

CSI OT 3D Platform Cyber Attack Demonstration PLC Control

Setup Manual

VERSION: CORPLAB-2021-T5F-A1

30/03/2021

Prepared by

Liu Yuancheng

Senior Security Development Engineer

<yuancheng.liu@trustwave.com>

Wong Jun Wen

Asst R&D Manager

<junwen.wong@trustwave.com>

Dr. Shantanu Chakrabarty

NUS Research Fellow

<shantanu1088@gmail.com>

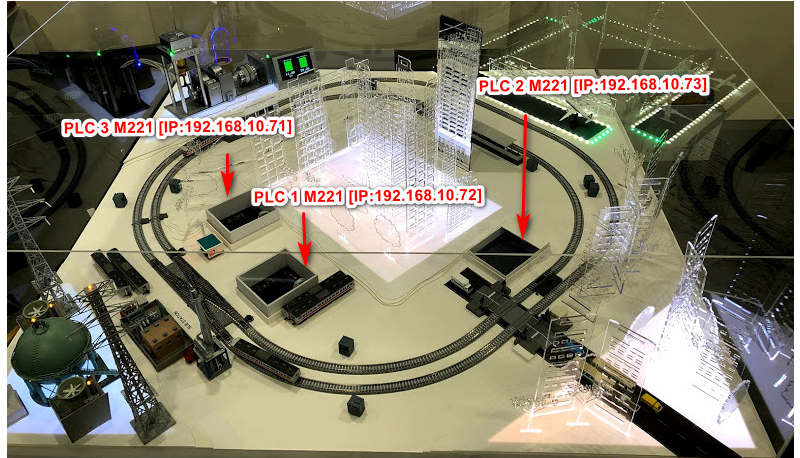
Confidentiality Notice: This document is confidential and contains proprietary information and intellectual property of Trustwave, a Singtel company, its parent company and its subsidiaries. Neither this document nor any of the information contained herein may be reproduced or disclosed under any circumstances. Please be aware that disclosure, copying, distribution or use of this document and the information contained therein is strictly prohibited.

**CSI OT 3D Platform Cyber Attack Demonstration PLC Control and Setup Manual**

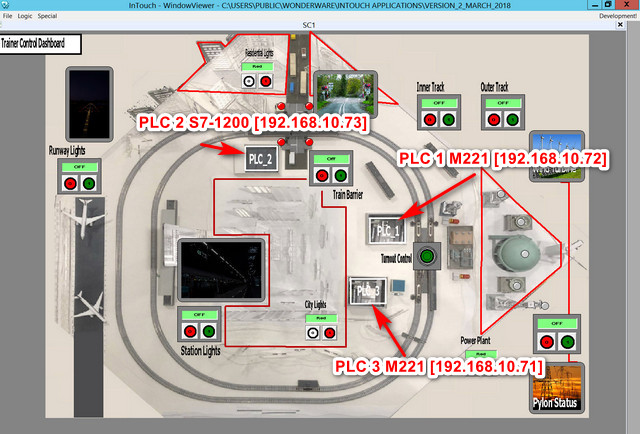
**1. Project Introduction**

This document will introduce how to program and control the Schneider M221 and the Siemens Simatic S7-1200 programmable logic controller (PLC) which are used in the CSI OT 3D Cyber Attack Demonstration Platform. The software Schneider PLC SoMachine Basic, Schneider Wonderware Maker.Viewer and Python3.7 will be used for the PCL control in this section.

PLC position view on the 3D platform and HMI: (remake the pic as the label M221 is wrong , should be s7-1200)



HMI User Interface on SCADA PC:



**1.1 PCL Coils Wires Connection to the System Components**

1.1.1 Siemens S7-1200 PLC Front View and Connection:

PLC 2 [Seimens S7-1200]

IP Address: 192.168.10.73

Modbus TCP port: 502

PLC Output Coils Connection:

[Q0] Qx0.0-> Q0.0 Station + sensor.

[Q1] Qx0.1-> Q0.1 level crossing power.

[Q2] Qx0.2-> Q0.2 Resident LED.

[Q3] Qx0.3-> Q0.3 generator Moto speed control idx0.

[Q4] Qx0.4-> Q0.4 generator Moto speed control idx1.

[Q5] not connected.

[Q6] not connected.

[Q7] not connected.

* S7-1200 PLC Product Information link: https://euroec.by/assets/files/siemens/s71200\_easy\_book\_en-US\_en-US.pdf

1.1.2 Schneider M221 Front View and Connection:

 PLC 1 [schneider M221]:

IP Address: 192.168.10.72

Modbus TCP port: 502

PLC Output Pin Connection:

[Q0] M10 -> Q0.0 Airport LED.

[Q1] M0 -> Q0.1 Power Plant.

[Q2] M60 -> Q0.2 Industrial LED.

[Q3] not connected.

[Q4] M4 -> Q0.4 generator pump speed control idx0.

[Q5] M5 -> Q0.5 generator pump speed control idx1.

[Q6] not connected.

[Q7] not connected.



PLC 3 [schneider M221]:

IP Address: 192.168.10.71

Modbus TCP port: 502

PLC Output Pin Connection:

[Q0] M0 -> Q0.0 fork turnout.

[Q1] M10 -> Q0.1 track A power.

[Q2] M20 -> Q0.2 track B power.

[Q3] M60 -> Q0.3 city LED.

[Q4] M4 -> Q0.4 Inner Train position sensor’s set power.

[Q5] M5 -> Q0.5 Outer Train position sensor’s set power.

[Q6] M50 -> Q0.6 System power switch.

[Q7] not connected.

* Schneider M221 PLC Production Information link: Prohttps://euroec.by/assets/files/siemens/s71200\_easy\_book\_en-US\_en-US.pd

**2. Schneider** **M221 PLC Setup**

This section will introduce how to use the SoMachine Basic, Schneider Wonderware Maker and python to connect and control the M221 PLC.

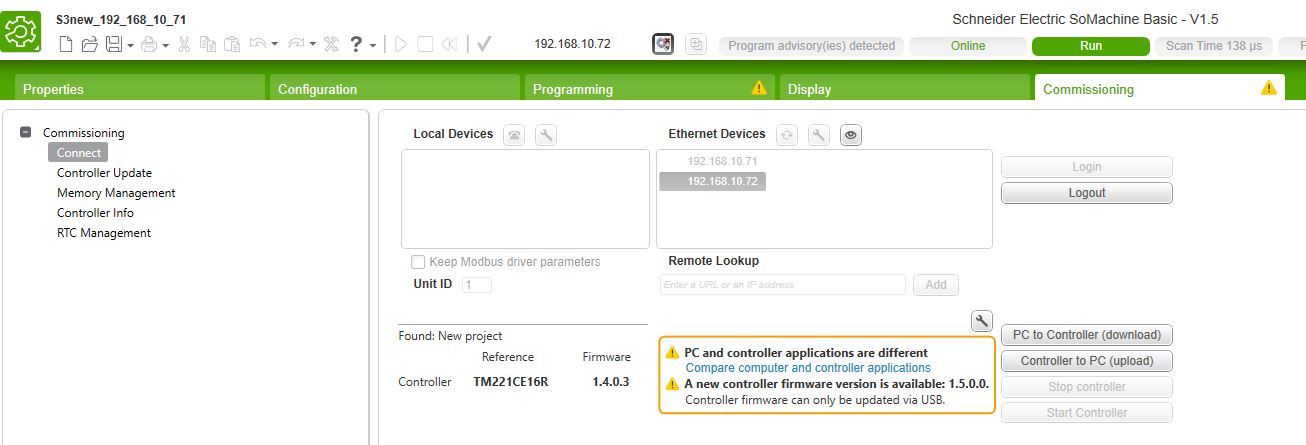
**2.1. Connect the PLC and control it by SoMachine Basic.**

Use Schneider SoMachine Basic, Schneider Wonderware Maker to control M221 PLC:

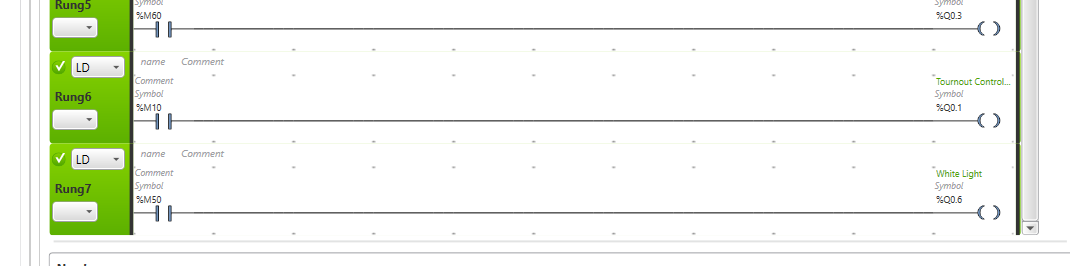
Download the SoMachine Basic 1.5 version and edit the ladder diagram:

<https://download.schneider-electric.com/files?p_File_Name=EIO0000001354.10.pdf>

* Connect (login) to the PLC and select “Controller to PC download” to load the PLC ladder diagram to the SoMachine Basic for further editing:

****

* Add one ladder diagram and set the register and output coils under the program page:



* After finished edit, press the “PC to Controller” and “Start controller” to upload the ladder diagram to the PLC for further testing.

Follow this manual to read and write related PLC register from the Wonderware Maker software: https://industrial-software.com/wp-content/uploads/SoMachine-OI-Server-G-2.1-Version-3.1-Help.pdf

**2.2 Connect the PLC and control it by Python program.**

Use the file module **M2PLC221.py** to connect the PLCs by TCP with port number 502. (The module will ping the PLC IP address first then do the connection initialization) Call **readMem()** function to read in the PLC coils output state string. Use the **writeMem(<register Idx(M##)>, <state(0/1)>)** function to setup the PLC coils state.

Related PLC setting link ( M221 function code):

* <https://www.schneider-electric.com/en/faqs/FA308725/>
* <https://www.schneider-electric.com/en/faqs/FA295250/>
* <https://www.schneider-electric.com/en/faqs/FA249614/>

Logic to check the related feedback string’s bytes to read the M221 PLC output coils states:

**2.3 Recover from the attack situation**.

#----------------------------------------------------------------------------- | Full Modbus TCP mem read message format:

M221 - S1(192.168.10.72) | TID = '0000'

Coils off: 08 01 00 00 40 00 00 00 10 | PROTOCOL\_ID = '0000'

Industry on: 08 01 00 00 40 00 00 00 [00] | R\_LENGTH = '0006' # read byte length

Airport on: 08 01 [04] 00 40 00 00 00 10 | UID = '01'

#----------------------------------------------------------------------------- | BYTE\_COUNT = '01'

M221 - S3(192.168.10.71) | M\_RD = '01' # mem state fetch internal bits %M

Coils off: 08 31 00 00 00 00 00 10 | PROTOCOL\_ID = '0000'

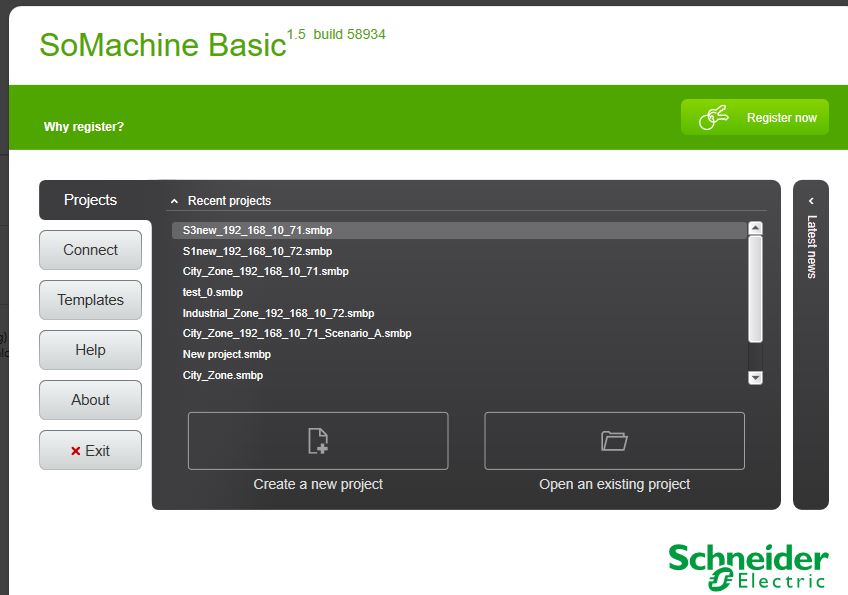
Track A on: 08 31 [04] 00 00 00 00 10 | FUNCTION\_CD= '003d'

Track B on: 08 31 00 [10] 00 00 00 10 |

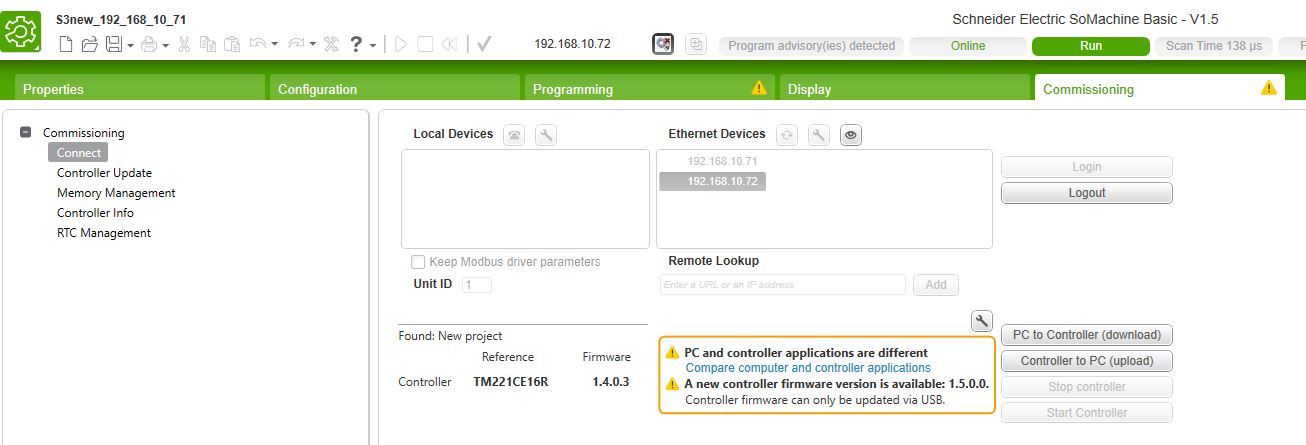
City on: 08 31 00 00 00 00 00 [00] |

# M221 function code link: https://www.se.com/ww/en/faqs/FA308725/

Some attack scenario will change the M221 PLC’s ladder diagram program, to recover the original ladder diagram, open the related \*.smbp file and connect to the PLC. Then press the “PC to controller button” as shown below.



After finished edit, press the “PC to Controller” to upload the ladder diagram to the PLC.

****

PLC 1 [schneider M221 IP Address: 192.168.10.72]: S1new\_192\_168\_10\_72.smbp

PLC 3 [schneider M221 IP Address: 192.168.10.71]: S3new\_192\_168\_10\_71.smbp

**3. Siemens S7-1200 PLC Setup**

The S7-1200 we do not need to edit the ladder diagram; we can use the Modbus TCP message to control/read the output coils directly. But the program needs to also install the related lib **snap7**. Snap7 official we side: <https://pypi.org/project/python-snap7/>

**3.1 Connect to the Siemens S7-1200 with Python**

3.1.1 Install Snap 7 on Linux System

If you have already installed the pip, run cmd: **pip install python-snap7** or follow the full installation steps with below cmd set:

#download and compile snap7 for rpi

wget http://sourceforge.net/projects/snap7/files/1.2.1/snap7-full-1.2.1.tar.gz/download

tar -zxvf snap7-full-1.2.1.tar.gz

cd snap7-full-1.2.1/build/unix

sudo make –f arm\_v6\_linux.mk all

#copy compiled library to your lib directories

sudo cp ../bin/arm\_v6-linux/libsnap7.so /usr/lib/libsnap7.so

sudo cp ../bin/arm\_v6-linux/libsnap7.so /usr/local/lib/libsnap7.so

#install python pip if you don't have it:

sudo apt-get install python-pip

sudo pip install python-snap7

After you have installed the snap7 lib on Linux by pip/pip3, there will be a running error if you want to run it on raspberry PI (arm CPU), please follow below step to solve the error. You will need to edit the **lib\_location** function on **common.py** in the **/usr/local/lib/python2.7(python3.x)/dist-packages/snap7/** directory.

Add a line in the **\_\_init\_\_** part of the Snap7Library class: **lib\_location='/usr/local/lib/libsnap7.so'**

class Snap7Library(object):

"""

Snap7 loader and encapsulator. We make this a singleton to make

sure the library is loaded only once.

"""

\_instance = None

def \_\_new\_\_(cls, \*args, \*\*kwargs):

if not cls.\_instance:

cls.\_instance = object.\_\_new\_\_(cls)

cls.\_instance.lib\_location = None

cls.\_instance.cdll = None

return cls.\_instance

def \_\_init\_\_(self, lib\_location=None):

lib\_location='/usr/local/lib/libsnap7.so' # add this line here

if self.cdll:

return

self.lib\_location = lib\_location or self.lib\_location or find\_library('snap7')

if not self.lib\_location:

msg = "can't find snap7 library. If installed, try running ldconfig"

raise Snap7Exception(msg)

self.cdll = cdll.LoadLibrary(self.lib\_location)

* Reference link: http://simplyautomationized.blogspot.com/2014/12/raspberry-pi-getting-data-from-s7-1200.html

3.1.2 Install snap7 on MS-Windows10 Sysetm:

If you have already installed the pip, run cmd: **pip install python-snap7**

Setup snap7 Used \*.dll and lib file system environment path otherwise you may get the error during running time as shown below:

To solve this problem, you should download the \*.dll file from this link (based on your system 32bits or 64 bits): <https://github.com/LiuYuancheng/Power_Generator_Manager/tree/master/lib/Windows>

Traceback (most recent call last):

File "Y:\Lonnox\Projekte\Bibliothek\Python und SPS\S7-1200 Test.py", line 6, in <module>

plc = snap7.client.Client()

File "C:\Python34\lib\site-packages\snap7\client.py", line 30, in \_\_init\_\_

self.library = load\_library()

File "C:\Python34\lib\site-packages\snap7\common.py", line 54, in load\_library

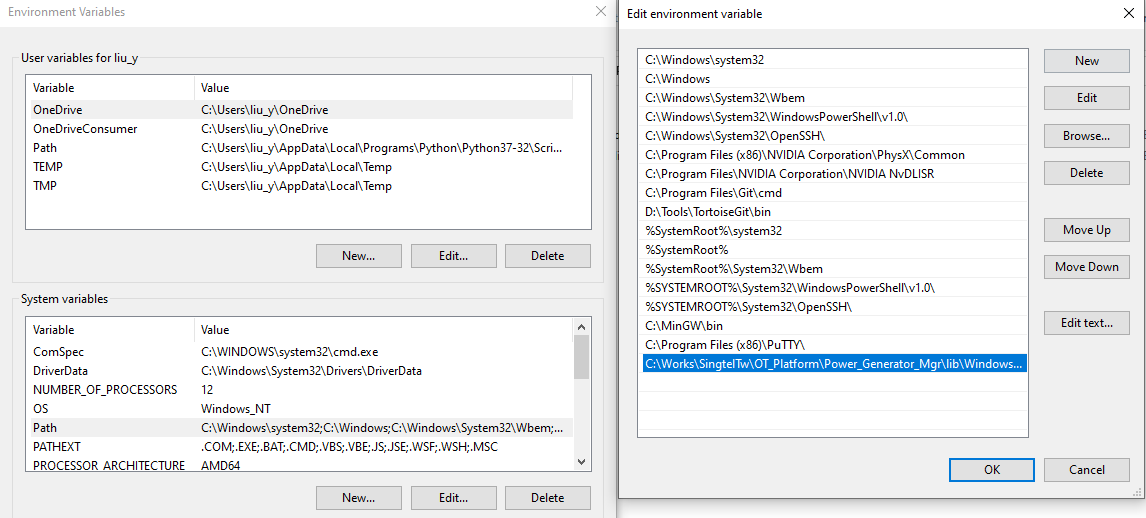
return Snap7Library(lib\_location).cdll

File "C:\Python34\lib\site-packages\snap7\common.py", line 46, in \_\_init\_\_

raise Snap7Exception(msg)

snap7.snap7exceptions.Snap7Exception: can't find snap7 library. If installed, try running ldconfig

Added the \*dll file to the path as show below: Computer > System Property >Advanced system settings > Environment Variables > System variable> New



* Reference link: https://stackoverflow.com/questions/33697263/python-snap7-windows-cant-find-snap7-library

**3.2. Connect the PLC and control it by Python program**

Use the file module **S7PLC1200.py** to connect the PLCs by TCP with port num 502. Call **getMem(self, mem, returnByte=False)** function to read in the PLC coils output state string. Use the **writeMem(self, mem, value)** function to setup the PLC coils state. During turn on/off one coil of the PLC, leave a short time interval such as 0.01 sec to let the PLC finish the signal change progress.

**4. Program File Structure**

All the related sources code/file can be download from:

<https://github.com/LiuYuancheng/Power_Generator_Manager>

Program File list:

| **Program File** | **Execution Env** | **Description** |
| --- | --- | --- |
| M2PLC221.py | python 2.7/3.x | This module is used to connect to the Schneider M2xx PLC. Run the module direct can check the testcase. |
| S7PLC1200.py | python 2.7/3.x | This module is used to connect to the siemens s7-1200 PLC. Run the module direct can check the testcase. |
| S1new\_192\_168\_10\_72.smbp | SoMachine Basic | PLC-1 ladder diagram. |
| S3new\_192\_168\_10\_71.smbp | SoMachine Basic | PLC-3 ladder diagram. |

**5 Reference.**

<https://www.se.com/sg/en/product-range/62128-logic-controller---modicon-m221/>

<https://new.siemens.com/global/en/products/automation/systems/industrial/plc/s7-1200.html>

End (last edited 08/04/2021)