Cloud Parametrization of CSI-Connect.

We use the Particle Cloud API: https://docs.particle.io/reference/device-cloud/api/

The CSI Connect uses HSI to communicate, both via USB and over the cloud.

- Particle allows the creation of functions on the device that can be called from the cloud using:
  - **POST /v1/products/:productIdOrSlug/devices/:deviceId/:functionName**
    Documentation: https://docs.particle.io/reference/device-cloud/api/#call-a-function
  - This request allows us to pass an argument to the function being called. As a string with a maximum of 63 characters.
  - The CSI Connect has a single registered function:
    - Hsi-call-from-Particle
    - Argument should be a HSI message in base64 string format. The HSI command can be one of many as described later in the document.
- Particle also allows us to register a variable that is readable from the cloud.
  - Link: https://docs.particle.io/reference/device-cloud/api/#get-a-variable-value
  - **GET /v1/products/:productIdOrSlug/devices/:deviceId/:varName**
  - We can return bool, double, int, String (up to 622 bytes of UTF-8 encoded characters).
  - It is also possible to make the device run a short function when the variable is read.
  - The CSI-Connect has two of these:
    - Bool HSI-Response-Ready
      - Returns false until a HSI-Response is ready.
      - Returns true When a response is ready.
    - String HSI-Response
      - Returns the latest response to a HSI call using the Hsi-call-from-Particle function described above.
        - The response is in base64 and contains a HSI response like you would get via USB as described later in the document.
      - When read the HIS-Response-Ready variable is set to false.
- It is possible to ping the device using:
  - **PUT /v1/products/:productIdOrSlug/devices/:deviceId/ping**
    - Link: https://docs.particle.io/reference/device-cloud/api/#ping-a-device

The flow of sending command and receiving response.

1. Send a HIS-Command to the device using **POST /v1/products/:productIdOrSlug/devices/:deviceId/Hsi-call-from-Particle**
2. Possibly wait some small amount of time as some commands might take some time for the CSI connect to perform. (Some require it to write to EEPROM which can take some time.)
    a. In my testing I wait 10 milliseconds. And have not yet had a problem.
3. Poll the boolian variable HSI-Response-Ready.
    a. If true proceed if not wait some more and try again.
4. Read the variable HSI-Response. Convert from Base64 to byte array and read the data.
5. Done

CSI-Connect HIS Commands.

A lot of the Comands are from the CSI-C11, and has been kept in place in the event that a tool should be created that can communicate with both devices. Some commands are only used for the Analog module internaly in the CSI-C11 and CSI Connect, these can be used by first stting passthrough mode with command 19. But for most applications this is not needed.

All comands consist of a start byte, a length byte(describing how many data bytes), an adress byte, a command number, x number of databytes, checksum byte, exit byte. Described in detail in "Quick Guide for HSI commands_R_F.pdf" on page 5

HIS comands consist of an array od bytes. For use via cloud with CSI-Connect, the array needs to be converted to a string, and encoded to base64 before it is sent. Responses is likevise sent as a string in base64 that needs to be decoded and and transformed to an array of bytes.

When sending and receiving data that consists of more than on byte the low byte is sent first. Example a 16 bit integer with the value 4370 decimal, hex1112, bin 0001 0010 0011 0100, will be sent as one byte integers 12, 11 in that order. A 32 bit integer example; decimal value 287454020, hex value 11223344 wil be sent 44, 33, 22, 11.

Command name	Number	Short Description
DeviceResultMessage	3	Will return the current value of all sensor channels, each as 32 bit integers.
SetPassThroughMode	19	Sets the passthrough mode, Mode 3 is the analog module 0 is off.
GetSetMaterialSerialNumber	140	Used to set or get the material and serial number of the device.
GetSetSmartsensorSettings	169	Used to set or get the smartsensor settings, options are CS1000 or HLB
GetSetAnalogSettings	173	Used to set or get parameters of a single analog channel.
GetSetAnalogCalibSettings	174	Used to set or get calibration parameters of all analog channels.
GetAnalogRawValueDigits	175	Returns the raw values measured by ADC of Analog module. Does not have to be sent wit passthrough active.
GetSetParticleStreamSettings	180	Used to get or set the Stream settings for particle. Modes are as of 05.02.21 Off=0, Continuos=1, threshold=2.
UpdateSettings	181	Writes settings from CSI-Connect to analog module. (It is not needed to do this)
GetParticleId	182	returns the particle id of the device
GetParticleAuthToken	183	Returns the Stored particle AuthToken. (Not used)
UpdateCMXSettings	184	Get or set the Stored CMX settings, Mostly used for creating a webhook with the labview tool.

In all cases where the command can either be write or read, the first value is a Uint16 that indicates direction. 0= read, 1=write.

DeviceResultMessage

Will return the current value of all sensor channels, each as 32 bit integers. Mostly used to get live data from the device.

- Expected data:
  - None
- Response data:
  - All measured values as uint32 values times 10.
  - Number of values depends on the settings, there ar 4 channels that can each be set to either:
    - Off (no data returned for this channel)
    - HLB (6 values)
    - CS1000 (4 values)
  - And always 4 analog values as the last

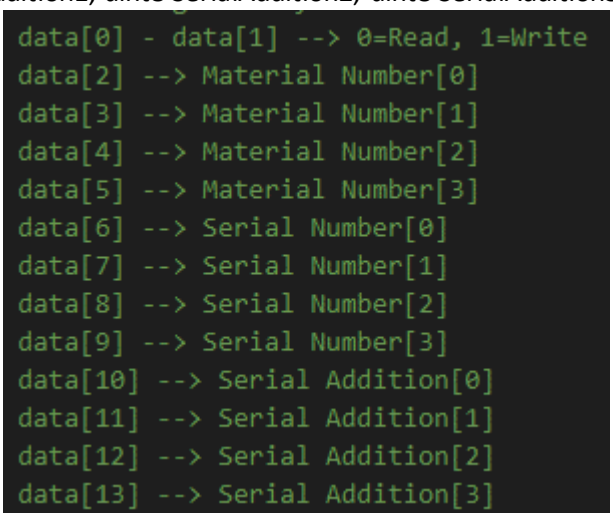
SetPassThroughMode

Sets the passthrough mode. This is only used during first setup of the device right after flashing of firmware, before calibration, not needed by the user.

- Expected data: uint8, 0 = 0ff and 3 = Analog module.
- Response data: uint8, mode

GetSetMaterialSerialNumber

Used to set or get the material and serial number of the device. Mostly used during first setup and calibration of the device.

- Expected data: uint16 direction, uint32 materialNumber, uint32 serialNumber, uint8 SerialAddition1, uint8 SerialAddition2, uint8 SerialAddition3, uint8 SerialAddition4.
  - 

- Response data: depends on direction:
  - Read:
    - uint32 materialNumber, uint32 serialNumber, uint8 SerialAddition1, uint8 SerialAddition2, uint8 SerialAddition3, uint8 SerialAddition4.
    -

```
data[2] --> Material Number[0]
data[3] --> Material Number[1]
data[4] --> Material Number[2]
data[5] --> Material Number[3]
data[6] --> Serial Number[0]
data[7] --> Serial Number[1]
data[8] --> Serial Number[2]
data[9] --> Serial Number[3]
data[10] --> Serial Addition[0]
data[11] --> Serial Addition[1]
data[12] --> Serial Addition[2]
data[13] --> Serial Addition[3]
```

    - Write:
        - int16 state
            - indicates if everything went well.

GetSetSmartsensorSettings

Used to set or get the smartsensor settings, options are CS1000 or HLB. When off the channels will work normally as analog channels. When either cs1000 or HLB, the channels will read the analog output of the chosen sensor.

- Expected data: uint16 direction, uint8 ch1Mode, uint8 ch2Mode, uint8 ch3Mode, uint8 ch4Mode, uint16 ch1SensorId, uint16 ch2SensorId, uint16 ch3SensorId, uint16 ch4SensorId.
    -
```
data[0] - data[1] --> 0=Read, 1=Write
data[2] - [5] --> Smartsensor Mode ch1-4
data[6] - data[13] --> SmartId ch1-4
```
- Response data: depends on direction:
    - Read:
        - uint8 ch1Mode, uint8 ch2Mode, uint8 ch3Mode, uint8 ch4Mode, uint16 ch1SensorId, uint16 ch2SensorId, uint16 ch3SensorId, uint16 ch4SensorId.
        -
```
data[2] - [5] --> Smartsensor Mode ch1-4
data[6] - data[13] --> SmartId ch1-4
```
    - Write:
        - int16 state
            - indicates if everything went well.

GetSetAnalogSettings

Used to set or get parameters of a single analog channel. To set all analog channels this will have to be sent 4 times with incrementing value of the channel number.

- Expected data: uint16 direction, uint8 channel, uint8 analogChannelType, string channelname (11 char), string channelUnit (9 char), int32 channelLowerRange, int32 channelUpperRange.

```
data[0] - data[1] --> 0=Read, 1=Write
data[2] --> 1=Channel1, 2=Channel2, 3=Channel3, 4=Channel4 //uint8
data[3] --> AnalogChannelxType                    //uint8
data[4] - data[14] --> ChannelxName               //string
data[15] - data[23] --> ChannlxUnit               //string
data[24]- data[27] --> ChannelxLowerRange         //int32
data[28]- data[31] --> ChannelxUpperRange         //int32
```

- Response data: depends on direction:
  - Read: uint8 channel, uint8 analogChannelType, string channelname (11 char), string channelUnit (9 char), int32 channelLowerRange, int32 channelUpperRange.

```
data[2] --> 1=Channel1, 2=Channel2, 3=Channel3, 4=Channel4 //uint8
data[3] --> AnalogChannelxType                    //uint8
data[4] - data[14] --> ChannelxName               //string
data[15] - data[23] --> ChannlxUnit               //string
data[24]- data[27] --> ChannelxLowerRange         //int32
data[28]- data[31] --> ChannelxUpperRange         //int32
```

  - Write: int16 state,
    - Indicates if everything went well.

GetSetAnalogCalibSettings

Used to set or get calibration parameters of all analog channels. Mostly used during first setup of device when calibration analog channels.

- Expected data: uint16 direction, int16 UGradient1, int16 UOffset1, int16 IGradient1, int16 IOffset1, int16 UGradient2, int16 UOffset2, int16 IGradient2, int16 IOffset2, int16 UGradient3, int16 UOffset3, int16 IGradient3, int16 IOffset3, int16 UGradient4, int16 UOffset4, int16 IGradient4, int16 IOffset4.

```
data[0] - data[1] --> 0=Read, 1=Write
data[2] - data[3] --> UGradient1
data[4] - data[5] --> UOffset1
data[6] - data[7] --> IGradient1
data[8] - data[9] --> IOffset1
data[10] - data[11] --> UGradient2
data[12] - data[13] --> UOffset2
data[14] - data[15] --> IGradient2
data[16] - data[17] --> IOffset2
data[18] - data[19] --> UGradient3
data[20] - data[21] --> UOffset3
data[22] - data[23] --> IGradient3
data[24] - data[25] --> IOffset3
data[26] - data[27] --> UGradient4
data[28] - data[29] --> UOffset4
data[30] - data[31] --> IGradient4
data[32] - data[33] --> IOffset4
```

- Response data: depends on direction:

- Read: int16 UGradient1, int16 UOffset1, int16 IGradient1, int16 IOffset1, int16 UGradient2, int16 UOffset2, int16 IGradient2, int16 IOffset2, int16 UGradient3, int16 UOffset3, int16 IGradient3, int16 IOffset3, int16 UGradient4, int16 UOffset4, int16 IGradient4, int16 IOffset4.

```
data[2] - data[3] --> UGradient1
data[4] - data[5] --> UOffset1
data[6] - data[7] --> IGradient1
data[8] - data[9] --> IOffset1
data[10] - data[11] --> UGradient2
data[12] - data[13] --> UOffset2
data[14] - data[15] --> IGradient2
data[16] - data[17] --> IOffset2
data[18] - data[19] --> UGradient3
data[20] - data[21] --> UOffset3
data[22] - data[23] --> IGradient3
data[24] - data[25] --> IOffset3
data[26] - data[27] --> UGradient4
data[28] - data[29] --> UOffset4
data[30] - data[31] --> IGradient4
data[32] - data[33] --> IOffset4
```

- Write: int16 state,
  - Indicates if everything went well.

GetAnalogRawValueDigits

Returns the raw values measured by ADC of Analog module. Does not have to be sent wit passthrough active.

- Expected data: none.
- Response data: uint16 ch1RawValue, uint16 ch2RawValue, uint16 ch3RawValue, uint16 ch4RawValue.

GetSetParticleStreamSettings

Used to get or set the Stream settings for particle. Modes are as of 05.02.21 Off=0, Continuos=1, threshold=2.

- Expected data: uint16 direction, uint8 particleStreamMode, uint32 SamplePeriod, uint8 ThresholdChannel, int32 ThresholdLowerRange, int32 ThresholdUpperRange.

```
data[0] - data[1] --> 0=Read, 1=Write
data[2] --> Particle Stream Mode //uint8
data[3] - data[6] --> sample Period //uint32
data[7] --> ThresholdChannel    // uint8
data[8] - data[11] --> Threshold Lower Range //int32
data[12] - data[15] --> Threshold upper Range //int32
```

- Response data: depends on direction:

- Read: uint8 particleStreamMode, uint32 SamplePeriod, uint8 ThresholdChannel, int32 ThresholdLowerRange, int32 ThresholdUpperRange.
  - ```
    data[2] --> Particle Stream Mode //uint8
    data[3] - data[6] --> sample Period //uint32
    data[7] --> ThresholdChannel    // uint8
    data[8] - data[11] --> Threshold Lower Range //int32
    data[12] - data[15] --> Threshold upper Range //int32
    ```
- Write: int16 state,
  - Indicates if everything went well.

UpdateSettings

Writes settings from CSI-Connect to analog module. (It is not needed to do this as settings are set in the analog module when setting them using other commands already)

- Expected data: none.
- Response data: none.

GetParticleId

returns the particle id of the device.

- Expected data: none
- Response data: string deviceId (25 char)

GetParticleAuthToken

Returns the Stored particle AuthToken. (Not used)

- Expected data: uint16 direction, string particleAuthToken (42 char).
  - ```
    data[0] - data[1] --> 0=Read, 1=Write
    data[2] - data[42]--> ParticleAuthToken //string
    ```
- Response data: depends on direction:
  - Read: string particleAuthToken (42 char).
    - ```
      data[2] - data[42]--> ParticleAuthToken //string
      ```
  - Write: int16 state,
    - Indicates if everything went well.
      ```

UpdateCMXSettings

Get or set the Stored CMX settings stored in the device, used for creating a webhook with the labview tool.

- Expected data: uint16 direction, uint32 IdCh1, uint32 stationIdCh1, uint32 DatapointIdCh1, uint32 IdCh2, uint32 stationIdCh2, uint32 DatapointIdCh2, uint32 IdCh3, uint32 stationIdCh3, uint32 DatapointIdCh3, uint32 IdCh4, uint32 stationIdCh4, uint32 DatapointIdCh4.
  - ```
    data[0] - data[1] --> 0=Read, 1=Write
    data[2] - data[129] --> URL //string
    data[130] - data[133] --> IdCh1                 //uint32
    data[134] - data[137] --> StationIdCh1          //uint32
    data[138] - data[141] --> DataPointIdCh1        //uint32
    data[142] - data[145] --> IdCh2                 //uint32
    data[146] - data[149] --> StationIdCh2          //uint32
    data[150] - data[153] --> DataPointIdCh2        //uint32
    data[154] - data[157] --> IdCh3                 //uint32
    data[158] - data[161] --> StationIdCh3          //uint32
    data[162] - data[165] --> DataPointIdCh3        //uint32
    data[166] - data[169] --> IdCh4                 //uint32
    data[170] - data[173] --> StationIdCh4          //uint32
    data[174] - data[177] --> DataPointIdCh4        //uint32
    ```
- Response data: Depends on direction:
  - Read:
    - ```
      data[2] - data[129] --> URL //string
      data[130] - data[133] --> IdCh1                 //uint32
      data[134] - data[137] --> StationIdCh1          //uint32
      data[138] - data[141] --> DataPointIdCh1        //uint32
      data[142] - data[145] --> IdCh2                 //uint32
      data[146] - data[149] --> StationIdCh2          //uint32
      data[150] - data[153] --> DataPointIdCh2        //uint32
      data[154] - data[157] --> IdCh3                 //uint32
      data[158] - data[161] --> StationIdCh3          //uint32
      data[162] - data[165] --> DataPointIdCh3        //uint32
      data[166] - data[169] --> IdCh4                 //uint32
      data[170] - data[173] --> StationIdCh4          //uint32
      data[174] - data[177] --> DataPointIdCh4        //uint32
      ```
  - Write: int16 state,
    - Indicates if everything went well.