

Modbus decoding

Generated on October 1, 2026

Modbus decoding

Double

`mbDouble` · `mbDouble(buffer: [], offset: int)`

Decodes a 64-bit IEEE 754 floating-point value from four consecutive registers, the first holding the highest word (byte order ABCDEFGH).

Parameters

Name	Type	Description
<code>buffer</code>	<code>[]</code>	The registers read from the device, as a list of whole numbers from 0 to 65535 - what a Modbus master's array collector writes.
<code>offset</code>	<code>int</code>	The index of the first register, counting from 0. Any whole number is accepted, a computed one such as 2^i included.

Returns `double` : The decoded number.

Examples

```
mbDouble([16393, 8699, 21572, 11544], 0) # 3.141592653589793
```

Data Orchestrator: available in plant formulas.

Float

`mbFloat` · `mbFloat(buffer: [], offset: int)`

Decodes a 32-bit IEEE 754 floating-point value from two consecutive registers, the first holding the high word (byte order ABCD, as the Modbus master's own float register type): `mbFloat([16256, 0], 0)` is 1.0.

Parameters

Name	Type	Description
<code>buffer</code>	<code>[]</code>	The registers read from the device, as a list of whole numbers from 0 to 65535 - what a Modbus master's array collector writes.
<code>offset</code>	<code>int</code>	The index of the first register, counting from 0. Any whole number is accepted, a computed one such as 2^i included.

Returns `double` : The decoded number, with the precision of a 32-bit float.

Examples

```
mbFloat(REGISTERS, 0) # 1.0
mbFloat([7, 16256, 0], 2 - 1) # 1.0
mbFloat(REGISTERS, 0.5) # error
```

Data Orchestrator: available in plant formulas.

Float (high/low)

`mbFloatHL` · `mbFloatHL(buffer: [], offset: int)`

Decodes a 32-bit IEEE 754 floating-point value from two consecutive registers, the first holding the high word, with the two bytes of each register swapped (byte order BADC): `mbFloatHL([32831, 0], 0)` is 1.0.

Parameters

Name	Type	Description
<code>buffer</code>	<code>[]</code>	The registers read from the device, as a list of whole numbers from 0 to 65535 - what a Modbus master's array collector writes.
<code>offset</code>	<code>int</code>	The index of the first register, counting from 0. Any whole number is accepted, a computed one such as $2*i$ included.

Returns `double` : The decoded number, with the precision of a 32-bit float.

Examples

```
mbFloatHL([32831, 0], 0) # 1.0
```

Data Orchestrator: available in plant formulas.

Int

`mbInt` · `mbInt(buffer: [], offset: int)`

Decodes a signed 32-bit whole number from two consecutive registers, the first holding the low word (word order CDAB): `mbInt([0, 1], 0)` is 65536.

Parameters

Name	Type	Description
<code>buffer</code>	<code>[]</code>	The registers read from the device, as a list of whole numbers from 0 to 65535 - what a Modbus master's array collector writes.
<code>offset</code>	<code>int</code>	The index of the first register, counting from 0. Any whole number is accepted, a computed one such as $2*i$ included.

Returns `int` : The decoded whole number.

Examples

```
mbInt(REGISTERS, 2) # 131071
mbInt([0, 1], 0) # 65536
```

Data Orchestrator: available in plant formulas.

Int (high/low)

`mbIntHL` · `mbIntHL(buffer: [], offset: int)`

Decodes a signed 32-bit whole number from two consecutive registers, the first holding the high word, with the two bytes of each register swapped (byte order BADC): `mbIntHL([256, 0], 0)` is 65536.

Parameters

Name	Type	Description
<code>buffer</code>	<code>[]</code>	The registers read from the device, as a list of whole numbers from 0 to 65535 - what a Modbus master's array collector writes.
<code>offset</code>	<code>int</code>	The index of the first register, counting from 0. Any whole number is accepted, a computed one such as <code>2*i</code> included.

Returns `int`: The decoded whole number.

Examples

```
mbIntHL([256, 0], 0) # 65536
mbIntHL(REGISTER, 2) # -65280
```

Data Orchestrator: available in plant formulas.

This page describes Amtiri Script 5.4.1 as shipped with Data Orchestrator engine 6.12.1 and Site 1.22.7.