HIEX

Data Q - DI

DataQ-DI/DO Serial Communication Protocol Manual

Revision 1.0

HIEX

1 - Introduction

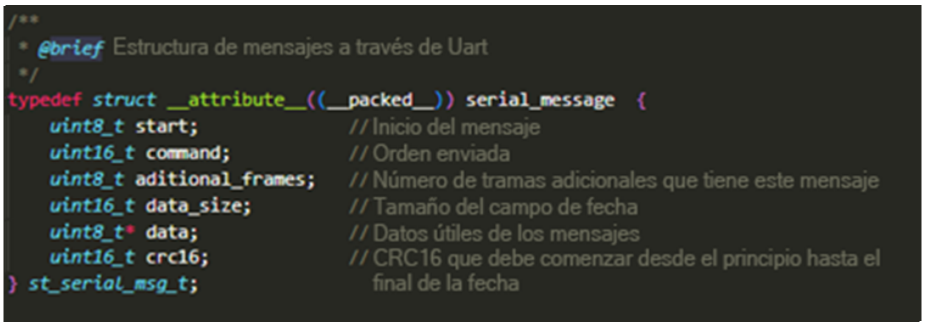
Communication between the configuration software and the DataQ takes place via a serial interface. This interface is available via a USB-C port and has a baudrate of 115200, bit parity disabled, data bit size 8, a stop bit and flow control disabled.

The data transferred over this interface respects a frame of varying size, which contains a Start Byte, two bytes indicating the command sent, a byte indicating whether this message is multipart (this property will be explained later), two bytes of the payload size, the payload (if any, its size being greater than zero) and two bytes of CRC16. This is calculated using the 0xA001 polynomial and has an initial value of 0x0000.

This protocol also has an ACK and NACK message. The NACK is sent if the message received by DataQ does not have a valid CRC16 and the expected CRC value for the message received is sent within the NACK payload. The ACK must be sent by both sides of the communication within a time limit of 500ms. If the ACK is not received by DataQ, the message will be sent again until the ACK is received within the time limit. Likewise, when DataQ receives a message with a valid CRC16, it sends an ACK.

2 - Message frame

As described above, messages travel according to a frame, which is divided into a few parts, each of which will be described in more detail in this topic. Figure 1 represents this frame in the C language:



```c
/**
 * @brief Estructura de mensajes a través de Uart
 */
typedef struct __attribute__((__packed__)) serial_message  {
    uint8_t start;                  //Inicio del mensaje
    uint16_t command;               //Orden enviada
    uint8_t aditional_frames;       //Número de tramas adicionales que tiene este mensaje
    uint16_t data_size;             //Tamaño del campo de fecha
    uint8_t* data;                  //Datos útiles de los mensajes
    uint16_t crc16;                 //CRC16 que debe comenzar desde el principio hasta el
} st_serial_msg_t;                    final de la fecha
```

Figure 1. Message frame

Start Byte: Un único byte que marca el inicio del mensaje, su valor es siempre 0x55.

Command: Field with Two Bytes, indicates the function of the message, it is from this field that the function of the message is defined. As the values are two bytes, the protocol always adopts the insertion of values as MSB First. Commands starting with 0x0XXX are those sent by DataQ, while those starting with 0xFXXX are those sent by the other communication pair. The last page of this document lists all the commands. All messages beginning with 0xFXXX are listed below.

Messages beginning with 0xF0XX are network configuration messages:

- 0xF000: EXTERNAL_MSG_SCAN_NETWORKS: Message responsible for requesting a search for all Wi-Fi networks in DataQ's range. This message does not require a payload. It implies a response message, which will be 0x0000.
- 0xF001: EXTERNAL_MSG_REQUEST_NETWORK_STATE: Message responsible for requesting the state of the network. This message does not require a payload and implies a response message, which will be 0x0001.
- 0xF002: EXTERNAL_MSG_SET_WIFI_CREDENTIALS: Message responsible for setting Wi-Fi credentials. This message requires two pieces of information in the

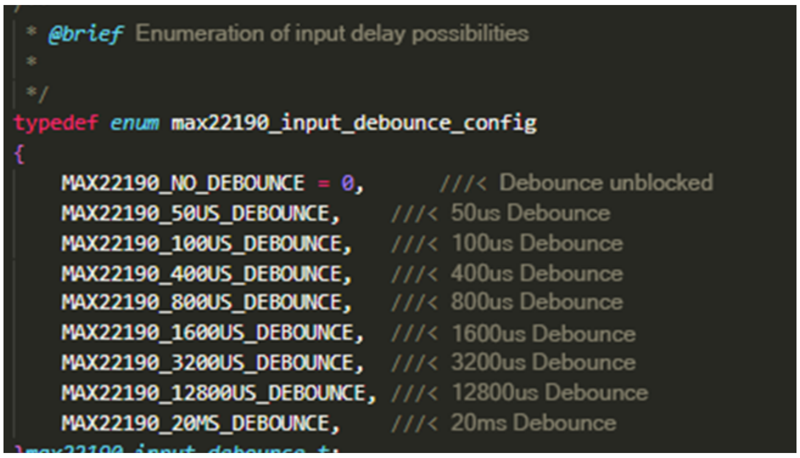
payload, the network SSID and the password. There is no reply message, only the ACK.

- 0xF003: EXTERNAL_MSG_REQUEST_WIFI_CREDENTIALS: Message responsible for requesting the saved network credentials. This message does not require a payload and implies a response message, which will be 0x0002.
- 0xF004: EXTERNAL_MSG_SET_NET_IP: Message responsible for setting the IP information for the network interface. This message requires three pieces of information in the payload: the IP to be set for the device, the IP of the network gateway and the IP of the desired netmask. There is no reply message, only the ACK.
- 0xF005: EXTERNAL_MSG_REQUEST_NET_IP: Message responsible for requesting saved IP addresses. This message does not require a payload and implies a response message, which will be 0x0003.
- 0xF006: EXTERNAL_MSG_REQUEST_MAC_ADDR: Message responsible for requesting the mac address of the interface used. This message does not require a payload and implies a response message, which will be 0x0004.
- 0xF007: EXTERNAL_MSG_SET_NET_INTERFACE: Message responsible for defining which network interface will be used. This message requires information in the payload, '1' (ASCII) to select the Wired interface and '2' (ASCII) to select the Wi-Fi interface. This message restarts the DataQ. It does not imply any response message, only the ACK.
- 0xF008: EXTERNAL_MSG_REQUEST_NET_INTERFACE: Message responsible for requesting which network interface is being used. This message does not require a payload and implies a response message, which will be 0x0005.

The messages beginning with 0xF1XX are the messages related to data collection and only have practical effect on the DataQ DI:

- 0xF100: EXTERNAL_MSG_REQUEST_DATA_COLLECT_INTERVAL, Message responsible for asking DataQ for the collection interval value of the digital pins. It can be sent with an empty payload. This message implies a 0x0100 command response message in addition to the ACK.
- 0xF101 to 0xF108: These messages are EXTERNAL_MSG_REQUEST_DATA_COLLECT_INX_CONFIGS, they are responsible for asking DataQ for the pin configuration values. It can be sent with an empty payload. This message implies a reply message with IDs 0x0101 to 0x0108, as well as the ACK.

- 0xF109 to 0xF110: EXTERNAL MSG REQUEST DATA COLLECT INX STATE: These are messages requesting the states of digital pins 1 to 8. Can be sent with an empty payload. It generates a reply message, which will contain the requested information, as well as the ACK. This reply message has the command 0x0109 to 0x0110, depending on the message sent.
- 0xF111:EXTERNAL_MSG_REQUEST_EXTERN_DATA_VIA_SERIAL_CONFIG: Message requesting data externalization settings via serial interface. Generates a response message with the command 0x0111.
- 0xF112:EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_INTERVAL: Message responsible for defining the interval between collections, in milliseconds. The payload of this message must contain one piece of information, which is the time between collections, expressed in ASCII characters. This message does not imply a reply message, only the ACK.
- 0xF113 to 0xF11A: Configuration messages for the digital pins to be collected, EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_INX. This message must contain 4 pieces of information in the payload, the first is the pin's enable, 1 or 0 to enable or disable the pin; the second is the pin's WireBreak enable, 1 or 0 to enable or disable; the third is the pin's debounce value, with the following values:



```
 * @brief Enumeration of input delay possibilities
 *
 */
typedef enum max22190_input_debounce_config
{
    MAX22190_NO_DEBOUNCE = 0,        ///< Debounce unblocked
    MAX22190_50US_DEBOUNCE,      ///< 50us Debounce
    MAX22190_100US_DEBOUNCE,     ///< 100us Debounce
    MAX22190_400US_DEBOUNCE,     ///< 400us Debounce
    MAX22190_800US_DEBOUNCE,     ///< 800us Debounce
    MAX22190_1600US_DEBOUNCE,    ///< 1600us Debounce
    MAX22190_3200US_DEBOUNCE,    ///< 3200us Debounce
    MAX22190_12800US_DEBOUNCE,   ///< 12800us Debounce
    MAX22190_20MS_DEBOUNCE,      ///< 20ms Debounce
}max22190_input_debounce_t;
```

And finally, the fourth piece of information is the definition of the pin as input or output, with 1 being the definition of an input pin and 2 for an output pin. All information must be sent in ASCII characters. Only one ACK is generated in response.

- 0xF11B: EXTERNAL_MSG_CONFIGURE_EXTERN_DATA_VIA_SERIAL: Message defining data externalization via serial interface. The payload must contain 4 pieces of

information, the first being the export enable, 1 to enable and 0 to disable. The second is the export delay in seconds, followed by the pin value export enable, either 1 or 0. Finally, the third piece of information is the error export enable. All information must be sent in ASCII code. The only message answered for this message is the ACK.
Messages beginning with 0xF2XX are messages related to the Web server:

● 0xF200: EXTERNAL_MSG_SEND_NEW_CA_FILE: Message used to send a new CA certificate to the WebServer. The payload must contain the CA certificate.

● 0xF201: EXTERNAL_MSG_SEND_NEW_CERT_FILE: Message responsible for sending the WebServer's new certificate. The payload must contain the certificate's new public key.

● 0xF202: EXTERNAL_MSG_SEND_NEW_KEY_FILE: Message responsible for sending the certificate's new private key.

Messages beginning with 0xF3XX are messages related to DataQ specifications:

● 0xF300: EXTERNAL_MSG_REQUEST_MODEL: Message responsible for requesting the Device Model. The payload can be empty. The return will be DI or DO, via message 0x0300.

● 0xF301: EXTERNAL_MSG_REQUEST_HW_VERSION: Hardware version request. This message may have an empty payload. Generates a command response message 0x0301.

● 0xF302: EXTERNAL_MSG_REQUEST_SW_VERSION: Software version request. This message may have an empty payload. Generates a response message with command 0x0302;

● 0xF303: EXTERNAL_MSG_REQUEST_SN: Message requesting the serial number of the device.

● 0xF304: EXTERNAL_MSG_REBOOT: Message requesting a reboot of the device.

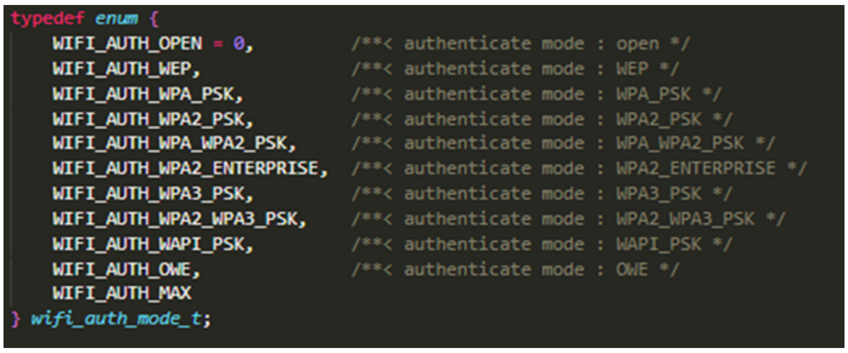
● 0xF305: EXTERNAL_MSG_FACTORY_RESET: Message that performs a factory reset of all configured parameters;

These are all messages that are interpreted by DataQ when sent by the external peer.
There are two messages that can be sent by both sides of the communication, which are ACK and NACK, their command codes are 0xFFFF and 0xFFFE, respectively.
There are also messages that are only sent by the DataQ to the communication pair. Almost all of these messages, which are sent only by the DataQ, are replies to information request messages. It is worth noting that the communication peer must send the ACK for each

of the messages sent by the DataQ, otherwise it will keep sending the same message over and over again.

The messages will be listed below, starting with the Network-related messages, which start at 0x00XX:

- 0x0000: DATAQ_MSG_SCAN_NETWORKS_RESULT: Message sent in response to command 0xF000. Its payload contains information about all the wireless networks in DataQ's range. The information is sent in the following order: SSID, Network security and Signal strength. If there are more networks in range, the information for the other networks follows the same order, sent sequentially. The network security values are numbers, in ASCII representation and the translation of the numbers is as follows:



```
typedef enum {
    WIFI_AUTH_OPEN = 0,         /**< authenticate mode : open */
    WIFI_AUTH_WEP,              /**< authenticate mode : WEP */
    WIFI_AUTH_WPA_PSK,          /**< authenticate mode : WPA_PSK */
    WIFI_AUTH_WPA2_PSK,         /**< authenticate mode : WPA2_PSK */
    WIFI_AUTH_WPA_WPA2_PSK,     /**< authenticate mode : WPA_WPA2_PSK */
    WIFI_AUTH_WPA2_ENTERPRISE,  /**< authenticate mode : WPA2_ENTERPRISE */
    WIFI_AUTH_WPA3_PSK,         /**< authenticate mode : WPA3_PSK */
    WIFI_AUTH_WPA2_WPA3_PSK,    /**< authenticate mode : WPA2_WPA3_PSK */
    WIFI_AUTH_WAPI_PSK,         /**< authenticate mode : WAPI_PSK */
    WIFI_AUTH_OWE,              /**< authenticate mode : OWE */
    WIFI_AUTH_MAX
} wifi_auth_mode_t;
```

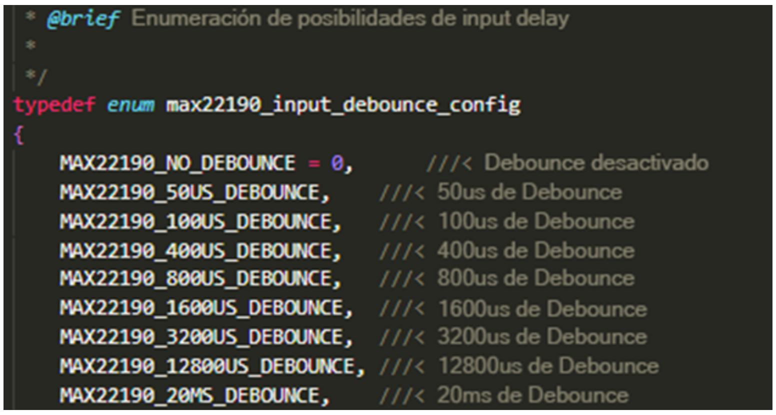
- 0x0001: DATAQ_MSG_RESPONSE_NET_STATE: Response message for request 0xF001. Its payload contains 2 pieces of information, the first is 1 or 0, indicating a connection or not. If there is a connection, the second information is 1 or 0 to indicate the presence of an IP address on the interface. If there is no connection, an error code is sent in the second piece of information.
- 0x0002: DATAQ_MSG_RESPONSE_WIFI_CREDENTIALS: Response message for the Wi-Fi credential request. Two pieces of information are sent in the payload, the first is the SSID and the second is the password.
- 0x0003: DATAQ_MSG_RESPONSE_NET_IP: Response message for the network interface IP request. This message has 3 pieces of information in the payload: the first is the device IP, the second is the gateway IP and the last is the adopted netmask.

- 0x0004: DATAQ_MSG_RESPONSE_MAC_ADDR: Message that is a response to the MAC Address request message. This message has only one piece of information in the payload, which is the MAC Address.
- 0x0005: DATAQ_MSG_RESPONSE_INTERFACE: Request response message for the network interface being used. It has only one piece of information in its payload, 1 for the ETH interface and 2 for the Wi-Fi interface. The content is always ASCII.

The next messages are related to data collection, so their use is only valid when a DataQ DI is involved. These messages start with byte 0x01.

- 0x0100: DATAQ_MSG_RESPONSE_DATA_COLLECT_INTERVAL: Collection time request response message. This message has only one piece of information in the payload, which is the collection time interval in milliseconds.
- 0x0101 to 0x0108: DATAQ_MSG_RESPONSE_DATA_COLLECT_INX_CONFI-GS: Configuration response message for each of the information collection pins. There are 4 pieces of information in the payload of this message, written in the following order: Pin Enable, 1 or 0 ASCII; WireBreak Enable, 1 or 0 ASCII; Debounce, has ASCII values, as listed below; Input Output, 1 for Input pin, 2 for Output pin, both ASCII.

a a continuación; Input Output, 1 para el pin Input, 2 para el pin Output, ambos ASCII.



```
 * @brief Enumeración de posibilidades de input delay
 *
 */
typedef enum max22190_input_debounce_config
{
    MAX22190_NO_DEBOUNCE = 0,         ///< Debounce desactivado
    MAX22190_50US_DEBOUNCE,       ///< 50us de Debounce
    MAX22190_100US_DEBOUNCE,      ///< 100us de Debounce
    MAX22190_400US_DEBOUNCE,      ///< 400us de Debounce
    MAX22190_800US_DEBOUNCE,      ///< 800us de Debounce
    MAX22190_1600US_DEBOUNCE,     ///< 1600us de Debounce
    MAX22190_3200US_DEBOUNCE,     ///< 3200us de Debounce
    MAX22190_12800US_DEBOUNCE,    ///< 12800us de Debounce
    MAX22190_20MS_DEBOUNCE,       ///< 20ms de Debounce
```

- 0x0109 to 0x0110: DATAQ_MSG_RESPONSE_DATA_COLLECT_IN1_STATE: Response message for the pin status request. The payload of this message contains two pieces of information. The pin state, either high or low, 1 or 0 ASCII; and the WireBreak state of that pin, 1 or 0 ASCII.
- 0x0111: DATAQ_MSG_RESPONSE_EXTERN_DATA_VIA_SERIAL_CONFIG: Response message to the request for data externalization settings via serial interface. In this message there are 4 pieces of information in the payload written in the following

- 0x0301: DATAQ_MSG_RESPONSE_HW_VERSION: Hardware version request response message. It has only one piece of information in the payload, which is the hardware version.
- 0x0302: DATAQ_MSG_RESPONSE_SW_VERSION: Software version request response message. It only has one piece of information in the payload, which is the software version.
- 0x0303: DATAQ_MSG_RESPONSE_SN: Response message for the product serial number request. It has only one piece of information in the payload, which is the serial number.

This left only ACK and NACK.

- 0xFFFF: ACK_COMMAND: ACK message, sent for all commands, from both parties. There is no information in the payload.
- 0xFFFE: NACK_COMMAND: NACK message, sent for messages received with an invalid CRC16. It has an information in the payload, which is the expected CRC16 for the received message.

So all the commands have been mentioned and explained, the next section will explain the addtional frames field.

Additional frames: In most cases, this field will have a value of 0. However, for messa-ges with very large payloads (exceeding 0xFFFF bytes in length), this field defines how many additional frames are needed to send the message.

Data size: Two bytes representing the size of the payload, which can be 0 or a maximum of 0xFFFF. Two-byte values are always inserted in the message as MSB First.

Payload: The payload field contains all the useful information in the message. If the message does not require any information to be sent, this field is non-existent. However, if there is information to be sent, it is organized according to the following structure:

For each piece of information to be written in the payload, the size of the information must be written first, in non-ASCII characters, followed by the information.

Below is a payload of the message sending the Wi-FI credentials:
Payload: 0x0B"Omega7Guest"0x0F"omega7guest1234"

The quotation marks do not exist in the message, they are only used to indicate ASCII characters.

For this message, the Data Size is 0x001C, i.e. 28.
Finally, the last remaining field is CRC16, all the bytes of the message go into the CRC16 account. An ACK message will be inserted below to check its CRC16. The CRC16 is inserted into the message as MSB First.
ACK message complete:

0xAA, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x3C, 0x0A
The first byte is the start byte;
The next two are command bytes;
The following is the additional frames byte
The next two are the size of the message;

And the last two the CRC16, since there is no payload, if there were, it would be present before the CRC16 bytes.

```
/**
 * @brief Enumeration with command description
 *
 */
typedef enum commands
{
   /*************** ESP MSG (SEND COMMANDS)*/
      /*********** Data send Msg 0x(0Fxx)*/
      DATAQ_MSG_EXTERN_DATA = 0x0F00,


      /*********** NetWork Related Msg 0x(00xx)*/
      DATAQ_MSG_SCAN_NETWORKS_RESULT = 0x0000,
      DATAQ_MSG_RESPONSE_WIFI_STATE,
      DATAQ_MSG_RESPONSE_WIFI_CREDENCIALS,
      DATAQ_MSG_RESPONSE_WIFI_IP,
      DATAQ_MSG_RESPONSE_MAC_ADDR,
      DATAQ_MSG_RESPONSE_INTERFACE,
      /*********** Pin Related Msg 0x(01xx)*/
      DATAQ_MSG_RESPONSE_DATA_COLLECT_INTERVAL = 0x0100,


      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN1_CONFIGS,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN2_CONFIGS,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN3_CONFIGS,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN4_CONFIGS,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN5_CONFIGS,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN6_CONFIGS,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN7_CONFIGS,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN8_CONFIGS,

      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN1_STATE,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN2_STATE,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN3_STATE,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN4_STATE,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN5_STATE,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN6_STATE,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN7_STATE,
      DATAQ_MSG_RESPONSE_DATA_COLLECT_IN8_STATE,


      DATAQ_MSG_RESPONSE_EXTERN_DATA_VIA_SERIAL_CONFIG,


      DATAQ_MSG_RESPONSE_MODEL = 0x0300,
      DATAQ_MSG_RESPONSE_HW_VERSION,
      DATAQ_MSG_RESPONSE_SW_VERSION,
      DATAQ_MSG_RESPONSE_SN,
   /**************************************************/


   /*************** External MSG (RECEIVE COMMANDS)*/
      /*********** NetWork Related Msg 0x(F0xx)*/
      EXTERNAL_MSG_SCAN_NETWORKS= 0xF000,
      EXTERNAL_MSG_REQUEST_WIFI_STATE ,
      EXTERNAL_MSG_SET_WIFI_CREDENCIALS,
      EXTERNAL_MSG_REQUEST_WIFI_CREDENCIALS,
      EXTERNAL_MSG_SET_WIFI_IP,
      EXTERNAL_MSG_REQUEST_WIFI_IP,
      EXTERNAL_MSG_REQUEST_MAC_ADDR,
      EXTERNAL_MSG_SET_NETWORK_INTERFACE,
      EXTERNAL_MSG_REQUEST_NETWORK_INTERFACE,

      /*********** Data Related Msg 0x(F1xx)*/
      EXTERNAL_MSG_REQUEST_DATA_COLLECT_INTERVAL = 0xF100,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN1_CONFIGS,  /**0xF101*/
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN2_CONFIGS,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN3_CONFIGS,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN4_CONFIGS,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN5_CONFIGS,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN6_CONFIGS,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN7_CONFIGS,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN8_CONFIGS,

        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN1_STATE /**= 0xF109*/,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN2_STATE,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN3_STATE,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN4_STATE,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN5_STATE,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN6_STATE,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN7_STATE,
        EXTERNAL_MSG_REQUEST_DATA_COLLECT_IN8_STATE,


        EXTERNAL_MSG_REQUEST_EXTERN_DATA_VIA_SERIAL_CONFIG  /**=
0xF111*/,


        EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_INTERVAL /**= 0xF112*/,


        EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_IN1 /**= 0xF113*/,
        EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_IN2,
        EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_IN3,
        EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_IN4,
        EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_IN5,
        EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_IN6,
        EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_IN7,
        EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_IN8,

        EXTERNAL_MSG_CONFIGURE_EXTERN_DATA_VIA_SERIAL /**= 0xF11B*/,


        /*********** Web_server Related Msg 0x(F2xx)*/
        EXTERNAL_MSG_SEND_NEW_CA_FILE = 0xF200,
        EXTERNAL_MSG_SEND_NEW_CERT_FILE,
        EXTERNAL_MSG_SEND_NEW_KEY_FILE,


        /*********** dataQ config Related Msg 0x(F3xx)*/
        EXTERNAL_MSG_REQUEST_MODEL = 0xF300,
        EXTERNAL_MSG_REQUEST_HW_VERSION,
        EXTERNAL_MSG_REQUEST_SW_VERSION,
        EXTERNAL_MSG_REQUEST_SN,
        EXTERNAL_MSG_REBOOT,
        EXTERNAL_MSG_FACTORY_RESET,


    /**Commom comands*/
    NACK_COMMAND = 0xFFFE,
    ACK_COMMAND = 0xFFFF
}serial_commands_t;
```