G3 Modbus configuration guide (v2.3)

Revision log:
v2.3>add support for #comment
>add support for different polling interval for each asset
>add [multiplier, adder] optional arguments for data type FLOAT
>add support for data type INT64, return hexadecimal value
v2.2> add support for data type INT8
> add support for Modbus/RTU over TCP
v2.1> add block marker MBM_START, MBM_STOP
>add [multiplier, adder] optional arguments for data type INT
v2.0> add support for data type FLOAT.
>add support for Modbus protocol 32bit registers (custom firmware required)
>revise definition of data type REGS.

Filename :	iotasset.txt
Location :	\user

1. Introduction

   The file 'iotasset.txt' contains the assets configuration that is required by the Modbus master program to acquire data from slave devices and also pre-process for downstream IoT clients.

2. IOT asset 'KEY,VALUE' general format

   Each IOT asset is defined by using a BLOCK of 'key, value' pairs (CSV format).
   There are four Modbus key names that must be present for each IOT asset.
   These Modbus key names are reserved and cannot be used for custom key names.

MODBUS KEYS	**Description**
TYPE	Define the type of Modbus communication
ADDR	Define the address of the Modbus slave device
MBFC	Define the Modbus function code for query device
REGS	Define the Modbus target registers and data type

Custom keys can be freely defined but limited to eight custom keys.
Each asset block must include the same set of custom keys.
Backslash (\) and quotations marks (") char cannot be used.
Comments can be inserted by using the hash (#) sign.

To ease parsing of different types of assets, the asset blocks need to be located between the start and end of block markers.

MODBUS BLOCK MARKER	**Description**
MBM_START	Define the start of Modbus assets
MBM_STOP	Define the end(stop) of Modbus assets

3. IOT asset 'KEY,VALUE' setup information

TYPE, m [, i]

Argument	Value	Description
m	R	Modbus/RTU standard protocol over RS-485
	T	Modbus/TCP standard protocol over Ethernet
	R32	Modbus/RTU 32bit protocol (MBFC=3,4 only) #1
	T32	Modbus/TCP 32bit protocol (MBFC=3,4 only) #1
i	1, 2, 3, 4, 5,….	Poll interval for each asset. #2

#1 Custom firmware required.

#2 Optional: to define poll interval of each asset, opt out is same as define i=1 (i.e. poll every interval).

Example of Poll Interval calculations with Modbus master Poll Period = 15 sec.

note: Modbus master Poll Period is the time interval between polling, refer web 'IoT Hardware' config.

Asset Poll Period	Calculation	Poll Interval (i)
1min	1*60/15	4
30min	30*60/15	120
1 hour	1*60*60/15	240
3 hour	3*60*60/15	720

ADDR, n0 Modbus/RTU

ADDR, n1.n2.n3.n4 : p Modbus/TCP

ADDR, n1.n2.n3.n4 : p : u Modbus/RTU over TCP

Argument	Value	Description	Notes
n0	1-247	Slave address	For Modbus/RTU
n1.n2.n3.n4:p	n1,n2,n3,n4=0-255 p=502 (default)	Slave IP address Slave port	For Modbus/TCP
n1.n2.n3.n4:p:u	n1,n2,n3,n4=0-255 p=502 (default) u=1-247	Slave IP address Slave port Unit ID	For Modbus/RTU over TCP setup using TYPE, T

MBFC, s

Argument	Value	Description
s	1	Read Coil Status (FC=01)
	2	Read Input Status (FC=02)
	3	Read Holding Registers (FC=03)
	4	Read Input Registers (FC=04)

REGS, t, u, v [, x, y]

Argument	Value	Description
t	0, 1, 2, 3, 4, 5,…	Data address of first register requested (dec) [#3]
u	Number of register	Number of register requested (dec)
v	Data Type	Data type conversion from Modbus hex data
x	Multiplier	Value = Value*Multiplier + Adder [#4]
y	Adder	Value = Value*Multiplier + Adder [#4]

#3 Depending on device model, may require a -1 offset of the register address.

#4 Optional: for Data Type INTEGER & FLOAT, **both** x & y arguments required when applied.

4. Data Type definitions for standard Modbus

DATA TYPE BOOLEAN FOR MODBUS 1-BIT REGISTER (FC=01/02)

v[Data Type]	u[Number of register]	Description
BOOL	1	Boolean value, ie 0 or 1

DATA TYPE INTEGER FOR MODBUS 16-BIT REGISTER (FC=03/04)

v[Data Type]	u[Number of register]	Description
UINT8H/UINT8L	1	16-bit register to **8-bit unsigned integer**, hi byte / lo byte
SINT8H/SINT8L		16-bit register to **8-bit signed integer**, hi byte / lo byte
UINT16HL	1	16-bit register to **16-bit unsigned integer**, big endian
UINT16LH		16-bit register to **16-bit unsigned integer**, little endian
SINT16HL		16-bit register to **16-bit signed integer**, big endian
SINT16LH		16-bit register to **16-bit signed integer**, little endian
UINT32HLhl	2	16-bit register pair to **32-bit unsigned integer**, big endian
UINT32hlHL		16-bit register pair to **32-bit unsigned integer**, Word – little endian, Byte – big endian
UINT32LHlh		16-bit register pair to **32-bit unsigned integer**, Word – big endian, Byte – little endian
UINT32lhLH		16-bit register pair to **32-bit unsigned integer**, little endian
SINT32HLhl		16-bit register pair to **32-bit signed integer**, big endian
SINT32hlHL		16-bit register pair to **32-bit signed integer**, Word – little endian, Byte – big endian
SINT32LHlh		16-bit register pair to **32-bit signed integer**, Word – big endian, Byte – little endian
SINT32lhLH		16-bit register pair to **32-bit signed integer**, little endian
HEX64BIGLIL	4	16-bit register dual-pair to **64-bit hexadecimal**, big endian
HEX64LILBIG		16-bit register dual-pair to **64-bit hexadecimal**, little endian

DATA TYPE STRING FOR MODBUS 16-BIT REGISTER (FC=03/04)

v[Data Type]	u[Number of register]	Description
STRING16	8	Set of eight 16-bit registers to **16 ASCII characters** (e.g. abcdefghijklmnop)
STRING16R		Set of eight 16-bit registers to **16 ASCII characters**, byte swapped (e.g. badcfehgjilknmpo)
STRING8	4	Set of four 16-bit registers to **8 ASCII characters** (e.g. abcdefgh)
STRING8R		Set of four 16-bit registers to **8 ASCII characters**, byte swapped (e.g. badcfehg)

DATA TYPE FLOAT FOR MODBUS 16-BIT REGISTER (FC=03/04)

v[Data Type]	u[Number of register]	Description
FLOAT32ABCD	2	16-bit register pair to IEEE-754 single precision floating point number. Byte orientation=ABCD,DCBA,BADC,CDAB AB=word data from first 16-bit register CD=word data from second 16-bit register
FLOAT32DCBA		
FLOAT32BADC		
FLOAT32CDAB		

5. Data type definitions for 32-bit Modbus

DATA TYPE INTEGER FOR MODBUS 32-BIT REGISTER (FC=03/04)*

v[Data Type]	u[Number of register]	Description
UINT32HLhl	1	32-bit register to **32-bit unsigned integer**, big endian
UINT32hlHL		32-bit register to **32-bit unsigned integer**, Word – little endian, Byte – big endian
UINT32LHlh		32-bit register to **32-bit unsigned integer**, Word – big endian, Byte – little endian
UINT32lhLH		32-bit register to **32-bit unsigned integer**, little endian
SINT32HLhl		32-bit register to **32-bit signed integer**, big endian
SINT32hlHL		32-bit register to **32-bit signed integer**, Word – little endian, Byte – big endian
SINT32LHlh		32-bit register to **32-bit signed integer**, Word – big endian, Byte – little endian
SINT32lhLH		32-bit register to **32-bit signed integer**, little endian
HEX64BIGLIL	2	32-bit register pair to **64-bit hexadecimal**, big endian
HEX64LILBIG		32-bit register pair to **64-bit hexadecimal**, little endian

*custom firmware required

DATA TYPE STRING FOR MODBUS 32-BIT REGISTER (FC=03/04)*

v[Data Type]	u[Number of register]	Description
STRING16	4	Set of four 32-bit registers to **16 ASCII characters** (e.g. abcdefghijklmnop)
STRING16R		Set of four 32-bit registers to **16 ASCII characters**, word swapped (e.g. dcbahgfelkjiponm)
STRING8	2	Set of two 32-bit registers to **8 ASCII characters** (e.g. abcdefgh)
STRING8R		Set of two 32-bit registers to **8 ASCII characters**, word swapped (e.g. dcbahgfe)

*custom firmware required

DATA TYPE FLOAT FOR MODBUS 32-BIT REGISTER (FC=03/04)*

v[Data Type]	u[Number of register]	Description
FLOAT32ABCD	1	32-bit register to IEEE-754 single precision floating point number. Byte orientation=ABCD,DCBA,BADC,CDAB ABCD=double word data from a 32-bit register
FLOAT32DCBA		
FLOAT32BADC		
FLOAT32CDAB		

*custom firmware required

6. Example for IOT asset configuration

#iotasset example for Modbus/RTU and Modbus/TCP

```
MBM_START                       #start of Modbus block

TYPE, R                         #Modbus/RTU
ADDR, 1                         #Slave address=1
MBFC, 2                         #Modbus function=02
REGS, 1, 1, BOOL                #Data address=1, Number of register=1, Data type=Boolean
Site, APC                       #Custom key1
Grp, PMS                        #Custom key2
Unit, status                    #Custom key3
Key, Door                       #Custom key4, max supported=8

TYPE, R, 5                      #Modbus/RTU, poll on every 5 polling interval
ADDR, 2                         #Slave address=2
MBFC, 3                         #Modbus function=03
REGS, 25, 1, UINT16HL, 0.5, 0.1 #Data type=unsigned 16-bit integer, value=value*0.5 + 0.1
Site, APC
Grp, PMS
Unit, V
Key, UPSVoltage

TYPE, T                         #Modbus/TCP
ADDR, 192.168.1.100:502         #Slave IP=192.168.1.100, Port=502
MBFC, 3
REGS, 100, 2, UINT32HLhl        #Data type=unsigned 32-bit integer
Site, APC
Grp, PMS
Unit, degC
Key, Temperature

TYPE, T, 10                     #Modbus/TCP, poll on every 10 polling interval
ADDR, 192.168.1.120:502
MBFC, 4
REGS, 200, 8, STRING16          #Data type=16 ascii chars
Site, APC
Grp, PMS
Unit, text
Key, Alarm

MBM_STOP                        #end of Modbus block
```

7. Methods to upload ‘iotasset.txt’ file to G3

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

-Put the iotasset.txt file in \user folder of USB drive (with label ‘FATBOX’).
Plug the USB drive into G3 and click the ‘Upload to FATBOX’ button in the ‘Management’ tab.

-Use SCP/Putty console or WinSCP.

<EOF>