RTDS Training course of IEPG
DAY 6: Application of GTNETx2 card

COORDINATOR: DR. IR. J.L. RUEDA TORRES
LAB INSTRUCTORS: dipl. ing Matija Naglic, msc. Arun Joseph

March 8, 2018

1. Introduction

The objective of this lab session is to provide a practical overview of the features offered by the GTNETx2 card of the RTDS. The first section of this document introduces GTNETx2 functionality, and supported protocols such as, SV, PMU and SKT, with example use cases to demonstrate their use. In the second part, the practical centralized based Under Frequency Load Shedding (UFLS) example, utilizing PMU and SKT is described and performed in real-time.

Note: This document is provided with MATLAB scripts and simple models in RTDS. You are expected to follow the instructions provided in this tutorial and carry out the specified tasks. At the end of this tutorial, you should be able to:

a. Running IEEE 9-bus system in RTDS.
b. Configuring SV and data visualisation using Wireshark.
c. Configuring PMU and data visualization using Wireshark, and in-house developed MATLAB based Synchro-measurement Application Development Framework (SADF).
d. Configuring SKT.
e. Using of PMUs, SKT, and SADF to preform closed-loop control UFLS scheme on IEEE 9-bus system in real-time.

2. Prerequisite Knowledge and software requirements

From previous lab sessions:

- You should be familiar with running basic cases in RSCAD
- Basic knowledge of TCP/IP and UDP protocol stack

Software requirements:

- MATLAB-2014a, with Instrument Control Toolbox
- OpenVPN client, with credentials to connect to RTDS VLAN

3. Attached folders

- \SV example – Test system for SV
- \UFLS example – 9 bus system with UFLS scheme using PMU data from the system and real time control of the system using SKT.

4. SV data visualisation using Wireshark

This section demonstrates the configuration of the GTNET SV unit and visualisation of sampled values from RTDS simulation using the Wireshark software. The figure 1 shows the RSCAD draft circuit used for this demonstration and the figure 2 shows the runtime of the same file. The assign control processor block is used to define the board number which is selected based on the config file (5 as for this experiment). The GTSYNC and GTNET-SV1 components from the Protection and Control library and paste them into the drawing canvas.

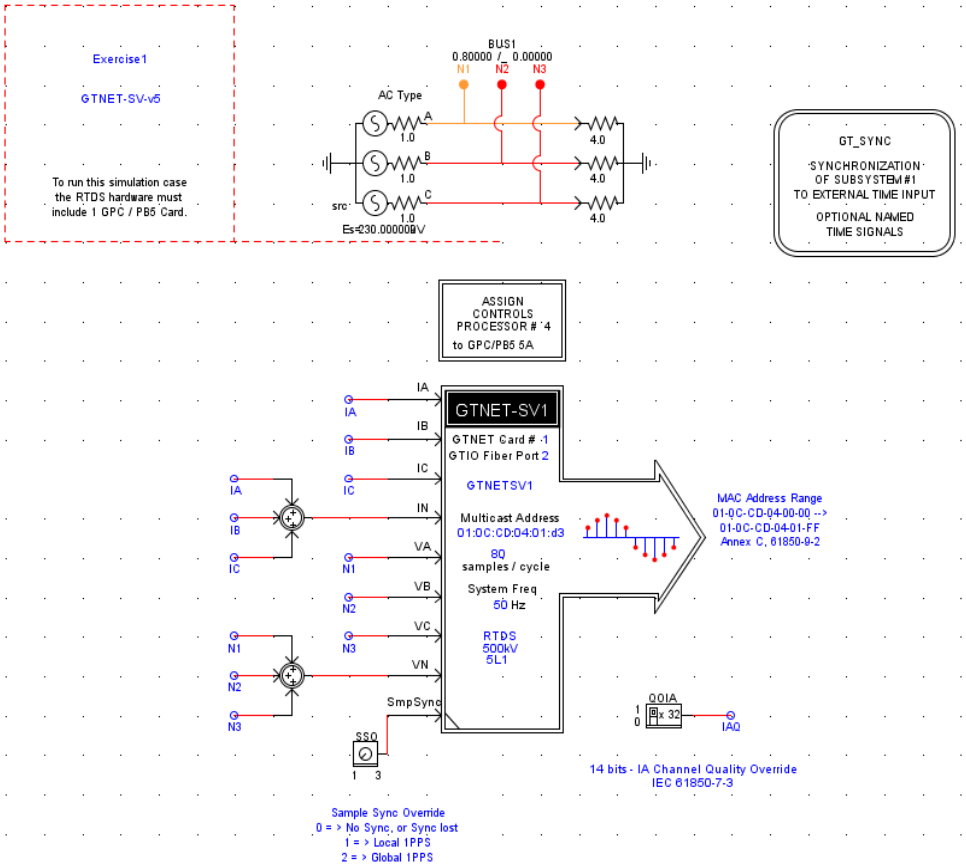

Figure 1 : RSCAD draft circuit of SV example

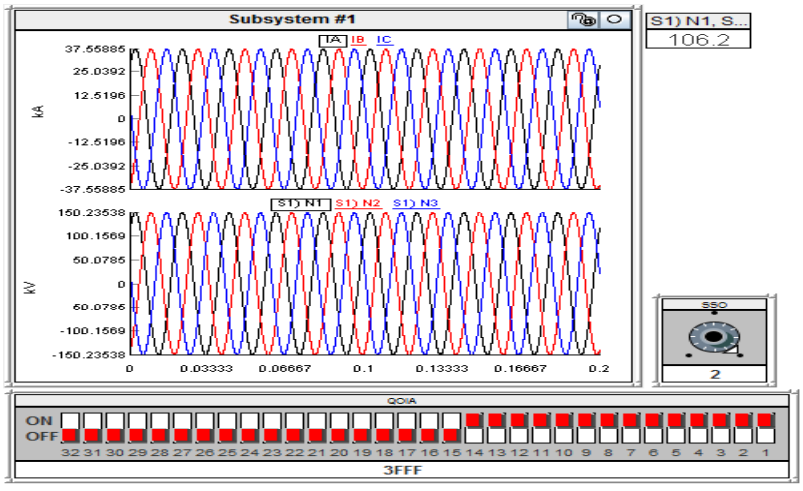

Figure 2 : RSCAD runtime of SV example.

The figure 3 shows the GTNET-SV block . The sampling period can be selected as 80 or 256 samples per second. The GTNET fiber port number and the card number values should be selected according to the config file.

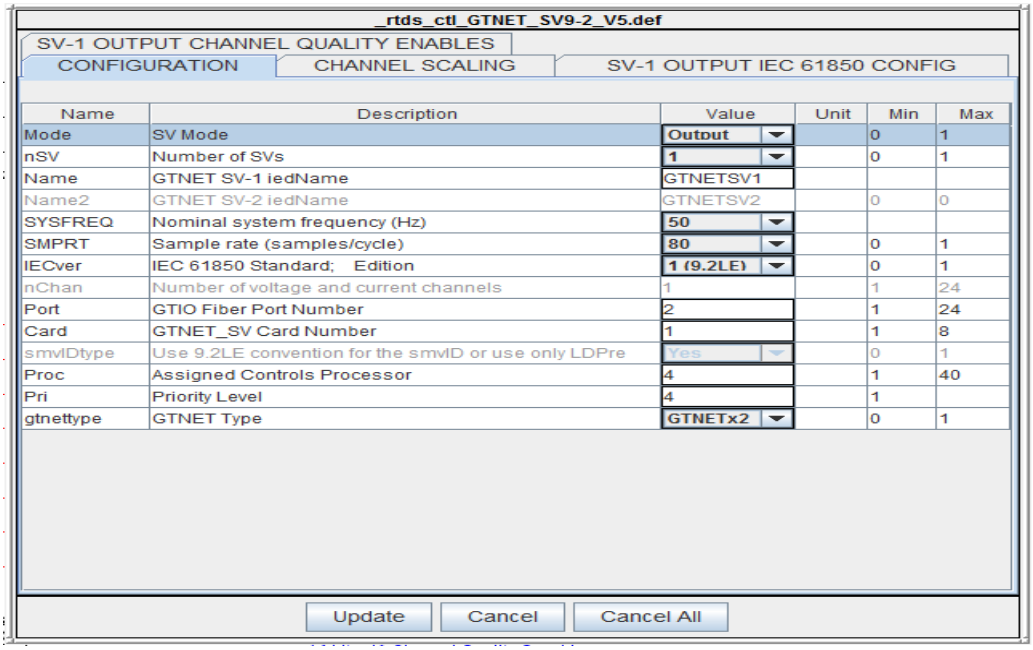

Figure 3: GTNET-SV block.

The following are the main steps for the experiment.

- Set up the SV example draft case.
- Open Wireshark and enter SV as filter .
- Start running the SV example.sib.
- Observe the “Time” (time slipped since last displayed packet) to verify if it matches the expected Sample Period. Note the time shown in Wireshark is not precise, but a good reference. SV packets being published at 80 samples/cycle with system frequency of 50 Hz is shown below.

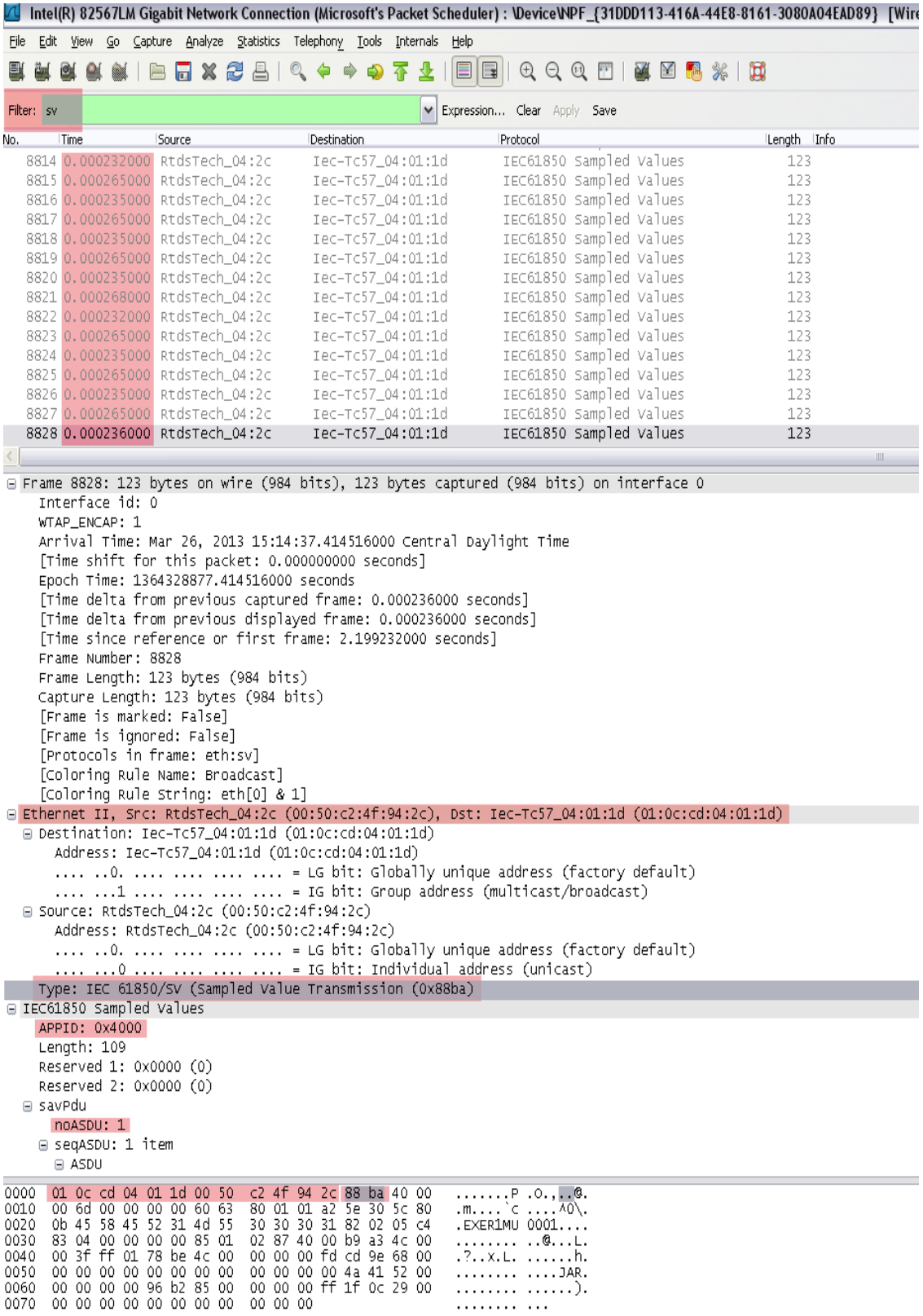

Figure 4: SV data visualization using Wireshark for 80 samples/sec.

- When set to 256 samples/cycle, according to the Implementation Guideline, 8 data units (1 ASDU contains 4 voltages and 4 currents) of SV data are sent at a time as ONE “packet”, so the packet period would be 1/(256*50) * 8 = 625 ms.

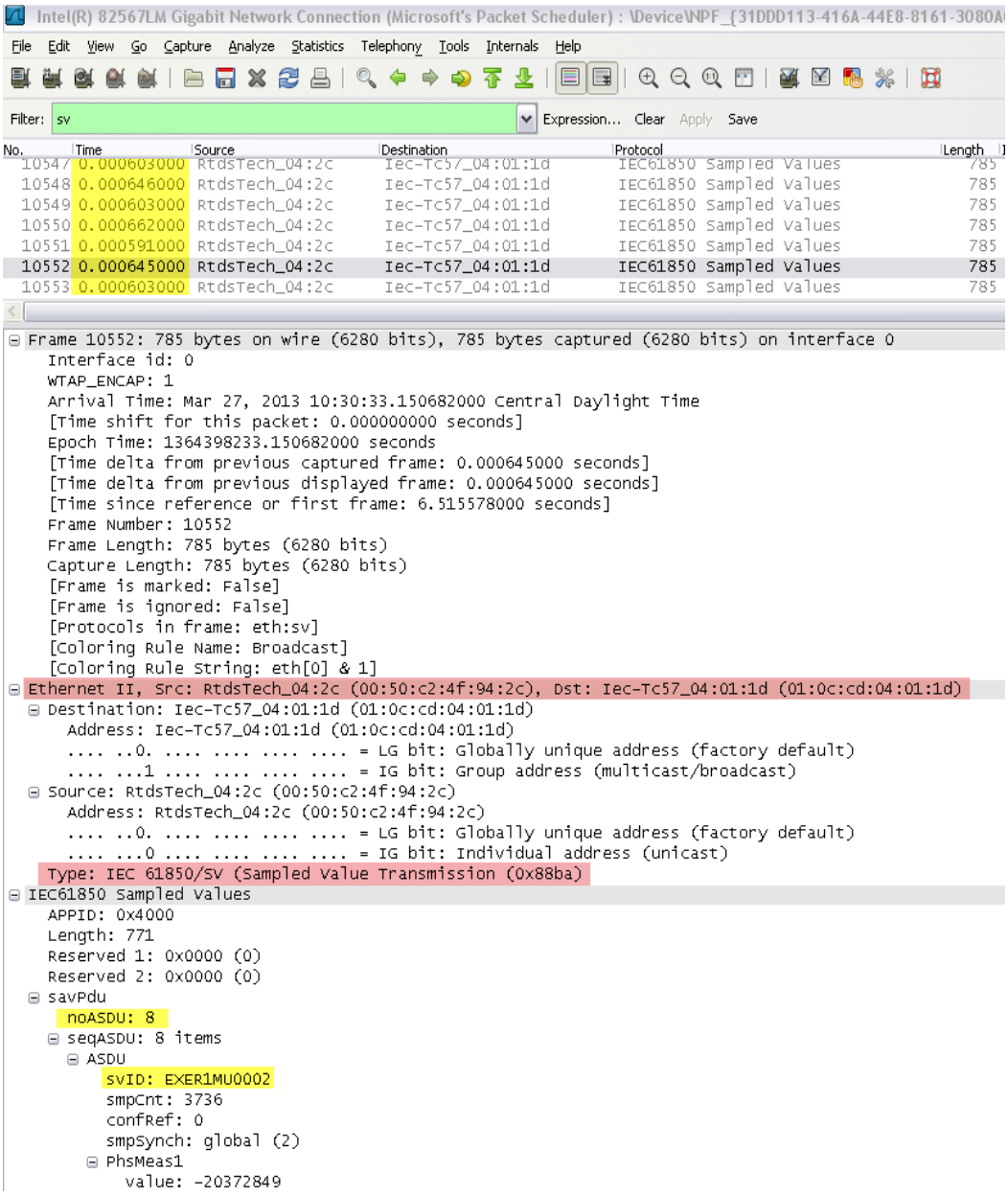

Figure 5: SV data visualization using Wireshark for 256 samples/sec.

5. PMU configuration and data inspection using Wireshark.

This section demonstrates the configuration of the GTNET-PMU and inspection of PMU communication frames and their values using the Wireshark packet inspection software and MATLAB supported SADF.

5.1 Configure draft parameters

Copy the GTSYNC and GTNET-PMU8 components from the Protection and Control library and paste them into the drawing canvas. Enter the parameters such as number of PMUs. The GTNET fiber port number and the card number values should be selected according to the config file.

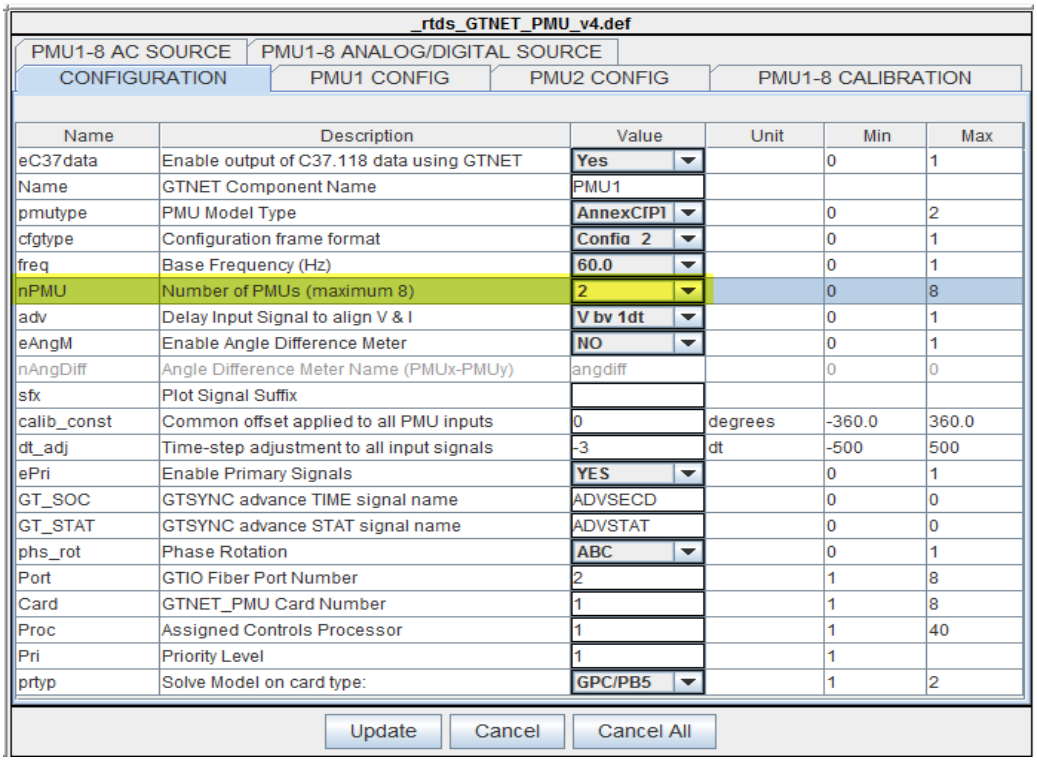

Figure 6: PMU block – CONFIGURATION tab.

Set up parameters for PMU as shown in figure 6, 7 and 8. Fill in voltage and current sources for PMUs as shown in figure 8

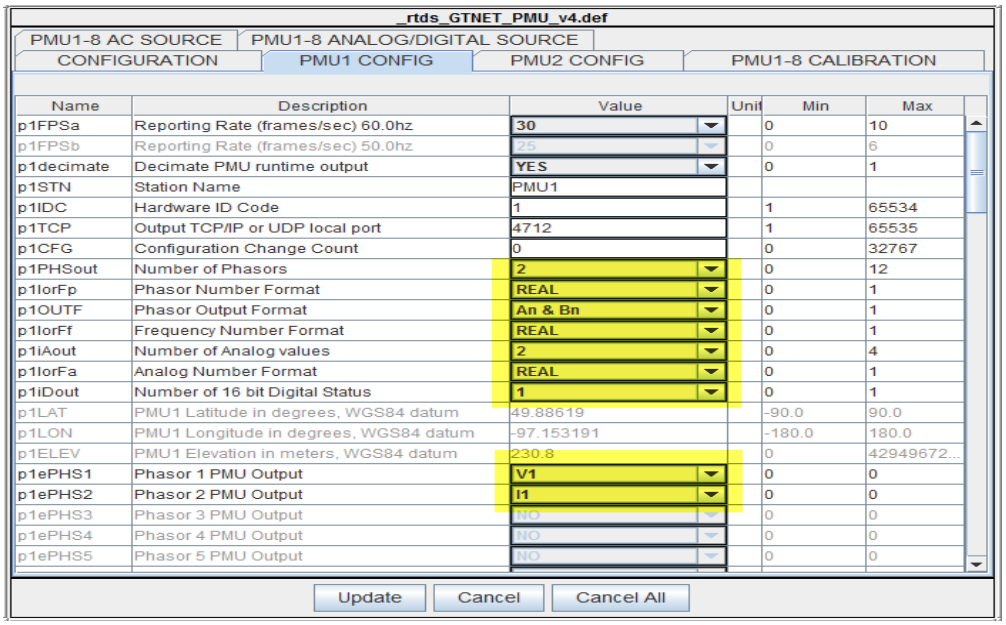

Figure 7: PMU block – PMU1 config tab.

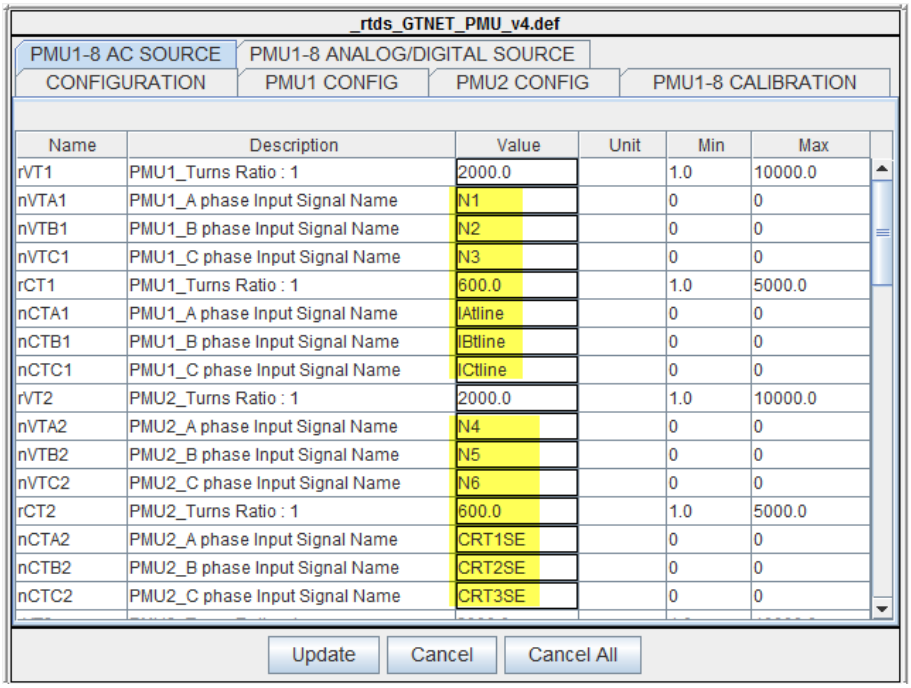

Figure 8: PMU block – PMU 1-8 AC Source tab.

5.2 Data visualisation using in Wireshark, PMU connection tester.

Make sure the case is running. Open Wireshark. Go to Edit -> Preferences. Find “SYNCHROPHASOR” under “Protocols” section as shown in figure 9. Make sure the port number filled in the “Synchrophasor TCP port” matches the setting in the GTNET-PMU component.

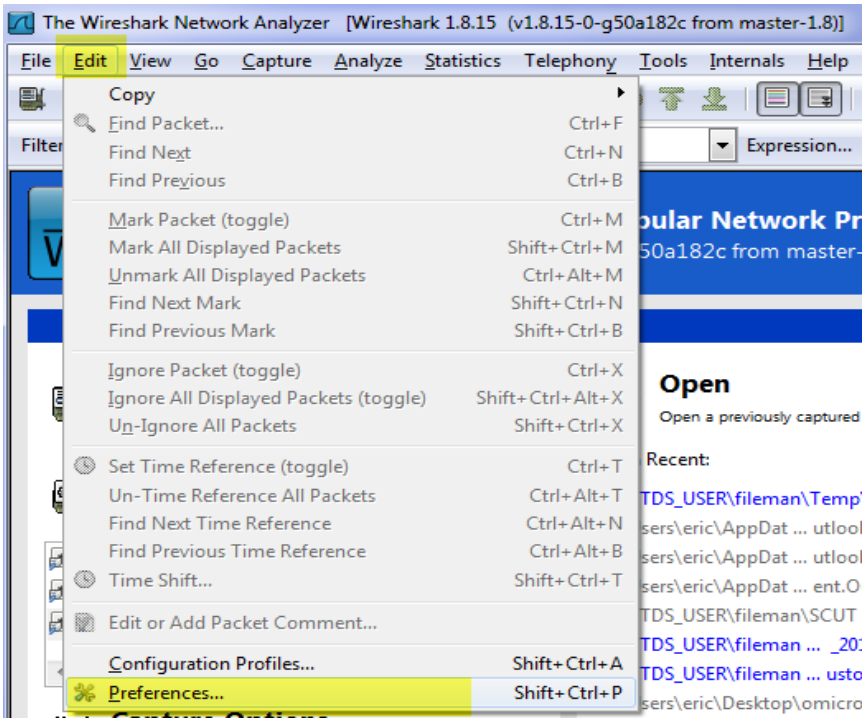

Figure 9: Wireshark Preferences selection

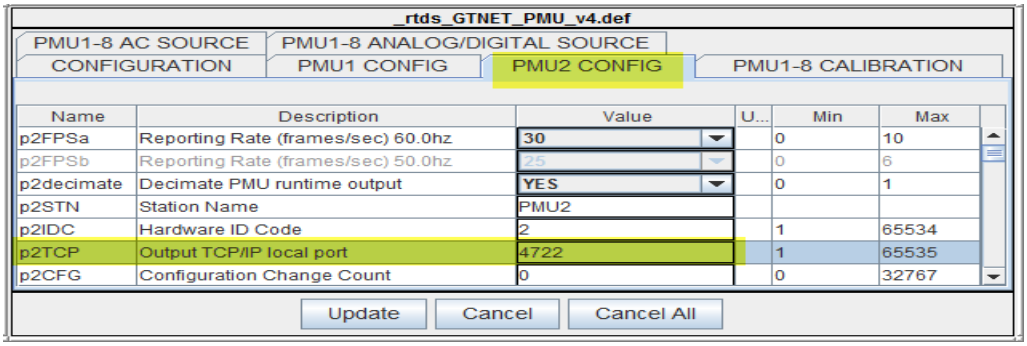

Figure 10: PMU block – PMU2 CONFIG.

Enter the port number in the Wireshark, as the same used for PMU as shown in figure 10 and 11.

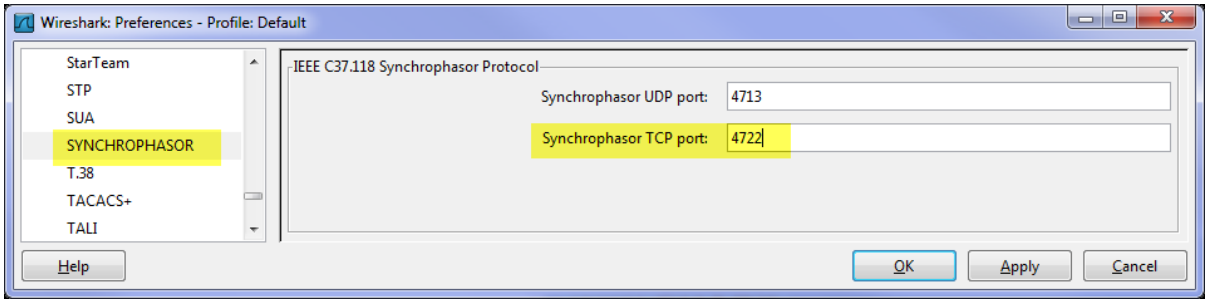

Figure 11: Port Setting in Wireshark for SYNCHROPASOR.

Select the proper Ethernet card as shown in figure 12, then click “Start” to start the Wireshark.

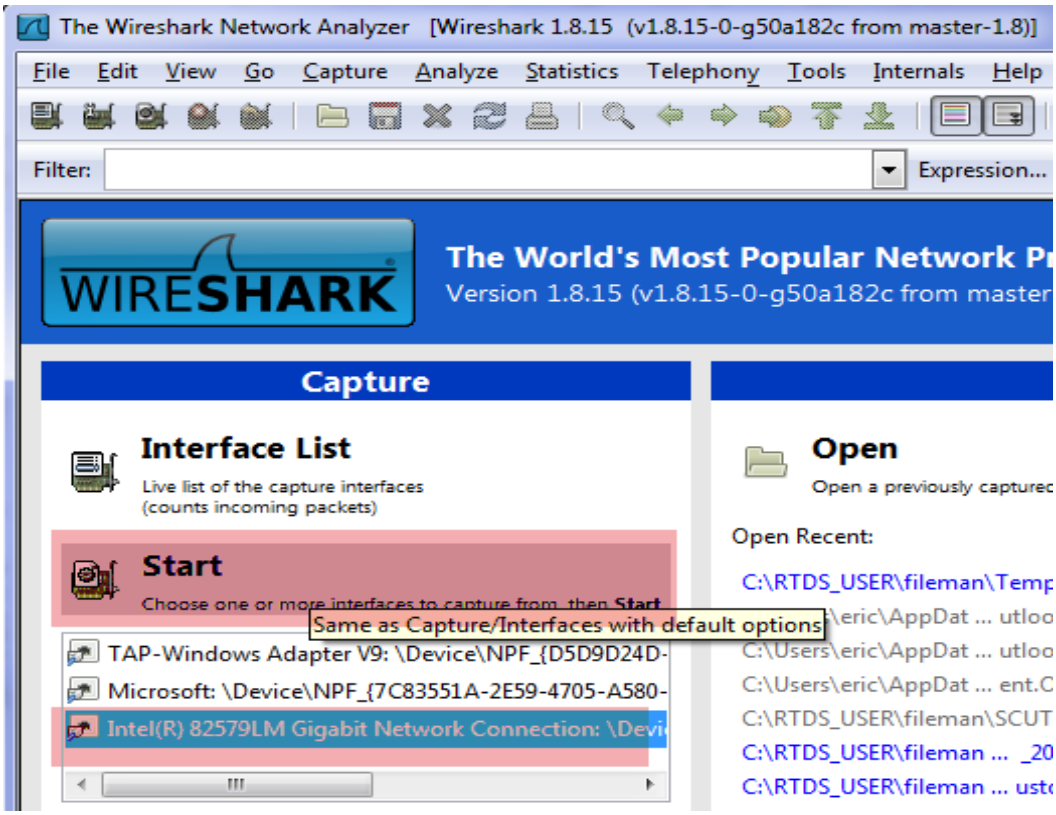

Figure 12: LAN selection in Wireshark.

Make Sure that you are capturing the data from the right LAN network using wireshark . Start the Wireshark and put “synphasor” as the Filter as ahown in figure 13 .

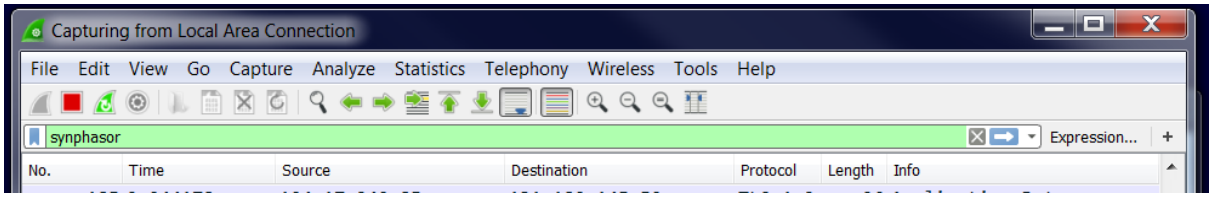

Figure 13: Starting Wireshark with “synchrophasor” filter enabled.

Enter all the IP, port and others details as shown in figure below in PMU connection tester software as shown in figure 14. Click the connect button after executing the Runtime file with PMU.

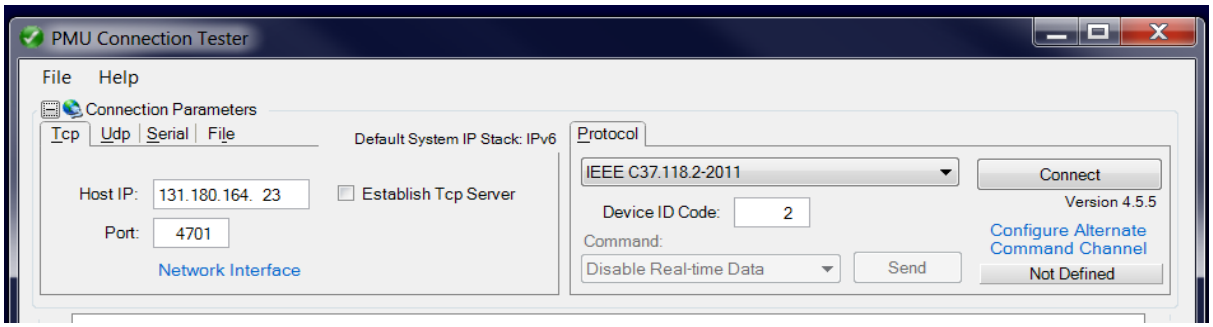

Figure 14: PMU Connection Tester settings

The PMU data exchange can be visualised in Wireshark as shown in figure 15.

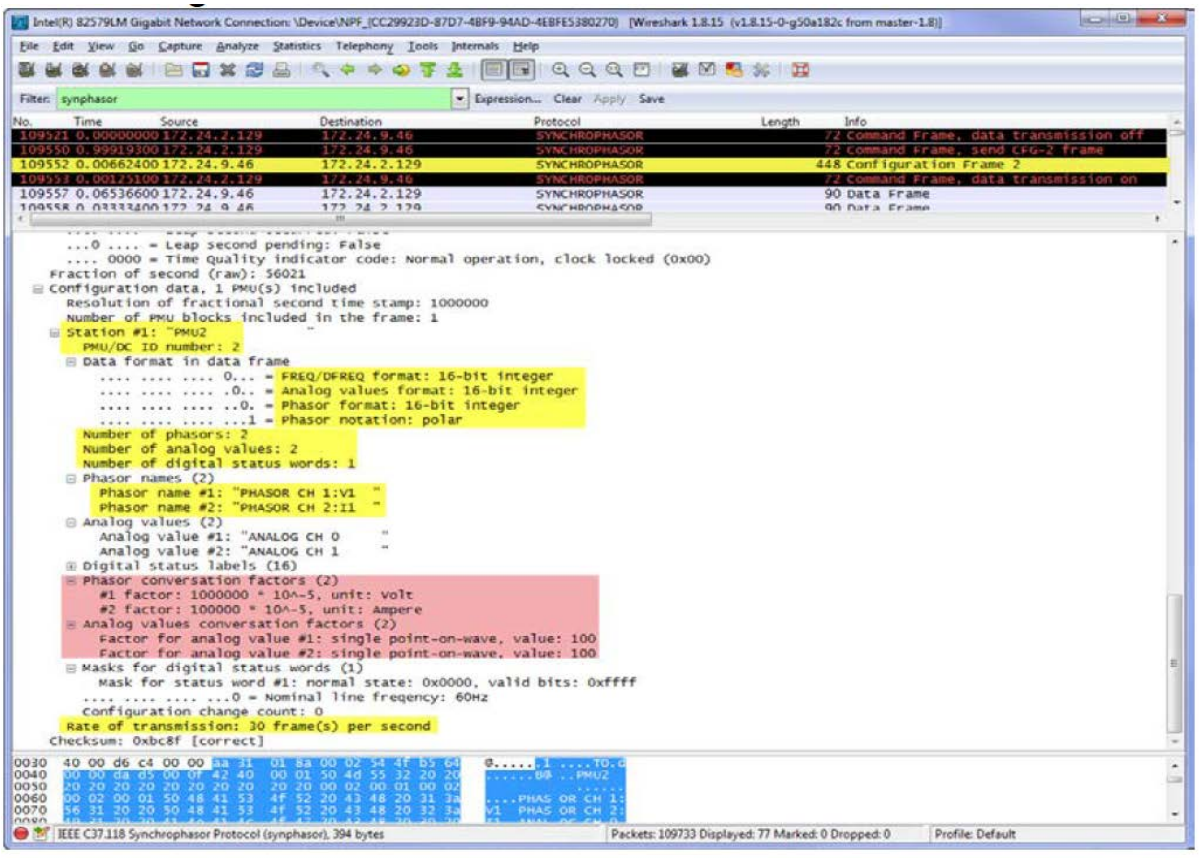

Figure 15 : PMU data visualisation using Wireshark.

6. SKT Configuration.

Using the GTNET-SKT component in RTDS, it is possible to send and receive socket data from external program during its runtime. The figure 16 illustrates the socket data transfer mechanism. The following text gives some basic terminology and definitions used:

Server/Client Application: The basic mechanisms of client-server setup are:

- A client send a request to a server
- The server returns a reply

1. Server Socket
   Create a Socket – get the file descriptor;
   Bind to an address – what port am I on?
   Listen on a port, and wait for a connection to be established;
   Accept the connection from a client;
   Send/recv;
   Shutdown to end read/write;
   Close to release data;

2. Client Socket
   Create a Socket;
   Connect to a server;
   Send/recv – repeat until the data is received;
   Shutdown to end read/write;
   Close to release data;

6.1 TCP vs UDP

Stream Sockets - TCP

Provides reliable two-way communication. One side initiates the connection to the other, and after the connection is established, either side can communicate to the other. In addition, there is immediate confirmation that what were sent actually reached its destination. Stream socket uses a Transmission Control Protocol (TCP), which exists on the transport layer of the Open Systems Interconnection (OSI) model. The data is usually transmitted in packets. TCP is designed so that the packets of data will arrive without errors and in sequence.

Datagram Sockets - UDP

One way communication, i.e. mailing a letter compared to making a phone call. The communication is unreliable. I.e. mailing several letters, we cannot be sure that they arrive in the same order, or even that they reached their destination at all.

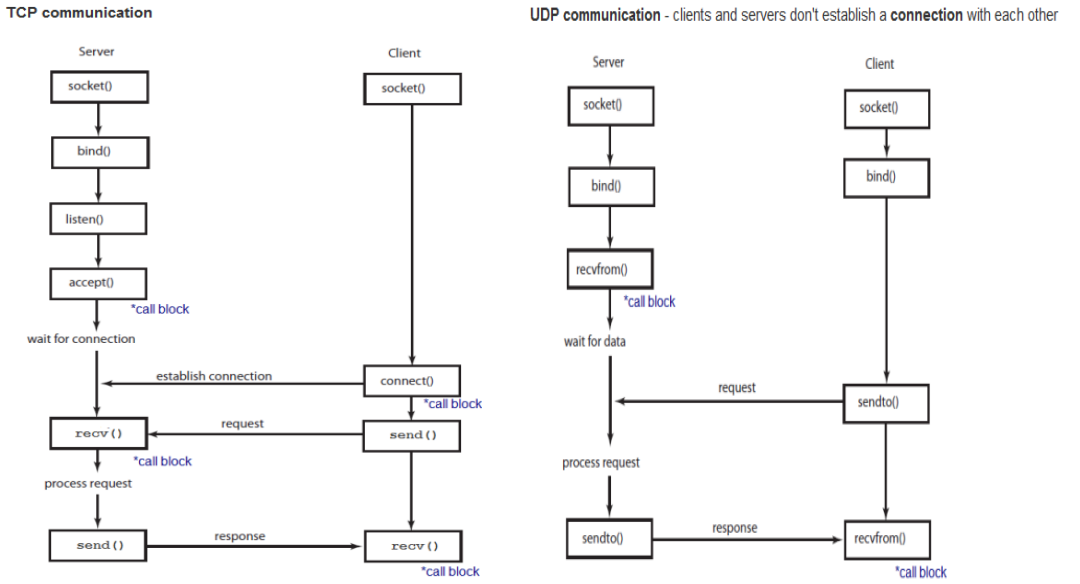

Figure 16 : Socket data transfer mechanism.

The figures 17-20 shows the GTNET-SKT configured in TCP server mode, which is further used in the test case explained in the later section.

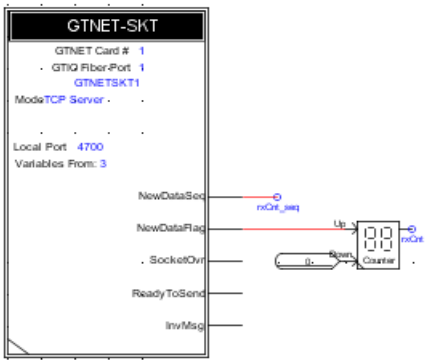

Figure 17 : GTNET-SKT BLOCK

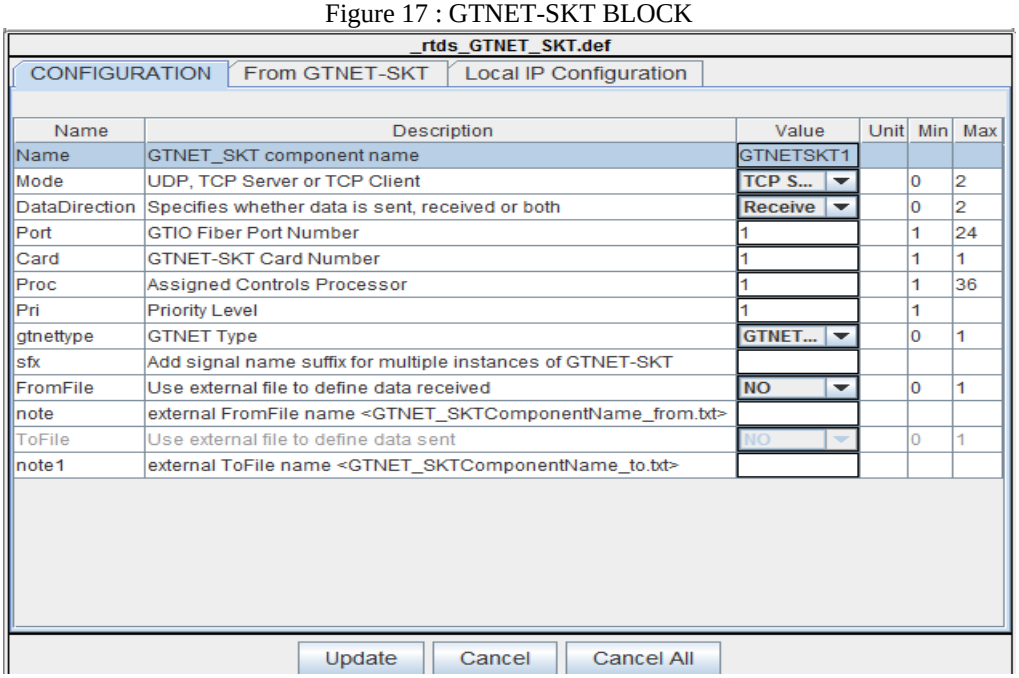

Figure 18 : GTNET-SKT BLOCK, CONFIGURATION tab.

Specify the input/output signals in the next windows as shown figure 19.

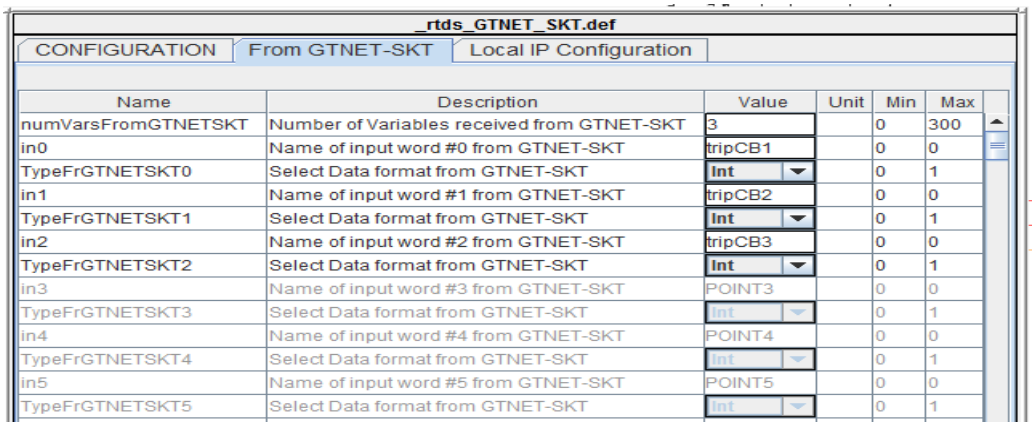

Figure 19 : GTNET-SKT BLOCK, From GTNET-SKT tab.

Enter the port details as shown in figure below as shown in figure 20.

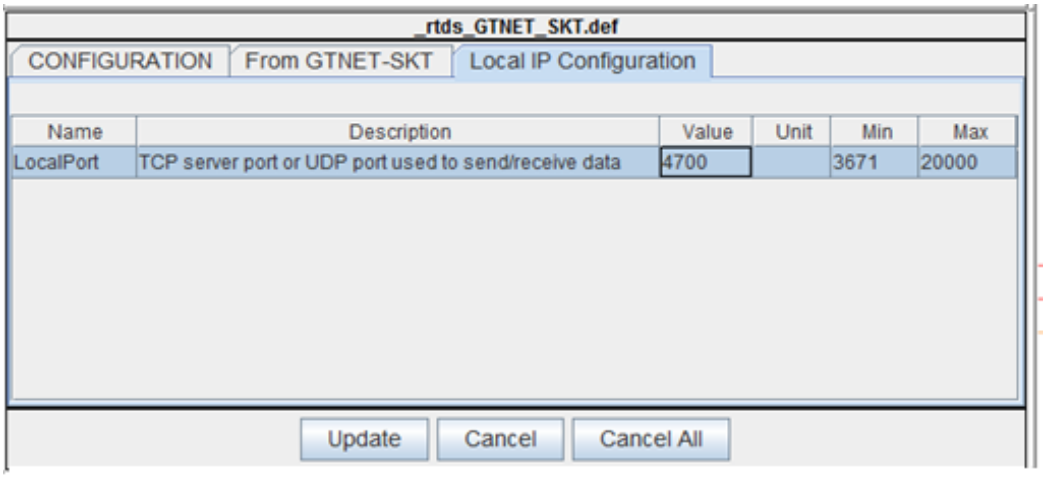

Figure 20 : GTNET-SKT BLOCK, Local IP Configuration tab.

The test case demonstrates the how an external program (Matlab) can be used to establish a socket connection with RTDS and used to receive trip signal from the external program.

7. Example application: IEEE 9-bus benchmark system with Under Frequency Load Shedding closed-loop control

7.1 Description

In this part the practical example of the centralized based Under Frequency Load Shedding (UFLS) corrective-control scheme utilizing PMU and SKT is described and performed in real-time. The UFLS is performed on the IEEE 9-bus system (use file “IEEE 9 Bus Power System_UFLS.dft”), where all 3 generator units are monitored using PMUs, installed on generator terminal bus, as illustrated on Fig. 21.

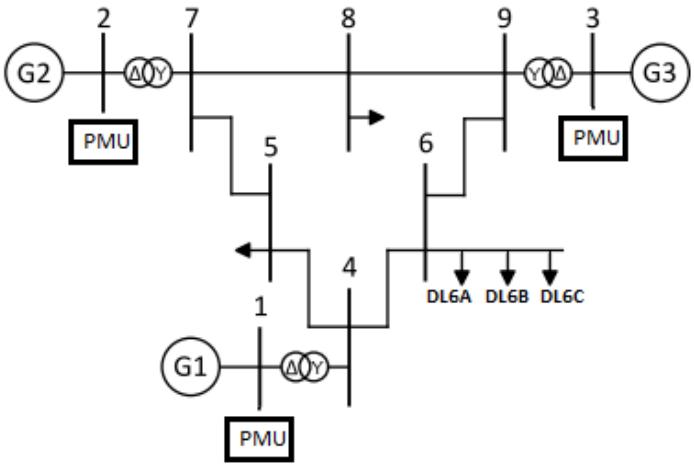

Figure 21 : IEEE 9-bus benchmark system model

Additionally, the dynamic load 6 (DL6) was replaced by 3 individual loads (DL6A, DL6B, DL6C), where the total sum of individual loads equals the original DL6 load, as illustrated on Fig. 21 and Fig. 22. Hereby, circuit breakers are installed to trip each 3 individual loads (DL6A, DL6B, DL6C) in case of preformed UFLS. The PMU measurements are first send to Phasor Data Concentrator, where the PMU measurements are time-aligned into one coherent data stream.

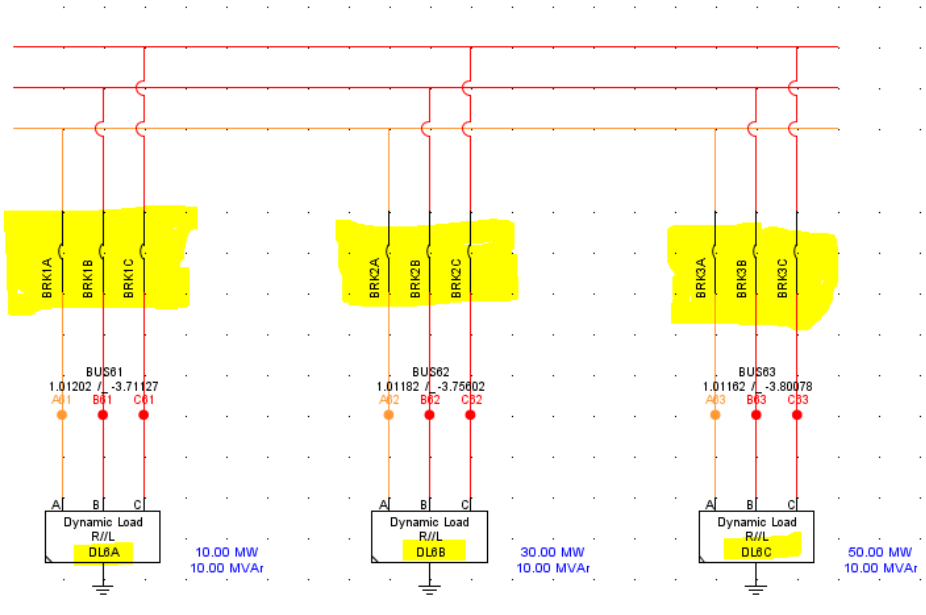

Figure 22 : Individual loads connected with the circuit breakers being controlled to enable UFLS.

The switching actions of the DL6A, DL6B, and DL6C circuit breakers are controlled by the received control signals from the SKT component, placed in the draft file, as illustrated on Fig. 23.

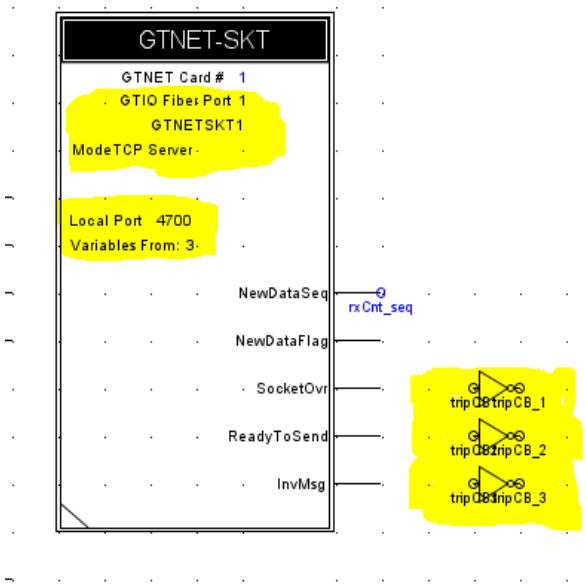

Figure 23 : SKT component with the negation of the received CBs control signals.

7.2 Synchro-measurement Application Development Framework

The UFLS scheme is implemented in MATLAB by using Synchro-measurement Application Development Framework (SADF) library and executed in online fashion. The SADF is a MATLAB supported library to facilitate simplified design and online validation of advanced closed-loop control Wide Area Monitoring, Protection, and Control (WAMPAC) applications, as well as PMU/PDC performance and compliance verification under realistic conditions. The SADF enables a seamless integration between the Synchronized Measurement Technology (SMT) supported electric power system and synchro-measurement supported user-defined applications. This is done by online receiving and parsing of IEEE Std. C37.118-2005 and C37.118.2-2011 specified machine-readable messages into a human-readable MATLAB format, as illustrated on Fig 24. SADF enables receiving of TCP, UDP, or TCP/UDP synchro-measurement data stream by using either "commanded" or "spontaneous" mode. Combining this library with MATLAB's signal processing and visualization functions allows mastering the design and validation of complex WAMAPC applications.

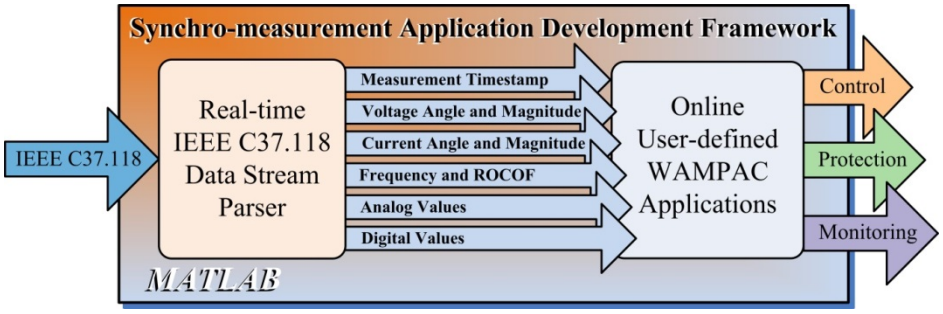

Figure 24 : Synchro-measurement Application Development Framework

7.3 Exercise task and expected results

The main goal of the exercise is to perform a simple 3 stage UFLS scheme on a IEEE 9-bus system by applying following 3 steps:

1. Shed the DL6A load (10MW) if PMU measured frequency of any generator drops below 59.2 Hz.
2. If the frequency is still dropping, shed DL6B load (30MW) if the frequency drops below 58.8 Hz.
3. If the frequency is still dropping, shed the DL6C load (50MW) if the frequency drops below 58 Hz.

The complete UFLS procedure is implemented in MATLAB by using SADF.

To edit the UFLS procedure execute following steps:

1. Open MATLAB and navigate to directory "/SADF_UFLS".
2. To edit the UFLS script type "edit UFLS" into MATLAB command window.

Finally, to perform the simulation, please execute the following steps:

3. Open and compile the "IEEE 9 Bus Power System_UFLS.dft" on a suitable rack, containing PMU and SKT functionality, typically by using a single GTNEx2 card.
4. Open the "IEEE 9 Bus Power System_UFLS.sib" and start the simulation
5. Open MATLAB and navigate to directory "SADF_UFLS".
6. Run the UFLS.m script in MATLAB command window by executing "run SADF_run".
7. Verify that the circuit breakers are closed. Lights should be turned ON as illustrated on Fig 25.

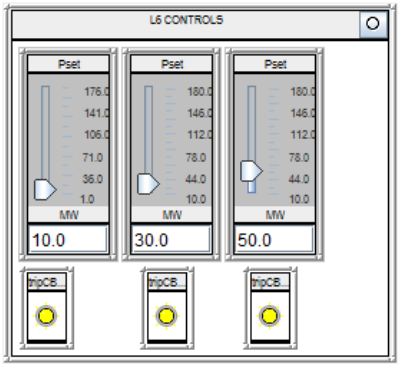

Figure 25 : LEDs on indicate that the CBs are closed.

8. You should be able to automatically receive the PMU measurements in MATLAB and plot the PMU frequency measurements as illustrated on Fig 26.

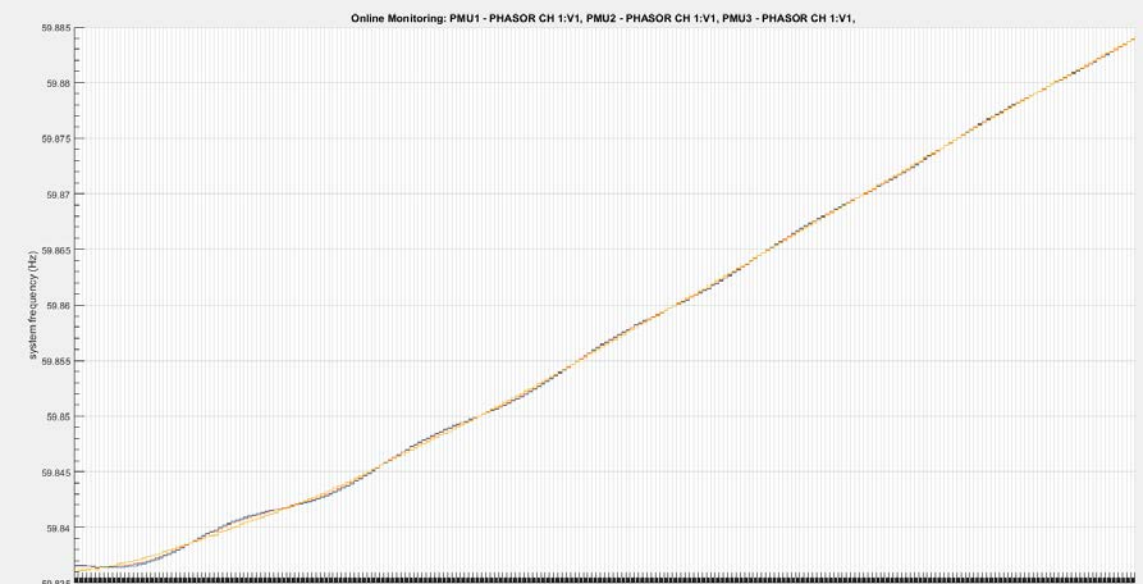

Figure 26 : IEEE 9-bus online generator terminal bus frequency measurements.

9. Increase the active power consumption of the DL8 load, by moving the slider to 200 MW. UFLS Action 1 and Action 2 should be executed as seen in the MATLAB command window. Observe Runtime and SADF command window output, frequency plot, and when the PMU measured frequency drops below predefined threshold, the tripping action occurs, which actions can be seen in the online updated frequency monitoring figure (Fig 27).

Figure 27 : Indication of UFLS actions observed in the PMU frequency measurements.

10. Decrease the the DL8 load slider to 100MW. As a result this will increase the system frequency, and if it goes above 60.5 Hz, then the UFLS is automatically reset, which re-closes all the opened CB of DL6A, DL6B, or DL6C, as seen in Fig 28.

Figure 28 : Reset of UFLS actions observed in the PMU frequency measurements.

11. Increase the active power consumption of the DL8 load, by increasing the slider to 300MW. UFLS Action 1, Action 2, and Action 3 should be executed as seen in the MATLAB command window and observed in PMU frequency measurements of the MATLAB plot (Fig 29).

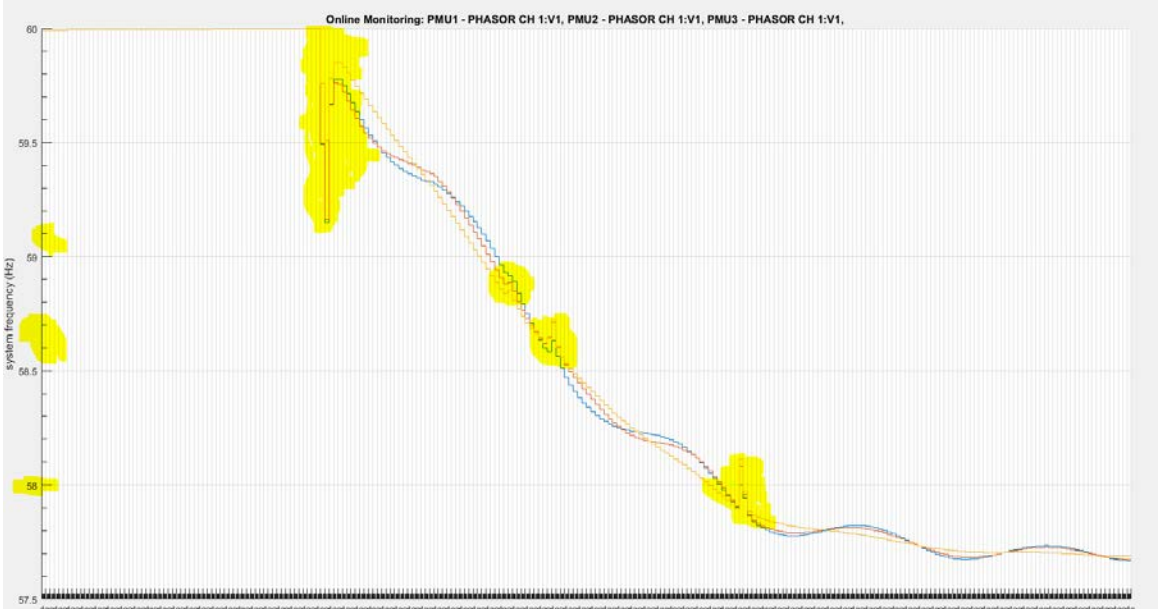

Figure 29 : Indication of the executed all UFLS actions, observed in the PMU frequency measurements.