Gerekli Kurulumların Yapılması :

Ben Ubuntu ortamında çalışma yaptım fakat Windows ortamında da kurulumu mümkündür. Windows tarafında çalıştırılacak ise Anaconda 'nın kullanılması ve DLL'lerin Path'e eklenmesi unutulmamalıdır.

Ubuntu ortamında hazırlamış olduğum runscripth.sh komutunu aşağıdaki görseldeki gibi çalıştırabilirsiniz.

```
ebb@ebb-VirtualBox: ~/Desktop/ebb
ebb@ebb-VirtualBox:~/Desktop/ebb$ ls
runscript.sh
ebb@ebb-VirtualBox:~/Desktop/ebb$ chmod +x runscript.sh
ebb@ebb-VirtualBox:~/Desktop/ebb$ ./runscript.sh
Ebb Simple Snap-7 Requirements Installation
 Python installation starting ..
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'python3.6-2to3' for regex 'python3.6'
Note, selecting 'libpython3.6-stdlib' for regex 'python3.6'
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
Python installed version 3.6 , If you use snap-7 , you have
 to python version 3.6+ version
0% [Connecting to tr.archive.ubuntu.com] [Connecting to security.ubuntu.com]
```

Temel mantıkta arka planda Profinet ve Byte dönüşümlerinin tamamını Snap-7 yapacaktır.

PLC'ye bağlanmak için biz client olarak davranacağımız için bir client objesi oluşturuyoruz :

```
create()    [source]
```

Creates a SNAP7 client.

Farklı şekillerde client oluşturabileceğimiz gibi clientı oluşturup parametreleri daha sonra ayarlamak daha efektif olacaktır . Burada PLC'nin çalışma Ipsi ve tsapleri ayarlanabilir.

set_connection_params(*address: str, local_tsap: int, remote_tsap: int*) [source]

Sets internally (IP, LocalTSAP, RemoteTSAP) Coordinates. This function must be called just before Cli_Connect().

Parameters:
- **address** – PLC/Equipment IPV4 Address, for example "192.168.1.12"
- **local_tsap** – Local TSAP (PC TSAP)
- **remote_tsap** – Remote TSAP (PLC TSAP)

Şimdi ayarlamalardan sonra bağlanmak için Eğer

get_connected() → bool [source]

Returns the connection status

Note

Sometimes returns True, while connection is lost.

Returns: True if is connected, otherwise false.

Eğer bağlantıyı sonlandırmak istersek ise :

destroy() → Optional[int] [source]

Destroys the Client object.

Returns: Error code from snap7 library.

Examples

```
>>> client.destroy()
640719840
```

İle sonlandırabiliriz bunu programın sonunda yapabiliriz eğer sonlandırmasak da program bitince otomatik kapanacaktır.

C'deki long , real , boolean vb veri tipleri pythonda otomatik cast edildiği için ve bunların dönüşümlerini manuel yapmamız gerekmekte bunun için python içinde default olarak gelen cytpe kullanıyorum.

Fundamental data types

`ctypes` defines a number of primitive C compatible data types:

ctypes type	C type	Python type
`c_bool`	`_Bool`	bool (1)
`c_char`	`char`	1-character bytes object
`c_wchar`	`wchar_t`	1-character string
`c_byte`	`char`	int
`c_ubyte`	`unsigned char`	int
`c_short`	`short`	int
`c_ushort`	`unsigned short`	int
`c_int`	`int`	int
`c_uint`	`unsigned int`	int
`c_long`	`long`	int
`c_ulong`	`unsigned long`	int
`c_longlong`	`__int64` or `long long`	int
`c_ulonglong`	`unsigned __int64` or `unsigned long long`	int
`c_size_t`	`size_t`	int
`c_ssize_t`	`ssize_t` or `Py_ssize_t`	int
`c_float`	`float`	float
`c_double`	`double`	float
`c_longdouble`	`long double`	float
`c_char_p`	`char *` (NUL terminated)	bytes object or `None`
`c_wchar_p`	`wchar_t *` (NUL terminated)	string or `None`
`c_void_p`	`void *`	int or `None`

1. The constructor accepts any object with a truth value.

Diğer kullandığım ve kodda açıklamasını yaptığım fonksiyonların detaylı anlatımlarını içeren dokümantasyondan aldığım kısımları ve detaylı kullanımlarını içeren görseller aşağıdaki gibidir.

get_block_info(*blocktype: str, db_number: int*) → snap7.types.TS7BlockInfo [source]

Returns detailed information about a block present in AG.

Parameters:
- **blocktype** – specified block type.
- **db_number** – number of db to get information from.

Returns: Structure of information from block.

Raises: `Snap7Exception` – if the *blocktype* is not valid.

Examples

```
>>> block_info = client.get_block_info("DB", 1)
>>> print(block_info)
Block type: 10
Block number: 1
Block language: 5
Block flags: 1
MC7Size: 100
Load memory size: 192
Local data: 0
SBB Length: 20
Checksum: 0
Version: 1
Code date: b'1999/11/17'
Interface date: b'1999/11/17'
Author: b''
Family: b''
Header: b''
```

db_read(*db_number: int, start: int, size: int*) → bytearray [source]

Reads a part of a DB from a PLC

> **Note**
>
> Use it only for reading DBs, not Marks, Inputs, Outputs.

Parameters:

- **db_number** – number of the DB to be read.
- **start** – byte index from where is start to read from.
- **size** – amount of bytes to be read.

Returns: Buffer read.

Example

```
>>> import snap7
>>> client = snap7.client.Client()
>>> client.connect("192.168.0.1", 0, 0)
>>> buffer = client.db_read(1, 10, 4)  # reads the db number 1 starting
>>> buffer
bytearray(b'\x00\x00')
```

db_get(*db_number: int*) → bytearray [source]

Uploads a DB from AG using DBRead.

> **Note**
>
> This method can't be use for 1200/1500 PLCs.

Parameters: **db_number** – db number to be read from.

Returns: Buffer with the data read.

Example

```
>>> import snap7
>>> client = snap7.client.Client()
>>> client.connect("192.168.0.1", 0, 0)
>>> buffer = client.db_get(1)  # reads the db number 1.
>>> buffer
bytearray(b"\x00\x00\x00\x00\x00\x00\x00\x00...<truncated>\x00\x00")
```

as_db_read(*db_number: int, start: int, size: int, data*) → _ctypes.Array [source]

Reads a part of a DB from a PLC.

Parameters:
- **db_number** – number of DB to be read.
- **start** – byte index from where start to read from.
- **size** – amount of bytes to read.
- **data** – buffer where the data read will be place.

Returns: Snap7 code.

Examples

```
>>> import ctypes
>>> data = (ctypes.c_uint8 * size_to_read)()  # In this ctypes array dat
>>> result = client.as_db_read(1, 0, size_to_read, data)
>>> result  # 0 = success
0
```

as_db_write(*db_number: int, start: int, size: int, data*) → int [source]

Writes a part of a DB into a PLC.

Parameters:
- **db_number** – number of DB to be write.
- **start** – byte index from where start to write to.
- **size** – amount of bytes to write.
- **data** – buffer to be write.

Returns: Snap7 code.

A snap7 client

Examples

```
>>> import snap7
>>> client = snap7.client.Client()
>>> client.connect("127.0.0.1", 0, 0, 1012)
>>> client.get_connected()
True
>>> data = client.db_read(1, 0, 4)
>>> data
bytearray(b"\x00\x00\x00\x00")
>>> data[3] = 0b00000001
>>> data
bytearray(b'\x00\x00\x00\x01')
>>> data.db_write(1, 0, data)
```

__init__(*lib_location: Optional[str] = None*) [source]

Creates a new *Client* instance.

Parameters: **lib_location** – Full path to the snap7.dll file. Optional.

Examples

```
>>> import snap7
>>> client = snap7.client.Client()  # If the `snap7.dll` file is in the path location
>>> client = snap7.client.Client(lib_location="/path/to/snap7.dll")  # If the `snap7.dll` file i
>>> client
<snap7.client.Client object at 0x0000028B257128E0>
```