nodeG5 – iotasset.json config guide

CAN-LISTEN

Firmware version: fw_nodeG5_v2.2
Guide release date: 8MAR2024

Filename :	iotasset.json
Location :	/user

1. Introduction
   The file 'iotasset.json' contains the assets configuration that is read by the CAN master program to parse specific data from CAN devices. Acquired data is then inserted into local database for downstream IoT cloud clients.

2. IoTasset name defiitions
   CAN-LISTEN IoTasset is defined by JSON objects inside JSON array "CANBUS".

   Below is the list of IoTasset names that should be defined in each JSON object.

IoTasset name	**Description**
Key	Define the data tagname
IOTMODE	Define the data handling mode

Below is the list of CAN-LISTEN names that should be defined in each JSON object.

CAN-LISTEN name	**Description**
CANBUSTYPE	Define the type of CAN bus communication
CANPORT	Define the physical port used for communication
CANIDLISTEN	Define the identifier for interesting CAN message
CANIDMASK	Define the acceptance mask for CAN message filtering
DATATYPE	Define data type to convert from CAN bus raw data

Data type = INTEGER, STRING, FLOAT, CANRAW	
BYTESTART	Define the position of starting byte to parse
BYTECOUNT	Define the number of bytes to parse

Data type = BITS	
BITSTART	Define the position of starting byte.bit to parse
BITCOUNT	Define the number of bits to parse

Optional names:

Data type = INTEGER, FLOAT only number=number*multiplier + adder	
SCALEMUL	Define both multiplier & adder when require scaling
SCALEADD	

3. IoTasset JSON object setup information

“Key”:”value”

value	Description
string	Unique name for this sensor data value eg temperature, voltage, pressure, rpm

“IOTMODE”:”value”

value	Description
0	Send to cloud immediately (priority data)
1	Store to local database for local IoT client processing

“CANBUSTYPE”:”value”

value	Description
LISTEN	CAN-LISTEN mode: listen to CAN messages

“CANPORT”:”value”

value	Description
canE	On-board CAN port (default)
canC	Expansion CAN port

Note: Refer to nodeG5 user manual for connector pin-out details.

“CANIDLISTEN”:”value”

value	Description
000-7FF 00000000-1FFFFFFF	11-bit identifier (CAN 2.0A) 29-bit identifier (CAN 2.0B)

“CANIDMASK:”value”

value	Description
000-7FF 00000000-1FFFFFFF	11-bit acceptance mask (CAN 2.0A) 29-bit acceptance mask (CAN 2.0B)

Note: By default mask=7FF if not defined in iotasset.json

“BYTESTART”:”value”

value	Description
1,2,3...8	Position of starting byte (1=left most byte)

“BYTECOUNT”:”value”

value	Description
1,2,3...8	Number of bytes is dependent on data type

“BITSTART”:”value”

value	Description
(byte#1-8).(bit#1-8)	Position of starting byte.bit (1=left most byte/bit)

“BITCOUNT”:”value”

value	Description
1,2,3...8	Number of bits on a single byte

Note: Bits parsing can only be applied on single byte of CAN data and not across multiple bytes.

4. Data Type definitions for CAN-LISTEN

“DATATYPE”:”value”
“BITCOUNT”:”count”

value	count	Description
BITS	1-8	1-8 bits to unsigned integer

Note: Binary value parsed will be converted to decimal value, eg 1110_2 will be reported as 14_{10}.

“DATATYPE”:”value”
“BYTECOUNT”:”count”

value	count	Description
UINT8	1	8-bit data to **8-bit unsigned integer**
SINT8		8-bit data to **8-bit signed integer**
UINT16HL	2	8-bit data pair to **16-bit unsigned integer**
UINT16LH		
SINT16HL		8-bit data pair to **16-bit signed integer**
SINT16LH		
UINT32HLhl	4	8-bit data quad to **32-bit unsigned integer**
UINT32hlHL		
UINT32LHlh		
UINT32lhLH		
SINT32HLhl		8-bit data quad to **32-bit signed integer**
SINT32hlHL		
SINT32LHlh		
SINT32lhLH		

Note: U=unsigned, S=signed, HLhl=big endian, lhLH=little endian
hlHL=word-little-endian, byte-big-endian
Lhlh=word-big-endian, byte-little-endian

“DATATYPE”:”value”
“BYTECOUNT”:”count”

value	count	Description
STRING8	8	Set of eight 8-bit data to **8 ASCII characters** (abcdefgh)
STRING8R		Set of eight 8-bit data to **8 ASCII characters**, reversed (hgfedcba)
STRING4	4	Set of four 8-bit data to **4 ASCII characters** (abcd)
STRING4R		Set of four 8-bit data to **4 ASCII characters**, reversed (dcba)

“DATATYPE”:”value”
“BYTECOUNT”:”count”

value	count	Description
FLOAT32ABCD	4	Set of four 8-bit data to IEEE-754 single precision floating point number. Byte orientation=ABCD,DCBA,BADC,CDAB A,B,C,D=canbyte1,canbyte2,canbyte3,canbyte4
FLOAT32DCBA		
FLOAT32BADC		
FLOAT32CDAB		

“DATATYPE”:”value”
“BYTECOUNT”:”count”

value	count	Description
CANRAW	8	String of 16 hexadecimal char

5. Example IOTasset JSON configuration


```
{
  "CANBUS":[
    {
      "Key":"SensorA",
      "IOTMODE":"1",
      "CANBUSTYPE":"LISTEN",
      "CANPORT":"canE",
      "CANIDLISTEN":"37F",
      "CANIDMASK":"7FF",
      "BYTESTART":"1",
      "BYTECOUNT":"2",
      "DATATYPE":"UINT16HL"
    },
    {
      "Key":"SensorB",
      "IOTMODE":"1",
      "CANBUSTYPE":"LISTEN",
      "CANPORT":"canE",
      "CANIDLISTEN":"38F",
      "CANIDMASK":"7FF",
      "BYTESTART":"1",
      "BYTECOUNT":"2",
      "DATATYPE":"UINT16HL"
    },
    {
      "Key":"SensorC",
      "IOTMODE":"1",
      "CANBUSTYPE":"LISTEN",
      "CANPORT":"canE",
      "CANIDLISTEN":"39F",
      "CANIDMASK":"7FF",
      "BYTESTART":"1",
      "BYTECOUNT":"2",
      "DATATYPE":"UINT16HL"
    }
  ]
}
```


6. Methods to upload ‘iotasset.json’ file to nodeG5

-Upload the iotasset.json file from your computer using the ‘Upload iotasset.json’ button in the ‘IoT Hardware’ tab.

-Put the iotasset.json file in \user folder of USB drive.
Plug the USB drive into any USB-A port and click the ‘Upload to nodeG5’ button in the ‘Management’ tab.

-Use SCP/Putty console or WinSCP.

<EOF>