail  1 – success |
| 9, 10 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 11, 12 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Get chiller set point temperature command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x000A | Packet length (hex value for length 12) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x0E | Message number |
| 7, 8 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 9, 10 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Get chiller set point temperature response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x0012 | Packet length (hex value for length 20) |
| 3, 4 | uint16\_t | 0x0000 | Master address in network byte order |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x0F | Message number |
| 7 - 14 | uint8\_t | 8 byte null term string | Chiller set point temperature in null terminated ASCII string, for example client will send something like the following, leading +/- sign supported, two decimal points supported.  - “-23.71”0x00 for -23.71C  - or “5”0x00 for +5.00C  - or “+10.83” for +10.83C |
| 15, 16 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 17, 18 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Enable TECs command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x0A | Packet length (hex value for length 10) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x10 | Message number |
| 7, 8 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 9, 10 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Enable TECs response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x0C | Packet length (hex value for length 16) |
| 3, 4 | uint16\_t | 0x0000 | Master address |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x11 | Message number |
| 7, 8 | uint16\_t | 0x0000  or  0x0001 | Result  - 0x0000 fail  - 0x0001 success |
| 9, 10 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 11, 12 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Disable TECs command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x0A | Packet length (hex value for length 10) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x12 | Message number |
| 7, 8 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 9, 10 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Disable TECs response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x0C | Packet length (hex value for length 16) |
| 3, 4 | uint16\_t | 0x0000 | Master address |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x13 | Message number |
| 7, 8 | uint16\_t | 0x0000  or  0x0001 | Result  - 0x0000 fail  - 0x0001 success |
| 9, 10 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 11, 12 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Start up command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x0A | Packet length (hex value for length 10) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x14 | Message number |
| 7, 8 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 9, 10 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Start up response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x0C | Packet length (hex value for length 16) |
| 3, 4 | uint16\_t | 0x0000 | Master address |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x15 | Message number |
| 7, 8 | uint16\_t | 0x0000  or  0x0001 | Result  - 0x0000 fail  - 0x0001 success |
| 9, 10 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 11, 12 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Shutdown command

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘#’ | Start of packet byte |
| 2 | uint8\_t | 0x0A | Packet length (hex value for length 10) |
| 3, 4 | uint16\_t | 0x0001 | Address in network byte order (assuming sending to slave ID 1) |
| 5 | uint8\_t | 0x01 | Sequence number (master can pick any number) |
| 6 | uint8\_t | 0x16 | Message number |
| 7, 8 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 9, 10 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |

### Shutdown response

|  |  |  |  |
| --- | --- | --- | --- |
| **Byte** | **Data type** | **Value** | **Description** |
| 1 | uint8\_t | ‘!’ | Start of packet byte |
| 2 | uint8\_t | 0x0C | Packet length (hex value for length 16) |
| 3, 4 | uint16\_t | 0x0000 | Master address |
| 5 | uint8\_t | 0x01 | Sequence number (from command) |
| 6 | uint8\_t | 0x17 | Message number |
| 7, 8 | uint16\_t | 0x0000  or  0x0001 | Result  - 0x0000 fail  - 0x0001 success |
| 9, 10 | uint16\_t | TODO: | CRC  - cyclic redundancy code over the packet not including this CRC and the end of packet byte |
| 11, 12 | uint16\_t | 0x000D | End of packet byte  - ‘\r’ (0x000D) |