ModbusCLC

Release 1.0.0

Dukkyu Lim

Jan 12, 2021

CONTENTS

CHAPTER

ONE

CONTENTS:

1.1 Installing

```
pip install modbusclc
```

1.2 Usages

You can get started fast, here you go.

1.2.1 Modbus Dummy Server

The dummy modbus server for these examples included in the repository.

```
# cloning the source code from the repository
$ git clone https://github.com/RavenKyu/modbus-command-line-client.git

# using docker if you want. the directory you are should be at the root of the␣
↪repository.
$ docker build -t dummy-modbus-server:latest -f ./dummy-modbus-server/Dockerfile .
$ docker run --rm dummy-modbus-server:latest
# or
# docker-compose would be easier.
$ docker-compose up -d dummy-modbus-server
```

1.2.2 Running ModbusCLC

```
# if you installed
$ modbusclc
# else, you better to make a virtualenv, activate it and install dependencies␣
↪according to requirements.txt
# and then,
$ python -m modbusclc
# also, you can use docker container after build it.
$ docker build -t modbusclc:latest .
$ docker run -it --rm modbusclc:latest
```

1.2.3 Options

All of the function commands has a help command. You can check how to use options supporting.

```
localhost:502> read_holding_register -h
usage:  read_holding_register [-h] [-t TEMPLATE] [-i UNIT_ID] [-v] [--b8str] [--
↪b16str] [--b32str] [--b64str] [--b8int] [--b8uint] [--b16int] [--b16uint]
                              [--b32int] [--b32uint] [--b64int] [--b64uint] [--
↪b16float] [--b32float] [--b64float] [-a ADDRESS] [-c COUNT]

optional arguments:
  -h, --help            show this help message and exit
  -t TEMPLATE, --template TEMPLATE
                        template name
  -i UNIT_ID, --unit-id UNIT_ID
                        unit id
  -v, --verbose
  --b8str
  --b16str
  --b32str
  --b64str
  --b8int
  --b8uint
  --b16int
  --b16uint
  --b32int
  --b32uint
  --b64int
  --b64uint
  --b16float
  --b32float
  --b64float
  -a ADDRESS, --address ADDRESS
                        address
  -c COUNT, --count COUNT
                        number of registers
localhost:502>
```

1.2.4 Setting Address

```
# setting the device address you connect to.
localhost:502> setting --ip 192.168.10.5 --port 502
192.168.10.5:502>
```

1.2.5 Data Type Directives

'read_holding_register' and 'read_input_register' shows data you queried in 16bit unsigned integer as default. Using 'data type directives' convert the data as readable for human instantly. It'll be useful for understanding what values of the data is.

Data Type	Directives
bits	--bits
8 bit string	--b8str
16 bit string	--b16str
32 bit string	--b32str
64 bit string	--b64str
8 Bit Signed Integer	--b8int
8 Bit Unsigned Integer	--b8uint
16 Bit Signed Integer	--b16int
16 Bit Unsigned Integer	--b16uint
32 Bit Signed Integer	--b32int
32 Bit Unsigned Integer	--b32uint
16 Bit Float	--b16float
32 Bit Float	--b32float
64 Bit Float	--b64float

```
# The data at the address 40022 is now showing it as `16bit unsigned integer`.
# But, in the datasheet of the device you use defined the data as `16bit signed␣
↪integer`.
localhost:502> read_holding_register -a40022 -c2
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         40022  cfc7      53191  -

# Using `Data Type Directives` convert it and show you them as you defined.
localhost:502> read_holding_register -a40022 -c2 --b16int
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_INT          40022  cfc7     -12345  -
localhost:502>
```

Also, *String Types* can be converted with the directives for the string type like below

```
localhost:502> read_holding_register -c8
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         40001  7765      30565  -
   1  B16_UINT         40002  6c63      27747  -
   2  B16_UINT         40003  6f6d      28525  -
   3  B16_UINT         40004  6521      25889  -

localhost:502> read_holding_register -c8 --b64str
  no  data type      address  data                value     note
----  -----------  ---------  ------------------  --------  ------
   0  B64_STRING       40001  7765 6c63 6f6d 6521  welcome!  -
localhost:502>
```

For converting the data more than one, Use the directives sequentially for the data.

```
localhost:502> read_holding_register -a40022 -c4
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         40022  cfc7      53191  -
   1  B16_UINT         40023  7b85      31621  -
```

(continues on next page)

(continued from previous page)

```
localhost:502> read_holding_register -a40022 -c4 --b16int --b8uint --b8int
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_INT          40022  cfc7     -12345  -
   1  B8_UINT          40023  7b          123  -
   2  B8_INT           40023  85         -123  -
localhost:502>
```

1.2.6 Template

This data is not applied any template and data type directives. We can add some data type directives for them. but, too many, every times.

```
localhost:502> read_holding_register -c30
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         40001  7765      30565  -
   1  B16_UINT         40002  6c63      27747  -
   2  B16_UINT         40003  6f6d      28525  -
   3  B16_UINT         40004  6521      25889  -
   4  B16_UINT         40005  4142      16706  -
   5  B16_UINT         40006  4344      17220  -
   6  B16_UINT         40007  4546      17734  -
   7  B16_UINT         40008  4748      18248  -
   8  B16_UINT         40009  ab54      43860  -
   9  B16_UINT         40010  a98c      43404  -
  10  B16_UINT         40011  eb1f      60191  -
  11  B16_UINT         40012  0ad2       2770  -
  12  B16_UINT         40013  eedd      61149  -
  13  B16_UINT         40014  ef0b      61195  -
  14  B16_UINT         40015  8216      33302  -
```

Using the template option makes it easier.

```
# `-t` : template option
# sample : template key name in the template data.

localhost:502> read_holding_register -c30 -tsample
  no  data type      address  data                value                 note
----  -----------  ---------  ------------------  --------------------  -------------
↪------
   0  B64_STRING       40001  7765 6c63 6f6d 6521  welcome!              64 bit string
   1  B32_STRING       40005  4142 4344            ABCD                  32 bit string
   2  B16_STRING       40007  4546                 EF                    16 bit string
   3  B8_STRING        40008  47                   G                     8 bit string
   4  B8_STRING        40008  48                   H                     8 bit string
   5  B64_UINT         40009  ab54 a98c eb1f 0ad2  12345678901234567890  64 bit␣
↪unsigned int
   6  B64_INT          40013  eedd ef0b 8216 7eeb  -1234567890123456789  64 bit int
   7  B32_UINT         40017  4996 02d2            1234567890            32 bit␣
↪unsigned int
   8  B32_INT          40019  b669 fd2e            -1234567890           32 bit int
   9  B16_UINT         40021  3039                 12345                 16 bit␣
↪unsigned int
  10  B16_INT          40022  cfc7                 -12345                16 bit int
```

(continues on next page)

(continued from previous page)

```
  11  B8_UINT          40023  7b                  123                   8 bit␣
↪unsigned int
  12  B8_INT           40023  85                  -123                  8 bit int
  13  B64_FLOAT        40024  419d 6f34 540c a458  123456789.01234567    64 bit float
  14  B32_FLOAT        40028  4b3c 614e           12345678.0            32 bit float
  15  B16_FLOAT        40030  64d2                1234.0                16 bit float
localhost:502>
```

The template option is very useful. It is a yaml format document described data type for the received data. The template file is located the directory name below.

```
~/.config/modbusclc/templates.yml
```

if the file or directory is not there, after running ModbusCLC once and then check it again.

Here is a sample template used above example.

```yaml
---
sample:
  - note: 64 bit string
    data_type: B64_STRING
  - note: 32 bit string
    data_type: B32_STRING
  - note: 16 bit string
    data_type: B16_STRING
  - note: 8 bit string
    data_type: B8_STRING
  - note: 8 bit string
    data_type: B8_STRING
  - note: 64 bit unsigned int
    data_type: B64_UINT
  - note: 64 bit int
    data_type: B64_INT
  - note: 32 bit unsigned int
    data_type: B32_UINT
  - note: 32 bit int
    data_type: B32_INT
  - note: 16 bit unsigned int
    data_type: B16_UINT
  - note: 16 bit int
    data_type: B16_INT
  - note: 8 bit unsigned int
    data_type: B8_UINT
  - note: 8 bit int
    data_type: B8_INT
  - note: 64 bit float
    data_type: B64_FLOAT
  - note: 32 bit float
    data_type: B32_FLOAT
  - note: 16 bit float
    data_type: B16_FLOAT
```

1.2.7 Verbose

It can also display all of the data sending both sides.

```
localhost:502> read_holding_register -c6 -v
2020-10-25 00:12:34 | send data | 192.168.200.185 > 00 01 00 00 00 06 00 03 9c 41 00␣
↪06
2020-10-25 00:12:34 | recv data | localhost:502   > 00 01 00 00 00 0f 00 03
2020-10-25 00:12:34 | recv data | localhost:502   > 0c 77 65 6c 63 6f 6d 65 21 41 42␣
↪43 44
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         40001  7765      30565  -
   1  B16_UINT         40002  6c63      27747  -
   2  B16_UINT         40003  6f6d      28525  -
localhost:502>
```

1.2.8 Read Coils (0x01)

```
# Without using the address option, the address starts from 1.
localhost:502> read_coils -c 8
  no  data type       address    data  value    note
----  ------------  ---------  ------  -------  ------
   0  BIT1_BOOLEAN          1       0  False    -
   1  BIT1_BOOLEAN          2       0  False    -
   2  BIT1_BOOLEAN          3       0  False    -
   3  BIT1_BOOLEAN          4       0  False    -
   4  BIT1_BOOLEAN          5       0  False    -
   5  BIT1_BOOLEAN          6       0  False    -
   6  BIT1_BOOLEAN          7       0  False    -
   7  BIT1_BOOLEAN          8       0  False    -
localhost:502>
```

1.2.9 Read Discrete Input (0x02)

```
# Without using the address option, the address starts from 10001.
localhost:502> read_discrete_inputs -c 8
  no  data type       address    data  value    note
----  ------------  ---------  ------  -------  ------
   0  BIT1_BOOLEAN      10001       0  False    -
   1  BIT1_BOOLEAN      10002       0  False    -
   2  BIT1_BOOLEAN      10003       0  False    -
   3  BIT1_BOOLEAN      10004       0  False    -
   4  BIT1_BOOLEAN      10005       0  False    -
   5  BIT1_BOOLEAN      10006       0  False    -
   6  BIT1_BOOLEAN      10007       0  False    -
   7  BIT1_BOOLEAN      10008       0  False    -
localhost:502>
```

1.2.10 Read Holding Register (0x03)

```
# Without using the address option, the address starts from 40001.
localhost:502> read_holding_register -c10
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         40001  7765      30565  -
   1  B16_UINT         40002  6c63      27747  -
   2  B16_UINT         40003  6f6d      28525  -
   3  B16_UINT         40004  6521      25889  -
   4  B16_UINT         40005  4142      16706  -
localhost:502>
```

1.2.11 Read Input Register (0x04)

```
# Without using the address option, the address starts from 30001.
localhost:502> read_input_register -a30003 -c10
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         30003  6f6d      28525  -
   1  B16_UINT         30004  6521      25889  -
   2  B16_UINT         30005  4142      16706  -
   3  B16_UINT         30006  4344      17220  -
   4  B16_UINT         30007  4546      17734  -
localhost:502>
```

1.2.12 Write Single Coil (0x05)

```
# before writing value
localhost:502> read_coils -a1 -c8
  no  data type       address    data  value    note
----  ------------  ---------  ------  -------  ------
   0  BIT1_BOOLEAN          1       0  False    -
   1  BIT1_BOOLEAN          2       0  False    -
   2  BIT1_BOOLEAN          3       0  False    -
   3  BIT1_BOOLEAN          4       0  False    -
   4  BIT1_BOOLEAN          5       0  False    -
   5  BIT1_BOOLEAN          6       0  False    -
   6  BIT1_BOOLEAN          7       0  False    -
   7  BIT1_BOOLEAN          8       0  False    -

# write 1 at the 3rd of coils
localhost:502> write_single_coil 3 1

localhost:502> read_coils -a1 -c8
  no  data type       address    data  value    note
----  ------------  ---------  ------  -------  ------
   0  BIT1_BOOLEAN          1       0  False    -
   1  BIT1_BOOLEAN          2       0  False    -
   2  BIT1_BOOLEAN          3       1  True     -
   3  BIT1_BOOLEAN          4       0  False    -
   4  BIT1_BOOLEAN          5       0  False    -
   5  BIT1_BOOLEAN          6       0  False    -
```

(continues on next page)

(continued from previous page)

```
   6  BIT1_BOOLEAN          7       0  False    -
   7  BIT1_BOOLEAN          8       0  False    -
localhost:502>
```

1.2.13 Write Single Register (0x06)

```
# before writing value
localhost:502> read_holding_register -c4
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         40001  7765      30565  -
   1  B16_UINT         40002  6c63      27747  -

# write a integer -999(0xfc19) at the register 40002
localhost:502> write_single_register 40002 --b16int -999

localhost:502> read_holding_register -c4
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         40001  7765      30565  -
   1  B16_UINT         40002  fc19      64537  -
localhost:502>
```

- Writing-Single-Register is allowed to write just for one register value only.
- The values out of range in the data type is not able to write.

Data Type	Directives	Examples	Description
String	--string	--string AB	You can put strings as much as 2 bytes
16 Bit Signed Integer	--b16int	--b16int -999	It allows only within 2 bytes much signed integer
16 Bit Unsigned Integer	--b16uint	--b16uint 65535	It allows only within 2 bytes much unsigned integer
16 Bit Float	--b16float	--16float 3.14	

1.2.14 Write Multiple Coils (0x0F)

```
# before writing values
localhost:502> read_coils -c8
  no  data type       address    data  value    note
----  ------------  ---------  ------  -------  ------
   0  BIT1_BOOLEAN          1       0  False    -
   1  BIT1_BOOLEAN          2       0  False    -
   2  BIT1_BOOLEAN          3       1  True     -
   3  BIT1_BOOLEAN          4       0  False    -
   4  BIT1_BOOLEAN          5       0  False    -
   5  BIT1_BOOLEAN          6       0  False    -
   6  BIT1_BOOLEAN          7       0  False    -
   7  BIT1_BOOLEAN          8       0  False    -

# writing the binaries from the start address to as many as the length of the value
localhost:502> write_multiple_coils 1 01101100

localhost:502> read_coils -c8
```

(continues on next page)

(continued from previous page)

```
  no  data type       address    data  value    note
----  ------------  ---------  ------  -------  ------
   0  BIT1_BOOLEAN          1       0  False    -
   1  BIT1_BOOLEAN          2       1  True     -
   2  BIT1_BOOLEAN          3       1  True     -
   3  BIT1_BOOLEAN          4       0  False    -
   4  BIT1_BOOLEAN          5       1  True     -
   5  BIT1_BOOLEAN          6       1  True     -
   6  BIT1_BOOLEAN          7       0  False    -
   7  BIT1_BOOLEAN          8       0  False    -
localhost:502>
```

1.2.15 Write Multiple Registers (0x10)

```
# before writing values
localhost:502> read_holding_register -c6
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         40001  7765      30565  -
   1  B16_UINT         40002  fc19      64537  -
   2  B16_UINT         40003  6f6d      28525  -

localhost:502> write_multiple_registers 40001 --b32uint 123456789 --string AB

localhost:502> read_holding_register -c6
  no  data type      address  data      value  note
----  -----------  ---------  ------  -------  ------
   0  B16_UINT         40001  075b       1883  -
   1  B16_UINT         40002  cd15      52501  -
   2  B16_UINT         40003  4142      16706  -
localhost:502>
```

Data Type	Directives	Examples	Description
String	--string	--string AB	You can put strings as much as 2 bytes
bits	--bits	--bits 1110 => 00001110 or 1111000010101010 or "1111 00 11"	
8 Bit Signed Integer	--b8int	--b8int -128	It allows only within 1 bytes much signed integer
8 Bit Unsigned Integer	--b8uint	--b8uint 255	It allows only within 1 bytes much unsigned integer
16 Bit Signed Integer	--b16int	--b16int -999	It allows only within 2 bytes much signed integer
16 Bit Unsigned Integer	--b16uint	--b16uint 65535	It allows only within 2 bytes much unsigned integer
32 Bit Signed Integer	--b32int	--b32int -2147483648	It allows only within 4 bytes much signed integer
32 Bit Unsigned Integer	--b32uint		It allows only within 4 bytes much unsigned integer
16 Bit Float	--b16float	--16float 3.14	
32 Bit Float	--b32float	--32float 3.14	
64 Bit Float	--b64float	--64float 3.14	

- **Usages**
  - Reading registers and coils
  - Writing registers and coils
  - Templates for data structures

```
sudo apt install modbusclc
```

```
pip install modbusclc
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search