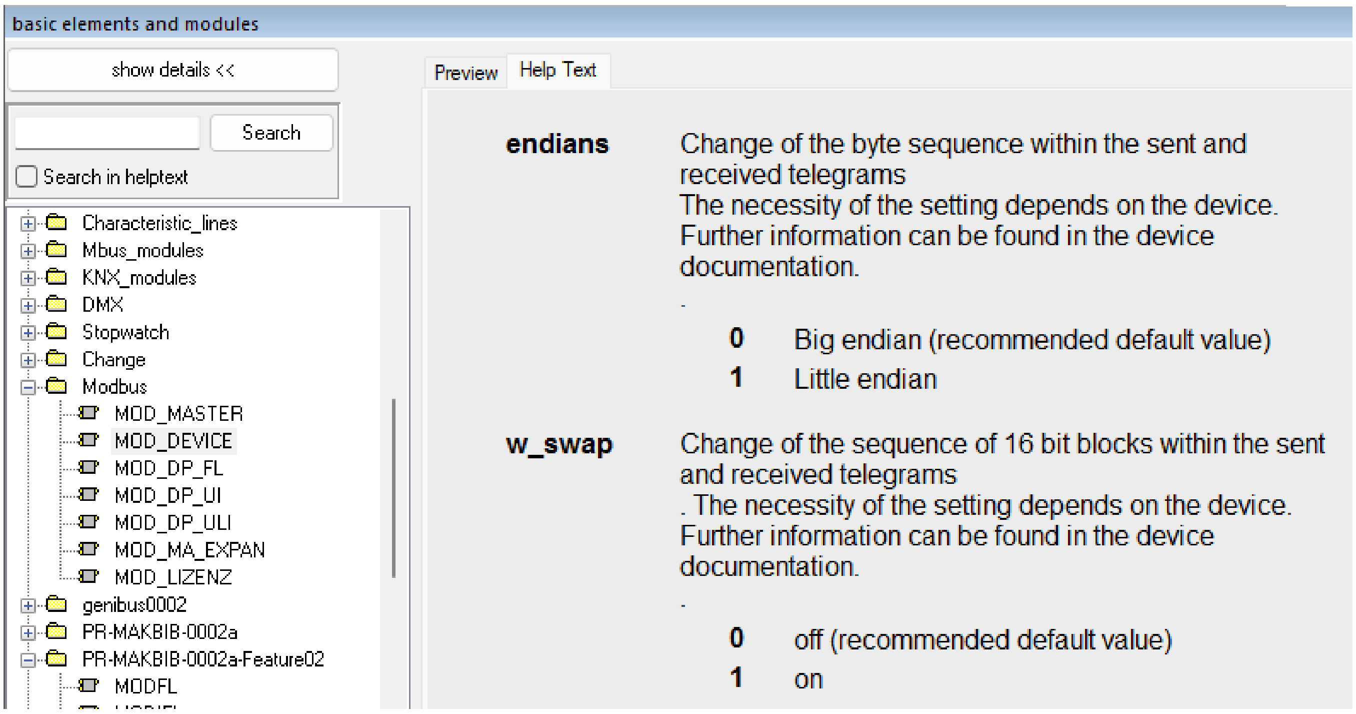
TT230603 – OFXL - Modbus Module Useful Settings

1. There are many settings in the Modbus modules. Let's take a look at the "MOD_DEVICE" module first.



endians Change of the byte sequence within the sent and received telegrams
The necessity of the setting depends on the device. Further information can be found in the device documentation.
-

0	Big endian (recommended default value)
1	Little endian

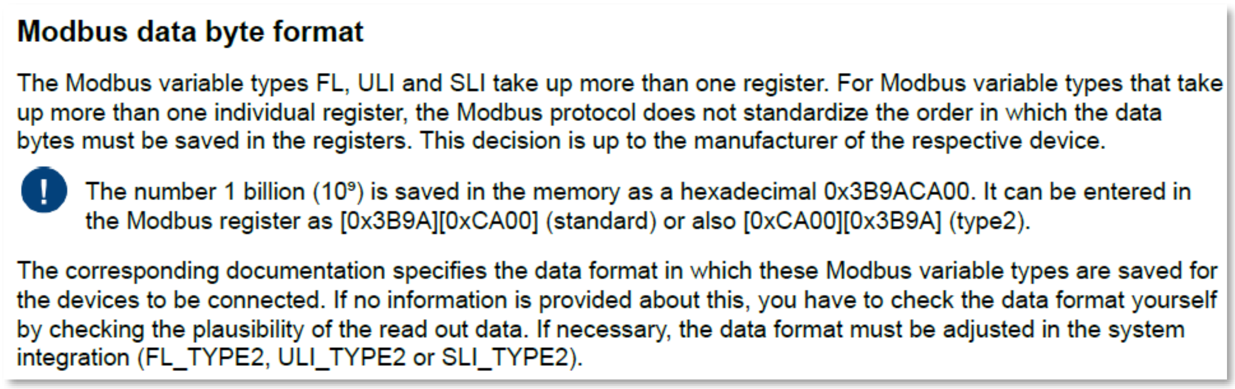
w_swap Change of the sequence of 16 bit blocks within the sent and received telegrams
. The necessity of the setting depends on the device. Further information can be found in the device documentation.
-

0	off (recommended default value)
1	on

2. Modbus itself does not define floating-point data types, but it is widely accepted that it implements 32-bit floating-point data using the IEEE-754 standard. However, the IEEE standard has no clear definition of the byte order. Therefore, the most important consideration when dealing with 32-bit data is that the data be addressed in the correct order. This is the same for 32-btt integer.
3. The following table shows two adjacent 16-bit registers conversion to a 32-bit floating point or 32-bit integer values.

Function Keyword	Swap Mode	Source Bytes	Target Bytes
2.i16-1.i32	N/A	[a b] [c d]	[a b c d]
2.i16-1.i32-s	byte and word swap	[a b] [c d]	[d c b a]
2.i16-1.i32-sb	byte swap	[a b] [c d]	[b a d c]
2.i16-1.i32-sw	word swap	[a b] [c d]	[c d a b]

4. You can see from the above table that a 32-bit Modbus register can be represented in 4 different ways (e.g. abcd). Therefore, we have the "endians" and "w_swap" settings to match any one of them.
5. These are the same as FL and FL_Type2 settings in FUP 2.



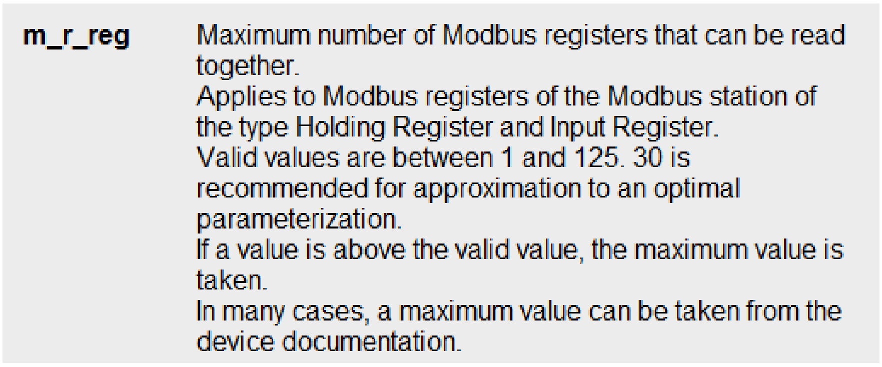
Modbus data byte format

The Modbus variable types FL, ULI and SLI take up more than one register. For Modbus variable types that take up more than one individual register, the Modbus protocol does not standardize the order in which the data bytes must be saved in the registers. This decision is up to the manufacturer of the respective device.

! The number 1 billion (10^9) is saved in the memory as a hexadecimal 0x3B9ACA00. It can be entered in the Modbus register as [0x3B9A][0xCA00] (standard) or also [0xCA00][0x3B9A] (type2).

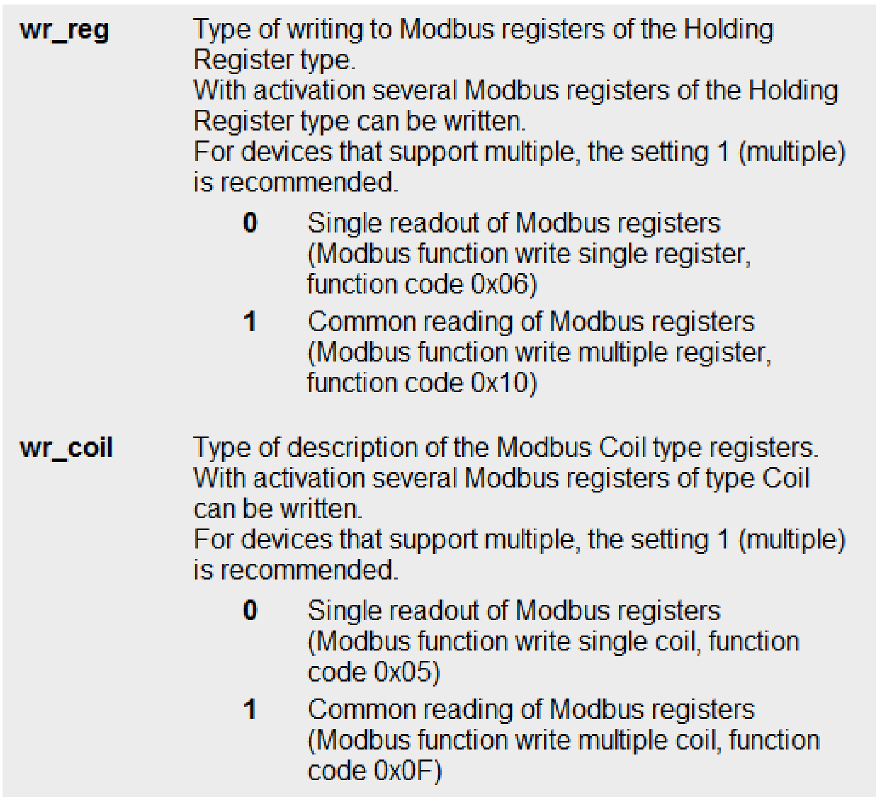
The corresponding documentation specifies the data format in which these Modbus variable types are saved for the devices to be connected. If no information is provided about this, you have to check the data format yourself by checking the plausibility of the read out data. If necessary, the data format must be adjusted in the system integration (FL_TYPE2, ULI_TYPE2 or SLI_TYPE2).

6. You can set either of them to 0 or 1 to see if you can read the correct value. Since most of the Modbus datasheet do not specify which one the Modbus device use, so you have to use trial and error if the default value does not work.
7. Thanks for the new Modbus module and the settings can be changed online, so you can easily test it with your Modbus device on-site.
8. The next are "m_r_reg" and "m_r_bit" settings. This allows the Modbus module to read multiple registers and bits from the Modbus devices automatically. Normally, the default value is fine so no need to change them. Similar for "m_w_reg" and "m_w_bit" settings which are for writing multiple registers and bits.



m_r_reg	Maximum number of Modbus registers that can be read together. Applies to Modbus registers of the Modbus station of the type Holding Register and Input Register. Valid values are between 1 and 125. 30 is recommended for approximation to an optimal parameterization. If a value is above the valid value, the maximum value is taken. In many cases, a maximum value can be taken from the device documentation.

9. Next are the "wr_reg" and "wr_coil" settings which enable/disable the multiple writing of Modbus holding register and coil. Since they use different Modbus function codes, you need to check if the Modbus device support the Modbus function code or not.

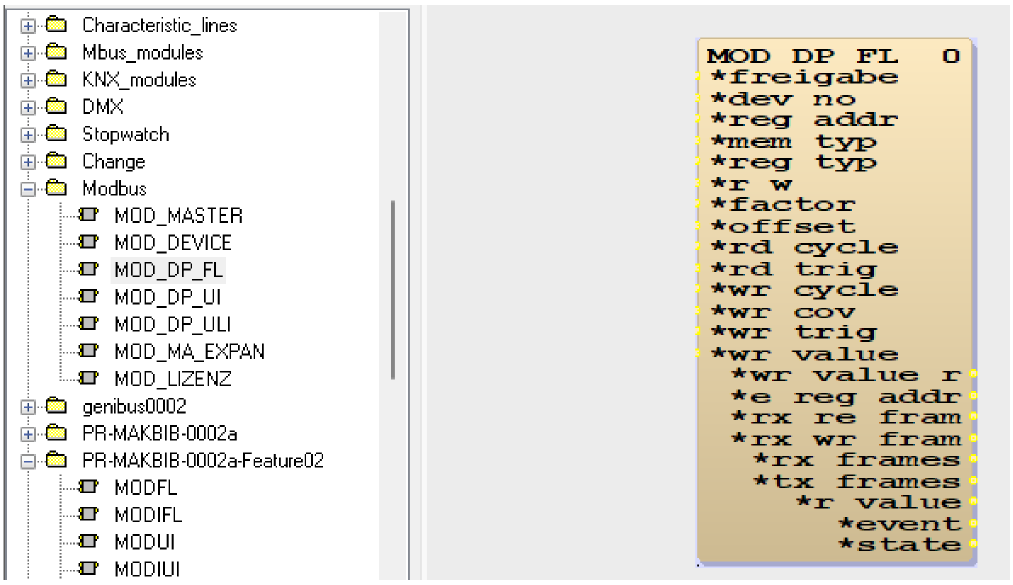


wr_reg	Type of writing to Modbus registers of the Holding Register type. With activation several Modbus registers of the Holding Register type can be written. For devices that support multiple, the setting 1 (multiple) is recommended. **0** Single readout of Modbus registers (Modbus function write single register, function code 0x06) **1** Common reading of Modbus registers (Modbus function write multiple register, function code 0x10)
wr_coil	Type of description of the Modbus Coil type registers. With activation several Modbus registers of type Coil can be written. For devices that support multiple, the setting 1 (multiple) is recommended. **0** Single readout of Modbus registers (Modbus function write single coil, function code 0x05) **1** Common reading of Modbus registers (Modbus function write multiple coil, function code 0x0F)

10. For Modbus device that use integer (e.g. Int, UI, SI, etc), they are 16-bit registers, and therefore we can set "wr_reg" to 0. For Modbus device that use floating point or 32-bit integer (e.g. FL, ULI, SLI, etc.), we have to set "wr_reg" to 1.
11. For "wr_coil", we can just leave it as default value 0.
12. For the other settings, normally you can also leave it as default value. Please refer to the below table for a description of each setting.

tel_delay	Minimum time interval in ms between two telegrams Valid values ≥ 50 ms is recommended for approximation to an optimal parameterization.
m_tout	Maximum waiting time in ms between a transmitted telegram and the response of the addressed Modbus station. If the answer is not given in time, the answer telegram is rejected. Valid values ≥ 100 ms. The value 200 ms is recommended for approximating an optimum parameterization.
lock_count	A Modbus station is locked if the number of consecutive and unanswered telegrams is ≥ 'lock_count'. The duration of the locking time is set via 'lock_t'. Valid values ≥ 0. The value 20 is recommended to approximate an optimum parameterisation. During commissioning, a value > 500 is recommended so that sufficient time is available for the initial configuration.
lock_t	Duration of the blocking time of a Modbus station in seconds. Valid values ≥ 30 seconds is recommended to approach an optimum parameterization.

13. Now take a look at the settings of the "MOD_DP_FL" module.



```
Characteristic_lines
Mbus_modules
KNX_modules
DMX
Stopwatch
Change
Modbus
    MOD_MASTER
    MOD_DEVICE
    MOD_DP_FL
    MOD_DP_UI
    MOD_DP_ULI
    MOD_MA_EXPAN
    MOD_LIZENZ
genibus0002
PR-MAKBIB-0002a
PR-MAKBIB-0002a-Feature02
    MODFL
    MODIFL
    MODUI
    MODIUI
```

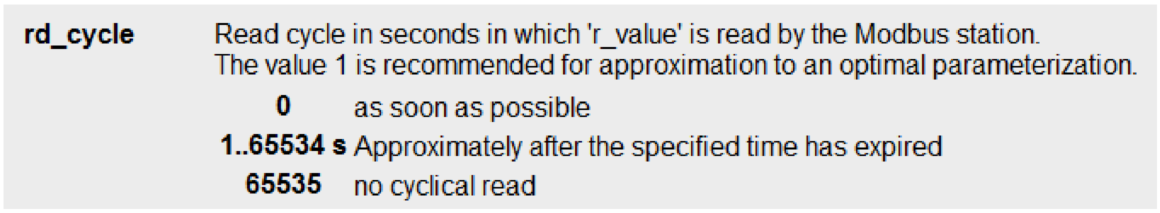
```
MOD DP FL    0
*freigabe
*dev no
*reg addr
*mem typ
*reg typ
*r w
*factor
*offset
*rd cycle
*rd trig
*wr cycle
*wr cov
*wr trig
*wr value
          *wr value r
          *e reg addr
          *rx re fram
          *rx wr fram
            *rx frames
            *tx frames
                *r value
                  *event
                  *state
```

14. The "reg_typ" setting set the different point type that we supported.

reg_typ Interpretation of the assigned Modbus register. Converts the content of the 'r_value' and the 'wr_value' into the specified format.
. The setting only affects the Holding Register and Input Register settings of the 'mem_typ' input. The necessity of the setting depends on the device. Further information can be found in the device documentation.
.

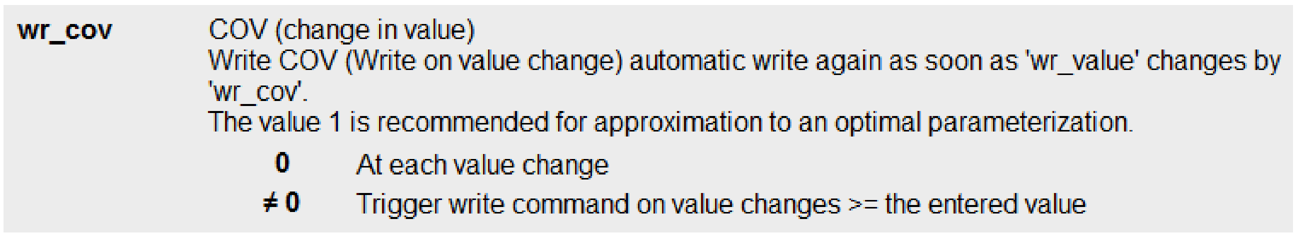
1	UI (UI16)	range of values 0 to 65535
2	SI (SI16)	range of values -32768 to +32767
3	ULI (UI32)	range of values 0 to 4294967295
4	SLI (SI32)	range of values -2147483648 to +2147483647
5	ULLI (UI64)	range of values 0 to 2ˆ64
6	SLLI (SI64)	range of values -2ˆ63 to +(2ˆ63)-1
7	SFL (FL16)	range of values -65504 to +65504
8	FL (FL32)	range of values -3.4*10ˆ38 to 3,4*10ˆ38
9	Double (FL64)	range of values -3.4*10ˆ38 to 3,4*10ˆ38 (Due to the conversion in FL)

15. Note: Do not write to the register for type 3, 4, 5, 6 and 9 using the “MOD_DP_FL” module.
16. In FUP 2, the Modbus values are always read as soon as possible, meaning that all values are read one by one sequentially. If you have many points to read, then it may become slow sometimes.



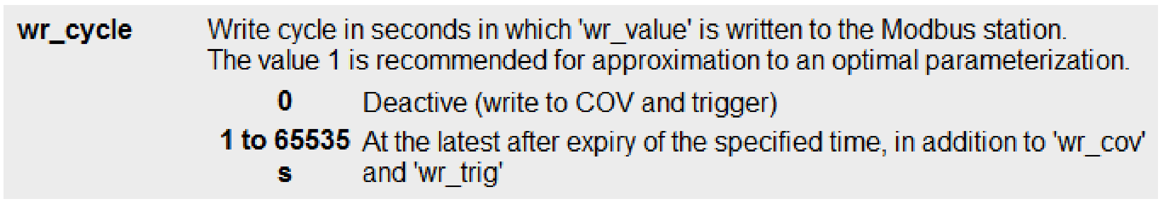
rd_cycle	Read cycle in seconds in which 'r_value' is read by the Modbus station. The value 1 is recommended for approximation to an optimal parameterization.	
	0	as soon as possible
	1..65534 s	Approximately after the specified time has expired
	65535	no cyclical read

17. Now with this new FUP module, you can set some points to read slower, and some points faster by adjusting the read cycle for each point. In this case, you can prioritize the points according to your requirements.
18. For example, you can read the “run hour” every 15 mins and the alarms every 1 or 2 sec. or set it to 0 to read the alarms as soon as possible.
19. Note: If you have many points and you set all of them to read every 2 seconds (for example), then it may overload the communication. So, if you’re unclear about the performance, it’s better to set “rd_cycle” to 0.
20. For writing, normally we use COV, meaning that the controller will write to the Modbus device when the value is change higher than “wr_cov”. For example, for temperature setpoint, we can set “wr_cov” to 0.1, and if the user changes the setpoint by more than 0.1, it is automatically written to the Modbus device.



wr_cov	COV (change in value) Write COV (Write on value change) automatic write again as soon as 'wr_value' changes by 'wr_cov'. The value 1 is recommended for approximation to an optimal parameterization.	
	0	At each value change
	≠ 0	Trigger write command on value changes >= the entered value

21. If you use write COV only, then you can send “wr_cycle” to 0. Sometime if you want to write to the Modbus device periodically, then you can specify the time in “wr_cycle”. You can have both “wr_cov” and “wr_cycle” at the same time.



wr_cycle	Write cycle in seconds in which 'wr_value' is written to the Modbus station. The value 1 is recommended for approximation to an optimal parameterization.	
	0	Deactive (write to COV and trigger)
	1 to 65535 s	At the latest after expiry of the specified time, in addition to 'wr_cov' and 'wr_trig'