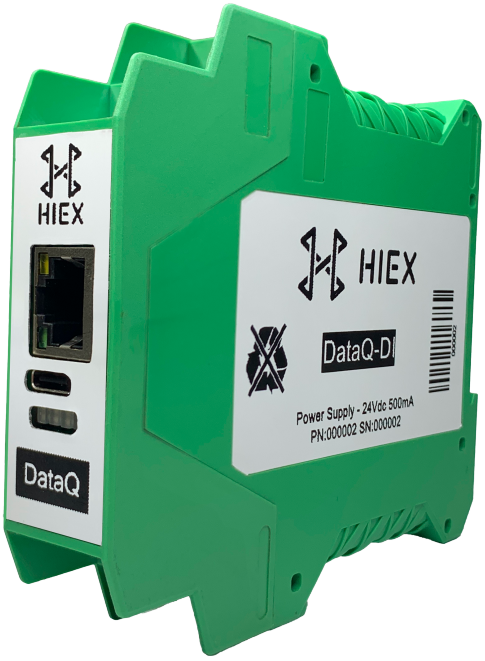
HIEX

Data Q - DI

Manual del protocolo de comunicación Serial DataQ-DI/DO



Revisión 1.0

1 - Introducción

La comunicación entre el software de configuración y el DataQ se realiza a través de una interfaz serie. Esta interfaz se proporciona a través de un puerto USB-C y tiene una velocidad en baudios de 115200, paridad de bits desactivada, tamaño de bits de datos 8, un bit de parada y control de flujo desactivado.

Los datos que viajan por esta interfaz respetan una trama de tamaño variable, que contiene un Start Byte, dos bytes que indican el comando enviado, un byte que indica si este mensaje es multiparte (esta propiedad se explicará más adelante), dos bytes del tamaño de payload, el payload (si la hay, siendo su tamaño mayor que cero) y dos bytes de CRC16. Éste se calcula utilizando el polinomio 0xA001 y tiene un valor inicial de 0x0000.

Este protocolo también tiene un mensaje ACK y NACK. El NACK se envía si el mensaje recibido por DataQ no tiene un CRC16 válido y el valor CRC esperado para el mensaje recibido se envía dentro del payload del NACK. El ACK debe ser enviado por ambos lados de la comunicación en un tiempo límite de 500ms. Si DataQ no recibe el ACK, el mensaje se enviará de nuevo hasta que se reciba el ACK dentro del tiempo límite. Del mismo modo, cuando DataQ recibe un mensaje con un CRC16 válido, envía un ACK.

2 - Frame de mensajes

Como se ha descrito anteriormente, los mensajes viajan según una trama, que se divide en varias partes, cada una de las cuales se describirán con más detalle en este tema. La figura 1 representa esta trama en el lenguaje C:

```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
```

Figura 1. Frame de mensajes

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

Comando: Este campo de dos bytes sirve para definir la función del mensaje. Al tratarse de dos bytes, el protocolo adopta siempre la inserción de valores como MSB First. Los comandos que empiezan por 0x0XXX son los enviados por DataQ, mientras que los que empiezan por 0xFXXX son los enviados por el otro par de comunicación. En la última página de este documento se enumeran todos los comandos. A continuación se enumeran todos los mensajes que empiezan por 0xFXXX.

Los mensajes que empiezan por 0xF0XX son mensajes de configuración de red:

- 0xF000: EXTERNAL_MSG_SCAN_NETWORKS: Mensaje responsable de solicitar una búsqueda de todas las redes Wi-Fi en el rango de DataQ. Este mensaje no requiere carga útil. Implica un mensaje de respuesta, que será 0x0000.
- 0xF001: EXTERNAL_MSG_REQUEST_NETWORK_STATE: Mensaje encargado de solicitar el estado de la red. Este mensaje no requiere payload e implica un mensaje de respuesta, que será 0x0001.
- 0xF002: EXTERNAL_MSG_SET_WIFI_CREDENTIALS: Mensaje responsable de definir las credenciales Wi-Fi. Este mensaje requiere dos datos en payload, el SSID de la red y la password. No hay mensaje de respuesta, sólo el ACK.

- 0xF003: EXTERNAL_MSG_REQUEST_WIFI_CREDENTIALS: Mensaje encargado de solicitar las credenciales de red guardadas. Este mensaje no requiere payload e implica un mensaje de respuesta, que será 0x0002..
- 0xF004: EXTERNAL_MSG_SET_NET_IP: Mensaje responsable de establecer la información IP para la interfaz de red. Este mensaje requiere tres datos en la payload: la IP a configurar para el dispositivo, la IP de la pasarela de red y la IP de la netmask deseada. No hay mensaje de respuesta, sólo el ACK.
- 0xF005: EXTERNAL_MSG_REQUEST_NET_IP: Mensaje encargado de solicitar direcciones IP guardadas. Este mensaje no requiere payload e implica un mensaje de respuesta, que será 0x0003.
- 0xF006: EXTERNAL_MSG_REQUEST_MAC_ADDR: Mensaje encargado de solicitar la dirección mac de la interfaz utilizada. Este mensaje no requiere payload e implica un mensaje de respuesta, que será 0x0004.
- 0xF007: EXTERNAL_MSG_SET_NET_INTERFACE: Mensaje encargado de definir qué interfaz de red se utilizará. Este mensaje requiere información en el payload, '1' (ASCII) para seleccionar la interfaz Cableada y '2' (ASCII) para seleccionar la interfaz Wi-Fi. Este mensaje reinicia el DataQ. No implica ningún mensaje de respuesta, sólo el ACK.
- 0xF008: EXTERNAL_MSG_REQUEST_NET_INTERFACE: Mensaje encargado de solicitar qué interfaz de red se está utilizando. Este mensaje no requiere payload e implica un mensaje de respuesta, que será 0x0005..

Los mensajes que empiezan por 0xF1XX son los relacionados con la recogida de datos y sólo tienen efecto práctico en el DataQ DI:

- 0xF100: EXTERNAL_MSG_REQUEST_DATA_COLLECT_INTERVAL, Mensaje encargado de solicitar a DataQ el valor del intervalo de recogida de los pines digitales. Puede enviarse con una payload vacía. Este mensaje implica un mensaje de respuesta de comando 0x0100 además del ACK.
- 0xF101 a 0xF108: Estos mensajes son EXTERNAL_MSG_REQUEST_DATA_-COLLECT_INX_CONFIGS, se encargan de solicitar a DataQ los valores de configuración de los pines. Se puede enviar con un payload vacío. Este mensaje implica un mensaje de respuesta con IDs 0x0101 a 0x0108, así como el ACK.
- 0xF109 a 0xF110: EXTERNAL MSG REQUEST DATA COLLECT INX STATE: Son mensajes que solicitan los estados de los pines digitales 1 a 8. Pueden enviarse con la carga útil vacía. Puede enviarse con un payload vacío. Genera un mensaje de

respuesta, que contendrá la información solicitada así como el ACK. Este mensaje de respuesta tiene el comando 0x0109 a 0x0110, dependiendo del mensaje enviado.

- 0xF111:EXTERNAL_MSG_REQUEST_EXTERN_DATA_VIA_SERIAL_CONFIG: Mensaje de solicitud de ajustes de externalización de datos a través de la interfaz serie. Genera un mensaje de respuesta con el comando 0x0111.
- 0xF112:EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_INTERVAL: Mensaje encargado de definir el intervalo entre recogidas, en milisegundos. El payload de este mensaje debe contener una información, que es el tiempo entre recolecciones, expresado en caracteres ASCII. Este mensaje no implica un mensaje de respuesta, sólo el ACK.
- 0xF113 a 0xF11A: Mensajes de configuración de los pines digitales a recoger, EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_INX. Este mensaje debe contener 4 piezas de información en la carga útil, la primera es la enable del pin, 1 o 0 para habilitar o deshabilitar el pin; la segunda es la enable WireBreak del pin, 1 o 0 para habilitar o deshabilitar; la tercera es el valor debounce del pin, con los siguientes valores:

```
 * @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
```

Y por último, el cuarto dato es la definición del pin como entrada o salida, siendo 1 la definición de pin de entrada y 2 para pin de salida. Toda la información debe enviarse en caracteres ASCII. Sólo se genera un ACK como respuesta.

- 0xF11B: EXTERNAL_MSG_CONFIGURE_EXTERN_DATA_VIA_SERIAL: Mensaje que define la externalización de datos a través de la interfaz serie. El payload debe contener 4 datos, el primero es el enable de exportación, 1 para habilitar y 0 para deshabilitar. El segundo es el delay de exportación en segundos, seguido de la

habilitación de exportación del valor del pin, 1 o 0. Por último, el tercer dato es el enable de exportación de errores. Toda la información debe enviarse en código ASCII. El único mensaje que se responde para este mensaje es el ACK.

Los mensajes que empiezan por 0xF2XX son mensajes relacionados con el servidor Web:

- 0xF200: EXTERNAL_MSG_SEND_NEW_CA_FILE: Mensaje utilizado para enviar un nuevo certificado CA al WebServer. El payload debe contener el certificado CA.
- 0xF201: EXTERNAL_MSG_SEND_NEW_CERT_FILE: Mensaje responsable del envío del nuevo certificado del WebServer. El payload debe contener la clave pública del nuevo certificado.
- 0xF202: EXTERNAL_MSG_SEND_NEW_KEY_FILE: Mensaje responsable del envío de la nueva clave privada del certificado.

Los mensajes que empiezan por 0xF3XX son mensajes relacionados con las especificaciones DataQ:

- 0xF300: EXTERNAL_MSG_REQUEST_MODEL: Mensaje encargado de solicitar el Modelo de Dispositivo. El payload puede estar vacío. El retorno será DI o DO, a través del mensaje 0x0300.
- 0xF301: EXTERNAL_MSG_REQUEST_HW_VERSION: Solicitud de versión de hardware. Este mensaje puede tener un payload vacío. Genera un mensaje de respuesta de comando 0x0301.
- 0xF302: EXTERNAL_MSG_REQUEST_SW_VERSION: Petición de versión de software. Este mensaje puede tener un payload vacío. Genera un mensaje de respuesta con el comando 0x0302;
- 0xF303: EXTERNAL_MSG_REQUEST_SN: Mensaje solicitando el número de serie del dispositivo.
- 0xF304: EXTERNAL_MSG_REBOOT: Mensaje solicitando el reinicio del dispositivo.
- 0xF305: EXTERNAL_MSG_FACTORY_RESET: Mensaje que realiza un reset de fábrica de todos los parámetros configurados;

Estos son todos los mensajes que son interpretados por DataQ cuando son enviados por el peer externo. Hay dos mensajes que pueden ser enviados por ambos lados de la comunicación, que son ACK y NACK, sus códigos de comando son 0xFFFF y 0xFFFE, respectivamente.

También hay mensajes que sólo son enviados por el DataQ al par de comunicación. Casi todos estos mensajes enviados únicamente por el DataQ son respuestas a mensajes de solicitud de información. Vale la pena señalar que el par de comunicación debe enviar el ACK para cada uno de los mensajes enviados por el DataQ, de lo contrario seguirá enviando el mismo mensaje una y otra vez.
Los mensajes se enumeran a continuación, comenzando por los mensajes relacionados con la red, que empiezan por 0x00XX:

- 0x0000: DATAQ_MSG_SCAN_NETWORKS_RESULT: Mensaje enviado en respuesta al comando 0xF000. Su payload contiene información sobre todas las redes inalámbricas en el rango de DataQ. La información se envía en el siguiente orden: SSID, Seguridad de la red e Intensidad de la señal. Si hay más redes dentro del alcance, la información de las otras redes sigue el mismo orden, enviada secuencialmente. Los valores de seguridad de la red son números, en representación ASCII y la traducción de los números es la siguiente:

```
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: Mensaje de respuesta para la petición 0xF001. Su payload contiene 2 piezas de información, la primera es 1 o 0, indicando si hay conexión o no. Si hay conexión, la segunda información es 1 o 0 para indicar la presencia de una dirección IP en la interfaz. Si no hay conexión, se envía un código de error en la segunda información.
- 0x0002: DATAQ_MSG_RESPONSE_WIFI_CREDENTIALS: Mensaje de respuesta para la solicitud de credenciales Wi-Fi. Se envían dos piezas de información en el payload, la primera es el SSID y la segunda es la contraseña.

- 0x0003: DATAQ_MSG_RESPONSE_NET_IP: Mensaje de respuesta para la solicitud de IP de interfaz de red. Este mensaje tiene 3 piezas de información en el payload: la primera es la IP del dispositivo, la segunda es la IP de la puerta de enlace y la última es la máscara de red adoptada.
- 0x0004: DATAQ_MSG_RESPONSE_MAC_ADDR: Mensaje de respuesta al mensaje de solicitud de dirección MAC. Este mensaje sólo contiene una información en el payload, que es la dirección MAC.
- 0x0005: DATAQ_MSG_RESPONSE_INTERFACE: Mensaje de solicitud de respuesta para la interfaz de red que se está utilizando. Sólo tiene una información en su payload, 1 para la interfaz ETH y 2 para la interfaz Wi-Fi. El contenido es siempre ASCII. Los siguientes mensajes están relacionados con la recogida de datos, por lo que su uso sólo es válido cuando se trata de un DataQ DI. Estos mensajes comienzan con el byte 0x01.
- 0x0100: DATAQ_MSG_RESPONSE_DATA_COLLECT_INTERVAL: Mensaje de respuesta a la solicitud de tiempo de recogida. Este mensaje sólo contiene una información en el payload, que es el intervalo de tiempo de recogida en milisegundos.
- 0x0101 a 0x0108: DATAQ_MSG_RESPONSE_DATA_COLLECT_INX_CONFIGS: Mensaje de respuesta de configuración para cada uno de los pines de recogida de información. Hay 4 piezas de información en el payload de este mensaje, escritas en el siguiente orden: Pin Enable, 1 o 0 ASCII; WireBreak Enable, 1 o 0 ASCII; Debounce, tiene valores ASCII, como se indica 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 a 0x0110: DATAQ_MSG_RESPONSE_DATA_COLLECT_IN1_STATE: Mensaje de respuesta para la solicitud de estado de pin. El payload de este mensaje contiene dos piezas de información. El estado del pin, alto o bajo, 1 o 0 ASCII; y el estado WireBreak de ese pin, 1 o 0 ASCII.
- 0x0111: DATAQ_MSG_RESPONSE_EXTERN_DATA_VIA_SERIAL_CONFIG: Mensaje de respuesta a la solicitud de configuración de externalización de datos a través de la interfaz serie. En este mensaje hay 4 piezas de información en el payload escritas en el siguiente orden: Habilitar exportación vía serie, 1 ó 0 ASCII; Retardo exportación en segundos ASCII; Habilitar exportación de valores de pines, 1 ó 0 ASCII; Habilitar exportación de bits de error, 1 ó 0 ASCII.

Los siguientes mensajes están relacionados con solicitudes de información sobre el DataQ.

- 0x0300: DATAQ_MSG_RESPONSE_MODEL: Mensaje de respuesta del modelo DataQ. Sólo tiene una información en el payload. Se trata de DI o DO.
- 0x0301: DATAQ_MSG_RESPONSE_HW_VERSION: Mensaje de respuesta de solicitud de versión de hardware. Sólo tiene una información en el payload, que es la versión de hardware.
- 0x0302: DATAQ_MSG_RESPONSE_SW_VERSION: Mensaje de respuesta de solicitud de versión de software. Sólo tiene un dato en el payload, que es la versión de software.
- 0x0303: DATAQ_MSG_RESPONSE_SN: Mensaje de respuesta para la solicitud del número de serie del producto. Sólo tiene un dato en el payload, que es el número de serie.

Sólo quedaban ACK y NACK.

- 0xFFFF: ACK_COMMAND: Mensaje ACK, enviado para todos los comandos, de ambas partes. No tiene información en el payload.
- 0xFFFE: NACK_COMMAND: Mensaje NACK, enviado para mensajes recibidos con un CRC16 inválido. Tiene información en el payload, que es el CRC16 esperado para el mensaje recibido.

Así que todos los comandos se han mencionado y explicado, la siguiente sección explicará el campo de marcos addtional.

Additional frames: En la mayoría de los casos, este campo tendrá un valor de 0. Sin embargo, para mensajes con payloads muy grandes (que superen los 0xFFFF bytes de

longitud), este campo define cuántas tramas adicionales son necesarias para enviar el mensaje.

Data size: Dos bytes que representan el tamaño del payload, que puede ser 0 o un máximo de 0xFFFF. Los valores de dos bytes siempre se insertan en el mensaje como MSB First.

Payload: El campo payload contiene toda la información útil del mensaje. Si el mensaje no requiere el envío de ninguna información, este campo es inexistente. Sin embargo, si hay información que enviar, se organiza según la siguiente estructura:

Para cada información que se escriba en el payload, debe escribirse primero el tamaño de la información, en caracteres no ASCII, seguido de la información.

A continuación se muestra un payload del mensaje que envía las credenciales Wi-FI:

Payload: 0x0B”Omega7Guest”0x0F”omega7guest1234”

Las comillas no existen en el mensaje, sólo se utilizan para indicar caracteres ASCII.

Para este mensaje, el tamaño de los datos es 0x001C, es decir, 28.

Finalmente, el último campo restante es CRC16, todos los bytes del mensaje van a la cuenta CRC16. A continuación se insertará un mensaje ACK para comprobar su CRC16. El CRC16 se inserta en el mensaje como MSB First.
Mensaje ACK completo:

0xAA, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x3C, 0x0A
El primer byte es el start byte;
Los dos siguientes son bytes de comando;
A continuación se muestra el byte de additional frames
Los dos siguientes son el tamaño del mensaje;

Y los dos últimos el CRC16, ya que no hay payload, si lo hubiera estaría presente antes de los bytes CRC16.

```
/**
 * @brief Enumeración con descripción del comando
 *
 */
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;
```