HIEX

# Data Q - DI

# Manual do Protocolo de Comunicação Serial DataQ-DI/DO


Revisão 1.0

# 1 - Introdução

A comunicação entre o software de configuração e o DataQ acontece por meio de uma interface serial. Essa interface é disponibilizada através de uma porta USB-C e possui um baudrate de 115200, paridade de bits desabilitada, data bit de tamanho 8, um stop bit e flow control desabilitado. Os dados trafegados sobre essa interface respeitam um frame de tamanho variado, o qual contém um Start Byte, dois bytes que indicam o comando enviado, um byte indicando se essa mensagem é multiparte (essa propriedade será explicada mais adiante), dois bytes de tamanho do payload, o payload (caso haja, sendo seu tamanho maior que zero) e dois bytes de CRC16. Este calculado usando o polinômio 0xA001 e com valor inicial 0x0000.
Além disso, esse protocolo conta com uma mensagem de ACK e NACK. O NACK é enviado caso a mensagem recebida pelo DataQ não possua um CRC16 válido e o valor de CRC esperado, para a mensagem recebida, é enviado dentro do payload desse NACK. Já o ACK deve ser enviado por ambos os lados da comunicação, em um tempo limite de 500ms. Caso o ACK não for recebido pelo DataQ a mensagem será enviada novamente, até receber o ACK dentro do tempo limite. Da mesma forma, ao receber uma mensagem com o CRC16 válido, um ACK é enviado por parte do DataQ.

## 2 - Frame de mensagem

Como descrito anteriormente, as mensagens trafegam de acordo com um frame, o qual é dividido em algumas partes, cada uma delas será descrita com mais detalhes nesse tópico. A Figura 1, representa esse frame na linguagem C:

```c
/**
 * @brief Estrutura de envio de mensagens via Uart
 */
typedef struct __attribute__((__packed__)) serial_message  {
    uint8_t start;              //Inicio da mensagem
    uint16_t command;           //Comando sendo enviado
    uint8_t aditional_frames;   //Numero de frames extras que essa mensagem possui
    uint16_t data_size;         //Tamanho do campo data
    uint8_t* data;              //Dados uteis da mensagem
    uint16_t crc16;             //CRC16 que deve comecar do start ate o fim de data
} st_serial_msg_t;
```

Figura 1. Frame da mensagem

**Start Byte:** Um único byte que marca o início da mensagem, o seu valor é sempre 0x55.

**Comando:** Campo que possui Dois Bytes, indica qual a função dessa mensagem, é a partir desse campo que a função da mensagem é definida. Como os valores são de dois bytes, o protocolo adota sempre a inserção dos valores como sendo MSB First. Os comandos que iniciam com 0x0XXX são os enviados pelo DataQ, já os iniciados com 0xFXXX são os comandos enviados pelo outro par da comunicação. Na última página desse documento existe uma enumeração com todos os comandos. Abaixo serão listadas todas as mensagens iniciadas por 0xFXXX.

As mensagens iniciadas por 0xF0XX são as mensagens de configuração de rede, sendo:

- 0xF000: EXTERNAL_MSG_SCAN_NETWORKS: Mensagem responsável por solicitar uma busca por todas as redes Wi-Fi no alcance do DataQ. Essa mensagem não necessita de nenhum payload. Ela implica em uma mensagem de resposta, que será a 0x0000.
- 0xF001: EXTERNAL_MSG_REQUEST_NETWORK_STATE: Mensagem responsável por solicitar o estado da rede. Essa mensagem não demanda de payload e implica em uma mensagem de resposta, que será a 0x0001.

- 0xF002: EXTERNAL_MSG_SET_WIFI_CREDENCIALS: Mensagem responsável por definir credenciais de Wi-Fi. Essa mensagem demanda duas informações no payload, o SSID da rede e o password. Não implica em nenhuma mensagem de resposta, apenas o ACK.
- 0xF003: EXTERNAL_MSG_REQUEST_WIFI_CREDENCIALS: Mensagem responsável por solicitar as credenciais de rede salvas. Essa mensagem não demanda de payload e implica em uma mensagem de resposta, que será a 0x0002.
- 0xF004: EXTERNAL_MSG_SET_NET_IP: Mensagem responsável por definir as informações de IP da interface de rede. Essa mensagem demanda três informações no payload, o IP a ser setado para o dispositivo, o IP do gateway da rede e o IP do netmask desejado. Não implica em nenhuma mensagem de resposta, apenas o ACK.
- 0xF005: EXTERNAL_MSG_REQUEST_NET_IP: Mensagem responsável por solicitar os endereços de IP salvos. Essa mensagem não demanda de payload e implica em uma mensagem de resposta, que será a 0x0003.
- 0xF006: EXTERNAL_MSG_REQUEST_MAC_ADDR: Mensagem responsável por solicitar o mac address da interface utilizada. Essa mensagem não demanda de payload e implica em uma mensagem de resposta, que será a 0x0004.
- 0xF007: EXTERNAL_MSG_SET_NET_INTERFACE: Mensagem responsável por definir qual a interface de rede será utilizada. Essa mensagem demanda de uma informação no payload, sendo ‘1’ (ASCII) para selecionar a interface Cabeada e ‘2’ (ASCII) para selecionar a interface Wi-Fi. Essa mensagem reinicia o DataQ. Não implica em nenhuma mensagem de resposta, apenas o ACK.
- 0xF008: EXTERNAL_MSG_REQUEST_NET_INTERFACE: Mensagem responsável por solicitar qual a interface de rede está sendo utilizada. Essa mensagem não demanda de payload e implica em uma mensagem de resposta, que será a 0x0005.

As mensagens iniciadas por 0xF1XX são as mensagens relacionadas a coleta de dados, tendo efeito prático somente nos DataQ DI, sendo:

- 0xF100: EXTERNAL_MSG_REQUEST_DATA_COLLECT_INTERVAL, Mensagem responsável por solicitar ao DataQ o valor de intervalo de coleta dos pinos digitais. Pode ser enviada com o payload vazio. Essa mensagem implica em uma mensagem de resposta de comando 0x0100, além do ACK.
- 0xF101 até 0xF108: Essas mensagens são EXTERNAL_MSG_REQUEST_DATA_COLLECT_INX_CONFIGS, elas são responsáveis por solicitar ao DataQ os valores de configuração dos pinos. Pode ser

enviada com o payload vazio. Essa mensagem implica em uma mensagem de resposta com os IDs 0x0101 até 0x0108, além do ACK.

- 0xF109 até 0xF110: EXTERNAL MSG REQUEST DATA COLLECT INX STATE: São as mensagens de requisição dos estados dos pinos digitais de 1 a 8. Pode ser envaido com o payload vazio. Gera uma mensagem de resposta, que conterá a informação solicitada, além do ACK. Essa mensagem respondida tem o comando de 0x0109 até 0x0110, dependendo da mensagem enviada.
- 0xF111:EXTERNAL_MSG_REQUEST_EXTERN_DATA_VIA_SERIAL_CONFIG: Mensagem de requisição das configurações de externalização de dados via interface serial. Gera uma mensagem de resposta com o comando 0x0111.
- 0xF112:EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_INTERVAL: Mensagem responsável por definir o intervalo entre as coletas, em milissegundos. O payload dessa mensagem deve conter uma informação, que é esse tempo entre coletas, expresso em caracteres ASCII. Essa mensagem não implica em uma mensagem de resposta, somente o ACK.
- 0xF113 até 0xF11A: Mensagens de configuração dos pinos digitais a serem coletados, EXTERNAL_MSG_CONFIGURE_DATA_COLLECT_INX. Essa mensagem deve conter no payload 4 informações, a primeira é o enable do pino, sendo 1 ou 0 para habilitar ou desabilitar o pino; a segunda é o enable do WireBreak do pino, sendo 1 ou 0 para habilitar ou desabilitar; a terceira é o valor de debounce do pino, sendo os seguintes valores:

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

E por fim, a quarta informação é a definição de pino como input ou output, sendo 1 a definição de um pino input e 2 para pino output. Todas as informações devem ser enviadas em caracteres ASCII. Gera somente um ACK como resposta.

- 0xF11B: EXTERNAL_MSG_CONFIGURE_EXTERN_DATA_VIA_SERIAL: Mensagem de definição da externalização dos dados via interface serial. O payload deve conter 4 informações, a primeira é o enable da exportação, sendo 1 para habilitar e 0 para desabilitar. A segunda é o delay de exportação em segundos, seguido pelo enable da exportação do valor dos pinos, sendo 1 ou 0. E por fim, a terceira informação é o enable da exportação dos erros. Todas as informações devem ser enviadas em código ASCII. A única mensagem respondida para essa mensagem é o ACK.

As mensagens iniciadas por 0xF2XX são as mensagens relacionadas ao Web server:

- 0xF200: EXTERNAL_MSG_SEND_NEW_CA_FILE: Mensagem utilizada para enviar um novo certificado de CA para o WebServer. No payload deve conter o certificado da CA.
- 0xF201: EXTERNAL_MSG_SEND_NEW_CERT_FILE: Mensagem responsável por enviar o novo certificado do WebServer. No payload deve conter a nova chave publica do certificado.
- 0xF202: EXTERNAL_MSG_SEND_NEW_KEY_FILE: Mensagem responsável pelo envio de nova chave privada do certificado.

As mensagens iniciadas por 0xF3XX são as mensagens relacionadas às especificações do DataQ:

- 0xF300: EXTERNAL_MSG_REQUEST_MODEL: Mensagem responsável por solicitar o Modelo de dispositivo. O payload pode ser vazio. O retorno será DI ou DO, através da mensagem 0x0300.
- 0xF301: EXTERNAL_MSG_REQUEST_HW_VERSION: Solicitação da versão de hardware. Essa mensagem pode ter o payload vazio. Gera uma mensagem de resposta de comando 0x0301.
- 0xF302: EXTERNAL_MSG_REQUEST_SW_VERSION: Solicitação da versão de software. Essa mensagem pode ter o payload vazio. Gera uma mensagem de resposta com o comando 0x0302;
- 0xF303: EXTERNAL_MSG_REQUEST_SN: Mensagem que solicita o número serial do dispositivo.
- 0xF304: EXTERNAL_MSG_REBOOT: Mensagem que solicita um reboot do aparelho.

- 0xF305: EXTERNAL_MSG_FACTORY_RESET: Mensagem que faz uma redefinição de fábrica de todos os parâmetros configurados;

Essas são todas as mensagens que são interpretadas pelo DataQ, quando enviadas pelo par externo. Existem duas mensagens que podem ser enviadas por ambos os lados da comunicação, que são ACK e NACK, seus códigos de comando são 0xFFFF e 0xFFFE, respectivamente.

Existem também mensagens que são somente enviadas pelo DataQ, para o par da comunicação. Praticamente todas essas mensagens, de envio único por parte do DataQ, são as respostas das mensagens de solicitações de informações. Vale destacar que o par da comunicação deve enviar o ACK para cada uma das mensagens enviadas pelo DataQ, caso contrário ele irá continuar enviando repetidamente a mesma mensagem.

As mensagens serão listadas abaixo, iniciando pelas mensagens relacionadas a Rede, as quais tem o início 0x00XX:

- 0x0000: DATAQ_MSG_SCAN_NETWORKS_RESULT: Mensagem enviada como resposta do comando 0xF000. O seu payload contém as informações sobre todas as redes sem fio no alcance do DataQ. As informações são enviadas na seguinte ordem: SSID, Segurança da rede, e potência de sinal. Se existirem mais redes no alcance, as informações das outras redes seguem essa mesma ordem, enviadas sequencialmente. Os valores de segurança de rede são números, em representação ASCII e a tradução dos números é a seguinte:

```
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;
```

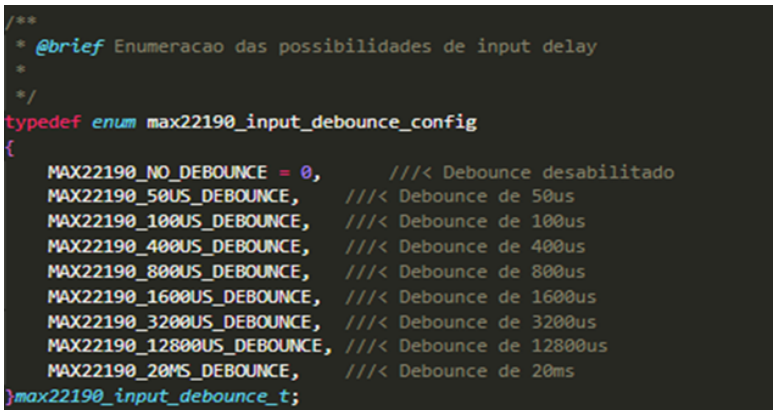
- 0x0001: DATAQ_MSG_RESPONSE_NET_STATE: Mensagem de resposta para a solicitação 0xF001. O seu payload contém 2 informações, a primeira é 1 ou 0,

indicando conexão ou não. Caso haja conexão, a segunda informação é 1 ou 0 para indicar a presença de endereço de IP na interface. Caso não haja conexão, será enviada na segunda informação um código de erro.

- 0x0002: DATAQ_MSG_RESPONSE_WIFI_CREDENCIALS: Mensagem de resposta para a solicitação de credenciais do Wi-Fi. São enviadas duas informações no payload, a primeira é o SSID e a segunda, o password.
- 0x0003: DATAQ_MSG_RESPONSE_NET_IP: Mensagem de resposta para a solicitação de IP da interface de rede. Essa mensagem possui 3 informações no payload: a primeira é o IP do dispositivo, a segunda é o IP do gateway e a última é a netmask adotada.
- 0x0004: DATAQ_MSG_RESPONSE_MAC_ADDR: Mensagem que é resposta da mensagem de solicitação do MAC Address. Essa mensagem possui apenas uma informação no payload, que é o MAC Address.
- 0x0005: DATAQ_MSG_RESPONSE_INTERFACE: Mensagem de resposta da solicitação da interface de rede sendo usada. Possui apenas uma informação no seu payload, sendo 1 para interface ETH e 2 para interface Wi-Fi. O conteúdo é sempre ASCII.

As próximas mensagens são relacionadas à coleta de dados, portanto seu uso só é válido quando se trata de um DataQ DI. Essas mensagens iniciam com o byte 0x01.

- 0x0100: DATAQ_MSG_RESPONSE_DATA_COLLECT_INTERVAL: Mensagem de resposta da solicitação do tempo de coleta. Essa mensagem possui apenas uma informação no payload, que é o intervalo do tempo de coleta em milissegundos.
- 0x0101 até 0x0108: DATAQ_MSG_RESPONSE_DATA_COLLECT_INX_CONFIGS: Mensagem de resposta da configuração de cada um dos pinos de coleta de informações. Existem 4 informações no payload dessa mensagem, escritos conforme a seguinte ordem: Enable do pino, 1 ou 0 ASCII; Enable do WireBreak, 1 ou 0 ASCII; Debounce, possui os valores ASCII, conforme a enumeração abaixo; Input Output, 1 para Input pin, 2 para Output pin, ambos ASCII.



```c
/**
 * @brief Enumeracao das possibilidades de input delay
 *
 */
typedef enum max22190_input_debounce_config
{
    MAX22190_NO_DEBOUNCE = 0,       ///< Debounce desabilitado
    MAX22190_50US_DEBOUNCE,     ///< Debounce de 50us
    MAX22190_100US_DEBOUNCE,    ///< Debounce de 100us
    MAX22190_400US_DEBOUNCE,    ///< Debounce de 400us
    MAX22190_800US_DEBOUNCE,    ///< Debounce de 800us
    MAX22190_1600US_DEBOUNCE,   ///< Debounce de 1600us
    MAX22190_3200US_DEBOUNCE,   ///< Debounce de 3200us
    MAX22190_12800US_DEBOUNCE, ///< Debounce de 12800us
    MAX22190_20MS_DEBOUNCE,     ///< Debounce de 20ms
}max22190_input_debounce_t;
```

- 0x0109 até 0x0110: DATAQ_MSG_RESPONSE_DATA_COLLECT_IN1_STATE: Mensagem de resposta da solicitação do estado do pino. No payload dessa mensagem existem duas informações. O estado do pino, seja alto ou baixo, 1 ou 0 ASCII; e o estado do WireBreak desse pino, 1 ou 0 ASCII.
- 0x0111: DATAQ_MSG_RESPONSE_EXTERN_DATA_VIA_SERIAL_CONFIG: Mensagem de resposta da solicitação das configurações de externalização de dados via interface serial. Nessa mensagem existem 4 informações no payload escritas na seguinte ordem: Enable da exportação via serial, 1 ou 0 ASCII; Delay da exportação em segundos ASCII; Enable da exportação dos valores dos pinos, 1 ou 0 ASCII; Enable da exportação dos bits de erro, 1 ou 0 ASCII.

As próximas mensagens são relacionadas a requisições de informações sobre o DataQ.

- 0x0300: DATAQ_MSG_RESPONSE_MODEL: Mensagem de resposta do modelo do DataQ. Possui apenas uma informação no payload. Que é DI ou DO.
- 0x0301: DATAQ_MSG_RESPONSE_HW_VERSION: Mensagem de resposta da solicitação de versão de hardware. Possui apenas uma informação no payload, que será a versão do Hardware.
- 0x0302: DATAQ_MSG_RESPONSE_SW_VERSION: Mensagem de resposta da solicitação de versão de software. Possui apenas uma informação no payload, que será a versão do Software.
- 0x0303: DATAQ_MSG_RESPONSE_SN: Mensagem de resposta da solicitação de número serial do produto. Possui apenas uma informação no payload, que será o número serial.

Com isso restaram somente o ACK e o NACK.

- 0xFFFF: ACK_COMMAND: Mensagem de ACK, enviado para todos os comandos, de ambas as partes. Não possui informações no payload.
- 0xFFFE: NACK_COMMAND: Mensagem de NACK, enviado para as mensagens recebidas com um CRC16 inválido. Possui uma informação no payload, que é o CRC16 esperado para a mensagem recebida.

Portanto todos os comandos foram citados e explicados, a próxima seção irá explicar o campo addtional frames.

Additional frames: Esse campo na maioria dos casos terá um valor 0, porém, para mensagens com payloads muito grandes (que excedam 0xFFFF bytes de comprimento), esse campo define quantos frames adicionais são necessários para o envio da mensagem.

Data size: Dois bytes que representam o tamanho do payload, podendo ser 0 e no máximo 0xFFFF. Os valores de dois bytes são sempre inseridos na mensagem como MSB First.

Payload: O campo do payload é o que contém toda a informação útil da mensagem. Caso a mensagem não demande o envio de nenhuma informação, esse campo será inexistente. Porém caso existam informações a serem enviadas, elas se organizam conforme a seguinte estrutura:

Para cada informação a ser escrita no payload deve ser escrito antes o tamanho da informação, em caracteres não ASCII, seguido então pela informação.

Abaixo está representado um payload da mensagem de envio das credenciais Wi-FI:

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

Sendo as aspas inexistentes na mensagem, usadas apenas para indicar os caracteres ASCII.

Para essa mensagem o Data Size será 0x001C, ou seja, vale 28.

Por fim, o último campo restante é o CRC16, todos os bytes da mensagem entram na conta do CRC16. Uma mensagem de ACK será inserida abaixo, para conferência de seu CRC16. O CRC16 é inserido na mensagem como MSB First.

Mensagem ACK completa:

0xAA, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x3C, 0x0A

Sendo o primeiro byte, o start byte;

Os dois próximos os bytes de comando;

O seguinte o byte de additional frames

Os dois seguintes o tamanho da mensagem;

E os dois últimos o CRC16, já que não há payload, se houvesse, ele seria presente antes dos bytes de CRC16.

```
/**
 * @brief Enumeracao com a descricao dos comandos
 *
 */
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;
```