

IEC 104 RTU Simulator – Ver 1.0

By M. Medhat

Contents

Introduction and why I wrote this program	2
Program arguments	3
Program operation.....	4
Initial file format	6
IOA signal database file format.....	7
Program GUI.....	8
Troubleshooting.....	9
Appendix A - Sample initial file	11
Appendix B - Sample IOA database file.....	13
Appendix C – GUI screenshots	17
Appendix D – Windows binary files	19
Appendix D - Other projects	19

If this project help you reduce time to develop, please spend some of your time to check link below:

The Fog is Lifting video

<https://www.youtube.com/watch?v=bdH16Hz1naI>

The Fog is Lifting' is one of the best holistic presentations a non Muslim can watch about Islam to learn about: its meaning, its beliefs and its concepts.

Introduction and why I wrote this program

IEC 104 RTU simulator is a program to simulate the operation of RTU (remote terminal unit) or server as defined by protocol IEC 60870-5-104. It can simulate any number of RTUs or servers. Simulated RTUs could be connected to different or same SCADA master station. IO signals are indexed and grouped by using index numbers. You can send IO signals from all RTUs to the connected SCADA master stations at once by using index number.

Program features:

- Simulation of any number of RTUs simultaneously.
- Connect to multiple SCADA systems at once.
- Can simulate redundant RTU ports. Program will send same IO signal to all RTUs with same RTU number.
- Can simulate redundant SCADA system connections.
- Easy IO database building by using spreadsheet programs such as MS Excel.
- IP address and network filtration for each RTU independently.
- Can repeat sending grouped (by index number) of IO signals for any period (in seconds). Also, a delay time in seconds could be applied after sending each IO.
- Can send any IO signal based on receiving filter conditions such as specific type ID, IOA, etc.
- Accordingly, could be used to simulate any site or factory acceptance tests (SAT/FAT) such as:
 - o End to end (E2E) test.
 - o Avalanche test (sending full IO signal list from all defined RTUs by using index 0).
 - o Worst case (SCADA system emergency case).
 - o Steady state test (normal operation).
 - o SCADA system switchover time.
 - o SCADA system time synchronization.
 - o Load shedding.
 - o FLISR (fault location, isolation, and service restoration).
 - o Self-healing.
 - o Etc.
- IEC 104 parameters (timing and k) for each RTU independently.
- Linux and windows compatible.
- Python native graphical user interface (GUI) (no need for third party solutions).
- Multithread operation.

- Time synchronization through multiple NTP servers.
- Detailed logging of all events and signals in comma delimited file format.

I wrote this program when at my company we replaced our legacy ABB SCADA system with new OSI SCADA system, so we wanted to test the new system in comparison with the old legacy system without taking shutdown on the power stations.

So, by indexing the IO signals in the IOA database files we succeeded to send same IO signals to both SCADA systems (old and new) at the same time and compare between them:

- IOA address of the signals.
- Signals' values for SPI, DPI, AMI.
- Testing commands and send back the status to both systems.
- Compare time tags and time synchronization between the two systems.
- Test IEC 104 protocol on the new system such as startdt, stopdt, GI, time tags, etc.
- Measure test frame period. For that reason, the simulator will not send "Testfr act" but it will wait until receiving "testfr act" from the connected SCADA system and will reply by "testfr con" and log the test frame period in the RTU log file.

Program is distributed under GPL license and could be found on GitHub:

<https://github.com/med7at69/IEC104-RTU-Simulator>

It is written in python3 language and code is supporting both Windows and Linux OS.

Package contains the following files:

iec104rs.py: The code in python 3 language.

iec104rs.csv: ini file in comma separated values. Must be in the same folder where program starts in.

"data" folder with samples "iodata.csv" files which are IOA data files in comma separated values. Must be kept in "data" folder. "data" folder must be in the same folder where program starts in.

iec104rs.pdf: Help file in pdf format.

Readme.txt

LICENSE file.

Program arguments

-h or --help display help message.

-i or --ini	specify init file name.
-t or --ntp_update_every_sec	NTP update interval (seconds). Default = 900 sec.
-s or --ntp_server	NTP server (may repeated for multiple servers).

Usage:

iec104rs [[-h][--help]] [[-i][--ini] init-file] [[-t][--ntp_update_every_sec seconds] sec] [[-s][--ntp_server ntpserver] server]

- Updating local time requires admin/root privilege.
- init file is a comma separated values format, default: iec104rs.csv
- “-s or --ntp_server” could be included multiple times for multiple servers.

example1:

```
iec104rs -i iec104rs1.csv
```

example2:

```
iec104rs --ntp_server pool.ntp.org --ntp_server time.windows.com
```

Program operation

Program operation depends on providing huge data either for RTUs that the program to simulate or for the IOA signals provided for each RTU. To make the program operation as simple as possible I tried to use the comma separated values file (csv) format to provide the data to the program for the following reasons:

- 1- “csv” format is simple and well known since long time.
- 2- Besides supported by Microsoft Excel, there are many freeware programs supporting editing “csv” files.
- 3- It is easy to add, delete, copy, and paste large number of data entries to the “csv” files without complications.

Program operation is based on the following files:

- 1- Initial file (default name is iec104rs.csv): It is a comma separated values file format or “.csv” which should be available in the same folder where program starts in. In the file you can define the following:
 - a. NTP servers to update local time of the PC (requires admin/root privilege).
 - b. Number of seconds to periodically update local time from NTP servers.

- c. Any number of RTUs or systems to be simulated by the program.
- 2- IOA database file (for example iodata.csv): It is a comma separated values file format or “.csv” which should be in folder “data”. Folder “Data” should be in the same folder where the program starts in. In the file you can define the IOA signals such that SPI, DPI, NVA, SVA, FLT, SCO, DCO, RCO. Each RTU may have separated IOA data file or multiple RTUs can share the same IOA signals data file. In the IOA database file, IO signals are indexed and grouped by index numbers. You can send IO signals from all RTUs to the connected SCADA master stations at once by using index numbers. For monitoring IO (SPI, DPI, AMI) a filter conditions could be added, and program will wait (for time out in ‘sec’ decimal point value) to receive filter conditions before sending the IO signal. Also, a delay time in seconds could be applied after sending each IO. Filter conditions will not be used during sending “GI” signals or command reply.
- 3- Log files for all RTUs are saved in folder “log”. Folder “log” will be created in the same folder where the program starts in.

When the program starts it will:

- 1- Read the program arguments if provided by user.
- 2- If initial file is not provided in program arguments, then the program will use the default iec104rs.csv
- 3- Read the initial file to get the NTP servers and RTUs as described later.
- 4- Each RTU should have name, RTU number, port number to listen on and IOA data file (in "data" folder).
- 5- To speed up the loading of RTUs, program will not start any RTU connection until load all RTUs in memory.
- 6- If user type index number and press “send” button, then the program will send all IO signals grouped by this index number for typed time duration in seconds in all IOA database files of different RTUs to the connected master stations. This function is helpful especially when you want to compare between two SCADA master stations, for example one new and the other is the legacy working system.
- 7- If index typed is “0” then program will send all IOA to all connected RTUs for the typed time duration at once.
- 8- If any monitoring IO (SPI, DPI, AMI) in the IOA database csv file has filter conditions defined then the program will wait (for time out in ‘sec’ decimal point value) to receive filter conditions before sending the IO signal. Also, if a delay time in seconds is defined then it will be applied after sending each IO.
- 9- Filter conditions will not be used during sending “GI” signals or command reply.
- 10- The simulator will not send “Testfr act” but it will wait until receiving “testfr act” from the connected SCADA system and will reply by “testfr con” and log the test frame period in the RTU log file.
- 11- If idle time (configured in the initial file for each RTU in seconds) passed without send/receive data, then the program simulator will disconnect the connection to restart working connection again.

Initial file format

General notes:

- Initial file format is comma separated values format (csv).
- Initial file default name is iec104rs.csv
- You can provide another name as program argument with "-i" or "--ini"
- Initial file should be in the same folder where the program starts in.
- If first character of first column in any row is "!" Then program will stop reading the initial file and cancel the rest of the rows.
- Initial file will start by defining the following parameters:
 - o "ntp_server": it could be repeated in multiple rows for multiple NTP servers. If program has admin/root privilege then it will try all the servers one by one to synchronize the local time.
 - o "ntp_update_every_sec": seconds to periodically update local time. If not provided, then the default is 900 seconds.
- Then initial file should contain all RTUs information one by one (each row contains one complete RTU information) such that each RTU is define the following parameters in separated rows:
 - o ID:
 - Better to be a unique sequential number for each RTU.
 - If "id" field is not number, then it will be considered as comment and will be neglected by the program.
 - If first character of "id" column in any row is "!" Then program will stop reading the rest of the initial file and will cancel the rest of the rows.
 - o System/RTU name: Name with maximum of 16 characters length.
 - o RTU number: RTU number (1-65535). RTU number is not unique and multiple RTUs can have the same RTU number. If multiple RTUs have the same RTU number, then it is supposed to be connected to different SCADA systems or IEC 104 clients.
 - o Port number: Unique port number (1-65535). Program will listen to this port to accept connection for the specified RTU.
 - o Accept hosts/network list: Accepted hosts or networks filters separated by ";". example: 192.168.1.0/24;10.10.1.2".
 - o IEC 104 "k": IEC 104 "k" constant which represent maximum number of packets could be transmitted without receiving acknowledge from the receiver. If not provided, then default value is "12" packets.
 - o Idle time: Idle time in seconds before disconnecting RTU connection. Default is "60" seconds.
 - o IOA signal database file: File contains the IOA signals for the specified RTU in comma separated values format (csv). Multiple RTU can share same IOA data file.

Please check appendix A for sample initial file format.

IOA signal database file format

General notes:

- IOA signal database file format is comma separated values format (csv).
- For each RTU, an existing file should be provided.
- Multiple RTUs can share the same IOA signal database file.
- All IOA database files should be in folder "data". Folder "data" should be in the same folder where the program starts in.
- If first character of first column in any row is "!" Then program will stop sending general interrogation "GI" signals and cancel the rest of the rows.
- IOA signal database file should define the following parameters for each signal row:
 - o Index number:
 - This index will be used to submit the signals from the file to the SCADA connected system.
 - Multiple signals could be grouped by same "index" number to submitted together when the specified index number given to the iec104rs program. Typing index "0" in program GUI will send all IOA to all connected RTUs for the typed time duration at once.
 - If first character of this column in any row is "!" Then program will stop sending general interrogation "GI" signals and cancel the rest of the rows.
 - o GI: If this field contains "Y" then the specified signal will be submitted during general interrogation to the connected SCADA system.
 - o Type ID: IEC 104 type ID of the signal in numeric value. All supported values are mentioned in the provided sample IOA database files:
 - SPI (single point indication) without time tag: 1
 - SPI (single point indication) with time tag: 30
 - DPI (double point indication) without time tag: 3
 - DPI (double point indication) with time tag: 31
 - NVA (normalized meas.) without time tag: 9
 - NVA (normalized meas.) with time tag: 34
 - SVA (Scaled meas.) without time tag: 11
 - SVA (Scaled meas.) with time tag: 35
 - FLT (Float meas.) without time tag: 13
 - FLT (Float meas.) with time tag: 36
 - SCO (single command) without time tag: 45
 - SCO (single command) without time tag: 45
 - SCO (single command) with time tag: 58
 - DCO (double command) without time tag: 46
 - DCO (double command) with time tag: 59
 - RCO (regulation command) without time tag: 47
 - DCO (regulation command) with time tag: 60

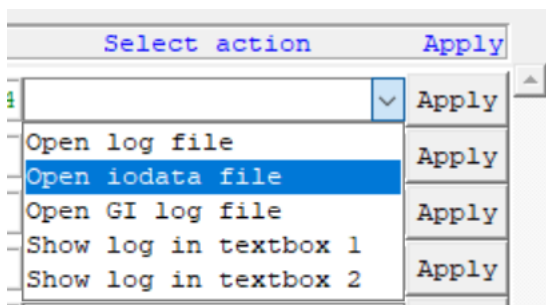
- Only “CP56Time2a “ time tag is supported.
- IOA address: Signal object address.
- Signal value: For command, the value should contain the status (SPI, DPI, or AMI) IOA address to be submitted as a response to the connected SCADA system when receiving the command. This way, full command simulation could be achieved.
- Wait (sec): delay time in seconds (decimal point value, 0.01, etc.) to be added after sending each IOA signal.
- Filter conditions: Program will wait to receive the defined filter conditions before sending the IOA signal. If any filter is empty, then it will not be used. For example, if “rtu” entry is empty then IOA signal will be submitted regardless of the RTU number. Filter conditions will not be used during sending “GI” signals or command reply.
 - RTU number: In case same IO database csv file is used for multiple RTUs then this field could be used to send the IO signal for specific RTU number only.
 - Type ID: received telegram should have this specific type-ID.
 - IOA: received telegram should have this specific type-ID.
 - Value: received telegram should have this specific signal “value”.
 - Time out (s): program will wait for “time out” in ‘sec’, decimal point seconds (0.02, etc.) to receive the filter conditions(s).
- Comment: Sequential columns (maximum 53 characters) could be used to represent signal and feeder names or any other comments to be written in the RTU log file.

Please check appendix B for sample IOA database file format.

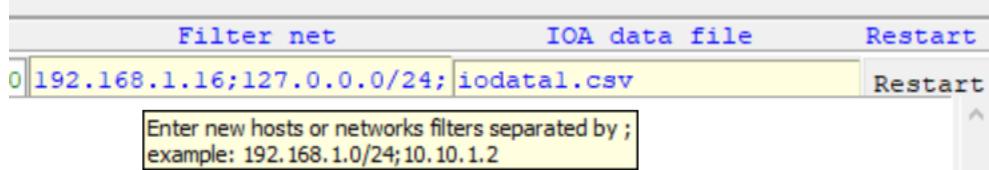
Program GUI

Trying to make the graphical user interface as simple as possible:

- 1- Configuration of the IOA signals is collected only from the IOA database files (example: iodata.csv). There is no place in the GUI to modify this part of the data. However, you can select to open the IOA database file in your default “csv” editor to modify on fly the database file and it will be reflected immediately in the “iec104rs” program. Please refer to screenshot below.



- 2- Configuration of all RTUs/Systems are initially read from the initial file (default: iec104rs.csv). However, you can select any RTU and display it in the “comparison tab” and modify its parameters then restart the RTU connection. Please refer to below screenshot (entries in light yellow color could be changed and it will take effect after restarting the connection).



- 3- Floating tooltips is displayed whenever possible to explain the GUI part.

More screenshots available in appendix “C”.

Troubleshooting

Program did not load one or more configured RTU(s) in the initial file:

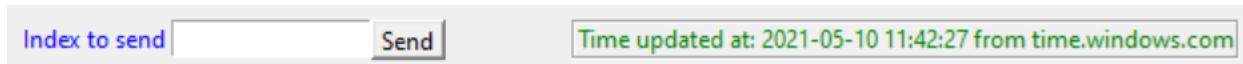
- If first character of “ID” field equal “!” then current row and all subsequent rows (RTUs) will not be loaded.
- If the “ID” field is not number, then RTU will not be loaded.
- If RTU has no name, then RTU will not be loaded. Program will read the first 16 characters of the name field.
- RTU number field should be in range 1 to 65535.
- Port number field should be unique (not used for any already loaded RTU) and in range 1 to 65535.
- IOA database file (example: iodata.csv) should be in folder “data” in the folder where the program starts in.

Some IO signals not submitted during general interrogation or during index sending process although the signal is configured in the IOA database file of the specific RTU.

- During general interrogation, if first character of “index” field equal “!” then current row and all subsequent rows (signals) will not be submitted, and general interrogation will stop.
- If “index” of the signal is not digits (numbers) then it will be considered as a comment and will not be submitted.
- If the “GI” field is not equal “Y” then the signal will not be submitted during GI.
- If signal “type ID” field is not in the supported type IDs then it will not be submitted.
- If signal value is not valid then it will not be submitted.
- If filter conditions(s) is not received before the time out period, then signal will not submitted during index send operation.

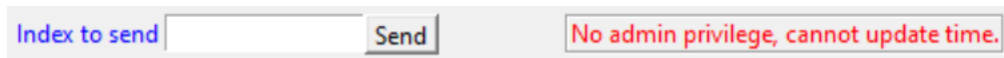
Local time not updated although NTP servers configured, and it is tested normally.

- Updating local time required admin/root privilege under both Windows and Linux so please be sure to start the application with admin/root privilege so it can update the local time normally.
- Program will try the NTP servers one by one then will sleep for the specified period (ntp_update_every_sec = 900 seconds by default) before trying again. So, maybe software couldn't reach to the servers at the first try so please wait until the software try next time.
- Please notice the local time update status as indicated in the screenshots below:



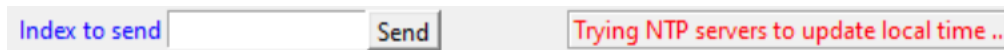
The screenshot shows a user interface with a label "Index to send" followed by an empty text input field and a "Send" button. To the right of these elements, a green-bordered box displays the message "Time updated at: 2021-05-10 11:42:27 from time.windows.com".

Local time updated successfully.



The screenshot shows the same user interface as the previous one, but the message box on the right now displays "No admin privilege, cannot update time." in red text.

No admin privilege so program cannot update the local time.



The screenshot shows the same user interface, but the message box on the right now displays "Trying NTP servers to update local time .." in red text.

Program is trying but cannot connect to the NTP servers so please check the network connection and the availability of the configured servers.

Appendix A - Sample initial file

	A	B	C	D	E	F	G	H
1	# ini file - iec104rs.csv							
2	# If first character of first column in any row is ! Then program will cancel the rest of the rows.							
3	#							
4	# starting by general settings in comma separated values.							
5	# parameter of ntp_server could be repeated multiple times in separated lines for multiple servers.							
6	ntp_server	10.1.1.15						
7	ntp_server	time.windows.com						
8	ntp_server	pool.ntp.org						
9	ntp_update_every_sec	900						
10	#							
11	# then all RTUs settings in comma separated values.							
12	# RTU number may be repeated for some RTUs.							
13	# port number should be unique for each RTU.							
14	# hosts: a list of hosts/net separated by ; which will only be accepted to connect to the specific RTU.							
15	# k: the IEC 104 k constant. Default is 12 packets.							
16	# idletime: time in seconds. If no data for idletime seconds the RTU connection will be disconnected.							
17	# io data files (iodata.csv) should be in "data" folder in the same folder where the program starts in							
18	# id	sys name	rtu no	port no	hosts	k	idletime	iodata.csv
19		1 ABB	32	2404	192.168.1.	12	60	iodata1.csv
20		2 ABB	32	2405	192.168.1.	12	60	iodata1.csv
21		3 OSI	105	2406	192.168.1.	12	60	iodata2.csv
22		4 OSI	105	2407	192.168.1.	12	60	iodata2.csv

ini file - iec104rs.csv,,,,,,,,

If first character of first column in any row is "!" Then program will cancel the rest of the rows,,,,,,,,

#,,,,,,,,,

starting by general settings in comma separated values,,,,,,,,

parameter of ntp_server could be repeated multiple times in separated lines for multiple servers,,,,,,,,

ntp_server,10.1.1.15,,,,,,,,

ntp_server,time.windows.com,,,,,,,,

ntp_server,pool.ntp.org,,,,,,,,

ntp_update_every_sec,900,,,,,,,,

#,,,,,,,,,

```

# then all RTUs settings in comma separated values.,,,,,,,
# RTU number may be repeated for some RTUs.,,,,,,,
# port number should be unique for each RTU.,,,,,,,
# hosts: a list of hosts/net separated by ; which will only be accepted to connect to the specific RTU.,,,,,,,
# k: the IEC 104 k constant. Default is 12 packets.,,,,,,,
# idletime: time in seconds. If no data for idletime seconds the RTU connection will be disconnected.,,,,,,,
"# io data files (iodata.csv) should be in ""data"" folder in the same folder where the program starts
in" ,,,,,,
# id,sys name,rtu no,port no,hosts, k,idletime, iodata.csv
1,ABB,32,2404,192.168.1.16;127.0.0.0/24;10.1.1.0/24,12,60,iodata1.csv
2,ABB,32,2405,192.168.1.16;127.0.0.0/24;10.1.1.0/24,12,60,iodata1.csv
3,OSI,105,2406,192.168.1.16;127.0.0.0/24;10.1.1.0/24,12,60,iodata2.csv
!4,OSI,105,2407,192.168.1.16;127.0.0.0/24;10.1.1.0/24,12,60,iodata2.csv
5,Siemens,178,2408,192.168.1.16;127.0.0.0/24;10.1.1.0/24,12,60,iodata1.csv

```

Appendix B - Sample IOA database file

```

#" IOA data file in comma separated values (.csv). Should be in folder ""data"" under main folder where the program starts in"
# If first character of first column in any row is ! Then GI will cancel rest of the rows.
# Supported types:
#   SPI (1,30) OFF = 0   ON = 1
#   DPI(3,31) OFF = 01  ON = 02  XX = 00/11
#   NVA(9,34) Example, if you want to send 11Kv then value=(11000/max. value) = (11000/13200) = 0.83
#   SVA(11,35) Example, if you want to send 11Kv then value = 11
#   FLT(13,36) Example, if you want to send 11.23Kv then value = 11.23
#   SCO(45,58) - value should equal the IOA of its status.
#   DCO(46,59) - value should equal the IOA of its status.
#   RCO(47,60) - value should equal the IOA of its status.
#
# send index entry from rtu if filter conditions received by connected master before timeout=T-out in second. Example: 0.002 = 2 ms, 3 = 3 sec.
# empty filter entry will not be used for filtering. Example: if rtuno is empty then index entry will be sent regardless the rtuno
# filters are not used during sending GI entries or during replying to commands.
# wait sec: Program will wait for 'wait sec' after sending the index entry. Not used in GI. Example: 0.01 = 10 ms, 5 = 5 sec.
#
#
# filter ----- conditions
# if      and if    and if    and if    filter
#index  GI      typeid  IOA      Value    wait sec  rtu no   type id   ioa      value    T-out(s)  Comment
#
#----- SCO command -----

```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
31	100008	N	46	21045	377	0						TL01-Q1 (1	TL01-Q1 (DCO)	
32	#----- SCO command with time tag -----													
33	#													
34	#----- DCO command with time tag -----													
35	#													
36	#----- DPI -----													
37	301	Y	31	301	1	0						DUMP	DUMP	
38	301	Y	31	303	1	0						spare	spare	
39	301	Y	31	305	1	0						H-TH01BA	H-TH01BAY CTRL	
40	301	Y	31	307	1	0						H-TH01Q0	H-TH01Q0	
41	301	Y	31	309	1	0						H-TH01Q1	H-TH01Q1	
42	301	Y	31	311	1	0						H-TH01Q4	H-TH01Q4	
43	301	Y	31	313	1	0						H-LN01BA	H-LN01BAY CTRL	
44	#!End GI	N	31	314	1	0								
45	301	Y	31	315	1	0						H-LN01Q0	H-LN01Q0	
46	317	Y	31	317	2	0		46	21001	2	5000	H-LN01Q1	H-LN01Q1	
47	317	Y	31	319	2	0						H-LN01Q3	H-LN01Q3	

#" IOA data file in comma separated values (.csv). Should be in folder ""data"" under main folder where the program starts in"" ,,,,,,,,,,

If first character of first column in any row is ! Then GI will cancel rest of the rows.,,,,,,,,,,

Supported types:,,,,,,,,,

#, "SPI (1,30)", OFF = 0, ON = 1, ,,,,,,,,,,

#, "DPI(3,31)", OFF = 01, ON = 02, XX = 00/11, ,,,,,,,,,,

#, "NVA(9,34)", "Example, if you want to send 11Kv then value=(11000/max. value) = (11000/13200) = 0.83" ,,,,,,,,,,

```

#,"SVA(11,35)","Example, if you want to send 11Kv then value = 11" ,,,,,,,,,
#,"FLT(13,36)","Example, if you want to send 11.23Kv then value = 11.23" ,,,,,,,,,
#,"SCO(45,58) - value should equal the IOA of its status." ,,,,,,,,,
#,"DCO(46,59) - value should equal the IOA of its status." ,,,,,,,,,
#,"RCO(47,60) - value should equal the IOA of its status." ,,,,,,,,,
#,,,,,,,,
"# send index entry from rtu if filter conditions received by connected master before timeout=T-out in
second. Example: 0.002 = 2 ms, 3 = 3 sec." ,,,,,,,,,
# empty filter entry will not be used for filtering. Example: if rtuno is empty then index entry will be sent
regardless the rtuno,,,,,,,,
# filters are not used during sending GI entries or during replying to commands.,,,,,,,,,
"# wait sec: Program will wait for 'wait sec' after sending the index entry. Not used in GI. Example: 0.01 =
10 ms, 5 = 5 sec." ,,,,,,,,,
#,,,,,filter ----- conditions,,,,,
#,,,,,if,and if,and if,and if,filter,,
#index,GI,typeid,IOA,Value,wait sec,rtu no,type id,ioa,value,T-out(s),Comment,
#,-,--,--,--,--,--,--,--,--,--,--,--,
#-----,,SCO command -----,,,,,,,,,
#,,,,,,,,
#-----,,DCO command + status -----,,,,,,,,,
100000,N,46,21001,301,0,,,,,dummy (DCO),dummy (DCO)
100001,N,31,301,1,0,,,,,dummy (dpi),dummy (dpi)
100002,N,46,21035,343,0,,,,,KLN01-Q0 (DCO),KLN01-Q0 (DCO)
100003,N,47,21007,15001,0,,,,,TX-(RCO),TX-(RCO)
100004,N,46,21039,355,0,,,,,KLN03-Q0 (DCO),KLN03-Q0 (DCO)
100005,N,46,21041,361,0,,,,,KLN05-Q0 (DCO),KLN05-Q0 (DCO)
100006,N,46,21043,367,0,,,,,KLN07-Q0 (DCO),KLN07-Q0 (DCO)
100007,N,46,21017,307,0,,,,,TH01-Q0 (DCO),TH01-Q0 (DCO)
100008,N,46,21045,377,0,,,,,TL01-Q1 (DCO),TL01-Q1 (DCO)
#-----,,SCO command with time tag -----,,,,,,,,,

```

```
#, , , , , , , , , ,  
#-----,,DCO command with time tag -----,,  
#, , , , , , , , , ,  
#-----,,DPI -----,,  
301,Y,31,301,1,0,,,,,,DUMP,DUMP  
301,Y,31,303,1,0,,,,,,spare,spare  
301,Y,31,305,1,0,,,,,,H-TH01BAY CTRL,H-TH01BAY CTRL  
301,Y,31,307,1,0,,,,,,H-TH01Q0,H-TH01Q0  
301,Y,31,309,1,0,,,,,,H-TH01Q1,H-TH01Q1  
301,Y,31,311,1,0,,,,,,H-TH01Q4,H-TH01Q4  
301,Y,31,313,1,0,,,,,,H-LN01BAY CTRL,H-LN01BAY CTRL  
#!End GI,N,31,314,1,0,,,,,,  
301,Y,31,315,1,0,,,,,,H-LN01Q0,H-LN01Q0  
317,Y,31,317,2,0,,46,21001,2,5000,H-LN01Q1,H-LN01Q1  
317,Y,31,319,2,0,,,,,,H-LN01Q3,H-LN01Q3  
321,Y,31,321,2,0,,,,,,K-C2 CAP CTRL,K-C2 CAP CTRL  
323,Y,31,323,2,0,,,,,,K-C2 CAP MODE,K-C2 CAP MODE  
#-----,,SPI -----,,  
445,Y,30,445,0,0,,,,,,110V DC FAULT,110V DC FAULT  
446,Y,30,446,0,0,,,,,,30V DC FAULT,30V DC FAULT  
447,Y,30,447,0,0,,,,,,48V DC BAT CIRC FAULT,48V DC BAT CIRC FAULT  
#-----,,Normalized meas. + CP56Time2a -----,,  
64,N,34,11002,-0.4,0,,,,,,W,W  
65,N,34,11002,0.6,0,,,,,,W,W  
66,N,34,11003,0.5,0,,,,,,VAR,VAR  
#-----,,Short floating point measurements -----,,  
88,N,13,11002,-10.456,0,,,,,,W,W  
89,Y,13,11002,5.435,0,,,,,,W,W
```


Appendix C – GUI screenshots

Tab1 – Full RTU list

IEC-104 RTU Simulator

Started at: 2021-05-12 09:58:29.050819 Index to send: Send Time updated at: 2021-05-12 09:58:31 from time.windows.com

Tab1: Full RTU list Tab2: Log files and data edit Duration in sec:

No.	System	Online	RTU	Port	GI	Last index	Connected at	Select action	Apply
1	ABB	YES	32	2404	RUN		2021-05-12 09:58:46.305639		Apply
2	ABB	NO	32	2405					Apply
3	OSI	NO	105	2406					Apply
4	OSI	NO	105	2407					Apply
5	Siemens	NO	178	2408				Select action to be applied/executed.	Apply
6	PSI	NO	251	2409					Apply
7	SELTA	NO	324	2410					Apply
8	Lucy	NO	397	2411					Apply
9	Schnieder	NO	470	2412					Apply
10	Multaqa	NO	543	2413					Apply

Tab2 – RTU comparisons and parameters editing.

IEC-104 RTU Simulator

Started at: 2021-05-12 09:58:29.050819 Index to send: [] Send Time updated at: 2021-05-12 10:01:32 from time.windows.com

Tab1: Full RTU list Tab2: Log files and data edit Duration in sec: []

No.	System	Online	RTU	Port	GI	Last index	Connected at	Filter net	IOA data file	Restart
1	ABB	YES	32	2404			2021-05-12 09:58:46.305639	192.168.1.16;127.0.0.0/24;	iodatal.csv	Restart

ABB log file .. RTU: 32, listen port: 2404
 2021-05-12 09:58:29.050819 : Initialized ..
 2021-05-12 09:58:46.236648 : Connected to IP: 10.1.1.15, Port: 1375
 2021-05-12 09:58:46.305639 : startdt act/con done.
 2021-05-12 09:58:46.305639 : End of initialization transmitted.
 2021-05-12 09:58:46.483987 : GI received.
 2021-05-12 09:58:50.473965 : GI finished.
 2021-05-12 09:58:51.390775 : GI received.
 2021-05-12 09:58:54.418993 : GI finished.
 2021-05-12 09:59:35.511918 : Received testfr act minimum period: 20.8 seconds.
 2021-05-12 09:59:55.657915 : Received testfr act minimum period: 20.1 seconds.
 2021-05-12 10:00:35.647872 : Received testfr act minimum period: 19.8 seconds.

Enter new hosts or networks filters separated by ;
 example: 192.168.1.0/24;10.10.1.2

No.	System	Online	RTU	Port	GI	Last index	Connected at	Filter net	IOA data file	Restart
2	ABB	YES	32	2405			2021-05-12 10:01:56.918088	192.168.1.16;127.0.0.0/24;	iodatal.csv	Restart

ABB log file .. RTU: 32, listen port: 2405
 2021-05-12 09:58:29.182327 : Initialized ..
 2021-05-12 10:01:56.871218 : Connected to IP: 127.0.0.1, Port: 51506
 2021-05-12 10:01:56.918088 : startdt act/con done.
 2021-05-12 10:01:56.918088 : End of initialization transmitted.

153 RTUs configured at once. It took about 1 minute to load all the 153 RTUs on a machine with 16 GB RAM and 3.8 GHZ Intel processor.

IEC-104 RTU Simulator

Started at: 2021-05-01 07:25:49.742867 Index to send: [] Send No admin privilege, cannot update time

Full RTU list Comparisons of log files Duration in sec: []

No.	System	Online	RTU	Port	GI	Last index	Connected at	Select action	Apply
136	Shinas	NO	9741	2539				▼ Apply	~
137	Tareef	NO	9814	2540				▼ Apply	
138	ABB	NO	9887	2541				▼ Apply	
139	OSI	NO	9960	2542				▼ Apply	
140	ABB	NO	10033	2543				▼ Apply	
141	ABB	NO	10106	2544				▼ Apply	
142	OSI	NO	10179	2545				▼ Apply	
143	OSI	NO	10252	2546				▼ Apply	
144	Siemens	NO	10325	2547				▼ Apply	
145	PSI	NO	10398	2548				▼ Apply	
146	SELTA	NO	10471	2549				▼ Apply	
147	Lucy	NO	10544	2550				▼ Apply	
148	Schnieder	NO	10617	2551				▼ Apply	
149	Multaqa	NO	10690	2552				▼ Apply	
150	Owinat	NO	10763	2553				▼ Apply	
151	DAS	NO	10836	2554				▼ Apply	
152	Shinas	NO	10909	2555				▼ Apply	
153	Tareef	NO	10982	2556				▼ Apply	

Normal text file length: 1,354 lines: 61 Ln: 1 Col: 1 Sel: 0 | 0 Windows (CR LF) UTF-8 INS

Type here to search

Appendix D – Windows binary files

Windows binary file is generated by nuitka Python compiler:

<https://nuitka.net/>

By using the following command:

```
python -m nuitka --windows-file-description="IEC104 RTU Simulator" --windows-file-version="1.0" --  
windows-product-version="1.0" --windows-company-name="M.M" --onefile --plugin-enable=tk-inter --  
standalone --mingw64 iec104rs.py
```

Appendix D - Other projects

IEC104 Multiple Masters to Single Slave

<https://github.com/med7at69/IEC104-MultipleMasters2SingleSlave>

This program will connect multiple masters (SCADA master stations) to single slave (RTU) as defined protocol IEC 60870-5-104. Although the protocol IEC 104 itself doesn't have a way to do that but the program will play the Man In The Middle role achieve this connection.

Any number of masters connected to a slave is forming a group. You can create any number of groups and the program will establish the communication among each group members independent on the other groups. Each master in each group can receive IO status and send commands to the slave RTU independent on the other masters.

Why may anyone need the IEC104MM2SS program?

For cyber security reasons, the slave RTU is configured to accept connection from specific number of masters' IPs. Some RTUs may have limited number of masters to be configured.

In my company, we were replacing our legacy SCADA system with new one. During the test period we need both two systems' servers to communicate to the RTUs and station control systems (SCS) simultaneously. We have 2 old servers + 2 new servers at the main control center + 1 server at the backup control center. These total of 5 servers so here is the problem:

- 1- Some RTUs and SCS stations doesn't have enough entries for all the servers.
- 2- Some old RTUs and SCS stations have many configuration difficulties.
- 3- For about 200 stations it is tedious to configure them many times to add the servers during the test period then to remove the old servers again after that.

For the above reasons, I wrote the IEC104MM2SS program.

Program features:

- Can easily create any number of groups of masters + slave to communicate among them.
- the program will establish the communication among each group members independent on the other groups.
- Each master in each group can receive IO status and send commands to the slave RTU independent on the other masters.
- To not affect the RTU availability reports, program will not accept connections from the master SCADA systems until the program connected first to the corresponding slave RTU. If disconnected from the RTU then it will disconnect all the masters in the same group.
- Easy configuration file building (.csv format) by using spreadsheet programs such as MS Excel.
- IP/Network filtration for each master entry independently.
- Multiple IP:PORT numbers for each slave (RTU/SCS). Program will try the IP:PORT one by one until establish the connection to the slave RTU/SCS.
- Linux and windows compatible.
- Python native graphical user interface (GUI) (no need for third party solutions).
- Multithread operation.
- Time synchronization through multiple NTP servers.
- Log file for each master/slave connection.
- No GUI mode of operation in which the program will work in background silently while only update the log files.