

Keysight PROPSIM

ATE Environment And Practices

Application Note

Notices

Copyright Notice

© Keysight Technologies, Inc. 2015-2021
No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Keysight Technologies, as governed by United States and international copyright laws.

Manual Part Number

F8800-93106

Revision

Revision 2.2, 29th June 2021

Published by:

Keysight Technologies Finland Oy
Elektroniikkatie 10
90590 Oulu, Finland

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Third Party Software

This software uses MATLAB Runtime.
MATLAB®. © 1984 – 2016. The MathWorks, Inc.

Open Source Licensing

Portions of this software are licensed by third parties including open source terms and conditions, copies of which may be found from "opensource" directory in PROPSIM software installation folder.

Declaration of Conformity

Declarations of Conformity for this product and for other Keysight products may be downloaded from the Web. Go to <http://www.keysight.com/go/conformity> and click on "Declarations of Conformity." You can then search by product number to find the latest Declaration of Conformity.

U.S.Government Rights

The Software is "commercial computer software," as defined by Federal Acquisition Regulation ("FAR") 2.101. Pursuant to FAR 12.212 and 27.405-3 and Department of Defense FAR Supplement ("DFARS") 227.7202, the U.S. government acquires commercial computer software under the same terms by which the software is customarily provided to the public. Accordingly, Keysight provides the Software to U.S. government customers under its standard commercial license, which is embodied in its End User License Agreement (EULA), a copy of which can be found at

<http://www.keysight.com/find/sweula>. The license set forth in the EULA represents the exclusive authority by which the U.S. government may use, modify, distribute, or disclose the Software. The EULA and the license set forth therein, does not require or permit, among other things, that Keysight: (1) Furnish technical information related to commercial computer software or commercial computer software documentation that is not customarily provided to the public; or (2) Relinquish to, or otherwise provide, the government rights in excess of those rights customarily provided to the public to use, modify, reproduce, release, perform, display, or disclose commercial computer software or commercial computer software documentation. No additional government requirements beyond those set forth in the EULA shall apply, except to the extent that those terms, rights, or licenses are explicitly required from all providers of commercial computer software pursuant to the FAR and the DFARS and are set forth specifically in writing elsewhere in the EULA. Keysight shall be under no obligation to update, revise or otherwise modify the Software. With respect to any technical data as defined by FAR 2.101, pursuant to FAR 12.211 and 27.404.2 and DFARS 227.7102, the U.S. government acquires no greater than Limited Rights as defined in FAR 27.401 or DFAR 227.7103-5 (c), as applicable in any technical data.

Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED "AS IS," AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, KEYSIGHT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. KEYSIGHT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. SHOULD KEYSIGHT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT WILL CONTROL.

Customer support

It is our goal to provide you with excellent Customer Support. To request assistance with any aspect of your test system, please create a Help Desk Request (HDR) using the NES Wireless Solutions Help Desk. For other queries, please email customersupport.di@keysight.com.

To access the NES Wireless Solutions Help Desk, and to download the latest releases of software and documentation, please log in to [Keysight Support](#). On the Keysight Support home page, in the Additional Support, click the link for the NES Wireless Solutions Help Desk, or in the Assets, click Keysight Software Manager for software downloads.

CONTENTS

1	INTRODUCTION	5
1.1	Connectivity methods	5
1.1.1	VISA.....	6
1.1.2	LAN Control.....	6
1.2	Command Parsing.....	7
1.3	Command Synchronization.....	8
1.4	10MHz Reference Source	9
1.4.1	Selecting the System 10MHz reference source	9
1.4.2	Switching the reference source	9
1.5	Terms and abbreviations.....	9
1.6	Document history.....	10
2	COMMAND SEQUENCING	12
2.1	Emulation states	12
2.2	Loading and storage of emulation files	12
2.2.1	File location tips.....	12
2.2.2	Loading procedures	13
2.2.3	Loading and opening an emulation file	13
2.2.4	Avoiding timeout errors during emulation file loading.....	13
2.3	Commands that shall be issued when emulation is in closed -state	13
2.3.1	Loading an emulation file	13
2.3.2	Multiemulator synchronization commands	14
2.4	Commands that shall be issued when emulation is in stopped -state.....	14
2.4.1	Creation / deletion of internal Interference sources (Option)	14
2.4.2	Setting Mobile speed	14
2.4.3	Setting RF local parameters	14
2.4.4	Input Level Control and Autoset-functions	14
2.4.5	Emulation control -commands	14
2.4.6	Trigger configuration -commands.....	14
2.5	Commands that can be issued when emulation is in running -state	14
3	EMULATION ARCHITECTURE.....	16
3.1	Terminology and concepts in Emulations	16
3.2	Traversing Emulation topology	17
4	INTERFERENCE SOURCES	19
4.1	CW interference	19
4.2	AWGN interference.....	20
4.3	Synchronized start-up of multiple Emulators	21
5	EXAMPLES	22
5.1	Establishing VISA session	22
5.2	Basic I/O using VISA	23
5.3	Loading an emulation file	24

5.4	Checking the error queue	25
5.5	Example case: throughput as function of output gain	26

1 INTRODUCTION

This document describes some aspects of general instrumentation; the focus is on using PROPSIM F-series device remotely through instrumentation interfaces, e.g. GPIB and LAN.

Primary audience for this document is measurement automation specialists, who already have some knowledge of instrumentation basics.

For complete list of PROPSIM commands, see PROPSIM User reference, chapter ‘Remote Control, ATE command interface’.

1.1 Connectivity methods

PROPSIM radio channel emulator features both GPIB and LAN connectivity. The emulator can be controlled through these interfaces to be used as a part of a larger test system or just automating simple measurements.

PROPSIM can be used with multiple vendors’ connectivity hardware and software, e.g. USB or PCI GPIB adapters, VISA or custom-made communication software.

The emulator can be used with multiple programming-, computing- and test development languages, including e.g., but not limited to: Microsoft Visual Studio, Microsoft Visual Basic for Applications, Python, Perl, Keysight VEE Pro, MathWorks MATLAB, National Instruments LabVIEW, etc.

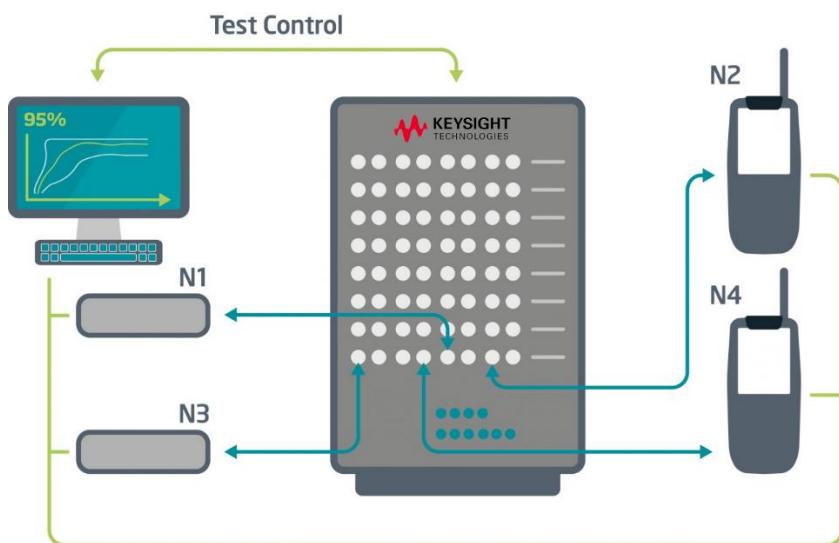


Figure 1. Example of laboratory setup

PROPSIM can be accessed remotely with various tools, e.g. VISA, telnet or custom-made internetworking software. Remoting interfaces are sometimes called ATE interfaces.

PROPSIM can be used either in:

- UI control (manual)
 - or -
- LAN Remote control
 - or -
- GPIB Remote control (F32/FS8/F8)

This means that when remotely controlling the emulator, emulation file must be closed in UI interface before attempting to load the emulation file through remote interface(s) – and vice versa.

Note: Starting from Release 2, PROPSIM F64 can be used in both local and remote modes, and it is possible to switch between the modes easily in normal use even when an emulation is open in the Emulation Control View.

In F32/FS8/F8, all commands related to a certain test sequence (including e.g. loading an emulation file, setting centre frequencies and gains, adding an interference source, etc.) have to be programmed through same remote interface (GPIB or LAN). This differs from traditional Remote/Local feature where the developer may change settings and parameters while having their system paused for debug purposes. If an emulation file is left open in UI control or other remote interface, it is not possible to load an emulation file through another remote interface.

If an emulation file has not been opened, its parameters cannot be set for emulator operation.

1.1.1 VISA

It is recommended to control the test environment using VISA as communication layer. As VISA is specified and structure maintained by IVI Foundation, changing VISA between different vendors does not impact program development; function prototypes stay the same.

Another major speed up using VISA to program communication is that connectivity through LAN / GPIB / Serial links etc. differ mostly in opening a session to instrument. After connectivity session is open, bus operations can be implemented using very same functions regardless of communication medium.

VISA also provides its own error handling, thus enabling the Test System designer to concentrate on progressing to actual system design, rather than writing and debugging low level I/O code.

1.1.2 LAN Control

LAN control has slight speed gain compared to GPIB, regarding raw time taken to do operations through command bus. Control can be implemented using standard networking I/O libraries, VISA or custom-made internetworking software.

1.1.2.1 TCP/IP socket

PROPSIM connectivity over LAN implements TCP/IP socket protocol. Fixed TCP/IP port for PROPSIM is 3334. To view the IP address of PROPSIM, select **Configuration > Device Information** in the navigation bar, see Figure 2. The IP address can be changed in **Network** section of Windows Control Panel. To open the Control Panel, select **System > Windows Control Panel** in the navigation bar.

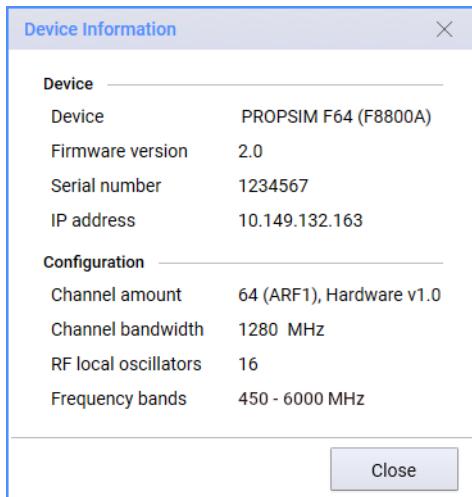


Figure 2. Device Information dialog

1.1.2.2 Telnet considerations

If Telnet is used to send ATE commands to PROPSIM, every line should be ended with CONTROL+ENTER. Only this way Telnet will send proper line feed; PROPSIM requires only Line Feed (LF) as EOL sequence.

1.1.2.3 VISA considerations

VISA resource identification for PROPSIM is of format TCPIP<interface number>::<ip address of PROPSIM>::3334::SOCKET, e.g. 'TCPIP0::192.32.6.100::3334::SOCKET'.

With TCPIP socket VISA session, the protocol does not implement EOI after message; VISA VI_ATTR_TERMCHAR_EN attribute state must be set to 1 for session, otherwise socket reads will result in timeout.

To avoid erroneous socket lockups during development, it is feasible to e.g. add a viClose call to VISA error handler when VI_ATTR_INTF_TYPE attribute value is VI_INTF_TCPIP and error is raised. The socket will stay open until closed; error situations might leave the socket open thus preventing further communication.

During instrument / class driver development, the developer might benefit also from a method that closes the VISA session on reopen condition e.g. in crash recovery. GPIB use does not require this.

Please note that successful reads from TCPIP interface using VISA socket session return VI_SUCCESS_TERM_CHAR or VI_SUCCESS_MAX_CNT depending on situation, instead of VI_SUCCESS.

1.1.2.4 LAN remote control general requirements

Operational LAN connectivity generally requires the following:

- The test system host pc is on same IP subnet and has physical connection with PROPSIM
- The IP address specified does not conflict with other IP addresses in same subnet
- The socket specified is not locked or in use by other software
- The VI_ATTR_TERMCHAR_EN attribute for opened VISA session is enabled
- The termination character is \n (10(dec), 0xa(hex), line feed)
- The socket that has successfully been in use has not been timed out / stuck from test SW side thus preventing further communication
- Emulation files are closed on UI control and GPIB remote interface
- Firewall / anti-virus SW does not interfere with communication
- Connectivity hardware (cables, Ethernet switches etc.) are operational

1.2 Command Parsing

When concatenating command strings to be sent to measurement equipment via communication layer, they should be parsed with care. If there is e.g. an unintentional whitespace in the end of command string, such as 'SYST:ERR? \n', or 'SYST:ERR?\r\n', this results as command error:

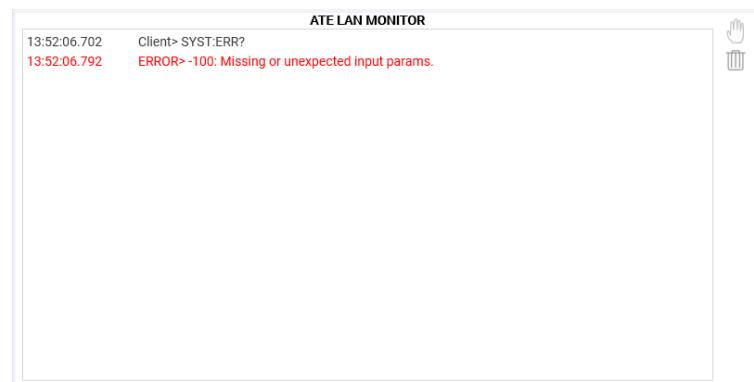


Figure 3. Command error caused by whitespace

These errors can be hard to trace down as whitespaces are not displayed in remote monitor. Various vendor provided I/O trace applications can help to trace down such issues:

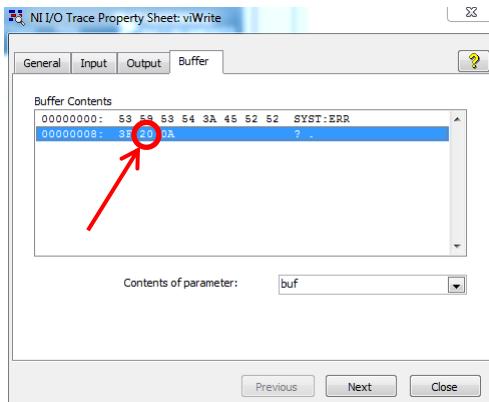


Figure 4. I/O monitor buffer displaying whitespace

In the example above, there is a space character (0x20, shown 2nd character on 2nd line of buffer) between question mark and line feed that was causing Command Error when querying for error queue.

It is also recommended to allocate enough space for instrument replies in I/O operations; using e.g. 256-character buffer in viRead might truncate part of long response:

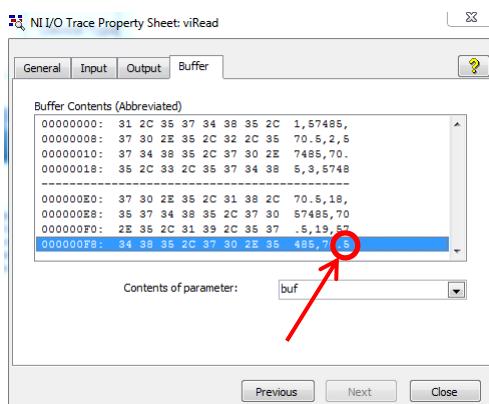


Figure 5. Truncated I/O read operation caused by inadequate buffer length

With adequate buffer length (4096 bytes in this case), complete response to a query can be read from PROPSIM:

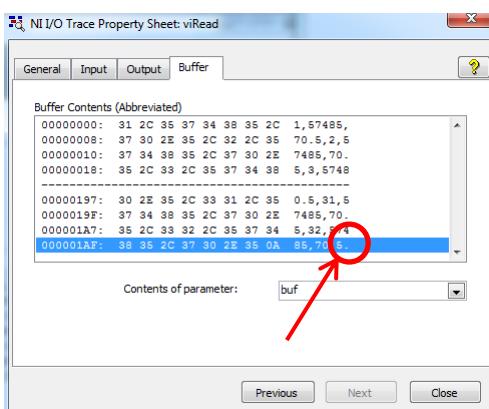


Figure 6. Complete I/O read operation

1.3 Command Synchronization

ATE commands are recommended to be synchronized. Most commonly used synchronization method is *OPC? Query.

Command synchronization procedures will enable good measurement repeatability. It is recommended to always use synchronization with every instrument, if applicable. Compromising synchronization methods by measurement speed is always a risk.

If synchronization methods are not used in measurement setup, especially in fast measurement loops, the designer might end up measuring controlled error instead of intended quantity; this may cause related measurement hardware being on (same) settling phase instead of fully settled during measurement.

Issuing "*OPC?" -query after command through remote interface will return '1' when last pending operation has been executed. This is the recommended method, which provides always adequate synchronization to test sequence. The "*OPC?" query can also be separated with semicolon, e.g. "CALC:FILT:FILE c:\Emulations\emulation.smu;*OPC?".

Note that "OPC?" -query may time out during lengthy operations; VISA default is usually 2000ms.

1.4 10MHz Reference Source

One cornerstone of automated measurement set-ups is a high-quality 10MHz reference chain. This is highly recommended to be taken care of when building a laboratory setup or test system.

PROPSIM shall be connected to same high quality 10MHz reference frequency source with other measurement equipment, either acting as source for all equipment or consuming same external reference. Reference source can be set by clicking the **INT REF / EXT REF** button in the status bar at the bottom of the PROPSIM GUI:

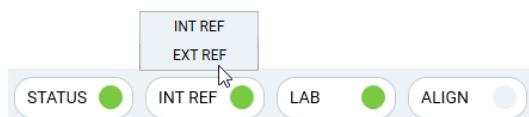


Figure 7. Reference Clock source selection in status bar

Reference source can also be set by ATE command:

- ROUT:PATH:REF <source>. Possible values for source are 'EXT' and 'INT'.

Used reference source can be queried by

- ROUT:PATH:REF?

1.4.1 Selecting the System 10MHz reference source

The selected 10MHz reference source shall have amplitude of >0dBm. To maximize accuracy in e.g. synchronized start-ups featuring multiple PROPSIMs, levels of ~+10dBm are recommended. High quality 10MHz sources having GPS lock ability will provide maximum accuracy.

When building a measurement setup, it is recommended to verify the 10MHz reference path with care. If the measurement setup is constructed so that the whole reference chain is daisy-chained i.e. there is only one chosen source/driver and all equipment are connected after each other, user has to make sure none of the instruments on the chain are set to use internal reference source apart from the primary 10MHz source device.

If e.g. one of the instruments in the middle of the reference chain has reference source set as internal, measurement setup will have multiple 10MHz sources; the measurement is not synchronized any more. This stands also for programmatical setups that do not use every instrument in the 10MHz reference chain. If such instruments will switch to internal 10MHz reference at power-up and the source is not changed programmatically or from instrument UI before doing the actual measurement, it will have impact on delay accuracy with certain setups and instrument configurations.

1.4.2 Switching the reference source

When changing PROPSIM 10MHz reference source from external to internal, it takes roughly 15 minutes for the internal reference oven oscillator to warm up adequately. During this period, the emulator will display warnings in ATE or UI window reading 'Warning: System clock unlocked'.

This is intended behaviour; if internal oscillator power would be on constantly, it would interfere with external 10MHz reference.

1.5 Terms and abbreviations

Name	Description
ATE	Automatic Test Equipment
AWGN	Additive White Gaussian Noise

Name	Description
C/I	Carrier-to-Interference
CR	Carriage Return, 13(dec), 0xd(hex)
CW	Carrier Wave = Continuous Wave = Sine Wave
EOI	End-or-identify, GPIB control signal
EOL	End Of Line sequence
GPIB	General Purpose Interface Bus, IEEE 488.2
GPS	Global Positioning System
GUI	Graphical User Interface
LAN	Local Area Network
LF	Line Feed, 10(dec), 0xa(hex)
MIMO	Multiple Input Multiple Output
PCI	Peripheral Component Interconnect, an industry-standard bus for attaching peripherals to computers
RX	Abbreviation for 'receive'
TCP/IP	Transmission Control Protocol / Internet Protocol
TX	Abbreviation for 'transmit'
UI	User Interface
USB	Universal Serial Bus
VISA	Virtual Instrument Software Architecture

1.6 Document history

The following table lists the main changes to issues of this document.

Issue	Date	Summary of Changes
1.0	Apr 2015	Release of PROPSIM – Application Note ATE environment and practices
1.1	Jan 2016	Template updated
1.2	Jan 2018	Rebranded using Keysight template.
1.3	Jun 2019	PROPSIM F64 changes.
2.0	Feb 2020	PROPSIM GUI changes.
2.1	Mar 2021	Terminology changes.
2.2	Jun 2021	Customer support information updated.

2 COMMAND SEQUENCING

2.1 Emulation states

Essential part of operating PROPSIM remotely is correct command sequencing. State of emulation is divided in four parts:

- Closed
When PROPSIM has just booted up or emulation file is manually closed after opening, emulation is in closed state.
- Stopped
When an emulation file has been loaded to PROPSIM but has not been started, or
- Running
When an emulation file has been issued a run command.
- Editing
When an emulation file has been opened but it has not been connected to PROPSIM hardware. This is useful when you want to change some settings of the emulation (for example, frequencies or connectors) before running it on the PROPSIM hardware.

In addition to these, Emulation Bypass –states are available. Issuing Butler bypass pauses the emulation and replaces all Fading Channels with static one-path model having average level of all paths in corresponding Channel Model. Delay of bypass model is the shortest delay in corresponding Channel Model.

Butler bypass also modifies phase components of individual Channels according to Butler matrix, which improves diversity so that MIMO links can be established.

Calibration bypass –state replaces Fading Channels with one-path model having -10dB Channel gain. With this state, it is possible to perform phase- and amplitude calibrations of Test Setup.

Part of the commands for remoting PROPSIM operation need to be issued when emulation is not open, or emulation is stopped.

Majority of commands can be issued during emulation run. These include e.g. setting channel center frequencies and input / output levels.

All emulation parameter commands need to be issued after emulation is loaded, otherwise they will result in an error.

Possible errors can be queried from PROPSIM by:

- SYST:ERR?
This command will return a string containing oldest error / event message in the error queue, or '0,"No Error"' when errors are not present.

Detailed list of errors and their descriptions are provided in PROPSIM User Reference, chapter 'Running view errors'.

2.2 Loading and storage of emulation files

2.2.1 File location tips

As PROPSIM is measurement equipment that requires annual calibration, one might want to consider portability point of view with their emulation files. If files reside on local hard drive, they need to be backed up when sending the emulator to calibration.

2.2.1.1 Local PROPSIM locations

If user prefers local placement of Emulation files, it is recommended that files would be placed in d:\User Emulations\ -folder, as this folder will be transferred in PROPSIM Service Centres if hard drive replacement is needed due to e.g. PROPSIM OS update.

2.2.1.2 Remote file locations

If user prefers remote placement of Emulation files, it is recommended that files would be stored and compiled in either corporate network share, network attached storage, or other removable media. Network share methods also allow the very same emulation files to be used in multiple PROPSIM setups.

2.2.2 Loading procedures

During development of a test system, the designer might run some parts of code instead of whole application. This might yield to a situation where different emulation file is open than intended. It is feasible for the designer to implement "DIAG:SIMU:CLOSE" to simulation loading function in the test system; closing an emulation file if it's not loaded will not raise errors in remote interface thus being the fail-safe way of loading procedure.

2.2.3 Loading and opening an emulation file

In order to operate as intended, PROPSIM requires an emulation file to be open. An emulation file can be opened by "CALC:FILT:FILE <filename>".

2.2.4 Avoiding timeout errors during emulation file loading

When loading up an emulation file to PROPSIM; especially emulation file being relatively large, the test system designer might run to timeout problems. This might show as error -400,"Query error":

```
Client> CALC:FILT:FILE c:\emulations\Emulation.smu
Client> *OPC?
Client> CALC:FILT:CENT:CH 1,2450.00
ERROR> -400,"Query error"
Client> SYST:ERR?
Server> -400,"Query error"
```

This is caused by VISA or socket timeout in reading response to "*OPC?" query; PROPSIM receives next command before test system has read the previous response from socket connection, thus throwing an error. This violates correct communication sequence. The response to a query shall be read in test system before sending other commands to test equipment.

Note: VISA default 2000ms timeout is not sufficient when loading large files; the timeout for PROPSIM VISA session / socket read has to be incremented at least temporarily during emulation file loading. The timeout value would be feasible to parameterize at least during development; the correct value for an arbitrary situation can be obtained with experience.

Loading emulation files shall be synchronized using "*OPC?" query, e.g.

1. Empty the error queue (if desired)
2. Close emulation file ("DIAG:SIMU:CLOSE")
3. Get VISA session / socket timeout, store as *val*
4. Set VISA session / socket timeout to sufficient value
5. Load an emulation file ("CALC:FILT:FILE <filename>")
6. Query operation complete-flag ("*OPC?")
7. Read response to query
8. Replace the VISA session / socket timeout to *val*
9. Check error queue

2.3 Commands that shall be issued when emulation is in closed -state

2.3.1 Loading an emulation file

- CALC:FILT:FILE <filename>

2.3.2 Multiemulator synchronization commands

- Setting synchronization cable length; SYST:MSIM:CABLE <cable length>
- Setting multiemulator configuration; SYST:MSIM:CONFIG <position>,<total>

Note: These commands can be queried during runtime.

2.4 Commands that shall be issued when emulation is in stopped state

2.4.1 Creation / deletion of internal Interference sources (Option)

- Creating an interference source; OUTP:INTERF:ADD <output number>,<interference identification>,<interference type>. Possible values for interference type are 1 (AWGN), 2 (CW).
- Removing an interference source; OUTP:INTERF:REM <name>

Note: Interference parameters (Interference strategy, power, level, bandwidth, data rate, Eb/No value, frequency, ratio and ratio mode) can be set during runtime where applicable.

2.4.2 Setting Mobile speed

- DIAG:SIMU:MOB:MAN:CH <channel>,<speed>
- DIAG:SIMU:MOB:MAN:CHG <channel>,<speed> <optional unit>

2.4.3 Setting RF local parameters (F32/FS8/F8)

- Set specified RFLO source to channel; ROUT:PATH:RFLO:CH <channel>,<RF local source>. Parameters for <RF local source> are INT or DIRECT.
- Set optimal RFLO sources automatically; ROUT:PATH:RFLO:AUTO

2.4.4 Input Level Control and Autoset-functions

- Automatic Input Level Control state; INP:LEV:AUTO:ENA <input number>,<state>
- Set average input level and crest factor values; INP:LEV:AUTOSET <input number>,<measurement time>. Measurement time can be 1, 3, 10 or 30 (seconds). If input number is set to 0, all inputs are autoset simultaneously.
- Get last input level measurement value; INP:MEAS:RES:GET? <input number>

2.4.5 Emulation control -commands

- Run emulation; DIAG:SIMU:GO
- Go to a CIR on specified channel; DIAG:SIMU:GOTO <channel number>,<CIR>
- Step emulation to next CIR; DIAG:SIMU:STEP
- Continue paused emulation; DIAG:SIMU:CONT
- Setting static model (bypass) states; DIAG:SIMU:MODEL:STATIC <state>. Valid parameters for state are 0 (bypass disabled), 1 (Channel Model bypass), 2 (Butler), 3 (Calibration bypass). Bypass can also be issued when emulation is running.

Note: If emulation was running before bypass was issued, disabling bypass continues the emulation.

2.4.6 Trigger configuration -commands

- Set emulator triggering configuration; DIAG:SIMU:TRIG:CONF <triggering mode>,<mode parameter>.
- Enable or disable external trigger; DIAG:SIMU:TRIG:SET <ON|OFF>

2.5 Commands that can be issued when emulation is in running -state

Majority of remoting commands can be issued during emulation run. See PROPSIM User Reference, chapter 'Remote Control, ATE command interface' for complete list of commands. Typical commands that are issued during Emulation run include e.g.

- Setting channel centre frequencies; CALC:FILT:CENT:CH <channel number>,<MHz>
- Closing an emulation file; DIAG:SIMU:CLOSE. Note! DIAG:SIMU:CLOSE can be always run before CALC:FILT:FILE <filename> to avoid errors in test system side.
- Pausing emulation; DIAG:SIMU:STOP. Note! When DIAG:SIMU:GO is issued, emulation will continue from CIR it was stopped on.
- Stop emulation, rewind to start; DIAG:SIMU:GOS. Note! After this command, emulation is in stopped state and needs to be resumed using DIAG:SIMU:GO.
- Enabling / disabling input / output; INP:EN <input number>,<set value> | (OUTP:EN ...)
- Setting input measurement mode; INP:MEAS:MODE:SET <input number>,<measurement mode>. Parameters for <measurement mode> are 0, 1, 2 and 3, denoting disabled, basic, continuous and burst.
- Setting input / output level value(s); INP:LEV:AMP:CH <input number>,<amplitude value> | (OUTP:LEV:AMP ...)
- Setting input / output phase(s); INP:PHA:CH <input number>,<phase register value> | (OUTP:PHA:CH ...)
- Setting input crest factor(s); INP:CRE:SET <input number>,<crest factor value>
- Setting output gain(s); OUTP:GAIN:CH <output number>,<gain value>
- Setting interference parameters; OUTP:INTERF: -branch, excluding creation / deletion
- All query-type commands, e.g. CALC:FILT:CENT:CH? <channel number>, OUTP:INTERF:POW:LIM? <interference identification>

3 EMULATION ARCHITECTURE

3.1 Terminology and concepts in Emulations

PROPSIM Emulations have architectural concepts Input, Output and Channel. Inputs and outputs are physical channel connections in PROPSIM, whereas ‘Channel’ refers to logical channel; a digital domain which implements majority of fading between emulator physical RF connections.

Note that the emulation diagrams in the Emulation Control View of the PROPSIM GUI (starting from Release 2) display links instead of individual channels. Links are higher-level radio links between devices and contain one or more channels.

Channels can be mapped between physical inputs and physical outputs; one logical channel can have only one logical input and one logical output. They are connected to physical inputs and outputs in emulation so, that one physical input can be connected to one or several logical channels and one or several logical channels can be connected to one physical output.

MIMO emulation topologies may contain different number of channels compared to physical input or output counts; it is recommended that channel routings are not hardcoded but parametrized or traversed in the test system side. This method will also eliminate errors in defining the emulation topology to test system.

In the Running View, MIMO emulation topology can be selected to be displayed as grouped or ungrouped by checking / unchecking checkbox in Running View > Settings > Grouped MIMO channels:

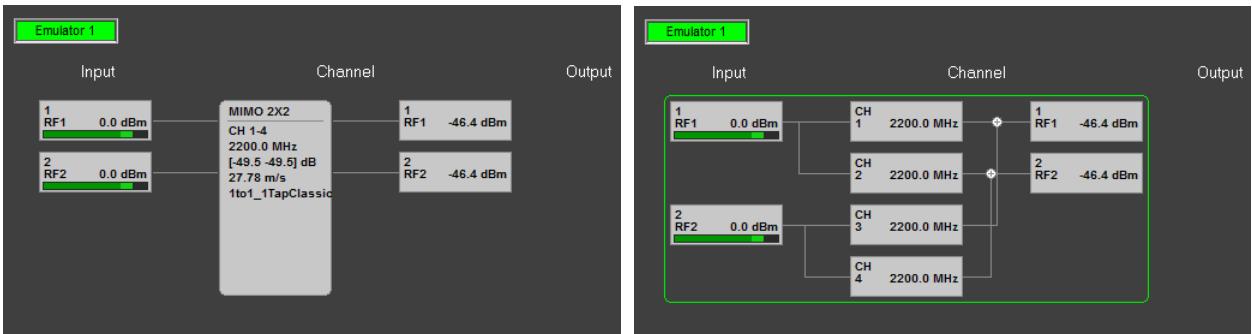


Figure 8. MIMO2x2 emulation, grouped (left) and ungrouped (right)

Figure 9 shows a corresponding MIMO 2x2 emulation in the Emulation Control View of the PROPSIM GUI:

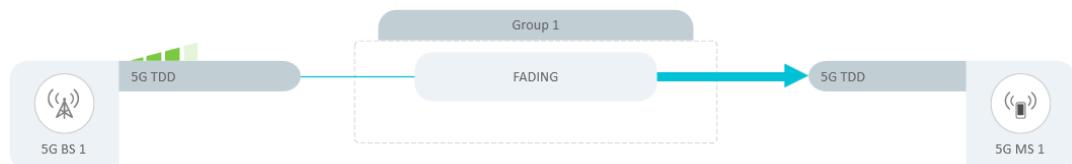


Figure 9. MIMO 2x2 emulation diagram in Emulation Control View

The MIMO 2x2 emulation shown in Figure 8 occupies 2 physical inputs, 2 physical outputs and consumes 4 logical channels. In the emulation, channels are mapped to physical inputs and outputs as follows:

- Channel 1 : input 1 → output 1
- Channel 2 : input 1 → output 2
- Channel 3 : input 2 → output 1
- Channel 4 : input 2 → output 2

For example, 4x2 full duplex MIMO group occupies 6 Full Duplex ports (or 6 inputs and 6 outputs with external summing) and 16 logical channels.

Emulation configuration can be traversed using remote commands specific to this purpose.

3.2 Traversing Emulation topology

The number of inputs, logical channels and outputs in loaded emulation can be queried with

- **DIAG:SIMU:MOD:INFO?**

The command returns information in format <number of inputs>,<number of logical channels>,<number of outputs>, response in e.g. MIMO4x4 is '4,16,4'.

Some remote commands are used to set properties of physical channels, such as setting input and output levels / phases, setting input crest factor etc. See chapters 'Channel Input Settings' and 'Channel Output Settings' in PROPSIM User Reference for details.

Some remote commands affect logical channel properties. These include e.g. setting mobile speed or gain imbalance. See chapter 'Channel Settings' in PROPSIM User Reference for details.

Loaded emulation may also contain channel groups instead of individual channels. This is typical for MIMO emulations:

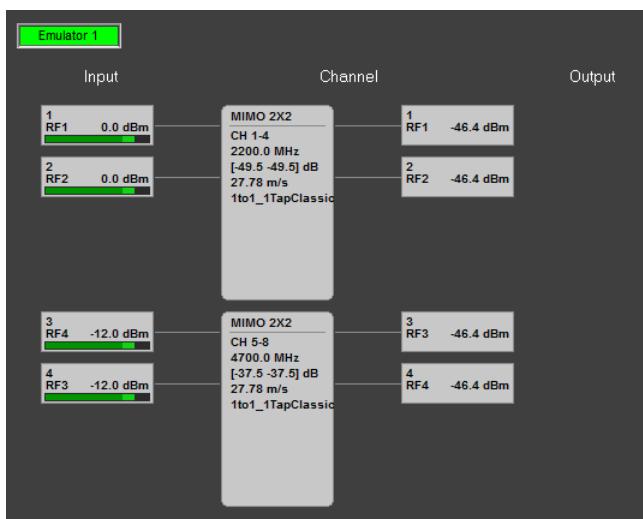


Figure 10. Emulation with 2 MIMO 2x2 groups in Running View



Figure 11. Emulation with 2 MIMO 2x2 groups in Emulation Control View

Group topology can be traversed with remote commands; first, retrieve number of groups in emulation:

- **GRO:GET?**

This command will return the number of groups, e.g. '2'. With group count, user can query for physical inputs, logical channels and physical outputs in the emulation by:

- **GRO:IN:GET? <group number>**
- **GRO:OUT:GET? <group number>**
- **GRO:CH:GET? <group number>**

These commands return the related values in CSV format, e.g. '1,2,3,4'.

Center frequency and channel group mobile speed –settings affects entire channel group the logical channel is member of, instead of single logical channel:

- CALC:FILT:CENT:CH <channel>,<frequency>
- DIAG:SIMU:MOB:MAN:CHG <channel>,<speed> <optional unit>

With these commands, parameter to be set will apply to all channels in the group that contains *channel* to be set. Channels belong to the same group if they have same physical input or same physical output.

After obtaining logical channel count, logical channels can be looped through to get channels' physical connections with query

- ROUT:PATH:CONN? <channel>

The query will return result in format: <emulator, in, out, inlo, outlo>, e.g. '1,RF-1,RF-1,-,-'. This information can be used in measurement function later on to e.g. drive RF matrices.

Note: The physical channel routing might use nonlinear physical channel mapping instead of linear; e.g. 1, 5, 2, 6, ...

4 INTERFERENCE SOURCES

PROPSIM features internal interference generators as options. Supported interference types are CW, AWGN, filtered AWGN (F8, F32, FS8 only) and Adjustable AWGN (F8, F32, FS8 only). All of these interferer types are available as constant or relative (in terms of interference level). The behaviour is set via interference strategy - property. The interference sources can also be set on and off remotely.

Note: In F8, F32, and FS8, AWGN is full bandwidth of PROPSIM, filtered AWGN is fixed 30MHz bandwidth AWGN, adjustable AWGN maximum bandwidth is (PROPSIM BW / 2). In F64 and FS16, all AWGN parameters are adjustable.

When controlling the emulator remotely, interference sources may have to be created and/or their properties set remotely. The settable interference properties differ by interference type and its interference strategy.

When building up a test system that will feature interference sources, it is feasible to add boundary checking for several parameters in the test system (or driver if applicable), because system design usually does not constrain the type, i.e. interference parameters might not be hard-coded. This may lead to situation, where caused by misparametrization, interference properties are set out of valid range thus rendering the measurement void.

4.1 CW interference

For constant CW interference where constant interference power is desired, interference strategy shall be set to 1:

- OUTP:INTERF:STRAT:SET <interference identification>,<strategy>
Valid parameters for <strategy> are
 - 0, constant carrier-to-interference ratio (C/I).
 - 1, constant interference power
- OUTP:INTERF:STRAT:GET? <interference identification>

For constant CW interference, parameters that can be set and limits for valid ranges queried are

- Interference frequency
 - OUTP:INTERF:FREQ:SET <interference identification>,<MHz>
 - OUTP:INTERF:FREQ:GET? <interference identification>
 - OUTP:INTERF:FREQ:LIM? <interference identification>
- Interference power
 - OUTP:INTERF:POWER:SET <interference identification>,<dBm>
 - OUTP:INTERF:POWER:GET? <interference identification>
 - OUTP:INTERF:POWER:LIM? <interference identification>

For relative CW interference where constant carrier-to-interference ratio is desired, interference strategy shall be set to 0. For relative CW interference (the emulator tracks input signal level and adjusts interference level accordingly), parameters that can be set and limits for valid ranges queried are

- Interference frequency
 - OUTP:INTERF:FREQ:SET <interference identification>,<MHz>
 - OUTP:INTERF:FREQ:GET? <interference identification>
 - OUTP:INTERF:FREQ:LIM? <interference identification>
- Interference ratio
 - OUTP:INTERF:RAT:SET <interference identification>,<dB>
 - OUTP:INTERF:RAT:GET? <interference identification>

- OUTP:INTERF:RAT:LIM? <interference identification>
- Interference ratio mode
 - OUTP:INTERF:RAT:MODE:SET <interference identification>,<mode>
 - OUTP:INTERF:RAT:MODE:GET? <interference identification>

Valid parameters for <mode> are

 - 0, Measure all input signals connected to output and adjust C/I ratio based on combined power of signals. Note that high correlation between input signals may distort measurement.
 - 1, Measure only input signal of the first channel and adjust C/I ratio based on that

4.2 AWGN interference

For constant power AWGN interference, interference strategy shall be set to 1:

- OUTP:INTERF:STRAT:SET <interference identification>,<strategy>
- Valid parameters for <strategy> are
 - 0, constant carrier-to-interference ratio (C/I)
 - 1, constant interference power
 - 2, constant C/I where the interference level is also user-given (AWGN only)
- OUTP:INTERF:STRAT:GET? <interference identification>

Parameters that can be set and limits for valid ranges queried are

- Interference level
 - OUTP:INTERF:LEV:SET <interference identification>,<dBm/Hz>
 - OUTP:INTERF:LEV:GET? <interference identification>
 - OUTP:INTERF:LEV:LIM? <interference identification>
- Interference bandwidth
 - OUTP:INTERF:BANDW:SET <interference identification>,<MHz>
 - OUTP:INTERF:BANDW:GET? <interference identification>
 - OUTP:INTERF:BANDW:LIM? <interference identification>
- Generated bandwidth
 - OUTP:INTERF:BANDW:GEN:SET <interference identification>,<generated bandwidth>
 - OUTP:INTERF:BANDW:GEN:GET? <interference identification>
 - OUTP:INTERF:BANDW:GEN:LIM? <interference identification>
- Frequency
 - OUTP:INTERF:FREQ:SET <interference identification>,<MHz>
 - OUTP:INTERF:FREQ:GET? <interference identification>
 - OUTP:INTERF:FREQ:LIM? <interference identification>
- Frequency offset
 - OUTP:INTERF:FREQ:OFFS:SET <interference identification>,<frequency offset>
 - OUTP:INTERF:FREQ:OFFS:GET? <interference identification>
 - OUTP:INTERF:FREQ:OFFS:LIM? <interference identification>

For relative AWGN interference, where constant carrier-to-interference ratio is desired, interference strategy shall be set to 0. For relative AWGN interference (the emulator tracks input signal level and adjusts interference level accordingly), parameters that can be set and limits for valid ranges queried are

- Interference ratio mode

- OUTP:INTERF:RAT:MODE:SET <interference identification>,<mode>
 - OUTP:INTERF:RAT:MODE:GET? <interference identification>
- Valid parameters for <mode> are
- 0, Measure all input signals connected to output and adjust C/I ratio based on combined power of signals. Note that high correlation between input signals may distort measurement.
 - 1, Measure only input signal of the first channel and adjust C/I ratio based on that
- Interference ratio
 - OUTP:INTERF:RAT:SET <interference identification>,<dB>
 - OUTP:INTERF:RAT:GET? <interference identification>
 - OUTP:INTERF:RAT:LIM? <interference identification>

4.3 Synchronized start-up of multiple Emulators

Multiple PROPSIMs can be started synchronously. In this setup, one PROPSIM acts as commander device and remaining devices act as followers. Connections for synchronization cables can be seen in Active Connectors-window.

See PROPSIM ‘Multi-emulator sync’ Application Note for details on this operation.

5 EXAMPLES

5.1 Establishing VISA session

In order to use VISA as communication layer, successfully opened resource manager and VISA session are required. Session to instrument can be opened using resource identifiers.

VISA resource identifiers are prefixed with interface identifier and suffixed with resource type, e.g.

- ‘GPIBO::01::INSTR’, where interface is GPIB, instrument address is 1, or
- ‘TCPIPO::192.32.6.100::3334::SOCKET’, where interface is TCP/IP, address is 192.32.6.100, socket is used instead of instrument protocol and port is 3334.

```
function r = visa_open(sAddress)

    global RM;
    global vi;
    r = 1;

    %if RM doesn't exist
    if isempty(RM) || RM == 0

        %initialize & open RM handle
        RM = int32(0);
        [A, B]= calllib('visa_dll', 'viOpenDefaultRM', RM);

        %error?
        if A ~= 0
            warndlg(['VISA error: Cannot open VISA Resource Manager!', ...
                'Is VISA installed?'], 'VISA error');
            return
        end
        %RM handle
        RM = B;
    end

    %open vi to instrument
    [C, ~, E] = calllib('visa_dll', 'viOpen', RM, sAddress, 0, 0, 0);

    %error?
    if C ~= 0
        warndlg(['VISA error: Cannot open VISA session to address ', ...
            "'", sAddress, "'!'], 'VISA error');
        return
    end
    %viSession handle
    vi = E;
    r = C;
end
```

Figure 12. Example VISA open function

5.2 Basic I/O using VISA

After viSession is successfully opened, the instrument I/O can be implemented with e.g. viWrite and viRead – functions.

The example below wraps VISA .dll write- and read calls, hiding unnecessary parameters from developer.

```
function r = visa_write(sCommand)

    global vi;
    %trim command, add linefeed
    sCommand = [strtrim(sCommand), char(10)];

    %write
    [A, ~, ~] = calllib('visa_dll', 'viWrite', vi, ...
        sCommand, length(sCommand), 0);

    if A ~= 0
        warndlg(['VISA error: Cannot write to VISA session "', ...
            "'", num2str(vi), "'!'], 'VISA error');
    end

    %return call result
    r = A;

end

function r = visa_read()

    global vi;

    %read from viSession
    [A, B, ~] = calllib('visa_dll', 'viRead', vi, blanks(4096), 4096, 0);

    if A ~= 0
        warndlg(['VISA error: Read error on VISA session "', ...
            "'", num2str(vi), "'!'], 'VISA error');
    end

    %trim response
    r = strtrim(B);

end
```

Figure 13. Example VISA write- and read functions

5.3 Loading an emulation file

Before PROPSIM can be used in a system, an Emulation File needs to be loaded. Below example illustrates conceptually how to ensure emulation file is closed, avoid timeout errors during loading, when to check error queue and how to revert timeout back to what it used to be.

```
function rslt = propsim_load_emulation(sFileName)
%This function loads emulation file to Propsim
%If Propsim is unable to load emulation, error is raised

%initialize output to error
rslt = 1;

%close emulation file
propsim_write('DIAG:SIMU:CLOSE');

%get VISA session timeout
tOut = visa_get_timeout();
%set timeout to temporary large value; 40sec in this case
visa_set_timeout(40000);

%load emulation file
propsim_write(['CALC:FILT:FILE ', sFileName]);
propsim_write('*OPC?');

%wait for *OPC? reply
propsim_read_str();

%replace original timeout
visa_set_timeout(tOut);

%check for loading errors
sErr = propsim_get_errors();

if sErr
    raise_error('Propsim Emulation File load error : "', sFileName, '"');
else
    rslt = 0;
end

end
```

Figure 14. MATLAB code example of loading an emulation file

5.4 Checking the error queue

When loading emulation to PROPSIM or setting emulator operation parameters, it is recommended to check error queue to e.g. avoid measurement loop running when emulation has not been successfully loaded to PROPSIM or emulation parameter settings are not valid.

```
function r = propsim_get_errors()
%This function reads Propsim error queue and outputs possible errors.
%If no errors are raised, the function returns empty string

    %initialize output to empty string
    r = '';

    %loop; if we run indefinitely, we might result in endless loop
    for i = 1 : 100

        %query error(s)
        sRes = propsim_write('SYST:ERR?');

        %check if VISA write was successful
        if strcmp(sRes,'ERROR') == 1
            raise_error('Propsim VISA write error');
            return
        end

        %read response
        sRes = propsim_read_str();

        %check for no error(s)
        if strcasecmp(sRes, '0,"No error"') == 1
            return
        else
            %concatenate error to output
            r = [r, sRes];
        end
    end
end
```

Figure 15. MATLAB code example of checking error queue

5.5 Example case: throughput as function of output gain

```
function test_gain_throughput()
    %initialize Propsim

    if propsim_connect('TCPIPO::192.32.6.100::3334::SOCKET')
        %errors? stop running.
        return
    end

    %reset Propsim, clear Status Register
    propsim_write('*RST');
    propsim_write('*CLS');

    %load emulation file
    if propsim_load_emulation('c:\User Emulations\ TestCase14b\TC14b.smu')
        %errors? stop running
        return
    end

    %cf 5705MHz
    dFreq = 5705;

    %initialize UE
    if ue_initialize(dFreq, 'TC14b.cfg');
        %errors? Stop running
        return
    end

    %initialize data store
    xml_initialize();

    %set Propsim ch1 - ch8 center frequencies
    for ch = 1 : 8
        %VISA write
        propsim_write(['CALC:FILT:CENT:CH ', ch, ',', dFreq, ';*OPC?']);
        %VISA read *OPC? response
        propsim_read_str();
    end

    %decrement gain from 0 to -50dB in -1dB steps
    for dGain = 0 : -1 : -50
        %set Propsim ch1 - ch8 gains
        for ch = 1 : 8
            propsim_write(['OUTP:GAIN:CH ', ch, ',', dGain, ';*OPC?']);
            propsim_read_str();
        end

        %measure throughput with UE, store results to XML DOM
        oTPData = ue_measure_throughput();
        xml_addtpnode(dGain, oTPData);
    end

    %save data to disk
    xml_save(['c:\test_results\ TestCase14b\tpdata_', getDateTime(), '.xml']);

    %close emulation
    propsim_write('DIAG:SIMU:CLOSE');

end
```

Figure 16. MATLAB example of throughput vs. gain